Contents

Articles

Manual:Customizing Hotspot 1
Manual:IP/Hotspot/User 12
Manual:IP/Hotspot/Walled Garden 15
Manual:Simple TE 17
Manual:API 25
User:Boen robot/API 39
Manual:IP/Address 52
Manual:IP/ARP 54
Manual:IPv6/Address 58
Manual:Router AAA 65
Manual:BGP based VPLS 71
Manual:BGP Best Path Selection Algorithm 78
Manual:BGP Case Studies 79
Manual:BGP HowTo & FAQ 86
Manual:BGP Load Balancing with two interfaces 91
Manual:BGP nexthop selection and validation in RouterOS 3.x 94
Manual:BGP soft reconfiguration alternatives in RouterOS 97
Manual:EBGP as PE-CE routing protocol 99
Manual:EoMPLS vs Cisco 105
Manual:Interface/Bonding 109
Manual:Interface/Bridge 117
Manual:MPLS L2VPN vs Juniper 126
Manual:Routing/BFD 137
Manual:Routing/BGP 140
Manual:Simple BGP Multihoming 147
Manual:BCP bridging (PPP tunnel bridging) 150
Manual:Bonding Examples 158
Manual:Bootloader upgrade 160
Manual:Console 161
Manual:Create Certificates 169
Manual:System/Certificates 171
Manual:CD Install 176
Manual:Configuration Management 182
Manual:Conformance Testing Mode 187

Manual:IP/Firewall/Connection tracking 187
Manual:Connection Rate 190
Manual:Console login process 193
Manual:CPU Usage 197
Manual:Default Configurations 198
Manual:IP/DHCP Client 203
Manual:IP/DHCP Relay 205
Manual:IP/DHCP Server 208
Manual:IP/DNS 215
Manual:Tools/Dynamic DNS 219
Manual:IPv6/DHCP Client 220
Manual:IPv6/DHCP Server 224
Manual:Interface/EoIP 227
Manual:Interface/Ethernet 231
Manual:Interface/Gre 235
Manual:Interface/Gre6 237
Manual:Tools/email 238
Manual:Tools/Ping 240
Manual:Routing/Routing filters 242
Manual:Tools/Fetch 245
Manual:Fast Path 247
Fast Path spanish 248
Manual:IP/Firewall 250
Manual:IPv6/Firewall 250
Manual:IP/Firewall/Address list 251
Manual:IP/Firewall/Filter 252
Manual:IPv6/Firewall/Filter 260
Manual:IP/Firewall/L7 261
Manual:IP/Firewall/Mangle 263
Manual:IP/Firewall/NAT 269
Manual:First time startup 275
Manual:Flashfig 279
Manual:FTP server 285
Manual:System/GPS 286
Manual:Tools/Graphing 288
Manual:Grounding 292
Manual:Hotspot Introduction 295
Manual:System/Health 299

Manual:IP/Hotspot 300
Manual:HTB 304
Manual:Interface/HWMPplus 313
Manual:Creating IPv6 loopback address 325
Manual:Interface 326
Manual:Interface/IPIP 328
Manual:IP 330
Manual:IPv6 330
Manual:IPv6 Overview 330
Manual:OSPFv3 with Quagga 335
Manual:Routing/IGMP-Proxy 338
Manual:Tools/IP-Scan 341
Manual:Initial Configuration 342
Manual:Internet access from VRF 364
Manual:Internet access from VRF with NAT 365
Manual:IP/Hotspot/Profile 368
Manual:IP/IPsec 371
Manual:IPv6/Firewall/Address-list 387
Manual:IPv6/Firewall/Mangle 387
Manual:IPv6/ND 387
Manual:IPv6/Neighbors 392
Manual:IPv6/Route 393
Manual:KVM 396
Manual:Entering a RouterOS License key 405
Manual:Interface/L2TP 408
Manual:License 414
Manual:Lua 420
Manual:System/LEDS 422
Manual:System/Log 423
Manual:System/UPS 430
Manual:Layer-3 MPLS VPN example 434
Manual:LCD TouchScreen 439
Manual:Limiting maximum number of prefixes accepted 441
Manual:Load balancing multiple same subnet links 442
MAC access 445
Manual:Making a simple wireless AP 448
Manual:Maximum Transmission Unit on RouterBoards 451
Manual:Metarouter 457

Manual:MLPPP over single and multiple links 465
Manual:MME wireless routing protocol 467
Manual:MPLS 470
Manual:MPLS over PPPoE 473
Manual:MPLS/EXP bit behaviour 478
Manual:MPLS/LDP 479
Manual:MPLS/Overview 481
Manual:MPLS/Traffic-eng 483
Manual:MPLSVPLS 487
Manual:Routing/Multicast 501
Manual:Multicast detailed example 507
Manual:Multicast SPT Switchover 514
Manual:My First IPv6 Network 516
Manual:IP/Neighbor discovery 520
Manual:Netinstall 522
Manual:Tools/Netwatch 529
Manual:NTH in RouterOS 3.x 531
Manual:Nv2 532
Manual:Interface/OVPN 537
Manual:OSPF as PE-CE routing protocol 540
Manual:OSPF Case Studies 544
Manual:OSPF Forwarding Address 560
Manual:OSPF-examples 562
Manual:Routing/OSPF 568
Manual:OSPF and Point-to-Point interfaces 577
Manual:Interface/PPPoE 578
Manual:Interface/PPTP 589
Manual:IP/Proxy 595
Manual:Packet Flow 606
Manual:Performance Testing with Traffic Generator 612
Manual:System/Packages 618
Manual:Tools/Profiler 621
Manual:IP/Packing 625
Manual:Password reset 627
Manual:PCC 630
Manual:PoE-Out 634
Manual:IP/Pools 641
Manual:IPv6/Pool 642

Manual:Port 644
Manual:IPv6 PD over PPP 646
Manual:PPP AAA 648
Manual:Routing/Prefix list 653
Proxylizer 654
Proxylizer/Getting Started 655
Proxylizer/Introduction 660
Manual:Purchasing a License for RouterOS 664
Manual:Replacement Key 666
Manual:Queue 667
Manual:Queues - Burst 677
Manual:Queues - PCQ 682
Manual:Queues - PCQ Examples 685
Manual:Queue Size 687
Manual:IP/Route 690
User:Prince500 698
Manual:RADIUS Client 706
Manual:Routing 716
Manual:Routing/RIP 716
Manual:RouterOS FAQ 719
Manual:RouterBOARD bad blocks 724
Manual:RouterOS features 725
Manual:Route Selection Algorithm in RouterOS 728
Manual:Routing Table Matcher 731
Manual:Routing/MME 733
Manual:Interface/SSTP 735
Manual:IP/Services 745
Manual:IP/Settings 748
Manual:Scripting 749
Manual:Scripting-examples 761
Manual:Simple Static IPv6 Routing 770
Manual:Simple Static Routing 772
Manual:Store 773
Manual:Support Output File 776
Manual:System 778
Manual:System/Scheduler 778
Manual:System/SSH client 781
Manual:Tools/Sigwatch 782

Manual:Tools/Sms 784
Manual:Using scope and target-scope attributes 787
Manual:SimTest 790
Manual:SNMP 799
Manual:IP/SOCKS 804
Manual:Special Login 807
Manual:Spectral scan 809
MikroTik Support 812
Manual:Switch Chip Features 814
Manual:System/File 821
Manual:System/LCD 822
Manual:System/Note 826
Manual:System/Resource 828
Manual:System/Serial Console 832
Manual:IP/SSH 836
Manual:IP/TFTP 837
Manual:System/Time 840
Manual:Tools 843
Manual:Tools/Traffic Generator 843
Manual:Connection oriented communication (TCP/IP) 853
Manual:TE tunnel auto bandwidth 859
Manual:TE Tunnels 862
Manual:TE Tunnels Example 867
Manual:The Dude/First use 872
Manual:The Dude/Installation 874
Manual:TOC 876
Manual:Tools/Bandwidth Test 879
Manual:Tools/Packet Sniffer 882
Manual:Tools/Traffic Monitor 889
Manual:Troubleshooting tools 890
Manual:Interface/Traffic Engineering 900
Manual:IP/Traffic Flow 902
Manual:IP/UPnP 906
Manual:User Manager 909
Manual:Upgrading RouterOS 912
Manual:Cisco VPLS 921
Manual:Interface/Virtual-ethernet 925
Manual:Interface/VLAN 926

Manual:Interface/VPLS 933
Manual:Interface/VRRP 936
Manual:Virtualization 943
Manual:VPLS Control Word 944
Manual:VRRP-examples 946
Manual:Virtual Routing and Forwarding 949
VRF Route Leaking 957
Manual:Interface/Wireless 959
Manual:System/Watchdog 989
Manual:Winbox 990
Manual:Wireless card diagnostics 1006
Manual:Wireless FAQ 1012
Manual:WMM 1016
Manual:Tools/Wake on lan 1018
Manual:Webfig 1018
Manual:Wireless Advanced Channels 1026
Manual:Wireless AP Client 1028
Manual:Wireless Debug Logs 1033
Manual:Wireless Station Modes 1037
Manual:Xen 1040

References

Article Sources and Contributors 1057
Image Sources, Licenses and Contributors 1062

Manual:Customizing Hotspot

Applies to RouterOS: v3, v4, v5+

HTML customizations

Summary

You can create a completely different set of servlet pages for each HotSpot server you have, specifying the directory it will be stored in html-directory property of a HotSpot server profile /ip hotspot profile. The default servlet pages are copied in the directory of your choice right after you create the profile. This directory can be accessed by connecting to the router with an FTP client. You can modify the pages as you like using the information from this section of the manual. Note that it is suggested to edit the files manually, as automated HTML editing tools may corrupt the pages by removing variables or other vital parts.

Available Pages

Main HTML servlet pages, which are shown to user:

- **redirect.html** - redirects user to another url (for example, to login page)
- **login.html** - login page shown to a user to ask for username and password. This page may take the following parameters:
  - **username** - username
  - **password** - either plain-text password (in case of PAP authentication) or MD5 hash of chap-id variable, password and CHAP challenge (in case of CHAP authentication). This value is used as e-mail address for trial users
  - **dst** - original URL requested before the redirect. This will be opened on successfull login
  - **popup** - whether to pop-up a status window on successfull login
  - **radius<id>** - send the attribute identified with <id> in text string form to the RADIUS server (in case RADIUS authentication is used; lost otherwise)
  - **radius<id>u** - send the attribute identified with <id> in unsigned integer form to the RADIUS server (in case RADIUS authentication is used; lost otherwise)
  - **radius<id>-<vnd-id>** - send the attribute identified with <id> and vendor ID <vnd-id> in text string form to the RADIUS server (in case RADIUS authentication is used; lost otherwise)
  - **radius<id>-<vnd-id>u** - send the attribute identified with <id> and vendor ID <vnd-id> in unsigned integer form to the RADIUS server (in case RADIUS authentication is used; lost otherwise)
- **md5.js** - JavaScript for MD5 password hashing. Used together with http-chap login method
- **alogin.html** - page shown after client has logged in. It pops-up status page and redirects browser to originally requested page (before he/she was redirected to the HotSpot login page)
- **status.html** - status page, shows statistics for the client. It is also able to display advertisements automatically
- **logout.html** - logout page, shown after user is logged out. Shows final statistics about the finished session. This page may take the following additional parameters:
  - **erase-cookie** - whether to erase cookies from the HotSpot server on logout (makes impossible to log in with cookie next time from the same browser, might be useful in multiuser environments)
- **error.html** - error page, shown on fatal errors only

Some other pages are available as well, if more control is needed:

- **rlogin.html** - page, which redirects client from some other URL to the login page, if authorization of the client is required to access that URL
- **rstatus.html** - similarly to rlogin.html, only in case if the client is already logged in and the original URL is not known
- **radvert.html** - redirects client to the scheduled advertisement link
- **flogin.html** - shown instead of login.html, if some error has happened (invalid username or password, for example)
- **fstatus.html** - shown instead of redirect, if status page is requested, but client is not logged in
- **flogout.html** - shown instead of redirect, if logout page is requested, but client is not logged in

Serving Servlet Pages

The HotSpot servlet recognizes 5 different request types:

1. **request for a remote host**
   - if user is logged in and advertisement is due to be displayed, radvert.html is displayed. This page makes redirect to the scheduled advertisment page
   - if user is logged in and advertisement is not scheduled for this user, the requested page is served
   - if user is not logged in, but the destination host is allowed by walled garden, then the request is also served
   - if user is not logged in, and the destination host is disallowed by walled garden, rlogin.html is displayed; if rlogin.html is not found, redirect.html is used to redirect to the login page
2. **request for "/" on the HotSpot host**
   - if user is logged in, rstatus.html is displayed; if rstatus.html is not found, redirect.html is used to redirect to the status page
   - if user is not logged in, rlogin.html is displayed; if rlogin.html is not found, redirect.html is used to redirect to the login page
3. **request for "/login" page**
   - if user has successfully logged in (or is already logged in), alogin.html is displayed; if alogin.html is not found, redirect.html is used to redirect to the originally requested page or the status page (in case, original destination page was not given)
   - if user is not logged in (username was not supplied, no error message appeared), login.html is showed
   - if login procedure has failed (error message is supplied), flogin.html is displayed; if flogin.html is not found, login.html is used
   - in case of fatal errors, error.html is showed
4. **request for "/status" page**
   - if user is logged in, status.html is displayed
   - if user is not logged in, fstatus.html is displayed; if fstatus.html is not found, redirect.html is used to redirect to the login page
5. **request for '/logout' page**
   - if user is logged in, logout.html is displayed
   - if user is not logged in, flogout.html is displayed; if flogout.html is not found, redirect.html is used to redirect to the login page

Note: If it is not possible to meet a request using the pages stored on the router's FTP server, Error 404 is displayed

There are many possibilities to customize what the HotSpot authentication pages look like:

- The pages are easily modifiable. They are stored on the router's FTP server in the directory you choose for the respective HotSpot server profile.
- By changing the variables, which client sends to the HotSpot servlet, it is possible to reduce keyword count to one (username or password; for example, the client's MAC address may be used as the other value) or even to zero (License Agreement; some predefined values general for all users or client's MAC address may be used as username and password)
- Registration may occur on a different server (for example, on a server that is able to charge Credit Cards). Client's MAC address may be passed to it, so that this information need not be written in manually. After the registration, the server should change RADIUS database enabling client to log in for some amount of time.

To insert variable in some place in HTML file, the $(var_name) syntax is used, where the "var_name" is the name of the variable (without quotes). This construction may be used in any HotSpot HTML file accessed as '/', '/login', '/status' or '/logout', as well as any text or HTML (.txt, .htm or .html) file stored on the HotSpot server (with the exception of traffic counters, which are available in status page only, and **error**, **error-orig**, **chap-id**, **chap-challenge** and **popup** variables, which are available in login page only). For example, to show a link to the login page, following construction can be used:

```
<a href="$(link-login)">login</a>
```

Variables

All of the Servlet HTML pages use variables to show user specific values. Variable names appear only in the HTML source of the servlet pages - they are automatically replaced with the respective values by the HotSpot Servlet. For most variables there is an example of their possible value included in brackets. All the described variables are valid in all servlet pages, but some of them just might be empty at the time they are accesses (for example, there is no uptime before a user has logged in).

List of available variables

Common server variables:

- **`hostname`** - DNS name or IP address (if DNS name is not given) of the HotSpot Servlet ("hotspot.example.net")
- **`identity`** - RouterOS identity name ("MikroTik")
- **`login-by`** - authentication method used by user
- **`plain-passwd`** - a "yes/no" representation of whether HTTP-PAP login method is allowed ("no")
- **`server-address`** - HotSpot server address ("10.5.50.1:80")
- **`ssl-login`** - a "yes/no" representation of whether HTTPS method was used to access that servlet page ("no")
- **`server-name`** - HotSpot server name (set in the /ip hotspot menu, as the name property)

Links:

- **`link-login`** - link to login page including original URL requested ("http://10.5.50.1/login?dst=http://www.example.com/")
- **`link-login-only`** - link to login page, not including original URL requested ("http://10.5.50.1/login")
- **`link-logout`** - link to logout page ("http://10.5.50.1/logout")
- **`link-status`** - link to status page ("http://10.5.50.1/status")
- **`link-orig`** - original URL requested ("http://www.example.com/")

General client information:

- **`domain`** - domain name of the user ("example.com")

- **interface-name** - physical HotSpot interface name (in case of bridged interfaces, this will return the actual bridge port name)
- **ip** - IP address of the client ("10.5.50.2")
- **logged-in** - "yes" if the user is logged in, otherwise - "no" ("yes")
- **mac** - MAC address of the user ("01:23:45:67:89:AB")
- **trial** - a "yes/no" representation of whether the user has access to trial time. If users trial time has expired, the value is "no"
- **username** - the name of the user ("John")
- **host-ip** - client IP address from /ip hotspot host table

User status information:

- **idle-timeout** - idle timeout ("20m" or "" if none)
- **idle-timeout-secs** - idle timeout in seconds ("88" or "0" if there is such timeout)
- **limit-bytes-in** - byte limit for send ("1000000" or "---" if there is no limit)
- **limit-bytes-out** - byte limit for receive ("1000000" or "---" if there is no limit)
- **refresh-timeout** - status page refresh timeout ("1m30s" or "" if none)
- **refresh-timeout-secs** - status page refresh timeout in seconds ("90s" or "0" if none)
- **session-timeout** - session time left for the user ("5h" or "" if none)
- **session-timeout-secs** - session time left for the user, in seconds ("3475" or "0" if there is such timeout)
- **session-time-left** - session time left for the user ("5h" or "" if none)
- **session-time-left-secs** - session time left for the user, in seconds ("3475" or "0" if there is such timeout)
- **uptime** - current session uptime ("10h2m33s")
- **uptime-secs** - current session uptime in seconds ("125")

Traffic counters, which are available only in the status page:

- **bytes-in** - number of bytes received from the user ("15423")
- **bytes-in-nice** - user-friendly form of number of bytes received from the user ("15423")
- **bytes-out** - number of bytes sent to the user ("11352")
- **bytes-out-nice** - user-friendly form of number of bytes sent to the user ("11352")
- **packets-in** - number of packets received from the user ("251")
- **packets-out** - number of packets sent to the user ("211")
- **remain-bytes-in** - remaining bytes until limit-bytes-in will be reached ("337465" or "---" if there is no limit)
- **remain-bytes-out** - remaining bytes until limit-bytes-out will be reached ("124455" or "---" if there is no limit)

Miscellaneous variables:

- **session-id** - value of 'session-id' parameter in the last request
- **var** - value of 'var' parameter in the last request
- **error** - error message, if something failed ("invalid username or password")
- **error-orig** - original error message (without translations retrieved from errors.txt), if something failed ("invalid username or password")
- **chap-id** - value of chap ID ("\371")
- **chap-challenge** - value of chap challenge ("\357\015\330\013\021\234\145\245\303\253\142\246\133\175\375\316")
- **popup** - whether to pop-up checkbox ("true" or "false")
- **advert-pending** - whether an advertisement is pending to be displayed ("yes" or "no")
- **http-status** - allows to set http status code and message
- **http-header** - allows to add http header

RADIUS-related variables:

- **radius<id>** - show the attribute identified with <id> in text string form (in case RADIUS authentication was used; "" otherwise)
- **radius<id>u** - show the attribute identified with <id> in unsigned integer form (in case RADIUS authentication was used; "0" otherwise)
- **radius<id>-<vnd-id>** - show the attribute identified with <id> and vendor ID <vnd-id> in text string form (in case RADIUS authentication was used; "" otherwise)
- **radius<id>-<vnd-id>u** - show the attribute identified with <id> and vendor ID <vnd-id> in unsigned integer form (in case RADIUS authentication was used; "0" otherwise)

Working with variables

$(if <var_name>) statements can be used in theses pages. Following content will be included, if value of <var_name> will not be an empty string. It is an equivalent to $(if <var_name> != "") It is possible to compare on equivalence as well: $(if <var_name> == <value>) These statements have effect until $(elif <var_name>), $(else) or $(endif). In general case it looks like this:

```
some content, which will always be displayed
$(if username == john)
Hey, your username is john
$(elif username == dizzy)
Hello, Dizzy! How are you? Your administrator.
$(elif ip == 10.1.2.3)
You are sitting at that crappy computer, which is damn slow...
$(elif mac == 00:01:02:03:04:05)
This is an ethernet card, which was stolen few months ago...
$(else)
I don't know who you are, so lets live in peace.
$(endif)
other content, which will always be displayed
```

Only one of those expressions will be shown. Which one - depends on values of those variables for each client.

Redirects and custom Headers

Starting from RouterOS 5.12 there are 2 new hotspot html page variables:

- **http-status** - allows to set http status code and message
- **http-header** - allows to add http header

Example:

```
      $(if http-status == 302)Hotspot login required$(endif)
      $(if http-header == "Location")$(link-redirect)$(endif)
```

In case if $(link-redirect) will evaluate to "http://192.168.88.1/login", then HTTP response will look like:

```
HTTP/1.0 302 Hotspot login required
<regular HTTP headers>
Location: http://192.168.88.1/login
```

http-status syntax:

```
      $(if http-status == XYZ)HTTP_STATUS_MESSAGE$(endif)
```

- *XYZ* - status code, should be 3 decimal digits, first one must not be 0
- *HTTP_STATUS_MESSAGE* - any text, will follow status code in HTTP reply

In HTTP response it will be on first line and will look like:

```
HTTP/1.0 XYZ HTTP_STATUS_MESSAGE
```

http-header syntax:

```
      $(if http-header == HTTP_HEADER_NAME)HTTP_HEADER_VALUE$(endif)
```

- *HTTP_HEADER_NAME* - name of the HTTP header to add
- *HTTP_HEADER_VALUE* - value of HTTP header with name HTTP_HEADER_NAME

In HTTP response it will look like:

```
HTTP_HEADER_NAME: HTTP_HEADER_VALUE
```

All variables and conditional expressions within HTTP_HEADER_VALUE and HTTP_STATUS_MESSAGE are processed as usual.

In case multiple headers with the same name are added, then only the last one will be used (previous ones will be discarded). It allows to override regular HTTP headers (for example, Content-Type and Cache-Control).

Customizing Error Messages

All error messages are stored in the errors.txt file within the respective HotSpot servlet directory. You can change and translate all these messages to your native language. To do so, edit the errors.txt file. You can also use variables in the messages. All instructions are given in that file.

Multiple Versions of HotSpot Pages

Multiple HotSpot page sets for the same HotSpot server are supported. They can be chosen by user (to select language) or automatically by JavaScript (to select PDA/regular version of HTML pages).

To utilize this feature, create subdirectories in HotSpot HTML directory, and place those HTML files, which are different, in that subdirectory. For example, to translate everything in Latvian, subdirectory "lv" can be created with login.html, logout.html, status.html, alogin.html, radvert.html and errors.txt files, which are translated into Latvian. If the requested HTML page can not be found in the requested subdirectory, the corresponding HTML file from the main directory will be used. Then main login.html file would contain link to "/lv/login?dst=$(link-orig-esc)", which then displays Latvian version of login page: Latviski . And Latvian version would contain link to English version: English

Another way of referencing directories is to specify 'target' variable:

```
        <a href="$(link-login-only)?dst=$(link-orig-esc)&target=lv">Latviski</a>
        <a href="$(link-login-only)?dst=$(link-orig-esc)&target=%2F">English</a>
```

After preferred directory has been selected (for example, "lv"), all links to local HotSpot pages will contain that path (for example, $(link-status) = "http://hotspot.mt.lv/lv/status"). So, if all HotSpot pages reference links using "$(link-xxx)" variables, then no more changes are to be made - each client will stay within the selected directory all the time.

Misc

If you want to use HTTP-CHAP authentication method it is supposed that you include the **doLogin()** function (which references to the **md5.js** which must be already loaded) before the **Submit** action of the login form. Otherwise, CHAP login will fail.

The resulting password to be sent to the HotSpot gateway in case of HTTP-CHAP method, is formed MD5-hashing the concatenation of the following: chap-id, the password of the user and chap-challenge (in the given order)

In case variables are to be used in link directly, then they must be escaped accordingly. For example, in login page, **link** will not work as intended, if username will be "123&456=1 2". In this case instead of $(user), its escaped version must be used: $(user-esc): **link**. Now the same username will be converted to "123%26456%3D1+2", which is the valid representation of "123&456=1 2" in URL. This trick may be used with any variables, not only with $(username).

There is a boolean parameter "erase-cookie" to the logout page, which may be either "on" or "true" to delete user cookie on logout (so that the user would not be automatically logged on when he/she opens a browser next time.

Examples

With basic HTML language knowledge and the examples below it should be easy to implement the ideas described above.

- To provide predefined value as username, in login.html change:

```
<type="text" value="$(username)>
```

to this line:

```
<input type="hidden" name="username" value="hsuser">
```

(where hsuser is the username you are providing)

- To provide predefined value as password, in login.html change:

```
<input type="password">
```

to this line:

```
<input type="hidden" name="password" value="hspass">
```

(where hspass is the password you are providing)

- To send client's MAC address to a registration server in form of:

https://www.example.com/register.html?mac=XX:XX:XX:XX:XX:XX

change the Login button link in login.html to:

```
https://www.example.com/register.html?mac=$(mac)
```

(you should correct the link to point to your server)

- To show a banner after user login, in alogin.html after

$(if popup == 'true') add the following line:

```
open('http://www.example.com/your-banner-page.html', 'my-banner-name','');
```

(you should correct the link to point to the page you want to show)

- To choose different page shown after login, in login.html change:

```
<input type="hidden" name="dst" value="$(link-orig)">
```

to this line:

```
<input type="hidden" name="dst" value="http://www.example.com">
```

(you should correct the link to point to your server)

- To erase the cookie on logoff, in the page containing link to the logout (for example, in status.html) change:

```
open('$(link-logout)', 'hotspot_logout', ...
```

to this:

```
open('$(link-logout)?erase-cookie=on', 'hotspot_logout', ...
```

or alternatively add this line:

```
<input type="hidden" name="erase-cookie" value="on">
```

before this one:

```
<input type="submit" value="log off">
```

An another example is making HotSpot to authenticate on a remote server (which may, for example, perform creditcard charging):

- Allow direct access to the external server in walled-garden (either HTTP-based, or IP-based)
- Modify login page of the HotSpot servlet to redirect to the external authentication server. The external server should modify RADIUS database as needed

  Here is an example of such a login page to put on the HotSpot router (it is redirecting to https://auth.example.com/login.php, replace with the actual address of an external authentication server):

```
<html>
<title>...</title>
<body>
<form name="redirect" action="https://auth.example.com/login.php" method="post">
<input type="hidden" name="mac" value="$(mac)">
<input type="hidden" name="ip" value="$(ip)">
<input type="hidden" name="username" value="$(username)">
<input type="hidden" name="link-login" value="$(link-login)">
<input type="hidden" name="link-orig" value="$(link-orig)">
<input type="hidden" name="error" value="$(error)">
</form>
<script language="JavaScript">
<!--
      document.redirect.submit();
//-->
</script>
</body>
</html>
```

- The external server can log in a HotSpot client by redirecting it back to the original HotSpot servlet login page, specifying the correct username and password

Here is an example of such a page (it is redirecting to https://hotspot.example.com/login, replace with the actual address of a HotSpot router; also, it is displaying www.mikrotik.com after successful login, replace with what needed):

```
<html>
<title>Hotspot login page</title>
<body>
<form name="login" action="https://hotspot.example.com/login" method="post">
<input type="text" name="username" value="demo">
<input type="password" name="password" value="none">
<input type="hidden" name="domain" value="">
<input type="hidden" name="dst" value="http://www.mikrotik.com/">
<input type="submit" name="login" value="log in">
</form>
</body>
</html>
```

- Hotspot will ask RADIUS server whether to allow the login or not. If not allowed, alogin.html page will be displayed (it can be modified to do anything). If not allowed, flogin.html (or login.html) page will be displayed, which will redirect client back to the external authentication server.

Note: as shown in these examples, HTTPS protocol and POST method can be used to secure communications.

Firewall customizations

Summary

Apart from the obvious dynamic entries in the /ip hotspot submenu itself (like hosts and active users), some additional rules are added in the firewall tables when activating a HotSpot service. Unlike RouterOS version 2.8, there are relatively few firewall rules added in the firewall as the main job is made by the one-to-one NAT algorithm.

NAT

From **/ip firewall nat print dynamic** command, you can get something like this (comments follow after each of the rules):

```
 0 D chain=dstnat action=jump jump-target=hotspot hotspot=from-client
```

Putting all HotSpot-related tasks for packets from all HotSpot clients into a separate chain.

```
 1 I chain=hotspot action=jump jump-target=pre-hotspot
```

Any actions that should be done before HotSpot rules apply, should be put in the pre-hotspot chain. This chain is under full administrator control and does not contain any rules set by the system, hence the invalid jump rule (as the chain does not have any rules by default).

```
 2 D chain=hotspot action=redirect to-ports=64872 dst-port=53 protocol=udp
 3 D chain=hotspot action=redirect to-ports=64872 dst-port=53 protocol=tcp
```

Redirect all DNS requests to the HotSpot service. The 64872 port provides DNS service for all HotSpot users. If you want HotSpot server to listen also to another port, add rules here the same way, changing dst-port property.

```
 4 D chain=hotspot action=redirect to-ports=64873 hotspot=local-dst dst-port=80
     protocol=tcp
```

Redirect all HTTP login requests to the HTTP login servlet. The 64873 is HotSpot HTTP servlet port.

```
 5 D chain=hotspot action=redirect to-ports=64875 hotspot=local-dst dst-port=443
     protocol=tcp
```

Redirect all HTTPS login requests to the HTTPS login servlet. The 64875 is HotSpot HTTPS servlet port.

```
 6 D chain=hotspot action=jump jump-target=hs-unauth hotspot=!auth protocol=tcp
```

All other packets except DNS and login requests from unauthorized clients should pass through the hs-unauth chain.

```
 7 D chain=hotspot action=jump jump-target=hs-auth hotspot=auth protocol=tcp
```

And packets from the authorized clients - through the hs-auth chain.

```
 8 D ;;; www.mikrotik.com
     chain=hs-unauth action=return dst-address=66.228.113.26 dst-port=80 protocol=tcp
```

First in the **hs-unauth** chain is put everything that affects TCP protocol in the `/ip hotspot walled-garden ip` submenu (i.e., everything where either protocol is not set, or set to TCP). Here we are excluding www.mikrotik.com from being redirected to the login page.

```
 9 D chain=hs-unauth action=redirect to-ports=64874 dst-port=80 protocol=tcp
```

All other HTTP requests are redirected to the Walled Garden proxy server which listens the 64874 port. If there is an allow entry in the `/ip hotspot walled-garden` menu for an HTTP request, it is being forwarded to the destination. Otherwise, the request will be automatically redirected to the HotSpot login servlet (port 64873).

```
10 D chain=hs-unauth action=redirect to-ports=64874 dst-port=3128 protocol=tcp
11 D chain=hs-unauth action=redirect to-ports=64874 dst-port=8080 protocol=tcp
```

HotSpot by default assumes that only these ports may be used for HTTP proxy requests. These two entries are used to "catch" client requests to unknown proxies (you can add more rules here for other ports). I.e., to make it possible for the clients with unknown proxy settings to work with the HotSpot system. This feature is called "Universal Proxy". If it is detected that a client is using some proxy server, the system will automatically mark that packets with the http hotspot mark to work around the unknown proxy problem, as we will see later on. Note that the port used (64874) is the same as for HTTP requests in the rule #9 (so both HTTP and HTTP proxy requests are processed by the same code).

```
12 D chain=hs-unauth action=redirect to-ports=64875 dst-port=443 protocol=tcp
```

HTTPS proxy is listening on the 64875 port.

```
13 I chain=hs-unauth action=jump jump-target=hs-smtp dst-port=25 protocol=tcp
```

Redirect for SMTP protocol may also be defined in the HotSpot configuration. In case it is, a redirect rule will be put in the hs-smtp chain. This is done so that users with unknown SMTP configuration would be able to send their mail through the service provider's (your) SMTP server instead of going to the [possibly unavailable outside their network of origin] SMTP server users have configured on their computers. The chain is empty by default, hence the invalid jump rule.

```
14 D chain=hs-auth action=redirect to-ports=64874 hotspot=http protocol=tcp
```

Providing HTTP proxy service for authorized users. Authenticated user requests may need to be subject to transparent proxying (the "Universal Proxy" technique and advertisement feature). This http mark is put automatically on the HTTP proxy requests to the servers detected by the HotSpot HTTP proxy (the one that is listening on the 64874 port) as HTTP proxy requests for unknown proxy servers. This is done so that users that have some proxy settings would use the HotSpot gateway instead of the [possibly unavailable outside their network of origin] proxy server users have configured in their computers. This mark is also applied when advertisement is due to be shown to the user, as well as on any HTTP requests done form the users whose profile is configured to transparently proxy their requests.

```
15 I chain=hs-auth action=jump jump-target=hs-smtp dst-port=25 protocol=tcp
```

Providing SMTP proxy for authorized users (the same as in rule #13).

Packet Filtering

From **/ip firewall filter print dynamic** command, you can get something like this (comments follow after each of the rules):

```
 0 D chain=forward action=jump jump-target=hs-unauth hotspot=from-client,!auth
```

Any packet that traverse the router from an unauthorized client will be sent to the **hs-unauth** chain. The hs-unauth implements the IP-based Walled Garden filter.

```
 1 D chain=forward action=jump jump-target=hs-unauth-to hotspot=to-client,!auth
```

Everything that comes to clients through the router, gets redirected to another chain, called **hs-unauth-to**. This chain should reject unauthorized requests to the clients.

```
 2 D chain=input action=jump jump-target=hs-input hotspot=from-client
```

Everything that comes from clients to the router itself, gets to yet another chain, called **hs-input**.

```
 3 I chain=hs-input action=jump jump-target=pre-hs-input
```

Before proceeding with [predefined] dynamic rules, the packet gets to the administratively controlled **pre-hs-input** chain, which is empty by default, hence the invalid state of the jump rule.

```
 4 D chain=hs-input action=accept dst-port=64872 protocol=udp
 5 D chain=hs-input action=accept dst-port=64872-64875 protocol=tcp
```

Allow client access to the local authentication and proxy services (as described earlier).

```
 6 D chain=hs-input action=jump jump-target=hs-unauth hotspot=!auth
```

All other traffic from unauthorized clients to the router itself will be treated the same way as the traffic traversing the routers.

```
 7 D chain=hs-unauth action=return protocol=icmp
 8 D ;;; www.mikrotik.com
     chain=hs-unauth action=return dst-address=66.228.113.26 dst-port=80 protocol=tcp
```

Unlike NAT table where only TCP-protocol related Walled Garden entries were added, in the packet filter **hs-unauth** chain is added everything you have set in the **/ip hotspot walled-garden ip** menu. That is why although you have seen only one entry in the NAT table, there are two rules here.

```
 9 D chain=hs-unauth action=reject reject-with=tcp-reset protocol=tcp
10 D chain=hs-unauth action=reject reject-with=icmp-net-prohibited
```

Everything else that has not been while-listed by the Walled Garden will be rejected. Note usage of TCP Reset for rejecting TCP connections.

```
11 D chain=hs-unauth-to action=return protocol=icmp
12 D ;;; www.mikrotik.com
     chain=hs-unauth-to action=return src-address=66.228.113.26 src-port=80 protocol=tcp
```

Same action as in rules #7 and #8 is performed for the packets destined to the clients (chain **hs-unauth-to**) as well.

```
13 D chain=hs-unauth-to action=reject reject-with=icmp-host-prohibited
```

Reject all packets to the clients with ICMP reject message.

[Top | Back to Content **]**

Manual:IP/Hotspot/User

Applies to RouterOS: v3, v4, v5+

Users

Sub-menu: `/ip hotspot user`

This is the menu, where client's user/password information is actually added, additional configuration options for HotSpot users are configured here as well.

Properties

Property	Description
address (*IP*; Default: **0.0.0.0**)	IP address, when specified client will get the address from the HotSpot one-to-one NAT translations. Address does not restrict HotSpot login only from this address
comment (*string*; Default:)	descriptive information for HotSpot user, it might be used for scripts to change parameters for specific clients
email (*string*; Default:)	HotSpot client's e-mail, informational value for the HotSpot user
limit-bytes-in (*integer*; Default: **0**)	Maximal amount of bytes that can be received from the user. User is disconnected from HotSpot after the limit is reached.
limit-bytes-out (*integer*; Default: **0**)	Maximal amount of bytes that can be transmitted from the user. User is disconnected from HotSpot after the limit is reached.
limit-bytes-total (*integer*; Default: **0**)	(limit-bytes-in+limit-bytes-out). User is disconnected from HotSpot after the limit is reached.
limit-uptime (*time*; Default: **0**)	Uptime limit for the HotSpot client, user is disconnected from HotSpot as soon as uptime is reached.
mac-address (*MAC*; Default: **00:00:00:00:00:00**)	Client is allowed to login only from the specified MAC-address. If value is *00:00:00:00:00:00*, any mac address is allowed.
name (*string*; Default:)	HotSpot login page username, when MAC-address authentication is used name is configured as client's MAC-address
password (*string*; Default:)	User password

profile (*string*; Default: **default**) User profile configured in */ip hotspot user profile*

routes (*string*; Default:) Routes added to HotSpot gateway when client is connected. The route format **dst-address gateway metric** (for example, *192.168.1.0/24 192.168.0.1 1*)

server (*string* | *all*; Default: **all**) HotSpot server's name to which user is allowed login

Read-only proterties

Property	Description
bytes-in (*integer*)	
bytes-out (*integer*)	
packets-in (*integer*)	
packets-out (*integer*)	
uptime (*time*)	

User Profile

Sub-menu: `/ip hotspot user profile`

User profile menu is used for common HotSpot client settings. Profiles are like User groups with the same set of settings, rate-limit, filter chain name, etc.

Properties

Property	Description
address-list (*string*; Default:)	Name of the address list in which users IP address will be added. Useful to mark traffic per user groups for queue tree configurations.
address-pool (*string* |*none*; Default: **none**)	IP pool name from which the user will get IP. When user has improper network settings configuration on the computer, HotSpot server makes translation and assigns correct IP address from the pool instead of incorrect one
advertise (*yes* | *no*; Default: **no**)	Enable forced advertisement popups. After certain interval specific web-page is being displayed for HotSpot users. Advertisement page might be blocked by browsers popup blockers.
advertise-interval (*time[,time[,..]]*; Default: **30m,10m**)	Set of intervals between advertisement popups. After the list is done, the last value is used for all further advertisements, 10 minutes
advertise-timeout (*time* | *immediately* | *never*; Default: **1m**)	How long advertisement is shown, before blocking network access for HotSpot client. Connection to Internet is not allowed, when advertisement is not shown.
advertise-url (*string[,string[,..]]*; Default:)	List of URLs that is show for advertisement popups. After the last URL is used, list starts from the begining.
idle-timeout (*time* | *none*; Default: **none**)	Maximal period of inactivity for authorized HotSpot clients. Timer is counting, when there is no traffic coming from that client and going through the router, for example computer is switched off. User is logged out, dropped of the host list, the address used by the user is freed, when timeout is reached.
incoming-filter (*string*; Default:)	Name of the firewall chain applied to incoming packets from the users of this profile, jump rule is required from built-in chain (input, forward, output) to chain=hotspot
incoming-packet-mark (*string*; Default:)	Packet mark put on incoming packets from every user of this profile
keepalive-timeout (*time* | *none*; Default:)	Keepalive timeout for authorized HotSpot clients. Used to detect, that the computer of the client is alive and reachable. User is logged out, when timeout value is reached
name (*string*; Default:)	Descriptive name of the profile

on-login (*string*; Default: "")	Script name to be executed, when user logs in to the HotSpot from the particular profile. It is possible to get username from internal **user** variable. For example, *:log info "User $user logged in!"*
on-logout (*string*; Default: "")	Script name to be executed, when user logs out from the HotSpot. It is possible to get username from internal **user** variable. For example, *:log info "User $user logged in!"*
open-status-page (*always* | *http-login*; Default: **always**)	Option to show status page for user authenticated with mac login method. For example to show advertisement on status page (alogin.html) • **http-login** - open status page only for HTTP login (includes cookie and HTTPS) • **always** - open HTTP status page in case of mac login as well
outgoing-filter (*string*; Default:)	Name of the firewall chain applied to outgoing packets from the users of this profile, jump rule is required from built-in chain (input, forward, output) to chain=hotspot
outgoing-packet-mark (*string*; Default:)	Packet mark put on outgoing packets from every user of this profile
rate-limit (*string*; Default: "")	Simple dynamic queue is created for user, once it logs in to the HotSpot. Rate-limitation is configured in the following form **[rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time] [priority] [rx-rate-min[/tx-rate-min]]]]**. For example, to set 1M download, 512k upload for the client, rate-limit=512k/1M
session-timeout (*time*; Default: **0s**)	Allowed session time for client. After this time, the user is logged out unconditionally
shared-users (*integer*; Default: **1**)	Allowed number of simultaneously logged in users with the same HotSpot username
status-autorefresh (*time* | *none*; Default: **none**)	HotSpot status page autorefresh interval
transparent-proxy (*yes* |; Default: **yes**)	Use transparent HTTP proxy for the authorized users of this profile

[Top | Back to Content]

Manual:IP/Hotspot/Walled Garden

Applies to RouterOS: v3, v4, v5+

Walled Garden

Sub-menu: `/ip hotspot walled-garden`

HTTP walled-garden, menu allows to set authentication bypass for HTTP and HTTPs resources

Properties

Property	Description
action (*allow* | *deny*; Default: **allow**)	Action to perform, when packet matches the rule • **allow** - allow access to the web-page without authorization • **deny** - the authorization is required to access the web-page
server (*string*; Default:)	Name of the HotSpot server, rule is applied to.
src-address (*IP*; Default:)	Source address of the user, usually IP address of the HotSpot client
method (*string*; Default:)	HTTP method of the request
dst-host (*string*; Default:)	Domain name of the destination web-server
dst-port (*integer*; Default:)	TCP port number, client sends request to
path (*string*; Default:)	The path of the request, path comes after '"http://dst_host/"'

Read-only properties

Property	Description
dst-address (*IP*)	
hits (*integer*)	

IP Walled Garden

Sub-menu: `/ip hotspot walled-garden ip`

Walled-garden menu for the IP requests (Winbox, SSH, Telnet, SIP, etc.)

Properties

Property	Description
action (*allow* | *deny* | *reject*; Default: **allow**)	Action to perform, when packet matches the rule • **allow** - allow access to the web-page without authorization • **deny** - the authorization is required to access the web-page • **reject** - the authorization is required to access the resource, ICMP reject message will be sent to client, when packet will match the rule
server (*string*; Default:)	Name of the HotSpot server, rule is applied to.
src-address (*IP*; Default:)	Source address of the user, usually IP address of the HotSpot client
dst-address (*IP*; Default:)	Destination IP address, IP address of the WEB-server. Ignored if **dst-host** is already specified.
dst-host (*string*; Default:)	Domain name of the destination web-server. When this parameter is specified dynamic entry is added to Walled Garden
dst-port (*integer*; Default:)	TCP port number, client sends request to
protocol (*integer* | *string*; Default:)	IP protocol

Example

When adding walled garden IP entry several dynamic rules are created. For example, lets add *www.paypalobject.com*

```
/ip hotspot walled-garden ip
add action=accept disabled=no dst-host=www.paypalobject.com
```

Now if you look at walled garden menu you will see dynamic entry for object we just added

```
[admin@493G] /ip hotspot walled-garden> print detail
Flags: X - disabled, D - dynamic
 0 D ;;; www.paypalobject.com
     dst-address=68.178.232.99 action=allow hits=0
```

Also dynamic firewall and NAT rules are added to allow paypalobject.com resolved address

```
[admin@493G] /ip firewall filter> print dynamic
Flags: X - disabled, I - invalid, D - dynamic
...
 7 D ;;; www.paypalobject.com
     chain=hs-unauth action=return dst-address=68.178.232.99
...

10 D ;;; www.paypalobject.com
     chain=hs-unauth-to action=return src-address=68.178.232.99
```

```
[admin@493G] /ip firewall nat> print dynamic
Flags: X - disabled, I - invalid, D - dynamic
...
 8 D ;;; www.paypalobject.com
     chain=hs-unauth action=return dst-address=68.178.232.99
...
```

[Top | Back to Content]

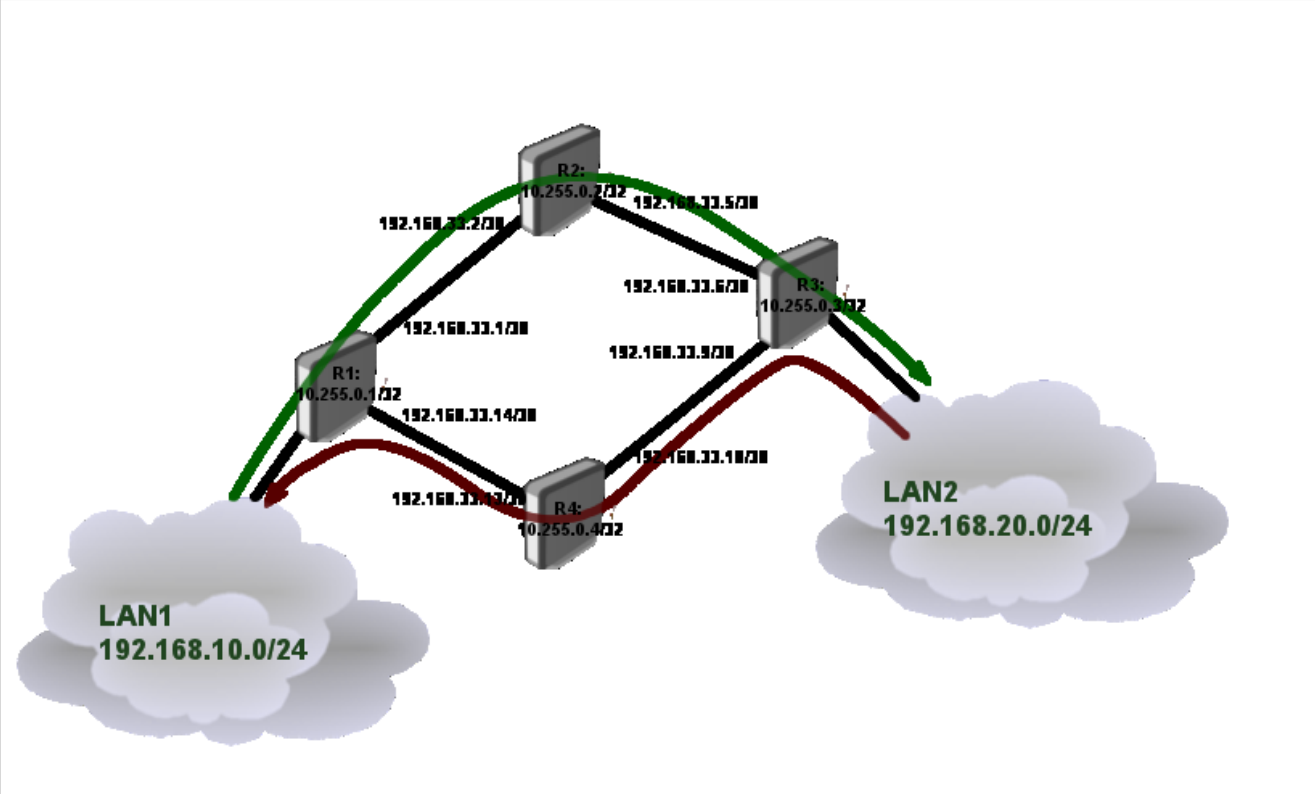
Manual:Simple TE

Summary

This article shows how to simply create traffic engineering tunnels using both dynamic and static tunnel paths.It also shows how to steer traffic over the tunnel.

Network Layout

We will create a network consisting of four routers connected in diamond shape as illustrated in diagram below.



Each router is connected to neighboring router using /30 network and each of them have unique loopback address form 10.255.0.x network. Loopback addresses will be used as tunnel source and destination.

The goal is to interconnect two LAN segments (Lan1, Lan2) using TE tunnels in the way that:

- traffic in direction from LAN1 to LAN2 goes over path through R2
- traffic in direction from LAN2 to LAN1 goes over path through R4

Router Configurations

Connectivity between routers and Loopback addresses

R1

```
/system identity set name=R1

/interface bridge add name=Loopback

/ip address
```

```
add address=192.168.33.1/30 interface=ether1
add address=192.168.33.14/30 interface=ether2
add address=192.168.10.1/24 interface=ether3
add address=10.255.0.1/32 interface=Loopback
```

R2

```
/system identity set name=R2

/interface bridge add name=Loopback

/ip address
add address=192.168.33.2/30 interface=ether1
add address=192.168.33.5/30 interface=ether2
add address=10.255.0.2/32 interface=Loopback
```

R3

```
/system identity set name=R3

/interface bridge add name=Loopback

/ip address
add address=192.168.33.6/30 interface=ether1
add address=192.168.33.9/30 interface=ether2
add address=192.168.20.1/24 interface=ether3
add address=10.255.0.3/32 interface=Loopback
```

R4

```
/system identity set name=R4

/interface bridge add name=Loopback

/ip address
add address=192.168.33.10/30 interface=ether1
add address=192.168.33.13/30 interface=ether2
add address=10.255.0.4/32 interface=Loopback
```

Loopback address reachability and CSPF setup

In this setup we will use OSPF dynamic routing protocol to distribute routing information between routers. To successfully complete the setup we need loopback reachability information on every router.

CSPF will also be configured (extension of OSPF) to carry TE reservation information.

R1

```
/routing ospf instance
set default router-id=10.255.0.1 mpls-te-area=backbone mpls-te-router-id=Loopback

/routing ospf network
add network=192.168.33.0/24 area=backbone
```

```
add network=10.255.0.1/32 area=backbone
```

R2

```
/routing ospf instance
set default router-id=10.255.0.2 mpls-te-area=backbone mpls-te-router-id=Loopback

/routing ospf network
add network=192.168.33.0/24 area=backbone
add network=10.255.0.2/32 area=backbone
```

R3

```
/routing ospf instance
set default router-id=10.255.0.3 mpls-te-area=backbone mpls-te-router-id=Loopback

/routing ospf network
add network=192.168.33.0/24 area=backbone
add network=10.255.0.3/32 area=backbone
```

R4

```
/routing ospf instance
set default router-id=10.255.0.4 mpls-te-area=backbone mpls-te-router-id=Loopback

/routing ospf network
add network=192.168.33.0/24 area=backbone
add network=10.255.0.4/32 area=backbone
```

After OSPF is set up verify that we have correct routing information in routing table of each router:

```
[admin@R1] /ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 0 ADS  0.0.0.0/0                          10.5.101.1                1
 1 ADC  10.255.0.1/32      10.255.0.1      lo                        0
 2 ADo  10.255.0.2/32                      192.168.33.2            110
 3 ADo  10.255.0.3/32                      192.168.33.2            110
                                           192.168.33.13
 4 ADo  10.255.0.4/32                      192.168.33.13           110
 5 ADC  192.168.10.0/30    192.168.10.1    ether3                    0
 6 ADC  192.168.33.0/30    192.168.33.1    ether1                    0
 7 ADo  192.168.33.4/30                    192.168.33.2            110
 8 ADo  192.168.33.8/30                    192.168.33.13           110
 9 ADC  192.168.33.12/30   192.168.33.14   ether2                    0
```

Setting Resource Reservation

Next step is to set up TE resource for every interface on which we might want to run TE tunnel.

Configuration on all the routers are the same:

```
/mpls traffic-eng interface
add interface=ether1 bandwidth=10Mbps
add interface=ether2 bandwidth=10Mbps
```

Since we are not using real bandwidth limitation on the tunnels in this example, bandwidth parameter is only used for administrative purposes and can be any value (it does not represent how much bandwidth will actually flow through the interface).

TE tunnel setup

Since our primary goal is to strictly forward traffic over specific path we will use static path configuration as primary, and dynamic (CSPF) as secondary path if primary fails.

R1

```
/mpls traffic-eng tunnel-path
add name=dyn use-cspf=yes
add name=tun-first-link use-cspf=no \
   hops=192.168.33.2:strict,192.168.33.5:strict,192.168.33.6:strict

/interface traffic-eng
add bandwidth=5Mbps name=TE-to-R3 to-address=10.255.0.3 primary-path=tun-first-link \
   secondary-paths=dyn record-route=yes from-address=10.255.0.1
```

R3

```
/mpls traffic-eng tunnel-path
add name=dyn use-cspf=yes
add name=tun-second-link use-cspf=no \
   hops=192.168.33.10:strict,192.168.33.13:strict,192.168.33.14:strict

/interface traffic-eng
add bandwidth=5Mbps name=TE-to-R1 to-address=10.255.0.1 primary-path=tun-second-link \
   secondary-paths=dyn record-route=yes from-address=10.255.0.3
```

Verify that TE tunnels are working

```
[admin@R1] /interface traffic-eng> monitor 0
             tunnel-id: 14
    primary-path-state: established
          primary-path: tun-first-link
  secondary-path-state: not-necessary
           active-path: tun-first-link
          active-lspid: 1
          active-label: 39
        explicit-route: S:192.168.33.2/32,S:192.168.33.5/32,S:192.168.33.6/32
    reserved-bandwidth: 5.0Mbps
```

Notice that running router will show assigned MPLS lables, whole tunnel path and reserved bandwidth. Reserved resources also can be monitored on each router:

```
[admin@R1] /mpls traffic-eng> path-state print
Flags: L - locally-originated, E - egress, F - forwarding, P - sending-path,
R - sending-resv
 #      SRC                  DST                  BANDWIDTH OUT.. OUT-NEXT-HOP
 0 LFP  10.255.0.1:1         10.255.0.3:15          5.0Mbps eth.. 192.168.33.2
 1  E R 10.255.0.3:1         10.255.0.1:8           5.0Mbps
[admin@R1] /mpls traffic-eng> resv-state print
Flags: E - egress, A - active, N - non-output, S - shared
 #    SRC                 DST                  BANDWIDTH LABEL            INT...
 0 AS 10.255.0.1:1        10.255.0.3:15          5.0Mbps 41               ether1
[admin@R1] /mpls traffic-eng>
[admin@R1] /mpls traffic-eng> interface print
Flags: X - disabled, I - invalid
 #   INTERFACE                                  BANDWIDTH  TE-METRIC REMAINING-BW
 0   ether1                                        10Mbps          1      5.0Mbps
 1   ether2                                        10Mbps          1     10.0Mbp
```

Notice that remaining bandwidth on interface decreased. It means that if multiple tunnels are created and whole bandwidth on that particular interface is used, then tunnel will try to look for different path.

Note: TE tunnels are unidirectional, meaning that tunnel may be running in one direction but fail in another direction if whole resources are reserved

Route Traffic over TE

To route LAN traffic over TE tunnel we will assign address 10.99.99.1/30 and 10.99.99.2/30 to each tunnel end.

R1

```
/ip address add address=10.99.99.1/30 interface=TE-to-R3

/ip route add dst-address=192.168.20.0/24 gateway=10.99.99.2
```

R3

```
/ip address add address=10.99.99.2/30 interface=TE-to-R1

/ip route add dst-address=192.168.10.0/24 gateway=10.99.99.1
```

To verify if traffic is actually going over TE tunnel and is label switched we can run traceroute:

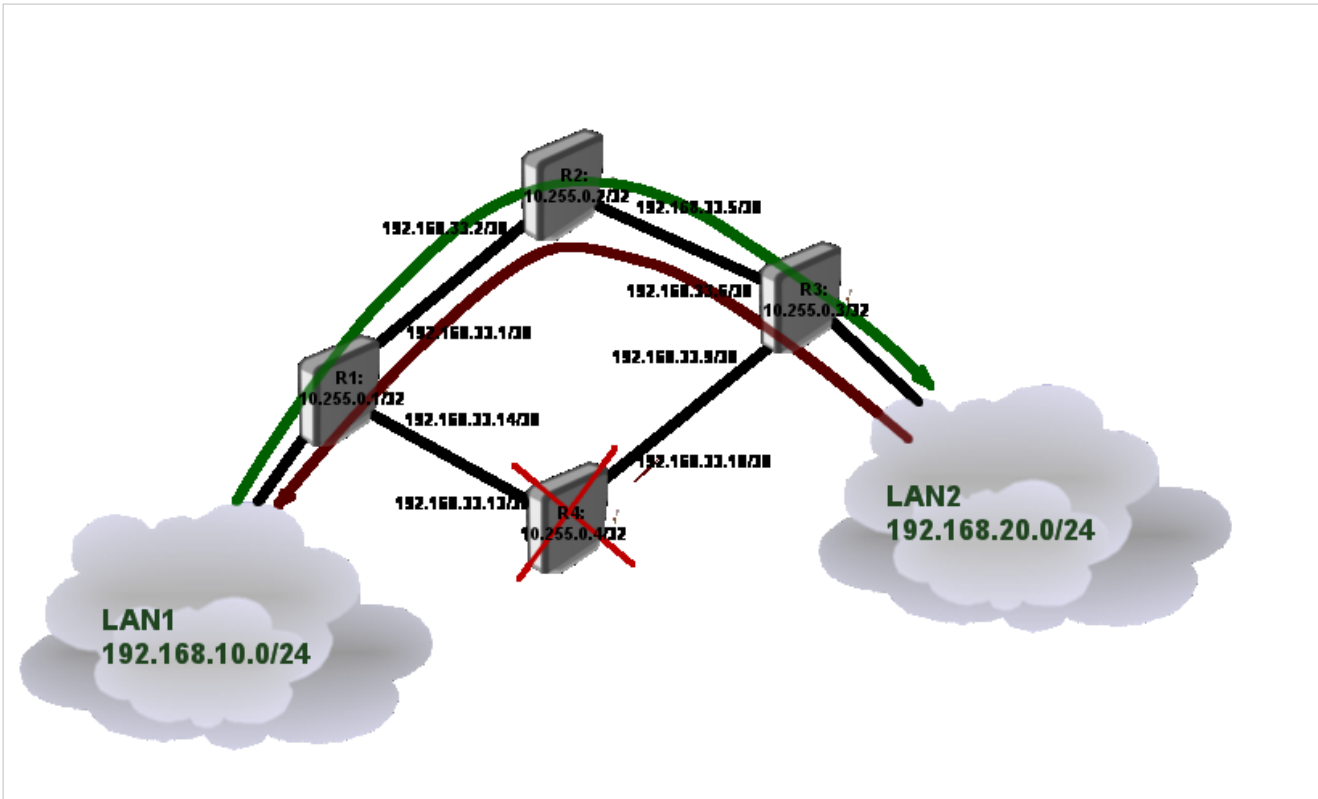
```
[admin@R1] /ip address> /tool traceroute  10.99.99.1
 # ADDRESS                                 RT1   RT2   RT3   STATUS
 1 192.168.33.2                            2ms   1ms   1ms   <MPLS:L=41,E=0>
 2 10.99.99.1                              3ms   1ms   1ms
```

As you can see traceroute recorded MPLS label in the path.

Congratulations our setup works.

Failover Testing

Lets consider that router R4 went down, and whole traffic needs to be switched over R2.



Traffic engineering does not switch paths automatically. If we use dynamic path then path relies on OSPF routing information and is recalculated whenever one of the link fails. In case of static primary paths as in our case, we need to re-optimize the tunnel. It can be done in two ways:

- manually - which is not what we need
- automatically - at specific interval

To set up path re-optimization we need to specify interval.

R1

```
/interface trafic-eng set TE-to-R3 reoptimize-interval=5s
```

R3

```
/interface trafic-eng set TE-to-R1 reoptimize-interval=5s
```

After a while tunnel will switch paths do secondary

```
[admin@R3] /interface traffic-eng> monitor 0
             tunnel-id: 10
    primary-path-state: trying-to-establish
          primary-path: tun-second-link
  secondary-path-state: established
        secondary-path: dyn
           active-path: dyn
         active-lspid: 3
         active-label: 45
        explicit-route: S:192.168.33.5/32,S:192.168.33.2/32,S:192.168.33.1/32
```

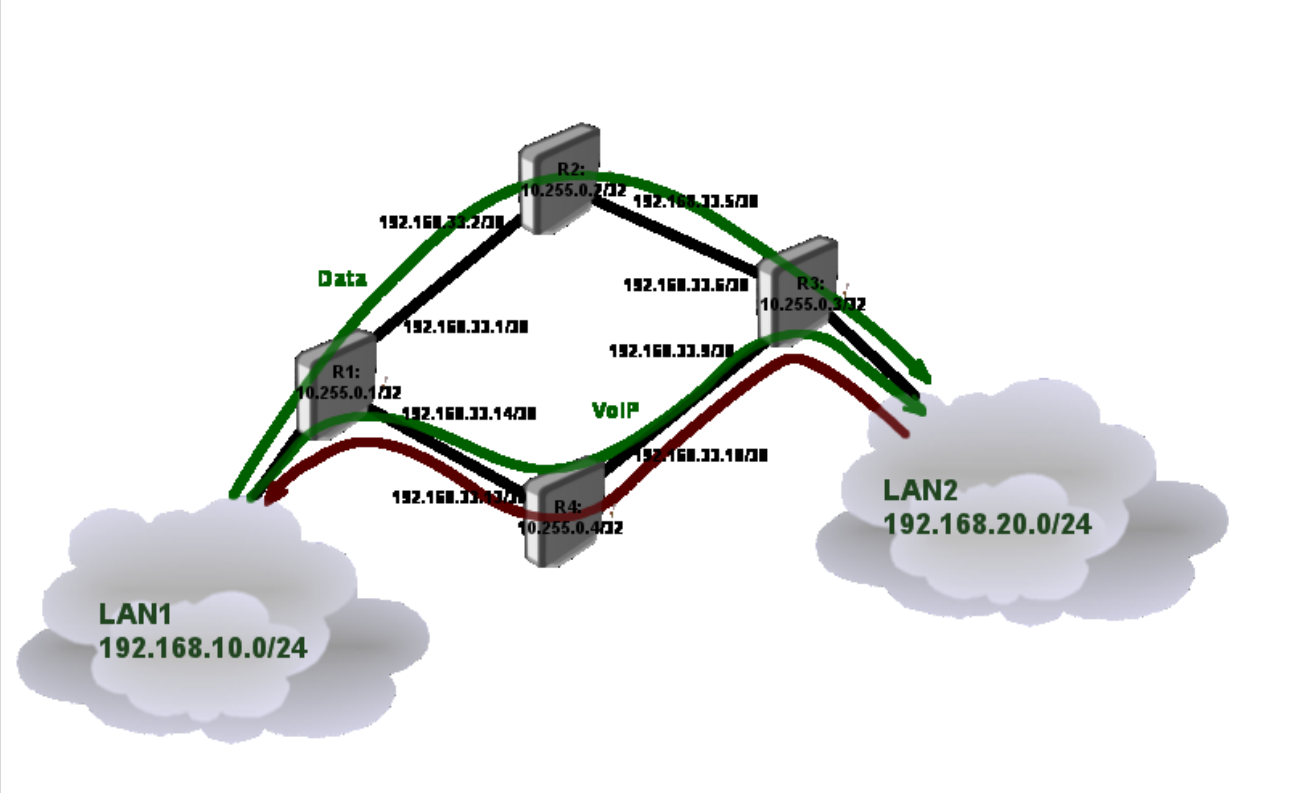
```
    reserved-bandwidth: 5.0Mbps
```

By default tunnel will try to switch back to primary path every minute. This setting can be changed with **primary-retry-interval** parameter.

Note: Switching from primary to secondary path may not be as fast as expected. It depends on OSPF dead timeouts, routing table updates and timeout settings in TE tunnel configuration.

Extended Tunnel for VoIP

Lets consider that in network that we made previously, path through R4 has lower latency which is good for VoIP traffic. Since VOIP server is on LAN2 we will create another TE tunnel from R1 to R3 explicitly for VoIP traffic.



Assuming that previous configuration is up and running, we will start with path definition for VOIP tunnel.

R1

```
/mpls traffic-eng tunnel-path
add name=tun-second-link use-cspf=no \
   hops=192.168.33.13:strict,192.168.33.10:strict,192.168.33.9:strict

/interface traffic-eng
add name=TE-to-R3-VOIP to-address=10.255.0.3 bandwidth=5Mbps record-route=yes \
   primary-path=tun-second-link secondary-paths=dyn reoptimize-interval=5s
```

Verify that tunnel is up and running

```
[admin@R1] /interface traffic-eng> monitor TE-to-R3-VOIP
              tunnel-id: 19
    primary-path-state: established
```

```
         primary-path: tun-second-link
  secondary-path-state: not-necessary
          active-path: tun-second-link
         active-lspid: 1
         active-label: 20
       explicit-route: S:192.168.33.13/32,S:192.168.33.10/32,S:192.168.33.9/32
       recorded-route: 192.168.33.10[20],192.168.33.9[0]
    reserved-bandwidth: 5.0Mbps
```

Notice that we are doing configuration only in one direction, since traffic coming back form the server will use the same tunnel as regular data.

Now it is time to set up routing.

Let's consider that VoIP server's IP address is 192.168.20.250. We will also need to set up IP addresses on tunnel ends.

R1

```
/ip address add address=10.100.100.1/30 interface=TE-to-R3-VOIP

/ip route add dst-address=192.168.20.250/32 gateway=10.100.100.2
```

R3

```
/ip address add address=10.100.100.2/30 interface=TE-to-R1
```

See More

- TE Tunnel Auto Bandwidth
- TE Tunnels

[Top | Back to Content]

Manual:API

Summary

Application Programmable Interface (API) allows users to create custom software solutions to communicate with RouterOS to gather information, adjust configuration and manage router. API closely follows syntax from command line interface (CLI). It can be used to create translated or custom configuration tools to aid ease of use running and managing routers with RouterOS.

To use API RouterOS version 3.x or newer is required.

By default API uses port #*8728* and service is disabled. More on service management see in corresponding manual section. Corresponding service name is **api**

Protocol

Communication with router is done by sending sentences to the router and receiving one or more sentences in return. Sentence is sequence of words terminated by zero length word. Word is part of sentence encoded in certain way - encoded length and data. Communication happen by sending sentences to the router and receiving replies to sent sentences. Each sentence sent to router using API should contain command as a first word followed by words in no particular order, end of sentence is marked by zero length word. When router receives full sentence (command word, no or more attribute words and zero length word) it is evaluated and executed, then reply is formed and returned.

API words

Words are part of sentence. Each word has to be encoded in certain way - length of the word followed by word content. Length of the word should be given as count of bytes that are going to be sent.

Length of the word is encoded as follows:

Value of length	# of bytes	Encoding
0 <= len <= 0x7F	1	len, lowest byte
0x80 <= len <= 0x3FFF	2	len | 0x8000, two lower bytes
0x4000 <= len <= 0x1FFFFF	3	len | 0xC00000, three lower bytes
0x200000 <= len <= 0xFFFFFFF	4	len | 0xE0000000
len >= 0x10000000	5	0xF0 and len as four bytes

- Each word is encoded as length, followed by that many bytes of content;
- Words are grouped into sentences. End of sentence is terminated by zero length word;
- Scheme allows encoding of length up to **0x7FFFFFFFFF**, only four byte length is supported;
- Bytes of **len** are sent most significant first (network order);
- If first byte of word is **>= 0xF8**, then it is a reserved control byte. After receiving unknown control byte API client cannot proceed, because it cannot know how to interpret following bytes;
- Currently control bytes are not used;

In general *words* can be described like this <<*encoded word length*><*word content*>>. *Word content* can be separated in 5 parts: *command word*, *attribute word*, *API attribute word*. *query word* and *reply word*

Command word

First word in sentence has to be command followed by attribute words and zero length word or terminating word. Name of command word should begin with '/'. Names of commands closely follow CLI, with spaces replaced with '/'. There are commands that are specific to API;

Command word structure in strict order:

- encoded length
- content prefix /
- CLI converted command

API specific commands:

```
getall
login
cancel
```

Command word concent examples:

```
/login
```

```
/ip/address/getall
```

```
/user/active/listen
```

```
/interface/vlan/remove
```

```
/system/reboot
```

Attribute word

Each *command word*have its own list of *attribute words* depending on content.

Atribute word structure consists of 5 parts in this order:

- encoded length
- content prefix equals sigh - =
- attribute name
- separating equals sign - =
- value of attribute if there is one. It is possible that attribute does not have a value

Note: Value can hold multiple *equal* signs in the value of *attribute word* since the way word is encoded

Note: Value can be empty

Examples without encoded length prefix:

```
=address=10.0.0.1
```

```
=name=iu=c3Eeg
```

```
=disable-running-check=yes
```

Warning: Order of attribute words and API parameters is not important and should not be relied on

API attribute word

API attribute word structure is in strict order:

- encoded length
- content prefix with dot .
- attribute name
- name postfixed with equals =sign
- attribute value

Currently the only such API attribute is *tag*.

Note: If sentence contain *API attribute word* tag then each returned sentence in reply from router to that tagged sentence will be tagged with same tag.

Query word

Senteces can have additional query paramteres that restrict their scope. They are explained in detail in separate section.

Example of sentence using query word attributes:

```
/interface/print
?type=ether
?type=vlan
?#|!
```

- Query words begin with '?'.
- Currently only *print* command handles query words.

Warning: Order of query words is significant

Reply word

It is sent only by the router. It is only sent in response to full sentence send by the client.

- First word of reply begins with '!';
- Each sentence sent generates at least one reply (if connection does not get terminated);
- Last reply for every sentence is reply that has first word `!done`;
- Errors and exceptional conditions begin with `!trap`;
- Data replies begin with `!re`
- If API connection is closed, RouterOS sends `!fatal` with reason as reply and then closes the connection;

API sentences

API sentence is main object of communication using API.

- Empty sentences are ignored.
- Sentence is processed after receiving zero length word.
- There is a limit on number and size of sentences client can send before it has logged in.
- Order of attribute words should not be relied on. As order and count is changeable by *.proplist* attribute.
- Sentence structure is as follows:
  - First word should contain *command word*;
  - Should contain *zero length word* to terminate the sentence;
  - Can contain none or several *attribute words*. There is no particular order at what attribute words has to be sent in the sentence, order is not important for *attribute words*;
  - Can contain none or several *query words*. Order of *query words* in the sentence is important.

Note: *Zero length word* terminates the sentence. If it is not provided router will not start to evaluate sent words and will consider all the input as part of the same sentence.

Initial login

```
/login
```

```
!done
=ret=ebddd18303a54111e2dea05a92ab46b4
```

```
/login
=name=admin
=response=001ea726ed53ae38520c8334f82d44c9f2
```

```
!done
```

Note: that each command and response ends with an empty word.

- First, clients sends `/login` command.
- Reply contains `=ret=`*`challenge`* argument.
- Client sends second `/login` command, with `=name=`*`username`* and `=response=`*`response`*.

- In case of error, reply contains `=ret=`*`error message`*.
- In case of successful login client can start to issue commands.

Tags

- It is possible to run several commands simultaneously, without waiting for previous one to complete. If API client is doing this and needs to differentiate command responses, it can use 'tag' API parameter in command sentences.
- If you include 'tag' parameter with non-empty value in command sentence, then 'tag' parameter with exactly the same value will be included in all responses generated by this command.
- If you do not include 'tag' parameter or it's value is empty, then all responses for this command will not have 'tag' parameter.

Command description

- `/cancel`
  - optional argument: `=tag=`*`tag of command to cancel`*, without it cancels all running commands
  - does not cancel itself
  - all canceled commands are interruped and in the usual case generate '!trap' and '!done' responses
  - please note that `/cancel` is separate command and can have it's own unique '.tag' parameter, that is not related to '=tag' argument of this command
- `listen`
  - `listen` command is avaliable where console print command is available, but it does not have expected effect everywhere (i.e. may not work)
  - `!re` sentences are generated as something changes in particular item list
  - when item is deleted or dissapears in any other way, the '!re' sentence includes value '=.dead=yes'
  - This command does not terminate. To terminate it use `/cancel` command.
- `getall`
  - `getall` command is available where console print command is available. Since version 3.21 `getall` is an alias for `print`.
  - replies contain `=.id=`*`Item internal number`* property.
- `print`
  - API `print` command differs from the console counterpart in the following ways:
    - `where` argument is not supported. Items can be filtered using query words (see below).
    - `.proplist` argument is a comma separated list of property names that should be included for the returned items.
      - returned items may have additional properties.
      - order of returned properties is not defined.
      - if list contains duplicate entries, handling of such entries is not defined.
      - if propery is present in `.proplist`, but absent from the item, then that item does not have this property value (?name will evaluate to false for that item).
      - if `.proplist` is absent, all properties are included as requested by print command, even those that have slow access time (such as file contents and perfomance counters). Thus use of `.proplist` is encouraged. Omission of `.proplist` may have high perfomance penalty if =detail= argument is set.

Queries

`print` command accepts query words that limit set of returned sentences. This feature is available since RouterOS 3.21.

- Query words begin with '?'.
- Order of query words is significant. Query is evaluated starting from the first word.
- Query is evaluated for each item in the list. If query succeeds, item is processed, if query fails, item is ignored.
- Query is evaluated using a stack of boolean values. Initially stack contains infinite amount of 'true' values. At the end of evaluation, if stack contains at least one 'false' value, query fails.
- Query words operate according to the following rules:

Query	Desciption
?name	pushes 'true' if item has value of property *name*, 'false' if it does not.
?-name	pushes 'true' if item does not have value of property *name*, 'false' otherwise.
?*name*=*x* **?=*name*=*x***	pushes 'true' if property *name* has value equal to *x*, 'false' otherwise.
?<name=''x''''' |style=''border-bottom:1px solid gray;'' valign=''top''| pushes 'true' if property ''name'' has value less than ''x'', 'false' otherwise. |- |style=''border-bottom:1px solid gray;'' valign=''top''|'''?>name=*x*	pushes 'true' if property *name* has value greater than *x*, 'false' otherwise.
?#*operations*	applies operations to the values in the stack. • operation string is evaluated left to right. • sequence of decimal digits followed by any other character or end of word is interpreted as a stack index. top value has index 0. • index that is followed by a character pushes copy of value at that index. • index that is followed by the end of word replaces all values with the value at that index. • **!** character replaces top value with the opposite. • **&** pops two values and pushes result of logical 'and' operation. • **|** pops two values and pushes result of logical 'or' operation. • **.** after an index does nothing. • **.** after another character pushes copy of top value.

Examples:

- Get all ethernet and VLAN interfaces:

```
/interface/print
?type=ether
?type=vlan
?#|
```

- Get all routes that have non-empty comment:

```
/ip/route/print
?>comment=
```

OID

`print` command can return OID values for properties that are available in SNMP. This feature appeared in 3.23 version.

In console, OID values can be seen by running 'print oid' command. In API, these properties have name that ends with ".oid", and can be retrieved by adding their name to the value of '.proplist'. An example:

```
/system/resource/print
=.proplist=uptime,cpu-load,uptime.oid,cpu-load.oid
```

```
!re
=uptime=01:22:53
=cpu-load=0
=uptime.oid=.1.3.6.1.2.1.1.3.0
=cpu-load.oid=.1.3.6.1.2.1.25.3.3.1.2.1

!done
```

Command examples

`/system/package/getall`

```
/system/package/getall
```

```
!re
=.id=*5802
=disabled=no
=name=routeros-x86
=version=3.0beta2
=build-time=oct/18/2006 16:24:41
=scheduled=

!re
=.id=*5805
=disabled=no
=name=system
=version=3.0beta2
=build-time=oct/18/2006 17:20:46
=scheduled=
```

... more !re sentences ...

```
!re
=.id=*5902
=disabled=no
=name=advanced-tools
```

```
=version=3.0beta2
=build-time=oct/18/2006 17:20:49
=scheduled=

!done
```

/user/active/listen

```
/user/active/listen
```

```
!re
=.id=*68
=radius=no
=when=oct/24/2006 08:40:42
=name=admin
=address=0.0.0.0
=via=console

!re
=.id=*68
=.dead=yes
```

... more !re sentences ...

/cancel, simultaneous commands

```
/login
```

```
!done
=ret=856780b7411eefd3abadee2058c149a3
```

```
/login
=name=admin
=response=005062f7a5ef124d34675bf3e81f56c556
```

```
!done
```

```
-- first start listening for interface changes (tag is 2)
```

```
/interface/listen
.tag=2
```

```
-- disable interface (tag is 3)
```

```
/interface/set
=disabled=yes
=.id=ether1
.tag=3
```

```
-- this is done for disable command (tag 3)
!done
.tag=3
-- enable interface (tag is 4)
/interface/set
=disabled=no
=.id=ether1
.tag=4
-- this update is generated by change made by first set command (tag 3)
!re
=.id=*1
=disabled=yes
=dynamic=no
=running=no
=name=ether1
=mtu=1500
=type=ether
.tag=2
-- this is done for enable command (tag 4)
!done
.tag=4
-- get interface list (tag is 5)
/interface/getall
.tag=5
-- this update is generated by change made by second set command (tag 4)
!re
=.id=*1
=disabled=no
=dynamic=no
=running=yes
=name=ether1
=mtu=1500
=type=ether
.tag=2
-- these are replies to getall command (tag 5)
!re
=.id=*1
```

```
=disabled=no
=dynamic=no
=running=yes
=name=ether1
=mtu=1500
=type=ether
.tag=5

!re
=.id=*2
=disabled=no
=dynamic=no
=running=yes
=name=ether2
=mtu=1500
=type=ether
.tag=5

-- here interface getall ends (tag 5)
!done
.tag=5

-- stop listening - request to cancel command with tag 2, cancel itself uses tag 7
/cancel
=tag=2
.tag=7

-- listen command is interrupted (tag 2)
!trap
=category=2
=message=interrupted
.tag=2

-- cancel command is finished (tag 7)
!done
.tag=7

-- listen command is finished (tag 2)
!done
.tag=2
```

Example client

- this is simple API client in Python2
- example for Python3
- usage: api.py *ip-address username password*
- after that type words from keyboard, terminating them with newline
- Since empty word terminates sentence, you should press enter **twice** after last word before sentence will be sent to router.

```
#!/usr/bin/python

import sys, posix, time, md5, binascii, socket, select

class ApiRos:
    "Routeros api"
    def __init__(self, sk):
        self.sk = sk
        self.currenttag = 0

    def login(self, username, pwd):
        for repl, attrs in self.talk(["/login"]):
            chal = binascii.unhexlify(attrs['=ret'])
        md = md5.new()
        md.update('\x00')
        md.update(pwd)
        md.update(chal)
        self.talk(["/login", "=name=" + username,
                   "=response=00" + binascii.hexlify(md.digest())])

    def talk(self, words):
        if self.writeSentence(words) == 0: return
        r = []
        while 1:
            i = self.readSentence();
            if len(i) == 0: continue
            reply = i[0]
            attrs = {}
            for w in i[1:]:
                j = w.find('=', 1)
                if (j == -1):
                    attrs[w] = ''
                else:
                    attrs[w[:j]] = w[j+1:]
            r.append((reply, attrs))
            if reply == '!done': return r

    def writeSentence(self, words):
        ret = 0
```

```
        for w in words:
            self.writeWord(w)
            ret += 1
        self.writeWord('')
        return ret

    def readSentence(self):
        r = []
        while 1:
            w = self.readWord()
            if w == '': return r
            r.append(w)

    def writeWord(self, w):
        print "<<< " + w
        self.writeLen(len(w))
        self.writeStr(w)

    def readWord(self):
        ret = self.readStr(self.readLen())
        print ">>> " + ret
        return ret

    def writeLen(self, l):
        if l < 0x80:
            self.writeStr(chr(l))
        elif l < 0x4000:
            l |= 0x8000
            self.writeStr(chr((l >> 8) & 0xFF))
            self.writeStr(chr(l & 0xFF))
        elif l < 0x200000:
            l |= 0xC00000
            self.writeStr(chr((l >> 16) & 0xFF))
            self.writeStr(chr((l >> 8) & 0xFF))
            self.writeStr(chr(l & 0xFF))
        elif l < 0x10000000:
            l |= 0xE0000000
            self.writeStr(chr((l >> 24) & 0xFF))
            self.writeStr(chr((l >> 16) & 0xFF))
            self.writeStr(chr((l >> 8) & 0xFF))
            self.writeStr(chr(l & 0xFF))
        else:
            self.writeStr(chr(0xF0))
            self.writeStr(chr((l >> 24) & 0xFF))
            self.writeStr(chr((l >> 16) & 0xFF))
            self.writeStr(chr((l >> 8) & 0xFF))
            self.writeStr(chr(l & 0xFF))
```

```
def readLen(self):
    c = ord(self.readStr(1))
    if (c & 0x80) == 0x00:
        pass
    elif (c & 0xC0) == 0x80:
        c &= ~0xC0
        c <<= 8
        c += ord(self.readStr(1))
    elif (c & 0xE0) == 0xC0:
        c &= ~0xE0
        c <<= 8
        c += ord(self.readStr(1))
        c <<= 8
        c += ord(self.readStr(1))
    elif (c & 0xF0) == 0xE0:
        c &= ~0xF0
        c <<= 8
        c += ord(self.readStr(1))
        c <<= 8
        c += ord(self.readStr(1))
        c <<= 8
        c += ord(self.readStr(1))
    elif (c & 0xF8) == 0xF0:
        c = ord(self.readStr(1))
        c <<= 8
        c += ord(self.readStr(1))
        c <<= 8
        c += ord(self.readStr(1))
        c <<= 8
        c += ord(self.readStr(1))
    return c

def writeStr(self, str):
    n = 0;
    while n < len(str):
        r = self.sk.send(str[n:])
        if r == 0: raise RuntimeError, "connection closed by remote end"
        n += r

def readStr(self, length):
    ret = ''
    while len(ret) < length:
        s = self.sk.recv(length - len(ret))
        if s == '': raise RuntimeError, "connection closed by remote end"
        ret += s
    return ret
```

```
def main():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((sys.argv[1], 8728))
    apiros = ApiRos(s);
    apiros.login(sys.argv[2], sys.argv[3]);

    inputsentence = []

    while 1:
        r = select.select([s, sys.stdin], [], [], None)
        if s in r[0]:
            # something to read in socket, read sentence
            x = apiros.readSentence()

        if sys.stdin in r[0]:
            # read line from input and strip off newline
            l = sys.stdin.readline()
            l = l[:-1]

            # if empty line, send sentence and start with new
            # otherwise append to input sentence
            if l == '':
                apiros.writeSentence(inputsentence)
                inputsentence = []
            else:
                inputsentence.append(l)

if __name__ == '__main__':
    main()
```

Example run:

```
debian@localhost:~/api-test$ ./api.py 10.0.0.1 admin ''
<<< /login
<<<
>>> !done
>>> =ret=93b438ec9b80057c06dd9fe67d56aa9a
>>>
<<< /login
<<< =name=admin
<<< =response=00e134102a9d330dd7b1849fedfea3cb57
<<<
>>> !done
>>>
/user/getall

<<< /user/getall
```

```
<<<
>>> !re
>>> =.id=*1
>>> =disabled=no
>>> =name=admin
>>> =group=full
>>> =address=0.0.0.0/0
>>> =netmask=0.0.0.0
>>>
>>> !done
>>>
```

User:Boen robot/API

Summary

This document describes the operation of MikroTik RouterOS API for RouterOS. The API (application programming interface) is a way to create your own versions of Winbox, or rather, to programmatically command RouterOS from a separate device. This guide will help you make simplified, or translated control applications for RouterOS v3 and newer.

By default, API uses TCP port 8728 and is disabled. To enable API use the following command:

```
/ip service enable api
```

If you need to use the API service with another TCP port, you can do so with the following command:

```
/ip service set api port="desired port"
```

For better security, you can limit access to the API service only to the IP addresses from which the API service will be accessed. This is done with the following command:

```
/ip service set api address="IP address(es) of known client(s)"
```

Protocol

Words

- Protocol stream (both incoming and outgoing) is formatted as a sequence of words.
- Each word is composed of an encoded length, followed by that many bytes of content.
- Length is encoded as follows:

Value of length	Number of bytes	Encoding
0 <= len <= 0x7F	1	len (lowest byte)
0x80 <= len <= 0x3FFF	2	len | 0x8000 (two lower bytes)
0x4000 <= len <= 0x1FFFFF	3	len | 0xC00000 (three lower bytes)
0x200000 <= len <= 0xFFFFFFF	4	len | 0xE0000000 (four lower bytes)
0x10000000 <= len <= 0x7FFFFFFF	5	len | 0xF000000000 (0xF0, followed by length as four bytes)

- Although this scheme allows encoding of length up to **0x7FFFFFFF**, only four byte length (**<= 0xFFFFFFF**) is currently supported by RouterOS.
- Bytes of **len** are sent most significant first (network order).
- If first byte of encoded length is **>= 0xF8**, then it is a reserved control byte. After receiving unknown control byte API client cannot proceed, because it cannot know how to interpret following bytes.
- Currently, control bytes are not used.

Sentences

- Words are grouped into sentences. End of sentence is specified with a zero length word.
- Sentences are processed after receiving a zero length word.
- Empty sentences are ignored.

Life cycle

- Once connected, the client sends one or more sentences (*Commands*) to RouterOS.
- For each command, RouterOS returns at least one sentence (*Reply*) to the client, unless the connection is terminated prematurely.
- There is a limit on number and size of sentences client can send before it has logged in (see `/login`).
- When sending multiple commands without waiting for each to finish, order of replies should not be relied on (use tags instead).

Syntax

Commands

- First word is name of command.
- Examples:

```
/login
/ip/address/getall
/user/active/listen
/interface/vlan/remove
/system/reboot
```

- Names of commands closely follow console, with spaces replaced by '/'. There are also some API specific commands.
- Name of command should begin with '/' and include the full menu name.
- Next, zero or more command arguments, API parameters and query words can be specified.
- Order of arguments and API parameters is not important and should not be relied on.

Replies

- First word of every reply begins with '!'.
- Last reply for every command is one that has first word `!done`.
- Replies that indicate errors and exceptional conditions have first word `!trap`.
- Data replies have first word `!re`.
- If API connection is about to be closed, RouterOS sends a final reply with first word `!fatal` with reason as next word and then closes the connection.
- After first word of every reply, zero or more arguments and/or API parameters may appear.
- Order of arguments and API parameters should not be relied on.

Arguments

- Examples:

```
=.id=*10F1
=address=10.0.0.1
=name=iu=c3Eeg
=disable-running-check=yes
```

- An argument should begin with '=' followed by name of argument, followed by another '=', followed by value of argument.
- There are API specific arguments, such as `.id`. Names of API specific arguments begin with a dot.
- Argument value can be empty and can contain '='.

API parameters

- An API parameter should begin with '.' followed by name of parameter, followed by '=', followed by value of parameter.
- Currently the only such parameter is 'tag'.

Query words

- Commands can also have additional query words that restrict their scope.
- Example:

```
/interface/print
?type=ether
?type=vlan
?#|!
```

- Query words begin with '?'.
- Order of query words is significant.
- Currently only 'print' command handles query words. See its query section for details.

Semantics

Tags

- It is possible to run several commands simultaneously, without waiting for previous one to complete. If API client is doing this and needs to differentiate command responses, it can use the 'tag' API parameter in command sentences.
- If you include 'tag' parameter with non-empty value in a command sentence, then 'tag' parameter with exactly the same value will be included in all responses generated for this command.
- If you do not include 'tag' parameter or its value is empty, then all replies for this command will not have a 'tag' parameter.

API specific commands

`/login`

- Used initially to login to RouterOS.
- Example session:

```
/login

!done
=ret=ebddd18303a54111e2dea05a92ab46b4

/login
=name=admin
=response=001ea726ed53ae38520c8334f82d44c9f2

!done
```

- First, clients sends `/login` command.
- Note that each command and response ends with an empty word.
- Reply contains `=ret=`*`challenge`* argument.
- Client sends second `/login` command, with `=name=`*`username`* and `=response=`*`response`*.
- A proper *response* is formed by two zeroes, followed by an MD5 sum of a NULL byte, the password, and the *challenge*.
- In case of error, reply contains `=ret=`*`error message`*.
- After a successful login, client can start to issue other commands.

`/cancel`

- optional argument: `=tag=`*`tag of command to cancel`*, without it cancels all running commands
- does not cancel itself
- all canceled commands are interruped and in the usual case generate '!trap' and '!done' responses
- please note that `/cancel` is separate command and can have it's own unique '.tag' parameter, that is not related to '=tag' argument of this command

`listen`

- `listen` command is avaliable where console print command is available, but it does not have expected effect (i.e. may not work) everywhere.
- `!re` sentences are generated as something changes in particular item list.
- when item is deleted or dissapears in any other way, the '!re' sentence includes value '=.dead=yes'
- This command does not terminate. To terminate it use `/cancel` command.

`print` and `getall`

- Since version 3.21 `getall` is an alias for `print`. The only difference prior to this version is that `getall` replies contained `=.id=`*`Item internal number`* property, while `print` did not.
- API `print` command differs from the console counterpart in the following ways:
  - `where` argument is not supported. Items can be filtered using query words (see below).
  - `.proplist` argument is a comma separated list of property names that should be included for the returned items.
    - returned items may have additional properties.
    - order of returned properties is not defined.
    - if list contains duplicate entries, handling of such entries is not defined.
    - if propery is present in `.proplist`, but absent from the item, then that item does not have this property value (?name will evaluate to false for that item).
    - if `.proplist` is absent, all properties are included as requested by print command, even those that have slow access time (such as file contents and perfomance counters). Thus use of `.proplist` is encouraged. Omission of `.proplist` may have high perfomance penalty if =detail= argument is set.

Queries

`print` command accepts query words that limit set of returned items. This feature appeared in the 3.21 version.

- Query is evaluated for each item in the list. If query succeeds, item is processed, if query fails, item is ignored.
- Query is evaluated using a stack of boolean values. Initially stack contains infinite amount of 'true' values. At the end of evaluation, if stack contains at least one 'false' value, query fails.
- Query words operate according to the following rules:

`?`*`name`*	pushes 'true' if item has value of property *name*, 'false' if it does not.
`?-`*`name`*	pushes 'true' if item does not have value of property *name*, 'false' otherwise.
`?`*`name`*`=`*`x`* `?=`*`name`*`=`*`x`*	pushes 'true' if property *name* has value equal to *x*, 'false' otherwise.
`?<name=`*`x`*	pushes 'true' if property *name* has value less than *x*, 'false' otherwise.
`?>name=`*`x`*	pushes 'true' if property *name* has value greater than *x*, 'false' otherwise.
`?#`*`operations`*	pops two values and pushes result of logical 'or' operation. • `.` after an index does nothing. • `.` after another character pushes copy of top value.

Examples:

- Get all ethernet and VLAN interfaces:

```
/interface/print
?type=ether
?type=vlan
?#|
```

- Get all routes that have non-empty comment:

```
/ip/route/print
?>comment=
```

OID

`print` command can return OID values for properties that are available in SNMP. This feature appeared in 3.23 version.

In console, OID values can be seen by running 'print oid' command. In API, these properties have name that ends with ".oid", and can be retrieved by adding their name to the value of '.proplist'. An example:

```
/system/resource/print
=.proplist=uptime,cpu-load,uptime.oid,cpu-load.oid

!re
=uptime=01:22:53
=cpu-load=0
=uptime.oid=.1.3.6.1.2.1.1.3.0
=cpu-load.oid=.1.3.6.1.2.1.25.3.3.1.2.1

!done
```

Command examples

`/system/package/getall`

```
/system/package/getall

!re
=.id=*5802
=disabled=no
=name=routeros-x86
=version=3.0beta2
=build-time=oct/18/2006 16:24:41
=scheduled=

!re
=.id=*5805
=disabled=no
=name=system
=version=3.0beta2
=build-time=oct/18/2006 17:20:46
=scheduled=

... more !re sentences ...

!re
```

```
=.id=*5902
=disabled=no
=name=advanced-tools
=version=3.0beta2
=build-time=oct/18/2006 17:20:49
=scheduled=

!done
```

/user/active/listen

```
/user/active/listen
```

```
!re
=.id=*68
=radius=no
=when=oct/24/2006 08:40:42
=name=admin
=address=0.0.0.0
=via=console

!re
=.id=*68
=.dead=yes
```

... more !re sentences ...

/cancel, simultaneous commands

```
/login
```

```
!done
=ret=856780b7411eefd3abadee2058c149a3
```

```
/login
=name=admin
=response=005062f7a5ef124d34675bf3e81f56c556
```

```
!done
```

```
-- first start listening for interface changes (tag is 2)
```

```
/interface/listen
.tag=2
```

```
-- disable interface (tag is 3)
```

```
/interface/set
```

```
=disabled=yes
=.id=ether1
.tag=3
-- this is done for disable command (tag 3)
!done
.tag=3
-- enable interface (tag is 4)
/interface/set
=disabled=no
=.id=ether1
.tag=4
-- this update is generated by change made by first set command (tag 3)
!re
=.id=*1
=disabled=yes
=dynamic=no
=running=no
=name=ether1
=mtu=1500
=type=ether
.tag=2
-- this is done for enable command (tag 4)
!done
.tag=4
-- get interface list (tag is 5)
/interface/getall
.tag=5
-- this update is generated by change made by second set command (tag 4)
!re
=.id=*1
=disabled=no
=dynamic=no
=running=yes
=name=ether1
=mtu=1500
=type=ether
.tag=2
```

```
-- these are replies to getall command (tag 5)
!re
=.id=*1
=disabled=no
=dynamic=no
=running=yes
=name=ether1
=mtu=1500
=type=ether
.tag=5

!re
=.id=*2
=disabled=no
=dynamic=no
=running=yes
=name=ether2
=mtu=1500
=type=ether
.tag=5
-- here interface getall ends (tag 5)
!done
.tag=5
-- stop listening - request to cancel command with tag 2, cancel itself uses tag 7
/cancel
=tag=2
.tag=7
-- listen command is interrupted (tag 2)
!trap
=category=2
=message=interrupted
.tag=2
-- cancel command is finished (tag 7)
!done
.tag=7
-- listen command is finished (tag 2)
!done
.tag=2
```

Example client

- this is simple API client in Python2
- example for Python3
- usage: api.py *ip-address username password*
- after that type words from keyboard, terminating them with newline
- Since empty word terminates sentence, you should press enter **twice** after last word before sentence will be sent to router.

```
#!/usr/bin/python

import sys, posix, time, md5, binascii, socket, select

class ApiRos:
    "Routeros api"
    def __init__(self, sk):
        self.sk = sk
        self.currenttag = 0

    def login(self, username, pwd):
        for repl, attrs in self.talk(["/login"]):
            chal = binascii.unhexlify(attrs['=ret'])
        md = md5.new()
        md.update('\x00')
        md.update(pwd)
        md.update(chal)
        self.talk(["/login", "=name=" + username,
                   "=response=00" + binascii.hexlify(md.digest())])

    def talk(self, words):
        if self.writeSentence(words) == 0: return
        r = []
        while 1:
            i = self.readSentence();
            if len(i) == 0: continue
            reply = i[0]
            attrs = {}
            for w in i[1:]:
                j = w.find('=', 1)
                if (j == -1):
                    attrs[w] = ''
                else:
                    attrs[w[:j]] = w[j+1:]
            r.append((reply, attrs))
            if reply == '!done': return r

    def writeSentence(self, words):
        ret = 0
```

```
        for w in words:
            self.writeWord(w)
            ret += 1
        self.writeWord('')
        return ret

    def readSentence(self):
        r = []
        while 1:
            w = self.readWord()
            if w == '': return r
            r.append(w)

    def writeWord(self, w):
        print "<<< " + w
        self.writeLen(len(w))
        self.writeStr(w)

    def readWord(self):
        ret = self.readStr(self.readLen())
        print ">>> " + ret
        return ret

    def writeLen(self, l):
        if l < 0x80:
            self.writeStr(chr(l))
        elif l < 0x4000:
            l |= 0x8000
            self.writeStr(chr((l >> 8) & 0xFF))
            self.writeStr(chr(l & 0xFF))
        elif l < 0x200000:
            l |= 0xC00000
            self.writeStr(chr((l >> 16) & 0xFF))
            self.writeStr(chr((l >> 8) & 0xFF))
            self.writeStr(chr(l & 0xFF))
        elif l < 0x10000000:
            l |= 0xE0000000
            self.writeStr(chr((l >> 24) & 0xFF))
            self.writeStr(chr((l >> 16) & 0xFF))
            self.writeStr(chr((l >> 8) & 0xFF))
            self.writeStr(chr(l & 0xFF))
        else:
            self.writeStr(chr(0xF0))
            self.writeStr(chr((l >> 24) & 0xFF))
            self.writeStr(chr((l >> 16) & 0xFF))
            self.writeStr(chr((l >> 8) & 0xFF))
            self.writeStr(chr(l & 0xFF))
```

```
    def readLen(self):
        c = ord(self.readStr(1))
        if (c & 0x80) == 0x00:
            pass
        elif (c & 0xC0) == 0x80:
            c &= ~0xC0
            c <<= 8
            c += ord(self.readStr(1))
        elif (c & 0xE0) == 0xC0:
            c &= ~0xE0
            c <<= 8
            c += ord(self.readStr(1))
            c <<= 8
            c += ord(self.readStr(1))
        elif (c & 0xF0) == 0xE0:
            c &= ~0xF0
            c <<= 8
            c += ord(self.readStr(1))
            c <<= 8
            c += ord(self.readStr(1))
            c <<= 8
            c += ord(self.readStr(1))
        elif (c & 0xF8) == 0xF0:
            c = ord(self.readStr(1))
            c <<= 8
            c += ord(self.readStr(1))
            c <<= 8
            c += ord(self.readStr(1))
            c <<= 8
            c += ord(self.readStr(1))
        return c

    def writeStr(self, str):
        n = 0;
        while n < len(str):
            r = self.sk.send(str[n:])
            if r == 0: raise RuntimeError, "connection closed by remote end"
            n += r

    def readStr(self, length):
        ret = ''
        while len(ret) < length:
            s = self.sk.recv(length - len(ret))
            if s == '': raise RuntimeError, "connection closed by remote end"
            ret += s
        return ret
```

```
def main():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((sys.argv[1], 8728))
    apiros = ApiRos(s);
    apiros.login(sys.argv[2], sys.argv[3]);

    inputsentence = []

    while 1:
        r = select.select([s, sys.stdin], [], [], None)
        if s in r[0]:
            # something to read in socket, read sentence
            x = apiros.readSentence()

        if sys.stdin in r[0]:
            # read line from input and strip off newline
            l = sys.stdin.readline()
            l = l[:-1]

            # if empty line, send sentence and start with new
            # otherwise append to input sentence
            if l == '':
                apiros.writeSentence(inputsentence)
                inputsentence = []
            else:
                inputsentence.append(l)

if __name__ == '__main__':
    main()
```

Example run:

```
debian@localhost:~/api-test$ ./api.py 10.0.0.1 admin ''
<<< /login
<<<
>>> !done
>>> =ret=93b438ec9b80057c06dd9fe67d56aa9a
>>>
<<< /login
<<< =name=admin
<<< =response=00e134102a9d330dd7b1849fedfea3cb57
<<<
>>> !done
>>>
/user/getall

<<< /user/getall
```

```
<<<
>>> !re
>>> =.id=*1
>>> =disabled=no
>>> =name=admin
>>> =group=full
>>> =address=0.0.0.0/0
>>> =netmask=0.0.0.0
>>>
>>> !done
>>>
```

Manual:IP/Address

Applies to RouterOS: 2.9, v3, v4 +

Summary

Sub-menu: `/ip address`
Standards: `IPv4 RFC 791`

IP addresses serve for a general host identification purposes in IP networks. Typical (IPv4) address consists of four octets. For proper addressing the router also needs the network mask value, id est which bits of the complete IP address refer to the address of the host, and which - to the address of the network. The network address value is calculated by binary AND operation from network mask and IP address values. It's also possible to specify IP address followed by slash "/" and the amount of bits that form the network address.
In most cases, it is enough to specify the address, the netmask, and the interface arguments. The network prefix and the broadcast address are calculated automatically.
It is possible to add multiple IP addresses to an interface or to leave the interface without any addresses assigned to it. In case of bridging or PPPoE connection, the physical interface may bot have any address assigned, yet be perfectly usable. Putting an IP address to a physical interface included in a bridge would mean actually putting it on the bridge interface itself. You can use /ip address print detail to see to which interface the address belongs to.
MikroTik RouterOS has following types of addresses:

- Static - manually assigned to the interface by a user
- Dynamic - automatically assigned to the interface by DHCP or an estabilished PPP connections

Properties

Property	Description
address (*IP/Mask*; Default:)	IP address
broadcast (*IP*; Default: **255.255.255.255**)	roadcasting IP address, calculated by default from an IP address and a network mask. Starting from v5RC6 this parameter is removed
interface (*name*; Default:)	Interface name the IP address is assigned to
netmask (*IP*; Default: **0.0.0.0**)	Delimits network address part of the IP address from the host part
network (*IP*; Default: **0.0.0.0**)	IP address for the network. For point-to-point links it should be the address of the remote end. Starting from v5RC6 this parameter is configurable only for addresses with /32 netmask (point to point links)

Read only properties

Property	Description
actual-interface (*name*)	Name of the actual interface the logical one is bound to. For example, if the physical interface you assigned the address to, is included in a bridge, the actual interface will show that bridge

Two IP addresses from the same network assigned to routers different interfaces are not valid unless VRF is used. For example, the combination of IP address 10.0.0.1/24 on the **ether1** interface and IP address 10.0.0.132/24 on the **ether2** interface is invalid, because both addresses belong to the same network 10.0.0.0/24. Use addresses from different networks on different interfaces, or enable proxy-arp on ether1 or ether2.

Example

```
[admin@MikroTik] ip address> add address=10.10.10.1/24 interface=ether2
[admin@MikroTik] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #   ADDRESS            NETWORK         BROADCAST       INTERFACE
  0   2.2.2.1/24         2.2.2.0         2.2.2.255       ether2
  1   10.5.7.244/24      10.5.7.0        10.5.7.255      ether1
  2   10.10.10.1/24      10.10.10.0      10.10.10.255    ether2

[admin@MikroTik] ip address>
```

[Top | Back to Content **]**

Manual:IP/ARP

Applies to RouterOS: 2.9, v3, v4 +

Summary

Sub-menu: `/ip arp`
Standards: `ARP RFC 826`

Even though IP packets are addressed using IP addresses, hardware addresses must be used to actually transport data from one host to another. Address Resolution Protocol is used to map OSI level 3 IP addresses to OSI level 2 MAC addreses. Router has a table of currently used ARP entries. Normally the table is built dynamically, but to increase network security, it can be partialy or completely built statically by means of adding static entries.

Properties

Property	Description
address (*IP*; Default:)	IP address to be mapped
interface (*string*; Default:)	Interface name the IP address is assigned to
mac-address (*MAC*; Default: **00:00:00:00:00:00**)	MAC address to be mapped to

Read only properties:

Property	Description
dhcp (*yes* | *no*)	Whether ARP entry is added by DHCP server
dynamic (*yes* | *no*)	Whether entry is dynamically created
invalid (*yes* | *no*)	Whether entry is not valid

Note: Maximal number of ARP entries is 8192.

ARP Modes

It is possible to set several ARP modes in interface configuration .....

Disabled

If ARP feature is turned off on the interface, i.e., arp=disabled is used, ARP requests from clients are not answered by the router. Therefore, static arp entry should be added to the clients as well. For example, the router's IP and MAC addresses should be added to the Windows workstations using the arp command:

```
C:\> arp -s 10.5.8.254  00-aa-00-62-c6-09
```

Enabled

This mode is enabled by default on all interfaces. ARPs will be discovered automatically and new dynamic entries will be added to ARP table.

Proxy ARP

A router with properly configured proxy ARP feature acts like a transparent ARP proxy between directly connected networks.

This behaviour can be usefull, for example, if you want to assign dial-in (ppp, pppoe, pptp) clients IP addresses from the same address space as used on the connected LAN.

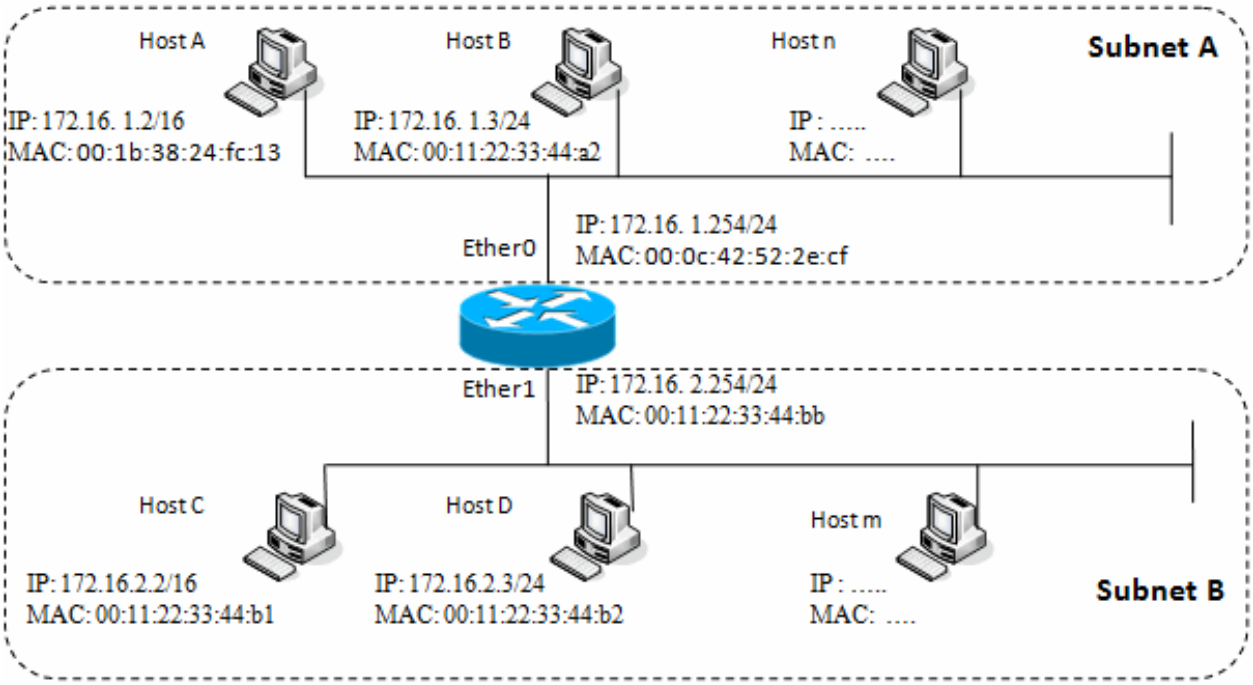


Figure 10.2. Network example for ARP proxy

Lets look at example setup from image above. Host A (172.16.1.2) on Subnet A wants to send packets to Host D (172.16.2.3) on Subnet B. Host A has a /16 subnet mask *which means that Host A believes that it is directly connected to all 172.16.0.0/16 network (the same LAN).* Since the Host A believes that is directly connected it sends an ARP request to the destination to clarify MAC address of Host D. (in case when Host A finds that destination IP address is not from the same subnet it send packet to default gateway.)

Host A broadcasts an ARP request on Subnet A:

Info from packet analyzer software:

```
No.     Time    Source            Destination       Protocol  Info

12    5.133205  00:1b:38:24:fc:13  ff:ff:ff:ff:ff:ff  ARP      Who has 173.16.2.3?  Tell 173.16.1.2
```

```
Packet details:

Ethernet II, Src: (00:1b:38:24:fc:13), Dst: (ff:ff:ff:ff:ff:ff)
    Destination: Broadcast (ff:ff:ff:ff:ff:ff)
    Source: (00:1b:38:24:fc:13)
    Type: ARP (0x0806)
Address Resolution Protocol (request)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (0x0001)
    [Is gratuitous: False]
    Sender MAC address: 00:1b:38:24:fc:13
    Sender IP address: 173.16.1.2
    Target MAC address: 00:00:00:00:00:00
    Target IP address: 173.16.2.3
```

With this ARP request, Host A (172.16.1.2) isasking Host D (172.16.2.3) to send its MAC address. The ARP request packet is then encapsulated in an Ethernet frame with the MAC address of Host A as the source address and a broadcast (FF:FF:FF:FF:FF:FF) as the destination address. Layer 2 broadcast means that frame will be sent to all hosts in the same layer 2 broadcast domain which includes the ether0 interface of the router, but does not reach Host D, because router by default does not forward layer 2 broadcast.

Since the router knows that the target address (172.16.2.3) is on another subnet but it can reach Host D, it replies with its own MAC address to Host A.

```
No.     Time   Source            Destination         Protocol   Info

13   5.133378  00:0c:42:52:2e:cf  00:1b:38:24:fc:13   ARP        172.16.2.3 is at 00:0c:42:52:2e:cf

Packet details:

Ethernet II, Src: 00:0c:42:52:2e:cf, Dst: 00:1b:38:24:fc:13
   Destination: 00:1b:38:24:fc:13
   Source: 00:0c:42:52:2e:cf
   Type: ARP (0x0806)
Address Resolution Protocol (reply)
   Hardware type: Ethernet (0x0001)
   Protocol type: IP (0x0800)
   Hardware size: 6
   Protocol size: 4
   Opcode: reply (0x0002)
   [Is gratuitous: False]
   Sender MAC address: 00:0c:42:52:2e:cf
   Sender IP address: 172.16.1.254
   Target MAC address: 00:1b:38:24:fc:13
   Target IP address: 172.16.1.2
```

This is the *Proxy ARP reply* that the router sends to Host A. Router sends back unicast proxy ARP reply with its own MAC address as the source address and the MAC address of Host A as the destination address, by saying *"send these packets to me, and I'll get it to where it needs to go."*

When Host A receives ARP response it updates its ARP table, as shown:

```
C:\Users\And>arp -a

Interface: 173.16.2.1 --- 0x8
  Internet Address      Physical Address      Type
  173.16.1.254          00-0c-42-52-2e-cf     dynamic
  173.16.2.3            00-0c-42-52-2e-cf     dynamic
  173.16.2.2            00-0c-42-52-2e-cf     dynamic
```

After MAC table update, Host A forwards all the packets intended for Host D (172.16.2.3) directly to router interface ether0 (00:0c:42:52:2e:cf) and the router forwards packets to Host D. The ARP cache on the hosts in Subnet A is populated with the MAC address of the router for all the hosts on Subnet B. Hence, all packets destined to Subnet B are sent to the router. The router forwards those packets to the hosts in Subnet B.

Multiple IP addresses by host are mapped to a single MAC address (the MAC address of this router) when proxy ARP is used.

Proxy ARP can be enabled on each interface individually with command **arp=proxy-arp**:

Setup proxy ARP:

```
 [admin@MikroTik] /interface ethernet> set 1 arp=proxy-arp

 [admin@MikroTik] /interface ethernet> print

 Flags: X - disabled, R - running
   #    NAME                 MTU   MAC-ADDRESS         ARP
   0  R ether1               1500  00:30:4F:0B:7B:C1 enabled
   1  R ether2               1500  00:30:4F:06:62:12 proxy-arp
 [admin@MikroTik] interface ethernet>
```

Reply Only

If arp property is set to `reply-only` on the interface, then router only replies to ARP requests. Neighbour MAC addresses will be resolved using /ip arp statically, but there will be no need to add the router's MAC address to other hosts' ARP tables like in case if arp is disabled.

Manual:IPv6/Address

Applies to RouterOS: v3, v4 +

Summary

Sub-menu: `/ipv6 address`
Standards: `RFC 4291`

IPv6 uses 16 bytes addresses compared to 4 byte addresses in IPv4. IPv6 address syntax and types are described in RFC 4291.

There are multiple IPv6 address types, that can be recognized by their prefix. RouterOS distinguishes the following:

- multicast (with prefix ff00::/8)
- link-local (with prefix fe80::/10)
- loopback (the address ::1/128)
- unspecified (the address ::/128)
- other (all other addresses, including the obsoleted site-local addresses, and RFC 4193 unique local addresses; they all are treated as global unicast).

One difference between IPv6 and IPv4 addressing is that IPv6 automatically generates a link-local IPv6 address for each active interface that has IPv6 support.

Address Expression

IPv6 addresses are represented a little bit different than IPv4 addresses. For IPv6, the 128-bit address is divided in eight 16-bit blocks, and each 16-bit block is converted to a 4-digit hexadecimal number and separated by colons. The resulting representation is called **colon-hexadecimal**.

In example above IPv6 address in binary format is converted to **colon-hexadecimal** representation

```
0010000000000001 0000010001110000 0001111100001001 0000000100110001
0000000000000000 0000000000000000 0000000000000000 0000000000001001
```

```
2001:0470:1f09:0131:0000:0000:0000:0009
```

IPv6 address can be further simplified by removing leading zeros in each block:

```
2001:470:1f09:131:0:0:0:9
```

As you can see IPv6 addresses can have long sequences of zeros. These contiguous sequence can be compressed to **::**

```
2001:470:1f09:131::9
```

Note: Zero compression can only be used once. Otherwise, you could not determine the number of 0 bits represented by each instance of a double-colon

Prefix

IPv6 prefix is written in `address/prefix-length` format. Compared to IPv4 decimal representation of network mask cannot be used. Prefix examples:

```
2001:470:1f09:131::/64
2001:db8:1234::/48
2607:f580::/32
2000::/3
```

Address Types

Several IPv6 address types exist:

- Unicast
- Anycast
- Multicast

As you can see there are no Broadcast addresses in ipv6 network, compared to IPv4 broadcast functionality was completely replaced with multicast.

Unicast Addresses

Packets addressed to a unicast address are delivered only to a single interface. To this group belong:

- globally unique addresses and can be used to connect to addresses with global scope anywhere.
- link-local addresses
- site-local addresses (FEC0::/48) - deprecated
- special purpose addresses
- compatibility addresses

Global unicast address can be automatically assigned to the node by ***Stateless Address auto-configuration***. `Read More >>`.

Link-local address

A link-local address is required on every IPv6-enabled interface, applications may rely on the existence of a link-local address even when there is no IPv6 routing, that is why link-local address is generated automatically for every active interface using it's interface identifier (calculated EUI-64 from MAC address if present). Address prefix is always FE80::/64 and IPv6 router never forwards link-local traffic beyond the link.

These addresses are comparable to the auto-configuration addresses 169.254.0.0/16 of IPv4.

A link-local address is also required for Neighbor Discovery processes.

Note: If interface is set as bridge port, interface specific link-local address is removed leaving only bridge link-local address

Special purpose address

Address	Description
Unspecified address (::/128)	Never assigned to an interface or used as a destination address, used only to indicate the absence of an address. Equivalent to IPv4 0.0.0.0 address.
loopback address (::1/128)	Used to identify a loopback interface, enabling a node to send packets to itself. It is equivalent to the IPv4 loopback address of 127.0.0.1.

Compatibility address

Address	Description
IPv4 compatible address	used by dual-stack nodes that are communicating with IPv6 over an IPv4 infrastructure. When the IPv4-compatible address is used as an IPv6 destination, IPv6 traffic is automatically encapsulated with an IPv4 header and sent to the destination by using the IPv4 infrastructure. Address is written in following format `::w.x.y.z`, where w.x.y.z is the dotted decimal representation of a public IPv4 address.
IPv4 mapped address	used to represent an IPv4-only node to an IPv6 node. It is used only for internal representation. The IPv4-mapped address is never used as a source or destination address for an IPv6 packet. The IPv6 protocol does not support the use of IPv4-mapped addresses. Address is written in following format: `::ffff:w.x.y.z`, where w.x.y.z is the dotted decimal representation of a public IPv4 address.
2002::/16	this prefix is used for 6to4 addressing. Here, an address from the IPv4 network 192.88.99.0/24 is also used.

Multicast address

Most important multicast aspects are:

- traffic is sent to a single address but is processed by multiple hosts;
- group membership is dynamic, allowing hosts to join and leave the group at any time;
- in IPv6, Multicast Listener Discovery (MLD) messages are used to determine group membership on a network segment, also known as a link or subnet;
- host can send traffic to the group's address without belonging to the corresponding group.

A single IPv6 multicast address identifies each multicast group. Each group's reserved IPv6 address is shared by all host members of the group who listen and receive any IPv6 messages sent to the group's address.

Multicast address consists of the following parts: [1]

- The first 8 bits in multicast address is always 1111 1111 (which is FF in hexadecimal format).
- *Flag* uses the 9th to 12th bit and shows if this multicast address is predefined (well-known) or not. If it is well-known, all bits are 0s.
- *Scope ID* indicates to which scope multicast address belongs, for example, Scope ID=2 is link-local scope.
- *Group ID* is used to specify a multicast group. There are predefined group IDs, such as Group ID=1 - all nodes. Therefore, if multicast address is ff02::1, that means Scope ID=2 and Group ID=1, indicating all nodes in link-local scope. This is analogous to broadcast in IPv4.

Here is the table of reserved IPV6 addresses for multicasting:

Address	Description
`FF02::1`	The all-nodes address used to reach all nodes on the same link.
`FF02::2`	The all-routers address used to reach all routers on the same link.
`FF02::5`	The all-Open Shortest Path First (OSPF) routers address used to reach all OSPF routers on the same link.
`FF02::6`	The all-OSPF designated routers address used to reach all OSPF designated routers on the same link.
`FF02::1:FFXX:XXXX`	The solicited-node address used in the address resolution process to resolve the IPv6 address of a link-local node to its link-layer address. The last 24 bits (XX:XXXX) of the solicited-node address are the last 24 bits of an IPv6 unicast address.

The following table is a partial list of IPv6 multicast addresses that are reserved for IPv6 multicasting and registered with the Internet Assigned Numbers Authority (IANA). For complete list of assigned addresses read IANA document [2].

Multicast addresses can be used to discover nodes in a network. For example, discover all nodes

```
mrz@bumba:/media/aaa/ver$ ping6 ff02::1%eth0
PING ff02::1%eth0(ff02::1) 56 data bytes
64 bytes from fe80::21a:4dff:fe5d:8e56: icmp_seq=1 ttl=64 time=0.037 ms
64 bytes from fe80::20c:42ff:fe0d:2c38: icmp_seq=1 ttl=64 time=4.03 ms (DUP!)
64 bytes from fe80::20c:42ff:fe28:7945: icmp_seq=1 ttl=64 time=5.59 ms (DUP!)
64 bytes from fe80::20c:42ff:fe49:fce5: icmp_seq=1 ttl=64 time=5.60 ms (DUP!)
64 bytes from fe80::20c:42ff:fe21:f1ec: icmp_seq=1 ttl=64 time=5.88 ms (DUP!)
64 bytes from fe80::20c:42ff:fe72:a1b0: icmp_seq=1 ttl=64 time=6.70 ms (DUP!)
```

discover all routers

```
mrz@bumba:/media/aaa/ver$ ping6 ff02::2%eth0
PING ff02::2%eth0(ff02::2) 56 data bytes
64 bytes from fe80::20c:42ff:fe28:7945: icmp_seq=1 ttl=64 time=0.672 ms
64 bytes from fe80::20c:42ff:fe0d:2c38: icmp_seq=1 ttl=64 time=1.44 ms (DUP!)
```

Anycast address

Anycast address is a new type of address incorporated in IPv6.

Anycasting is a new networking paradigm supporting service–oriented Addresses where an identical address can be assigned to multiple nodes providing a specific service. An anycast packet (i.e., one with an anycast destination address) is delivered to one of these nodes with the same anycast address.

Anycast address is not assigned a specific address range. It is assigned from unicast address range.

Interface Identifier

The last 64 bits of an IPv6 address are the interface identifier that is unique to the 64-bit prefix of the IPv6 address. There are several ways how to determine interface identifier:

- EUI-64;
- randomly generated to provide a level of anonymity;
- manually configured.

EUI-64

Traditional interface identifiers for network adapters are 48-bit MAC address. This address consists of a 24-bit manufacturer ID and a 24-bit board ID.

IEEE EUI-64 is a new standard for network interface addressing. The company ID is still 24-bits in length, but the extension ID is 40 bits, creating a much larger address space for a network adapters.

To create an EUI-64 address from the interface MAC address:

- 0xFFFE is inserted into the MAC address between the manufacturer ID and the board ID.
- seventh bit of the first byte is reversed.

Lets make an example with following MAC address `00:0C:42:28:79:45`.

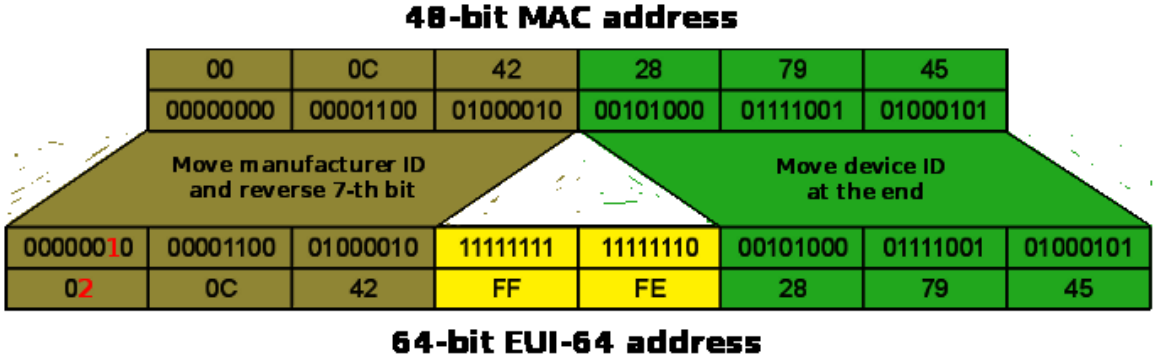


Image above illustrates conversation process. When the result is converted to colon-hexadecimal notation, we get the interface identifier `20C:42FF:FE28:7945`. As the result, corresponds link-local address is

```
FE80::20C:42FF:FE28:7945/64
```

In RouterOS, if the *eui-64* parameter of an address is configured, the last 64 bits of that address will be automatically generated and updated using interface identifier. The last bits must be configured to be zero for this case. Example:

```
[admin@MikroTik] > ipv6 address add address=fc00:3::/64 interface=ether3 eui-64=yes
[admin@MikroTik] > ipv6 address print
Flags: X - disabled, I - invalid, D - dynamic, G - global, L - link-local
 #    ADDRESS                                     INTERFACE                   ADVERTISE
 ...
 5  G fc00:3::20c:42ff:fe1d:3d4/64                ether3                      yes
[admin@MikroTik] > interface ethernet set ether3 mac-address=10:00:00:00:00:01
[admin@MikroTik] > ipv6 address print
Flags: X - disabled, I - invalid, D - dynamic, G - global, L - link-local
 #    ADDRESS                                     INTERFACE                   ADVERTISE
 ...
 5  G fc00:3::1200:ff:fe00:1/64                   ether3                      yes
```

Properties

Property	Description
address (*Address/Netmask*; Default:)	Ipv6 address. Allowed netmask range is 0..128. Address can also be constructed from the pool if **from-pool** property is specified. For example if address is set to ::1/64 then address will be constructed as follows <prefix_from_pool>::1/64
advertise (*yes* | *no*; Default: **no**)	Whether to enable stateless address configuration. The prefix of that address is automatically advertised to hosts using ICMPv6 protocol. The option is set by default for addresses with prefix length 64. Read more >>
comment (*string*; Default:)	Descriptive name of an item
disabled (*yes* | *no*; Default: **no**)	Whether address is disabled or not. By default it is disabled
eui-64 (*yes* | *no*; Default: **no**)	Whether to calculate EUI-64 address and use it as last 64 bits of the IPv6 address. Read more >>
from-pool (*string*; Default:)	Name of the pool from which prefix will be taken to construct IPv6 address taking last part of the address from **address** property. See example >>
interface (*string*; Default:)	Name of an interface on which Ipv6 address is set.

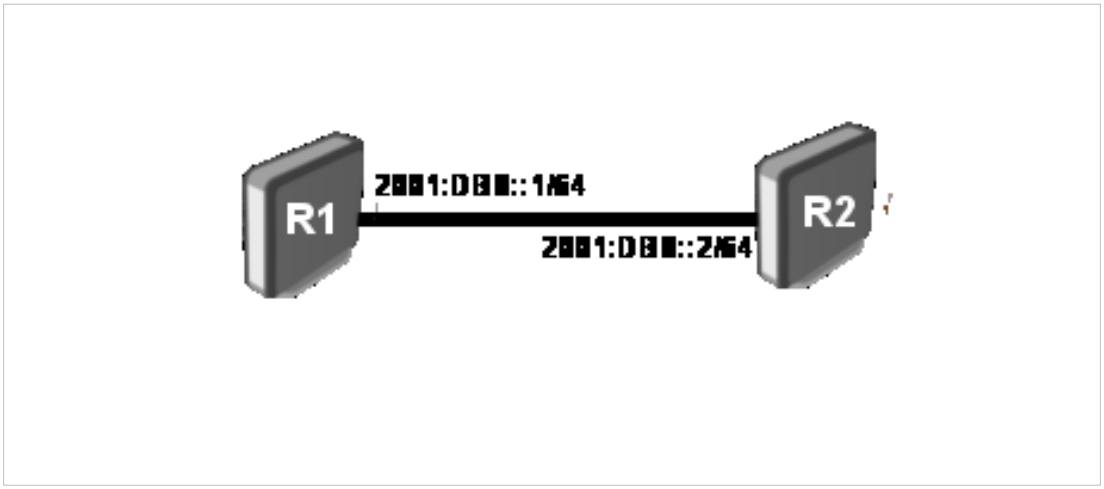
Read-only properties

Property	Description
actual-interface (*string*)	Actual interface on which address is set up. For example, if address was configured on ethernet interface and ethernet interface was added to bridge, then actual interface is bridge not ethernet.
dynamic (*yes* | *no*)	Whether address is dynamically created
global (*yes* | *no*)	Whether address is global
invalid (*yes* | *no*)	
link-local (*yes* | *no*)	Whether address is link local

Examples

Manual address configuration

This example shows how to set up simple addressing with global IPv6 addresses between two routers.



R1 configuration:

```
/ipv6 address
add address=2001:DB8::1/64 interface=ether1 advertise=no
```

R2 configuration:

```
/ipv6 address
add address=2001:DB8::2/64 interface=ether1 advertise=no
```

Check address list

```
[admin@R1] /ipv6 address> print
Flags: X - disabled, I - invalid, D - dynamic, G - global, L - link-local
 #    ADDRESS                                     FROM-POOL INTERFACE     ADVERTISE
 0  G 2001:db8::1/64                                       ether1        no
 3 DL fe80::219:d1ff:fe39:3535/64                           ether1        no
```

Notice that our added address has G flag indicated that this address can be globally routed. We also have link local address on the interface which is created automatically for every IPv6 capable interface.

Test connectivity

```
[admin@R1] /ipv6 address> /ping 2001:DB8::2
HOST                                     SIZE TTL TIME  STATUS
2001:db8::2                                56  64 12ms  echo reply
2001:db8::2                                56  64 0ms   echo reply
    sent=2 received=2 packet-loss=0% min-rtt=0ms avg-rtt=6ms max-rtt=12ms
```

[Top | Back to Content **]**

References

[1] http://www.ipv6style.jp/files/ipv6/en/tech/20030228/images/1.gif
[2] http://www.iana.org/assignments/ipv6-multicast-addresses/

Manual:Router AAA

Applies to RouterOS: 2.9, v3, v4, v5+

Summary

Sub-menu: `/user`

MikroTik RouterOS router user facility manage the users connecting the router from the local console, via serial terminal, telnet, SSH or Winbox. The users are authenticated using either local database or designated RADIUS server.

Each user is assigned to a user group, which denotes the rights of this user. A group policy is a combination of individual policy items.

In case the user authentication is performed using RADIUS, the RADIUS Client should be previously configured.

User Groups

Sub-menu: `/user group`

The router user groups provide a convenient way to assign different permissions and access rights to different user classes.

Properties

Property	Description
name (*string*; Default:)	The name of the user group
policy (*local* | *telnet* | *ssh* | *ftp* | *reboot* | *read* | *write* | *policy* | *test* | *web* | *sniff* | *api* | *winbox* | *password* | *sensitive*; Default:)	List of allowed policies:

- `local` - policy that grants rights to log in locally via console
- `telnet` - policy that grants rights to log in remotely via telnet
- `ssh` - policy that grants rights to log in remotely via secure shell protocol
- `ftp` - policy that grants full rights to log in remotely via FTP and to transfer files from and to the router. Users with this policy can both read, write and erase files, regardless of "read/write" permission, as that deals only with RouterOS configuration.
- `reboot` - policy that allows rebooting the router
- `read` - policy that grants read access to the router's configuration. All console commands that do not alter router's configuration are allowed. Doesn't affect FTP
- `write` - policy that grants write access to the router's configuration, except for user management. This policy does not allow to read the configuration, so make sure to enable read policy as well
- `policy` - policy that grants user management rights. Should be used together with write policy
- `test` - policy that grants rights to run ping, traceroute, bandwidth-test and wireless scan, sniffer and snooper commands
- `web` - policy that grants rights to log in remotely via WebBox
- `winbox` - policy that grants rights to log in remotely via WinBox
- `password` - policy that grants rights to change the password
- `sensitive` - grants rights to see sensitive information in the router, see below list as to what is regarded as sensitive.
- `api` - grants rights to access router via API.
- `sniff` - policy that grants rights to use packet sniffer tool.

Sensitive information

Starting with RouterOS v3.27, the following information is regarded as sensitive, and can be hidden from certain user groups with the **'sensitive**' policy unchecked.

Also, since RouterOS v4.3, backup files are considered sensitive, and users without this policy will not be able to download them in any way.

system package

```
/radius: secret
/snmp/community: authentication-password, encryption-password
```

advanced-tools package

```
/tool/sms: secret
```

wireless package

```
/interface/wireless/security-profiles: wpa-pre-shared-key,
wpa2-pre-shared-key, static-key-0, static-key-1, static-key-2,
static-key-3, static-sta-private-key
/interface/wireless/access-list: private-key, private-pre-shared-key
```

wireless-test package

```
/interface/wireless/security-profiles: wpa-pre-shared-key, wpa2-pre-shared-key,
static-key-0, static-key-1, static-key-2, static-key-3, static-sta-private-key, management-protection-key
/interface/wireless/access-list: private-key, private-pre-shared-key, management-protection-key
```

user-manager package

```
/tool/user-manager/user: password
/tool/user-manager/customer: password
```

hotspot package

```
/ip/hotspot/user: password
```

ppp package

```
/ppp/secret: password
```

security package

```
/ip/ipsec/installed-sa: auth-key, enc-key
/ip/ipsec/manual-sa: ah-key, esp-auth-key, esp-enc-key
/ip/ipsec/peer: secret
```

routing package

```
/routing/bgp/peer: tcp-md5-key
/routing/rip/interface: authentication-key
/routing/ospf/interface: authentication-key
/routing/ospf/virtual-link: authentication-key
```

routing-test package

```
/routing/bgp/peer: tcp-md5-key
/routing/rip/interface: authentication-key
/routing/ospf/interface: authentication-key
/routing/ospf/virtual-link: authentication-key
```

Notes

There are three system groups which cannot be deleted:

```
[admin@rb13] > /user group print
 0 name="read" policy=local,telnet,ssh,reboot,read,test,winbox,password,web,!ftp,!write,!policy

 1 name="write" policy=local,telnet,ssh,reboot,read,write,test,winbox,password,web,!ftp,!policy

 2 name="full" policy=local,telnet,ssh,ftp,reboot,read,write,policy,test,winbox,password,web

 3 name="test" policy=ssh,read,policy,!local,!telnet,!ftp,!reboot,!write,!test,!winbox,!password,!web
[admin@rb13] >
```

Exclamation sign '!' just before policy item name means NOT.

Example

To add reboot group that is allowed to reboot the router locally or using telnet, as well as read the router's configuration, enter the following command:

```
[admin@rb13] user group> add name=reboot policy=telnet,reboot,read,local
[admin@rb13] user group> print
 0 name="read" policy=local,telnet,ssh,reboot,read,test,winbox,password,web,!ftp,!write,!policy

 1 name="write" policy=local,telnet,ssh,reboot,read,write,test,winbox,password,web,!ftp,!policy
```

```
 2 name="full" policy=local,telnet,ssh,ftp,reboot,read,write,policy,test,winbox,password,web

 3 name="reboot" policy=local,telnet,reboot,read,!ssh,!ftp,!write,!policy,!test,!winbox,!password,!web
[admin@rb13] user group>
```

Router Users

Sub-menu: `/user`

Router user database stores the information such as username, password, allowed access addresses and group about router management personnel.

Properties

Property	Description
address (*IP/mask | IPv6 prefix*; Default:)	Host or network address from which the user is allowed to log in
group (*string*; Default:)	Name of the group the user belongs to
name (*string*; Default:)	User name. Although it must start with an alphanumeric character, it may contain "*", "_", "." and "@" symbols.
password (*string*; Default:)	User password. If not specified, it is left blank (hit [Enter] when logging in). It conforms to standard Unix characteristics of passwords and may contain letters, digits, "*" and "_" symbols.

Notes

There is one predefined user with full access rights:

```
[admin@MikroTik] user> print
Flags: X - disabled
  #    NAME                                              GROUP ADDRESS
  0    ;;; system default user
       admin                                             full  0.0.0.0/0

[admin@MikroTik] user>
```

There always should be at least one user with fulls access rights. If the user with full access rights is the only one, it cannot be removed.

Monitoring Active Users

Sub-menu: `/user active`

`/user active print` command shows the currently active users along with respective statisics information.

Properties

All properties are read-only.

Property	Description
address (*IP/IPv6 address*)	Host IP/IPv6 address from which the user is accessing the router. 0.0.0.0 means that user is logged in locally
group (*string*)	Group that user belongs to.
name (*string*)	User name.
radius (*true* | *false*)	Whether user is authenticated by RADIUS server.
via (*console* | *telnet* | *ssh* |*winbox* | *api* | *web*)	User's access method
when (*time*)	Time and date when user logged in.

Example

To print currently active users, enter the following command:

```
[admin@dzeltenais_burkaans] /user active> print detail
Flags: R - radius
 0   when=dec/08/2010 16:19:24 name="admin" address=10.5.8.52 via=winbox

 2   when=dec/09/2010 09:23:04 name="admin" address=10.5.101.38 via=telnet

 3   when=dec/09/2010 09:34:27 name="admin" address=fe80::21a:4dff:fe5d:8e56 via=api
```

Remote AAA

Sub-menu: `/user aaa`

Router user remote AAA enables router user authentication and accounting via RADIUS server. The RADIUS user database is consulted only if the required username is not found in the local user database

Properties

Property	Description
accounting (*yes* | *no*; Default: **yes**)	
exclude-groups (*list of group names*; Default:)	Exclude-groups consists of the groups that should not be allowed to be used for users authenticated by radius. If radius server provides group specified in this list, default-group will be used instead. This is to protect against privilege escalation when one user (without policy permission) can change radius server list, setup it's own radius server and log in as admin.
default-group (*string*; Default: **read**)	User group used by default for users authenticated via RADIUS server.
interim-update (*time*; Default: **0s**)	Interim-Update time interval
use-radius (*yes* |*no*; Default: **no**)	Enable user authentication via RADIUS

Note: If you are using RADIUS, you need to have CHAP support enabled in the RADIUS server for Winbox to work

SSH Keys

Sub-menu: `/user ssh-keys`

This menu allows to import public keys used for ssh authentication.

Warning: User is not allowed to login via ssh by password if ssh-keys for the user is added

Properties:

Property	Description
user (*string*; Default:)	username to which ssh key is assigned.

Read-only properties:

Property	Description
key-owner (*string*)	

When importing ssh key by `/user ssh-keys import` command you will be asked for two parameters:

- **public-key-file** - file name in routers root directory containing the key.
- **user** - name of the user to which key will be assigned

Private keys

Sub-menu: `/user ssh-keys private`

This menu is used to import and list imported private keys. Private keys are used to authenticate remote login attempts using certificates.

Read-only properties:

Property	Description
user (*string*)	
key-owner (*string*)	

When importing ssh keys from this sub menu using `/user ssh-keys private` import command you will be asked for three parameters:

- **private-key-file** - file name in routers root directory containing private key.
- **public-key-file** - file name in routers root directory containing public key.
- **user** - name of the user to which key will be assigned

Example

Read full example >>

Manual:BGP based VPLS

Overview

MPLSVPLS page covers general introduction to VPLS service and configuration of LDP based VPLS tunnels. Due to their static nature LDP based VPLS tunnels have scalability issues that arise when number of VPLSes and sites participating in VPLSes grow. One of the problems is the requirement to maintan full mesh of LDP tunnels between sites forming VPLS. In case number of sites in VPLS is high, adding new site to existing VPLS can become burdensome for network administrator.

BGP based autodiscovery and signaling of VPLS tunnels can help to avoid complexity of configuration at the expense of running BGP protocol between VPLS routers. In general, BGP based VPLS serves two purposes:

- autodiscovery: there is no need to configure each VPLS router with all remote endpoints of VPLS tunnels, provided there are means to deliver BGP multiprotocol NLRIs between them - routers figure out remote endpoints of tunnels from received BGP Updates;
- signaling: labels used for VPLS tunnels by remote endpoints are distributed in the same BGP Updates, this means there is no need for targeted LDP sessions between tunnel endpoints as in case of LDP signaled VPLS.

For example, if LDP signaled VPLS is used, adding new site to existing VPLS would mean configuring router that connects new site to establish tunnels with the rest of sites and also configure all other routers to establish tunnels with router connecting this new site. BGP based VPLS, if configured properly eliminates need to adjust configuration on all routers forming VPLS.

The requirement to exchange BGP NLRIs between VPLS routers means that either full mesh of BGP sessions need to be established among routers forming VPLS or route reflector must be used. In case full mesh of BGP sessions are established between VPLS routers, the benefits of BGP based VPLS over LDP signaled VPLS are questionable - when new site is added to VPLS, BGP peer configuration still needs to be entered on every router forming given VPLS. When BGP route reflector is used, adding new site to VPLS becomes more simple - router connecting new site must only peer with route reflector and no additional configuration is required on other routers. Taking into account that route reflector can also be one of routers forming VPLS, there is no need for additional separate equipment. Of course, scalability and availability concerns still must be taken into account - multiple route reflectors can be used for backup purposes as well as for distributing information load.

The drawback of running BGP based VPLS is requirement to configure BGP which requires that network administrator has at least basic understanding of BGP, its multiprotocol capabilities and route reflectors. Therefore it is advised to implement LDP signaled VPLS if amount of sites and VPLS networks is small, topology is more static - that is, benefits of using BGP are not obvious.

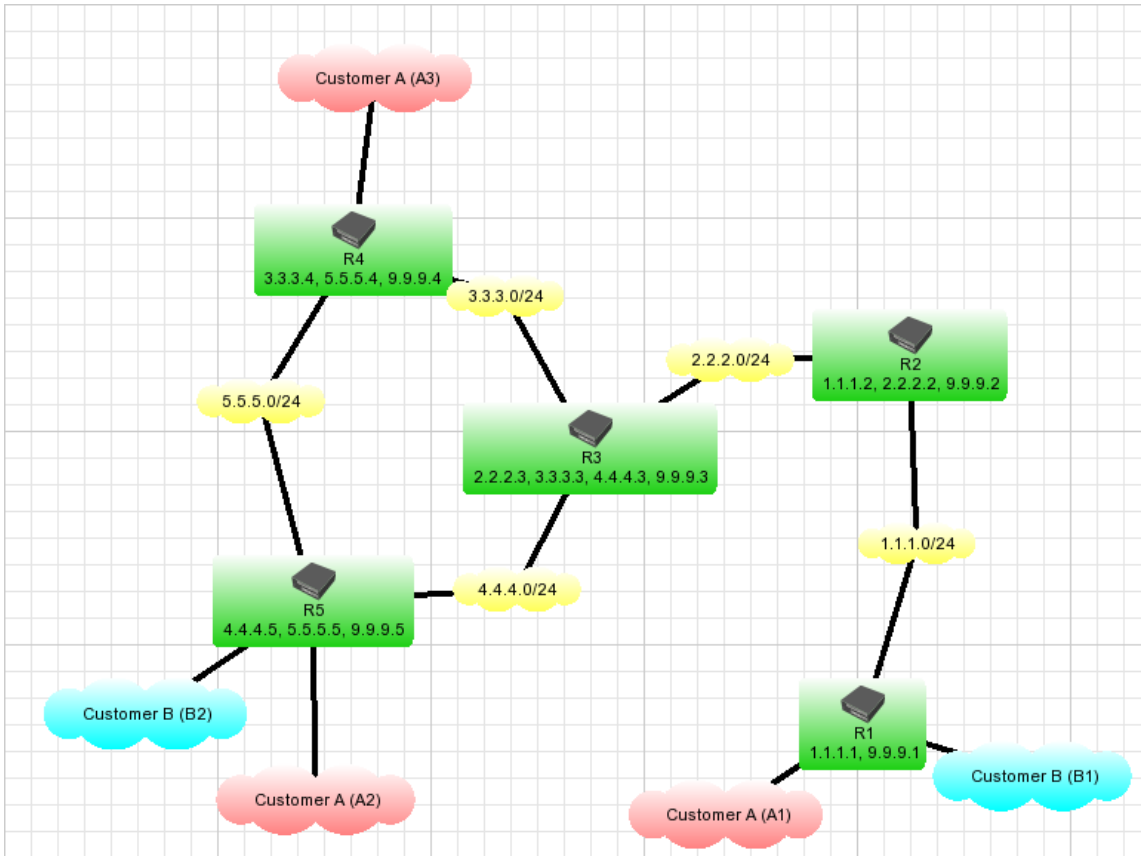
Note that BGP based VPLS is a method only for VPLS tunnel label exchange, it does not deal with delivery of traffic between VPLS tunnel endpoints, so general MPLS frame delivery between tunnel endpoints must be ensured as discussed in MPLSVPLS.

Suggested reading material:

- RFC 4761, Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling
- RFC 4456, BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)

Example network

Consider the same network as used for LDP signaled VPLS example in MPLSVPLS:



The requirements of customers A and B are the same - ethernet segments must be transparently connected. Taking into account simplicity of given network topology Service Provider has decided to use R5 as route reflector and to have no backup route reflector. Consider that MPLS switching is configured and running, as discussed in MPLSVPLS, but no any VPLS configuration has been applied yet. the rest of this document deals with specifics that are introduced by use of BGP for VPLS signaling.

Configuring IBGP session for VPLS signaling

At first, BGP instance must be configured, default instance can also be used:

```
[admin@R1] /routing bgp instance> print
Flags: X - disabled
 0   name="default" as=65530 router-id=0.0.0.0 redistribute-connected=no redistribute-static=no
     redistribute-rip=no redistribute-ospf=no redistribute-other-bgp=no out-filter=""
     client-to-client-reflection=yes ignore-as-path-len=no
```

To enable VPLS NLRI delivery across BGP, BGP multiprotocol capability must be used. This is enabled by specifying **l2vpn** in BGP peer's **address-families** setting.

For example, to configure BGP connection between R1 and R5, the following commands should get issued.

On R1:

```
[admin@R1] /routing bgp peer> add remote-address=9.9.9.5 remote-as=65530 address-families=l2vpn \
  update-source=lobridge
```

and on R5:

```
[admin@R5] /routing bgp peer> add remote-address=9.9.9.1 remote-as=65530 address-families=l2vpn \
  update-source=lobridge
```

BGP connection should get established between R1 and R5. This can be confirmed by:

```
[admin@R1] /routing bgp peer> print status
Flags: X - disabled
 0   name="peer1" instance=default remote-address=9.9.9.5 remote-as=65530 tcp-md5-key=""
     nexthop-choice=default multihop=no route-reflect=no hold-time=3m ttl=255 in-filter=""
     out-filter="" address-families=l2vpn update-source=lobridge remote-id=4.4.4.5
     local-address=9.9.9.1 uptime=3s prefix-count=0 updates-sent=0 updates-received=0
     withdrawn-sent=0 withdrawn-received=0 remote-hold-time=3m used-hold-time=3m
     used-keepalive-time=1m refresh-capability=yes state=established
```

There are several things to note about BGP peer configuration:

- there is no need to distribute any IP or IPv6 routes and even no need have IP or IP6 support over BGP connection at all to be able to exchange VPLS NLRIs, it is sufficient to specify **address-families=l2vpn**
- "loopback" addresses of routers are used as BGP peer addresses (local address is configured by means of **update-source** setting). BGP peer, when originating VPLS NLRI, specifies its local address as BGP NextHop (for example, in given setup R1 originating BGP NLRIs will use address 9.9.9.1 as BGP NextHop address), receiving VPLS router uses received BGP NextHop address as tunnel endpoint address and therefore uses transport label that ensures delivery to BGP NextHop. In order for penultimate hop popping to work properly, it is advised to use loopback IP address for this. See penultimate hop popping related discussion in MPLSVPLS.

Configuring Route Reflector

In its simplest sense BGP Route Reflector re-advertises received IBGP routes without changing BGP NextHop for route. This feature can be used to avoid setting up full mesh of BGP connections. Note that for router be able to operate as route reflector for VPLS NLRIs, it is not necessary for it to participate in any VPLS, it is even not necessary for it to have MPLS support. Still it is mandatory for VPLS routers to be able to establish BGP sessions with route reflector, therefore IP connectivity is a must.

Route reflector's BGP instance must be configured with **client-to-client-reflection=yes** setting:

```
[admin@R5] /routing bgp instance> print
Flags: X - disabled
 0   name="default" as=65530 router-id=0.0.0.0 redistribute-connected=no redistribute-static=no
     redistribute-rip=no redistribute-ospf=no redistribute-other-bgp=no out-filter=""
     client-to-client-reflection=yes ignore-as-path-len=no
```

Additionaly, peers on route reflector must be configured with **route-reflect=yes** setting:

```
[admin@R5] /routing bgp peer> print
Flags: X - disabled
 0   name="peer1" instance=default remote-address=9.9.9.1 remote-as=65530 tcp-md5-key=""
     nexthop-choice=default multihop=no route-reflect=no hold-time=3m ttl=255 in-filter=""
     out-filter="" address-families=l2vpn update-source=lobridge
[admin@R5] /routing bgp peer> set 0 route-reflect=yes
[admin@R5] /routing bgp peer> print
Flags: X - disabled
 0   name="peer1" instance=default remote-address=9.9.9.1 remote-as=65530 tcp-md5-key=""
```

```
     nexthop-choice=default multihop=no route-reflect=yes hold-time=3m ttl=255 in-filter=""
     out-filter="" address-families=l2vpn update-source=lobridge
```

To enable R5 to operate as route reflector, all its peers should get added with **route-reflect=yes** setting. So to enable proper VPLS NLRI distribution, R5 must be configured with 2 BGP peers - R1 and R4:

```
[admin@R5] /routing bgp peer> print status
Flags: X - disabled
 0   name="peer1" instance=default remote-address=9.9.9.1 remote-as=65530 tcp-md5-key=""
     nexthop-choice=default multihop=no route-reflect=yes hold-time=3m ttl=255 in-filter=""
     out-filter="" address-families=l2vpn update-source=lobridge remote-id=1.1.1.1
     local-address=9.9.9.5 uptime=5m55s prefix-count=0 updates-sent=0 updates-received=0
     withdrawn-sent=0 withdrawn-received=0 remote-hold-time=3m used-hold-time=3m
     used-keepalive-time=1m refresh-capability=yes state=established

 1   name="peer2" instance=default remote-address=9.9.9.4 remote-as=65530 tcp-md5-key=""
     nexthop-choice=default multihop=no route-reflect=yes hold-time=3m ttl=255 in-filter=""
     out-filter="" address-families=l2vpn update-source=lobridge remote-id=3.3.3.4
     local-address=9.9.9.5 uptime=23s prefix-count=0 updates-sent=0 updates-received=0
     withdrawn-sent=0 withdrawn-received=0 remote-hold-time=3m used-hold-time=3m
     used-keepalive-time=1m refresh-capability=yes state=established
```

But R1 and R4 must only peer with R5. On R1:

```
[admin@R1] /routing bgp peer> print status
Flags: X - disabled
 0   name="peer1" instance=default remote-address=9.9.9.5 remote-as=65530 tcp-md5-key=""
     nexthop-choice=default multihop=no route-reflect=no hold-time=3m ttl=255 in-filter=""
     out-filter="" address-families=l2vpn update-source=lobridge remote-id=4.4.4.5
     local-address=9.9.9.1 uptime=6m33s prefix-count=0 updates-sent=0 updates-received=0
     withdrawn-sent=0 withdrawn-received=0 remote-hold-time=3m used-hold-time=3m
     used-keepalive-time=1m refresh-capability=yes state=established
```

and on R4:

```
[admin@R4] /routing bgp peer> print status
Flags: X - disabled
 0   name="peer1" instance=default remote-address=9.9.9.5 remote-as=65530 tcp-md5-key=""
     nexthop-choice=default multihop=no route-reflect=no hold-time=3m ttl=255 in-filter=""
     out-filter="" address-families=l2vpn update-source=lobridge remote-id=4.4.4.5
     local-address=9.9.9.4 uptime=3s prefix-count=0 updates-sent=0 updates-received=0
     withdrawn-sent=0 withdrawn-received=0 remote-hold-time=3m used-hold-time=3m
     used-keepalive-time=1m refresh-capability=yes state=established
```

Using route reflector means that in order to add new site to some VPLS, e.g. connected by router Ry, would mean adding Ry as BGP peer to R5 (with route-reflect=yes setting) and adding R5 as BGP peer to Ry.

Configuring BGP signaled VPLS

Configuring ethernet bridging

BGP signalled VPLS tunnels are created dynamically when proper BGP NLRIs are received. Therefore there is no need to configure any VPLS interfaces. Still, to transparently deliver packets from ethernet segment across VPLS bridging must be configured. For example, on R1 two bridges are created, named "A" and "B" with appropriate customer-facing ethernet interfaces added to them:

```
[admin@R1] /interface bridge> print
Flags: X - disabled, R - running
 0  R name="lobridge" mtu=1500 arp=enabled mac-address=00:00:00:00:00:00 protocol-mode=none
      priority=0x8000 auto-mac=yes admin-mac=00:00:00:00:00:00 max-message-age=20s
      forward-delay=15s transmit-hold-count=6 ageing-time=5m

 1  R name="A" mtu=1500 arp=enabled mac-address=00:01:50:E7:00:09 protocol-mode=none
      auto-mac=yes admin-mac=00:00:00:00:00:00 max-message-age=20s forward-delay=15s
      priority=0x8000 transmit-hold-count=6 ageing-time=5m

 2  R name="B" mtu=1500 arp=enabled mac-address=00:01:50:E7:00:08 protocol-mode=none
      auto-mac=yes admin-mac=00:00:00:00:00:00 max-message-age=20s forward-delay=15s
      priority=0x8000 transmit-hold-count=6 ageing-time=5m
[admin@R1] /interface bridge> port print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE                           BRIDGE PRIORITY PATH-COST            HORIZON
 0    ether2                              A      0x80     10                   none
 1    ether1                              B      0x80     10                   none
```

Configuring BGP signaled VPLS instances

Configuring BGP signaled VPLS instance makes router advertise VPLS BGP NLRI that advertises that particular router belongs to some VPLS. Upon receiving such advertisement, other members of same VPLS know to establish VPLS tunnel with this router.

To configure VPLS for customers A and B, on R1 the following commands should be issued:

```
[admin@R1] /interface vpls bgp-vpls> add bridge=A bridge-horizon=1 route-distinguisher=1:1 \
   site-id=1 import-route-targets=1:1 export-route-targets=1:1
[admin@R1] /interface vpls bgp-vpls> add bridge=B bridge-horizon=1 route-distinguisher=2:2 \
   site-id=1 import-route-targets=2:2 export-route-targets=2:2
```

Note: Since v3.20 **vpls-id** was replaced with separate import/export-route-targets to provide more flexibility.

route-distinguisher setting specifies value that gets attached to VPLS NLRI so that receiving routers can distinguish advertisements that may otherwise look the same. This implies that unique route-distinguisher for every VPLS must be used. It is not necessary to use the same route distinguisher for some VPLS on all routers forming that VPLS as distinguisher is not used for determining if some BGP NLRI is related to particular VPLS (Route Target attribute is used for this), but it is mandatory to have different distinguishers for different VPLSes.

export-route-targets setting is used for tagging BGP NLRI

import-route-targets setting is used to determine if BGP NLRI is related to particular VPLS

site-id setting must be unique among members of particular VPLS. It is advisable although not mandatory to allocate site-id values in as narrow range as possible as that increases efficency of BGP (for details see RFC 4761).

bridge setting specifies bridge to which dynamically created VPLS tunnels should get added.

bridge-horizon specifies horizon value to be used for ports added to bridge (see Split horizon bridging discussion in MPLSVPLS).

After configuring R4 as member of VPLS 1:1 (used for customer A) with command:

```
[admin@R4] /interface vpls bgp-vpls> add bridge=A bridge-horizon=1 route-distinguisher=1:1 \
   site-id=4 import-route-targets=1:1 export-route-targets=1:1
```

Dynamic VPLS tunnel gets created on both R1 and R4. On R1 this can be confirmed:

```
[admin@R1] > /interface vpls print
Flags: X - disabled, D - dynamic, R - running, B - bgp-signaled
 0 RDB name="vpls1" mtu=1500 mac-address=02:FA:33:C4:7A:A9 arp=enabled
       disable-running-check=no remote-peer=9.9.9.4 cisco-style=no
       cisco-style-id=0 vpls=bgp-vpls1
[admin@R1] > /interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE                           BRIDGE PRIORITY PATH-COST            HORIZON
 0    ether2                              A      0x80     10                   none
 1    ether1                              B      0x80     10                   none
 2  D vpls1                               A      0x80     50                   1
```

Here we have confirmed also that route reflection as configured on R5 works as expected as there is no BGP peer relationship between R1 and R4.

Additionally we must configure R5 to participate in VPLS for customer A:

```
[admin@R5] /interface vpls bgp-vpls> add bridge=A bridge-horizon=1 route-distinguisher=1:1 \
   site-id=5 import-route-targets=1:1 export-route-targets=1:1
```

This causes R1 and R4 to establish additional VPLS tunnel with R5. For example on R1:

```
[admin@R1] > /interface vpls print
Flags: X - disabled, D - dynamic, R - running, B - bgp-signaled
 0 RDB name="vpls1" mtu=1500 mac-address=02:FA:33:C4:7A:A9 arp=enabled
       disable-running-check=no remote-peer=9.9.9.4 cisco-style=no
       cisco-style-id=0 vpls=bgp-vpls1
 1 RDB name="vpls2" mtu=1500 mac-address=02:FF:B7:0E:4B:97 arp=enabled
       disable-running-check=no remote-peer=9.9.9.5 cisco-style=no
       cisco-style-id=0 vpls=bgp-vpls1
```

And bridge port to get added with proper horizon value:

```
[admin@R1] > /interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE                           BRIDGE PRIORITY PATH-COST            HORIZON
 0    ether2                              A      0x80     10                   none
 1    ether1                              B      0x80     10                   none
 2  D vpls1                               A      0x80     50                   1
 3  D vpls2                               A      0x80     50                   1
```

To complete the setup, necessary configuration for customer B VPLS should be applied to R5:

```
[admin@R5] /interface vpls bgp-vpls> add site-id=5 route-distinguisher=2:2 bridge=B \
    bridge-horizon=1 import-route-targets=2:2 export-route-targets=2:2
```

As the result we get full mesh of VPLS tunnels established, for example on R5:

```
[admin@R5] /interface vpls> print
Flags: X - disabled, D - dynamic, R - running, B - bgp-signaled

 0 RDB name="vpls1" mtu=1500 mac-address=02:FA:5C:28:29:D3 arp=enabled
       disable-running-check=no remote-peer=9.9.9.1 cisco-style=no
       cisco-style-id=0 vpls=bgp-vpls1
 1 RDB name="vpls2" mtu=1500 mac-address=02:EA:51:31:3E:2B arp=enabled
       disable-running-check=no remote-peer=9.9.9.4 cisco-style=no
       cisco-style-id=0 vpls=bgp-vpls1
 2 RDB name="vpls3" mtu=1500 mac-address=02:F6:CF:06:1E:CB arp=enabled
       disable-running-check=no remote-peer=9.9.9.1 cisco-style=no
       cisco-style-id=0 vpls=bgp-vpls2
```

Note that remote-peer for VPLS tunnels is BGP NextHop address as received in BGP Update. For example BGP logs on R5 when receiving Update for VPLS 2:2 (customer B), say:

```
11:24:06 route,bgp,debug,packet UPDATE Message
11:24:06 route,bgp,debug,packet     RemoteAddress=9.9.9.1
11:24:06 route,bgp,debug,packet     MessageLength=79
11:24:06 route,bgp,debug,packet
11:24:06 route,bgp,debug,packet     PathAttributes
11:24:06 route,bgp,debug,packet         bgp-origin=INCOMPLETE
11:24:06 route,bgp,debug,packet         bgp-nexthop=9.9.9.1
11:24:06 route,bgp,debug,packet         bgp-localpref=100
11:24:06 route,bgp,debug,packet         bgp-extended-communities=RT:2:2
11:24:06 route,bgp,debug,packet
11:24:06 route,bgp,debug,packet     NLRI= rd
11:24:06 route,bgp,debug,packet         type=0
11:24:06 route,bgp,debug,packet         administrator=2
11:24:06 route,bgp,debug,packet         assigned-number=2 veId=1 veBlockOffset=0 veBlockSize=16
                            labelBase=40
```

This is reflected for dynamic VPLS tunnel, where remote-peer for tunnel with export-route-targer 2:2 is 9.9.9.1. This implies that R5 uses IGP route that leads to 9.9.9.1 to decide what transport label to use. In given case there are /32 IGP routes distributed in the network by means of OSPF, therefore:

```
[admin@R5] /interface vpls> monitor 2 once
    remote-label: 45
     local-label: 40
   remote-status:
      igp-prefix: 9.9.9.1/32
     igp-nexthop: 4.4.4.3
  imposed-labels: 17,45
```

Shows that 9.9.9.1/32 route is used and immediate nexthop is 4.4.4.3. Labels attached to VPLS packets are 17 and 45 where 45 is label mapping received with BGP Update, and 17 is label assigned by R3 for prefix 9.9.9.1/32:

```
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 ...
14 AD 9.9.9.1/32         4.4.4.3         17         9.9.9.3:0
 ...
```

Manual:BGP Best Path Selection Algorithm

Introduction

With the full Internet BGP routing table being upward of 300K routes and with a BGP router having the potential to be receiving multiple copies of that routing table from multiple providers, it has to have some way to compare those multiple BGP routing tables and select only the best route to go into the IP routing table on the router. It uses the BGP Best Path Selection Algorithm to do this.

You should note that MikroTik and Cisco BGP routers have weight as the first criteria in the table where other brands do not.

Best path algorithm compares routes received by a **single BGP instance**. Routes installed by different BGP instances are compared by the general algorithm, i.e. route distances are compared and the route with lower distance is preferred.

BEST PATH ALGORITHM

1. Router is ignoring received path if the route is not **valid**. Route is valid if:
   - NEXT_HOP of the route is valid and reachable
   - AS_PATH received from external peers does not contain the local AS
   - route is not rejected by routing filters

   For more information read nexthop selection and validation.
2. The first path received is automatically considered 'best path'. Any further received paths are compared to first received to determine if the new path is better.
3. Prefer the path with the highest **WEIGHT**.

   WEIGHT parameter is local to the router on which it is configured. A route without assigned WEIGHT have a default value of 0.
4. Prefer the path with the highest **LOCAL_PREF**. It is used only within an AS.

   A path without **LOCAL_PREF** attribute have a value of 100 by default.
5. Prefer the path with the shortest **AS_PATH**. (skipped if `ignore-as-path-len` set to **yes**)

   Each **AS_SET** counts as 1, regardless of the set size. The **AS_CONFED_SEQUENCE** and **AS_CONFED_SET** are not included in the **AS_PATH** length.
6. Prefer the path that was locally originated via aggregate or BGP network
7. Prefer the path with the lowest **ORIGIN** type.

   Interior Gateway Protocol (IGP) is lower than Exterior Gateway Protocol (EGP), and EGP is lower than INCOMPLETE

in other words **IGP < EGP < INCOMPLETE**

8. Prefer the path with the lowest **multi-exit discriminator** (MED).

   The router compare MED attribute only for paths that have the same neighboring (leftmost) AS. Paths without explicit MED value are treated as with MED of 0
9. Prefer **eBGP** over **iBGP** paths
10. Prefer the route that comes from the BGP router with the lowest **router ID**. If a route carries the **ORIGINATOR_ID** attribute, then the **ORIGINATOR_ID** is used instead of router ID.
11. Prefer the route with the shortest **route reflection cluster list**. Routes without a cluster list are considered to have a cluster list of length 0.
12. Prefer the path that comes from the lowest neighbor address

Manual:BGP Case Studies

A good place to start learning about BGP in MikroTik RouterOS.

What is BGP?

The Border Getaway Protocol (BGP) is an inter-autonomous system routing protocol based on distance-vector algorithm. It is used to exchange routing information across the Internet and is the only protocol that is designed to deal with a network of the Internet's size and the only protocol that can deal well with having multiple connections to unrelated routing domains.

BGP is designed to allow for sophisticated administrative routing policies to be implemented. BGP does not exchange information about network topology but rather reachability information. As such, BGP is better suited to inter-AS environments and special cases like informational feeds. If you just need to enable dynamic routing in your network, consider OSPF instead.

How Does BGP Work?

BGP operates by exchanging network layer reachability information (NLRI). This information contains an indication to a what sequence of full paths (BGP AS numbers) the route should take in order to reach destination network (NLRI prefix).

BGP routers exchange reachability information by means of a transport protocol, which in case of BGP is TCP (port 179). Upon forming a TCP connection these routers exchange initial messages to negotiate and confirm connection parameters.

Any two routers that have established TCP connection to exchange BGP routing information are called peers, or neighbors. The peers initially exchange their full routing tables. After the initial exchange incremental updates are sent as the routing tables change. Thus, BGP does not require periodic refresh of the entire BGP routing table. BGP maintains routing table version number which must be the same between any two given peers for the duration of the connection. KeepAlive messages are sent periodically to ensure that the connection is up and running. BGP sends notification messages in response to errors or special conditions.

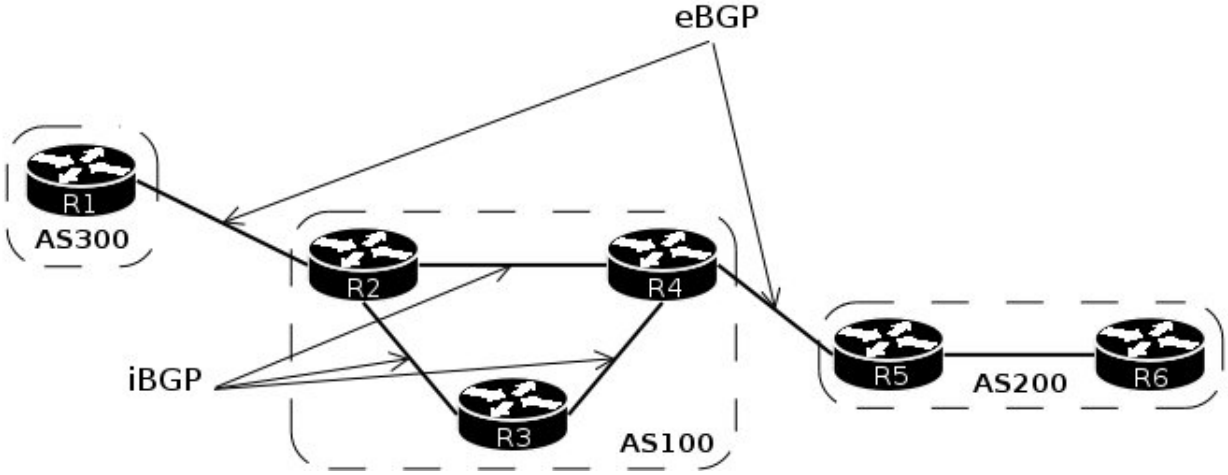
TCP protocol connection between two peers is closed when either an error has occured or no update messages or KeepAlive messages has been received during the period of BGP Hold Timer.

iBGP and eBGP

A particular AS might have multiple BGP speakers and provide transit service to other ASs. This implies that BGP speakers must maintain a consistent view of routing within the AS. A consistent view of the interior routes of the AS is provided by the interior routing protocol such as OSPF or RIP. A consistent view of the routes exterior to the AS is provided by having all BGP routers within the AS establishing direct BGP connections with each other.

Using a set of administrative policies BGP speakers within the AS arrive to an agreement as to which entry/exit point to use for a particular destination. This information is communicated to the interior routers of the AS using interior routing protocol.

Two BGP neighbors from different ASs are said to maintain an "external" link. Similarly, a BGP peer in a different AS is referred to as an external peer. BGP connections between peers within the same AS are known as "internal" links. BGP speakers that are connected by internal link are referred as internal peers. As far as this paper is concerned, iBGP refers to the BGP session between two peers in the same AS, or internal link. In turn, eBGP refers to the links between external BGP peers (these that are in different ASs).



Enabling BGP

To enable BGP assuming only one BGP process will be present in the system, it is enough to do the following:

- modify configuration of the default BGP instance. In particular, change instance AS number to the desired ASN:

```
[admin@rb11] > /routing bgp instance set default as=100 redistribute-static=no
[admin@rb11] > /routing bgp instance print Flags: X - disabled
 0   as=100 router-id=0.0.0.0 redistribute-static=no redistribute-connected=no
     redistribute-rip=no redistribute-ospf=no redistribute-other-bgp=no
     name="default" out-filter=""
[admin@rb11] >
```

Note, that, unless explicitly specified, BGP router ID is set as the least IP address on the router.

- add at least one BGP peer. Refer to the next section for more information on how to configure BGP peers.

BGP Peers

Two BGP routers have to establish TCP connection between each other to be considered as BGP peers. Since BGP requires a reliable transport for routing information, a TCP connection is essential for it to operate properly.

Once TCP connection is up, routers exchange some initial information such as the BGP router ID, the BGP version, the AS number and the Hold Time interval value in the OPEN message. After these values are communicated and agreed upon, the BGP session is established and the routers are ready to exchange routing information via BGP UPDATE messages.

To establish TCP connection to another BGP router, issue the following command:

```
[eugene@SM_BGP] > /routing bgp peer add remote-address=10.20.1.210 remote-as=65534
[eugene@SM_BGP] > /routing bgp peer print
Flags: X - disabled
 0   instance=default remote-address=10.20.1.210 remote-as=65534 tcp-md5-key=""
     multihop=no route-reflect=no hold-time=3m ttl=3 in-filter=""
     out-filter=""

[eugene@SM_BGP] >
```

Issue the following command to verify the connection is established:

```
[eugene@SM_BGP] > /routing bgp peer print status
Flags: X - disabled
 0   instance=default remote-address=10.20.1.210 remote-as=65534 tcp-md5-key=""
     multihop=no route-reflect=no hold-time=3m ttl=3 in-filter=""
     out-filter="" remote-id=10.20.1.210 uptime=1d1h43m16s
     prefix-count=180000 remote-hold-time=3m used-hold-time=3m
     used-keepalive-time=1m refresh-capability=yes state=established
[eugene@SM_BGP] >
```

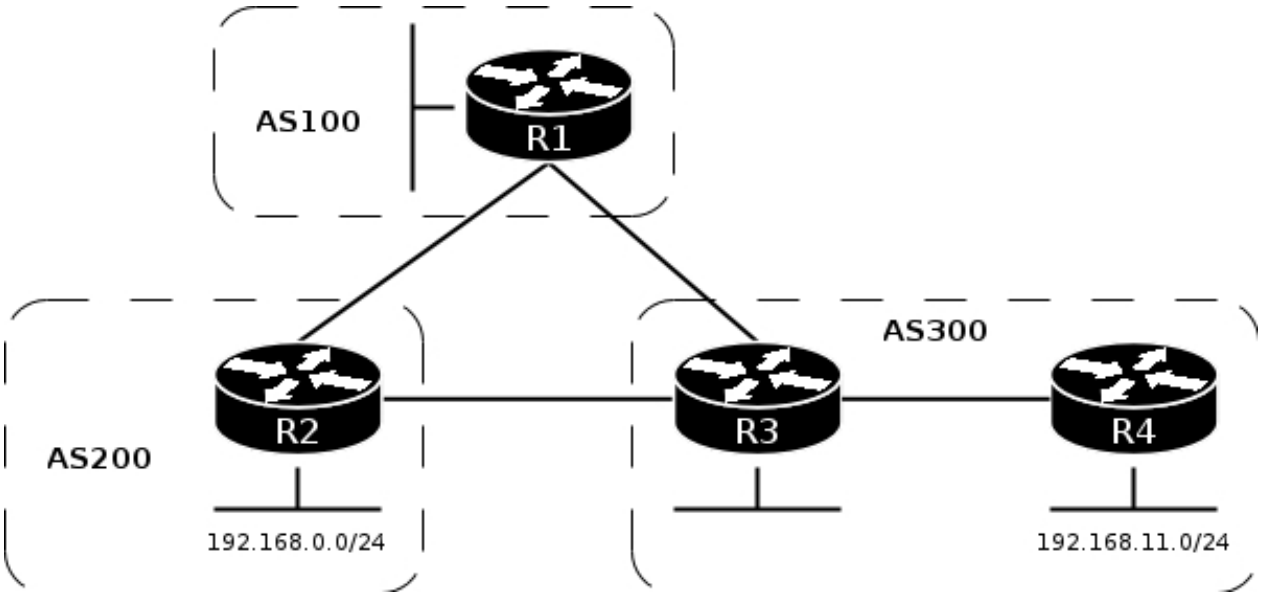
The BGP connection between two peers is up (**state=established**) with used value of Hold Time of 3 minutes. The **prefix-count** parameter indicates the total number of prefixes received from this particular peer. In case a peer later withdraws some prefixes from its routing announcements, the total number of prefixes is reduced by the appropriate value.

Route Redistribution

BGP process does not redistribute routes by default. You need to set one or more of the **redistribute-connected**, **redistribute-static**, **redistribute-rip**, **redistribute-ospf** and **redistribute-other-bgp** BGP instance parameters to **yes** to enable redistribution of the routes of the particular type. Thus issuing the /routing bgp instance set default redistribute-static=yes redistribute-connected=yes command enables redistribution of static and connected routes to all BGP peers that are configured to use default BGP instance. This might not be the desired behavior, since now you are announcing all of your internal routes into BGP. Moreover, some of the advertised prefixes might be too small and should be substituted with larger ones. You need to configure routing filters and route aggregation to avoid these problems.

Routing Filters

Unfiltered redistribution of routes might lead to undesired results. Consider the example below. R3 has a static route to the 192.168.0.0/24 network and since it has redistribute-static set to yes it announces the route to its BGP peer R1. This makes R1 believe that the AS300 is the source of the 192.168.0.0/24 network, which is misleading. To avoid this problem a routing filter that permits redistribution only of the 192.168.11.0/24 network must be applied on the R3.



- To enable the router R3 to advertise static networks to its peers:

```
/routing bgp instance set default redistribute-static=yes
```

- To filter out all prefixes except the 192.168.11.0/24 network:

```
/routing filter add chain=to_R1 prefix=192.168.11.0/24 invert-match=yes action=discard
/routing bgp peer set R1 out-filter=to_R1
```

Note the **invert-match** parameter. It makes the rule to match everything except the 192.168.11.0/24 prefix and discard it.

Routing filters are accessible through **/routing filter** menu. A routing filter consists of one or more filter rules identified by common **chain**. Rules are processed from top to bottom. Each rule consists of condition(s) to be satisfied in order for rule to match and action(s) to be performed on the matched prefixes. To enable routing filter, specify corresponding chain name as either **in-filter** or **out-filter** for BGP peer, or as **out-filter** for BGP instance.

Routing Filter Example

```
[eugene@SM_BGP] routing filter> print chain=Latnet-in
Flags: X - disabled
 0   chain=Latnet-in prefix=10.0.0.0/8 prefix-length=8-32 invert-match=no action=discard

 1   chain=Latnet-in prefix=192.168.0.0/16 invert-match=no action=discard

 2   chain=Latnet-in prefix=169.254.0.0/16 invert-match=no action=discard

 3   chain=Latnet-in prefix=4.23.113.0/24 invert-match=no action=passthrough
     set-bgp-communities=64550:14
```

```
 4   chain=Latnet-in prefix=4.36.116.0/23 invert-match=no action=passthrough set-routing-mark="LAN"
     set-route-comment="Remote offices"

 5   chain=Latnet-in prefix=8.8.0.0/16 prefix-length=16-32 bgp-communities=2588:800 invert-match=no
     action=discard
[eugene@SM_BGP] routing filter>
```

- rule #0 matches prefix 10.0.0.0/8 and more specific prefixes like 10.0.1.0/24, 10.1.23.0/28, etc. and discards them (these prefixes are silently dropped from inbound update messages and do not appear in memory)
- rule #3 sets BGP COMMUNITY attribute for prefix 4.23.113.0/24
- rule #4 has two actions. It simultaneously sets routing mark and comment for route to 4.36.116.0/23
- rule #5 discards prefix 8.8.0.0/16 and more specific ones, if they have COMMUNITY attribute of 2588:800

To use the filter above, add it as **in-filter** to the **Latnet** peer:

```
[eugene@SM_BGP] routing bgp peer> set Latnet in-filter=Latnet-in
[eugene@SM_BGP] routing filter> print
Flags: X - disabled
 0   name="C7200" instance=latnet remote-address=10.0.11.202 remote-as=64527 tcp-md5-key=""
     nexthop-choice=default multihop=no route-reflect=no hold-time=3m ttl=1 in-filter=""
     out-filter=to_C7200

 1   name="Latnet" instance=latnet remote-address=10.0.11.55 remote-as=2588 tcp-md5-key=""
     nexthop-choice=default multihop=yes route-reflect=no hold-time=3m ttl=5 in-filter="Latnet-in"
     out-filter=to_Latnet

 8   name="gated" instance=latnet remote-address=10.0.11.20 remote-as=64550 tcp-md5-key=""
     nexthop-choice=default multihop=no route-reflect=no hold-time=3m ttl=1 in-filter=""
     out-filter=""

[eugene@SM_BGP] routing bgp peer>
```

BGP Networks

The information in this article may be deprecated, and is described better elsewhere in the Wiki.

BGP allows to specify some arbitrary prefixes to be unconditionally advertised. These prefixes should be added to the **/routing bgp networks** list. The prefixes in this list are advertised as IGP routes. The redistribution of the BGP networks is affected by peer's routing filters. On the other hand, BGP networks are not installed in main routing table. As a consequence, they are not considered in best path selection algorithm, and do not affect aggregate processing.

Issue the following command to make the router advertise the 192.168.0.0/24 network to its peers:

```
[eugene@SM_BGP] > /routing bgp network add network=192.168.0.0/24
[eugene@SM_BGP] > /routing bgp network print
Flags: X - disabled
 #   NETWORK
 0   192.168.0.0/24
[eugene@SM_BGP] >
```

Note: consider aggregates as an alternative to BGP networks.

Static Routes

You could always use a static route to originate a subnet. With the routing-test package bringing many bgp-related enhancements into the **/ip route** menu, the static routes become a more powerful tool to originate prefixes. For example, you could add a static route to the 10.8.0.0/16 network and set BGP Local Preference attribute value for this route simultaneously:

```
/ip route add dst-address=10.8.0.0/16 gateway=10.0.11.1 bgp-local-pref=110
[admin@MikroTik] > /ip ro print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf
 #     DST-ADDRESS        PREF-SRC        G GATEWAY          DISTANCE INTERFACE
 0 A S 0.0.0.0/0                          r 10.0.11.1        1        ether1
 1 ADC 10.0.11.0/24       10.0.11.51                         0        ether1
 2 A S 10.8.0.0/16                        r 10.0.11.1        1        ether1
 3 ADC 10.12.0.0/24       10.12.0.2                          0        bonding1
[admin@MikroTik] >
```

BGP Advertisements

RouterOS provides a way to view what prefixes the router is redistributing to its peers. Issue **/routing bgp advertisements print <peer's address>** command to view prefixes sent to this peer.

```
[eugene@SM_BGP] routing bgp advertisements> print 10.0.11.20
 # DST-ADDRESS        NEXTHOP         AS-PATH                         ORIGIN     LOCAL-PREF MED
 0 3.0.0.0/8          159.148.254.250 2588,6747,1299,701,703,80       igp        100
 1 4.0.0.0/8          10.0.11.155     2588,6747{174,1273,1299,2914... igp        100
 2 6.0.0.0/8          10.0.11.155     2588,6747,1299,701,668          igp        100
 3 8.0.0.0/8          159.148.254.250 2588,6747,1299,3356             igp        100
 4 8.0.0.0/9          159.148.254.250 2588,6747,1299,3356             igp        100
 5 8.2.64.0/23        159.148.254.250 2588,6747,1299,3356,16803       igp        100
 6 8.2.144.0/22       159.148.254.250 2588,6747,1299,3356,36394       igp        100
 7 8.3.12.0/24        159.148.254.250 2588,6747,1299,3356,14711       igp        100
 8 8.3.13.0/24        159.148.254.250 2588,6747,1299,3356,26769       igp        100
 9 8.3.15.0/24        159.148.254.250 2588,6747,1299,3356,14711       igp        100
10 8.3.17.0/24        159.148.254.250 2588,6747,1299,25973            igp        100
11 8.3.19.0/24        159.148.254.250 2588,6747,1273,22822,26769      igp        100
12 8.3.37.0/24        159.148.254.250 2588,6747,1299,3356,3356,21640  igp        100
13 8.3.38.0/23        159.148.254.250 2588,6747,1299,3549,16420       igp        100
14 8.3.46.0/24        159.148.254.250 2588,6747,1299,3356,3356,21640  igp        100
15 8.3.208.0/24       159.148.254.250 2588,6747,1299,3549,36431       igp        100
16 8.3.209.0/24       159.148.254.250 2588,6747,1273,22822,26769      igp        100
17 8.3.210.0/24       159.148.254.250 2588,6747,1299,27524            igp        100
18 8.3.216.0/24       159.148.254.250 2588,6747,1299,3356,15170       igp        100
19 8.4.86.0/24        159.148.254.250 2588,6747,1299,3356,14627       igp        100
20 8.4.96.0/20        159.148.254.250 2588,6747,1299,3356,15162       igp        100
21 8.4.113.0/24       159.148.254.250 2588,6747,1299,3356,15162       igp        100
```

```
22 8.4.224.0/24       159.148.254.250 2588,6747,1299,3356,13546       igp        100
23 8.5.192.0/22       159.148.254.250 2588,6747,1299,209,13989        igp        100
24 8.6.48.0/21        159.148.254.250 2588,6747,1299,3356,36492       igp        100
25 8.6.89.0/24        159.148.254.250 2588,6747,1299,3356,11734       igp        100
26 8.6.90.0/24        159.148.254.250 2588,6747,1299,3356,16541       igp        100
27 8.6.220.0/22       159.148.254.250 2588,6747,1299,3356,13680       igp        100

[eugene@SM_BGP] routing bgp advertisements>
```

BGP Aggregates

This feature allows to redistribute one big prefix instead of many smaller ones.

```
[eugene@SM_BGP] routing bgp aggregate> print
Flags: X - disabled
 0   prefix=3.0.0.0/8 summary-only=yes inherit-attributes=yes attribute-filter="" suppress-filter=""
     advertise-filter=""

 1   prefix=6.0.0.0/8 summary-only=yes inherit-attributes=yes attribute-filter="" suppress-filter=""
     advertise-filter=""

 2   prefix=4.0.0.0/8 summary-only=yes inherit-attributes=yes attribute-filter="" suppress-filter=""
     advertise-filter=""
[eugene@SM_BGP] routing bgp aggregate>
```

The rules above suppress specific prefixes in ranges 3.0.0.0/8, 6.0.0.0/8 and 4.0.0.0/8 from being advertised:

```
[eugene@SM_BGP] routing bgp advertisements> print 10.0.11.20
 # DST-ADDRESS        NEXTHOP         AS-PATH                         ORIGIN     LOCAL-PREF MED
 0 3.0.0.0/8          159.148.254.250 2588,6747,1299,701,703,80       igp        100
 1 4.0.0.0/8          10.0.11.155     2588,6747{174,1273,1299,2914... igp        100
 2 6.0.0.0/8          10.0.11.155     2588,6747,1299,701,668          igp        100
 3 8.0.0.0/8          159.148.254.250 2588,6747,1299,3356             igp        100
 4 8.0.0.0/9          159.148.254.250 2588,6747,1299,3356             igp        100
 5 8.2.64.0/23        159.148.254.250 2588,6747,1299,3356,16803       igp        100
```

Manual:BGP HowTo & FAQ

Problem: **BGP session is not established**

BGP uses TCP, so to discover the cause of the problem, you can start with testing TCP connectivity. One way to do that is as simple as **/system telnet <remote-ip> 179** and check if the TCP connection can be established, and BGP port 179 is open and reachable.

If this is eBGP, make sure you have configured **multihop=yes** and TTL settings as needed. Use **/routing bgp peer print status** to see the current state of BGP connection.

Also note that if the remote peer is not supporting BGP Capabilities Advertisement (RFC 2842), some extra time will be needed for session establishment. The establishment will fail at the first time in this case, because of unknown options in BGP OPEN message. It should succeed at second attempt (i.e. after about a minute) and in any further attempts, because RouterOS will remember the offending options for that peer and not include them in BGP OPEN messages anymore.

Problem: **BGP session has been established, but routing updates are ignored**

NLRI (Network Layer Reachability Information) is ignored if path attributes are invalid. Turn on BGP debug logs to see the exact cause of the problem. (*/system logging add topics=bgp,!raw*).

One frequent case is unacceptable BGP next-hop. (Read here more about RouterOS and BGP next-hops.) In this case you must fix the next-hop on the sending side. In case the sender also is MT, you can use **nexthop-choice** peer setting to modify default next-hop selection preferences. If that fails, specify next-hop manually using **set-out-nexthop** routing filter.

Question: **How to check if a specific route exists in IP routing table?**

Finding a route by prefix is pretty fast:

```
/ip route print where dst-address = 193.23.33.0/24
```

To find all routes with prefixes falling in a range:

```
/ip route print where dst-address in 193.23.0.0/16
```

You can also search routes by other attributes, but it will be much slower and can take some time on a router having full BGP feed.

For example, since RouterOS 3.23 you can use this syntax to match routes having originated from a specific AS 30621:

```
[atis@SM_BGP] > /ip route print detail where bgp-as-path ~ "30621\$"
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
0 ADb  dst-address=12.151.74.0/23
       gateway=x.x.x.x recursive via y.y.y.y ether1 distance=20
       scope=40 target-scope=10 bgp-as-path="2588,42979,702,701,7018,30621"
       bgp-origin=igp received-from=x.x.x.x

1 ADb  dst-address=12.151.76.0/22
       gateway=x.x.x.x recursive via y.y.y.y ether1 distance=20
       scope=40 target-scope=10 bgp-as-path="2588,42979,702,701,7018,30621"
```

```
bgp-atomic-aggregate=yes bgp-origin=igp received-from=x.x.x.x
```

Problem: **Routes are exchanged and installed in IP route table, but they stay inactive**

Routes must be resolved to become active; it's possible that you need to change scope or target-scope attributes for some routes.

Question: **How to filter out something?**

Use routing filters. For example, to filter out routes with a specific BGP community, add this rule:

```
/routing filter add bgp-communities=111:222 chain=bgp-in action=discard
```

Then tell BGP peer to use that filter chain:

```
/routing bgp peer set peer in-filter=bgp-in
```

There is also an **out-filter** BGP peer parameter for filtering outgoing BGP updates.

In recent RouterOS versions **bgp-as-path** filter accepts regular expressions. Community filtering by regular expressions is not yet possible.

Question: **How to quickly check how many routes there are in route table?**

For all routes use:

```
ip route print count-only
```

To see route count from a particular peer look at **prefix-count** property in:

```
route bgp peer print status
```

Question: **How to seen routes advertised to, and routes received from a particular peer?**

To see routes advertised to a particular peer (similar to Cisco command *show ip bgp neighbor x.x.x.x advertised-routes*) use:

```
routing bgp advertisements print
```

Or

```
routing bgp advertisements print <peer_name>
```

Note: At the moment AS-PATH attribute is displayed without prepends!

To see routes received from a particular peer (similar to Cisco command *show ip bgp neighbor x.x.x.x received-routes*) use:

```
ip route print where received-from=<peer_name>
```

Note: Routes that were discarded (with action *discard*) in incoming filters, or ignored because of invalid attributes (e.g. not directly reachable next-hop for EBGP) will not be displayed!

Question: **Is load balancing possible with MT BGP?**

Yes. Even though BGP itself cannot propagate multiple next-hops for a single route through the network, there are ways how to have routes with multiple next-hops on a router.

One way is to set multiple next-hops with routing filter.

```
routing filter add chain=bgp-in set-in-nexthop=10.0.1.1,10.0.2.1
```

Another way is to resolve BGP next-hop (if it is not directly reachable) through a static or OSPF route with multiple next-hops.

```
ip route add dst-address=x.x.x.x/y gateway=10.0.1.1,10.0.2.1
```

See also: BGP Load Balancing with two interfaces.

Question: **How to announce routes?**

If your don't have many routes to announce and want the best control over them, use BGP networks or aggregates. Note that both maximal BGP network and aggregate count is limited to 200.

Otherwise use route redistribution options, configurable under BGP instance settings.

Question: **What does BGP network *synchronize* option exactly mean?**

Since version 3.30 routing-test it means "do not announce this network, unless there is a matching active IGP or connected route in IP route table". "Matching" in this case means: with exactly the same prefix.

Question: **How to control advertised routing information?**

Use routing filters.

To advertise the same information (e.g. some BGP attribute value) to all peers, use BGP instance out-filter:

```
/routing filter add set-bgp-communities=111:222 chain=bgp-out
/routing bgp instance set default out-filter=bgp-out
```

To send routing information to different peers, use peer specific filters. For example, if you want to advertise a lower preference value (higher path cost) to one of the peers, you can prepend your AS number multiple times to the BGP AS_PATH attribute:

```
/routing filter add set-bgp-prepend=4 chain=bgp-out-peer1
/routing bgp peer set peer1 out-filter=bgp-out-peer1
```

Use */routing bgp advertisements print* to see what routing information exactly is advertised to peers.

Problem: **Looks like my routing filter isn't working**

Most likely prefix matcher is configured incorrectly. For example, say that you want to configure filter that will discard all routes falling under prefix 1.1.1.0/24.

The ***correct way to do this*** is with specifying *prefix-length* matcher:

```
add prefix=1.1.1.0/24 prefix-length=24-32 action=discard chain=bgp-in
```

This rule is incorrect (default netmask is /32, so it will match only prefix 1.1.1.0/32):

```
add prefix=1.1.1.0 prefix-length=24-32 action=discard chain=bgp-in
```

This is incorrect too (because it will match only route with netmask 255.255.255.0)

```
add prefix=1.1.1.0/24 action=discard chain=bgp-in
```

Use filter action *log* to see which routes are matched by a routing filter.

Question: **How to announce just a single large IP prefix instead of many smaller (i.e. more specific) prefixes?**

Use BGP aggregates if you need to aggregate multiple routes in a single one. An aggregate will be announced one if there are some active routes with more specific netmasks falling under it. When an aggregate becomes active, a corresponding blackhole route is a automatically created.

By default, BGP aggregates take in account only BGP routes. To also include IGP and connected routes in consideration, use **include-igp** configuration option.

Question: **How to aggregate IGP routes?**

Since 3.30 you can specify **include-igp** in BGP aggregate configuration. Example:

```
ip route add dst-address=10.9.9.0/25 gateway=10.0.0.1
ip route add dst-address=10.9.9.128/25 gateway=10.0.0.2
routing bgp aggregate add instance=default prefix=10.9.9.0/24 include-igp=yes
```

Results:

```
[admin@MikroTik] > routing bgp advertisements print
PEER     PREFIX              NEXTHOP          AS-PATH                                 ORIGIN     LOCAL-PREF
peer1    10.9.9.0/24         10.0.0.131                                               incomplete
```

Use routing filters to control which routes are aggregated. For example, if you don't want to aggregate connected routes:

```
routing filter add chain=aggregate-out protocol=connect action=discard
routing bgp aggregate set [find] advertise-filter=aggregate-out
```

Question: **How to advertise the default route?**

To send default route to a particular peer, set **default-originate=always** or **if-installed** for that peer.

Problem: **Routes are announced, but with attributes not from IP routing table**

There exists a limitation in MT BGP operation: if a BGP network with synchronization turned off, or default route generated by **default-originate=always** configuration statement is announced, the attributes of that route will not be taken from routing table.

If **synchronize=yes** or **default-originate=if-installed** is used, the attributes of the announced route will be taken from routing table.

Question: **Can MT propagate BGP route updates without installing them in IP route table (i.e. serve as a pure route reflector)?**

No, it's not possible.

Question: **Does MT BGP support 4-octet AS numbers?**

Yes. For input, both ASPLAIN (i.e. xxxxxx) and ASDOT (i.e. xxx.xxx) formats are supported; for output, ASPLAIN only.

Question: **What are the specifics of MT BGP route selection algorithm?**

The algorithm is described here. The algorithm follows BGP RFC closely, with a few differences:

- Cisco-style weight is used as the first and most important selection criteria;
- AS path length comparison can be turned off by a configuration parameter;
- locally originated BGP routes are preferred in case of same AS path length, weight, and local-preference values;

- interior cost calculation and comparison step is skipped.

The algorithm is used only to compare BGP routes from the same BGP instance. For different instances, only "distance" attributes are compared.

Question: **How much memory is required to keep the global BGP route table?**

Our recommendations are at least 256 MB RAM for a single copy of the table and at least 512 MB RAM for two or three copies.

Assuming the Internet route table size ~300 000 routes, for the first copy of the table, with routes resolved and active, about 155 MB extra memory is needed. This is only for the first copy specifically, the amount of RAM needed for each additional copy of the table is significantly less than that number.

RAM usage on RB1000 (BGP feed size 301 480 routes, no redistribution):

- No BGP routes: 26 MB
- Single copy: 181 MB
- Two copies: 241 MB
- Three copies: 299 MB

Memory requirements will increase if incoming routing filters that change route attributes are used. That happens because unchanged copy of the route attributes received also will be stored in RAM, to be used in case of later routing filter change.

The requirements will also increase depending on count of peers to which routes are advertised.

It is not recommended to turn on SNMP on routers with full BGP feed!

Question: **How to hide my own AS?**

To hide your own AS you need to set up routing filter in output chain and set *set-bgp-prepend*. If value is set to 0 then peer's own AS is removed from AS_PATH.

Manual:BGP Load Balancing with two interfaces

Applies to RouterOS: 3, v4

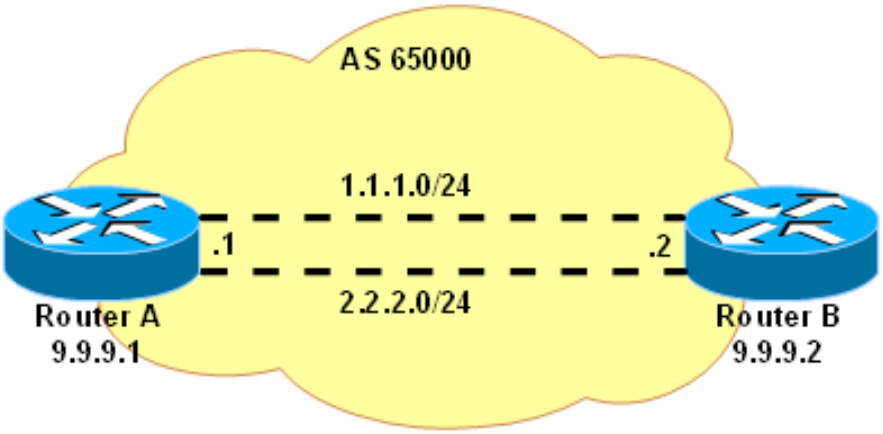
NB: RouterOS version 3.13 or later with routing-test package is required for this to work

In these examples we show how to do load balancing when there are multiple equal cost links between two BGP routers. The "multiple recursive next-hop resolution" feature is used to achieve that.

The BGP session is established between loopback interfaces; **update-source** configuration setting is used to bind the BGP connection to the right interface.

Example with iBGP

Network Diagram



Configuration

On Router A:

```
# loopback interface
/interface bridge add name=lobridge

# addresses
/ip address add address=1.1.1.1/24 interface=ether1
/ip address add address=2.2.2.1/24 interface=ether2
/ip address add address=9.9.9.1/32 interface=lobridge

# ECMP route to peer's loopback
/ip route add dst-address=9.9.9.2/32 gateway=1.1.1.2,2.2.2.2

# BGP
/routing bgp instance set default as=65000
/routing bgp add name=peer1 remote-address=9.9.9.2 remote-as=65000 update-source=lobridge
```

On Router B:

```
# loopback interface
/interface bridge add name=lobridge
```

```
# addresses
/ip address add address=1.1.1.2/24 interface=ether1
/ip address add address=2.2.2.2/24 interface=ether2
/ip address add address=9.9.9.2/32 interface=lobridge

# ECMP route to peer's loopback
/ip route add dst-address=9.9.9.1/32 gateway=1.1.1.1,2.2.2.1

# BGP
/routing bgp instance set default as=65000
/routing bgp add name=peer1 remote-address=9.9.9.1 remote-as=65000 update-source=lobridge

# a route to advertise
/routing bgp network add network=4.4.4.0/24
```

Results

Check that BGP connection is established:

```
[admin@B] > /routing bgp peer print status
Flags: X - disabled
0   name="peer1" instance=default remote-address=9.9.9.1 remote-as=65000
    tcp-md5-key="" nexthop-choice=default multihop=no route-reflect=no hold-time=3m
    ttl=255 in-filter="" out-filter="" address-families=ip
    update-source=lobridge default-originate=no remote-id=1.1.1.1
    local-address=9.9.9.2 uptime=28s prefix-count=0 updates-sent=1
    updates-received=0 withdrawn-sent=0 withdrawn-received=0 remote-hold-time=3m
    used-hold-time=3m used-keepalive-time=1m refresh-capability=yes
    as4-capability=yes state=established
```

Route table on Router A:

```
[admin@A] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
#      DST-ADDRESS        PREF-SRC        G GATEWAY               DISTANCE INTER...
0 ADC  1.1.1.0/24         1.1.1.1                                 0        ether1
1 ADC  2.2.2.0/24         2.2.2.1                                 0        ether2
2 ADb  4.4.4.0/24                         r 9.9.9.2               200      ether1
                                                                           ether2
3 ADC  9.9.9.1/32         9.9.9.1                                 0        lobridge
4 A S  9.9.9.2/32                         r 1.1.1.2               1        ether1
                                          r 2.2.2.2                        ether2
```

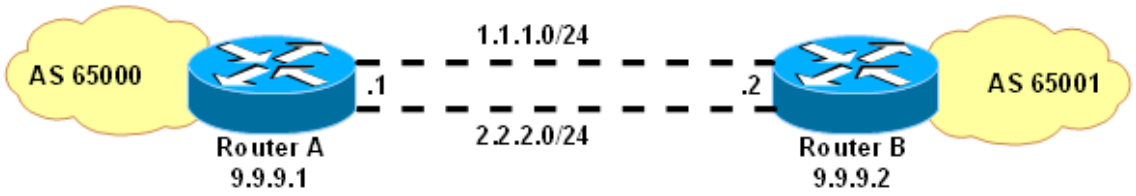
```
[admin@A] > /ip route print detail
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
```

```
0 ADC  dst-address=1.1.1.0/24 pref-src=1.1.1.1 interface=ether1 distance=0 scope=10

1 ADC  dst-address=2.2.2.0/24 pref-src=2.2.2.1 interface=ether2 distance=0 scope=10

2 ADb  dst-address=4.4.4.0/24 gateway=9.9.9.2 interface=ether1,ether2
       gateway-state=recursive distance=200 scope=40 target-scope=30
       bgp-local-pref=100 bgp-origin=igp received-from=9.9.9.2

3 ADC  dst-address=9.9.9.1/32 pref-src=9.9.9.1 interface=lobridge distance=0 scope=10

4 A S  dst-address=9.9.9.2/32 gateway=1.1.1.2,2.2.2.2 interface=ether1,ether2
       gateway-state=reachable,reachable distance=1 scope=30 target-scope=10
```

The route 4.4.4.0./24 is installed in Linux kernel now with two nexthops: 1.1.1.2 (on ether1) and 2.2.2.2 (on ether2).

Example with eBGP

Network Diagram



Configuration

Here the example given above is further developed for eBGP case. By default, eBGP peers are required to be directly reachable. If we are using loopback interfaces, they technically are not, so **multihop=yes** configuration setting must be specified.

On Router A:

```
/routing bgp instance set default as=65000
/routing bgp set peer1 remote-address=9.9.9.2 remote-as=65001 update-source=lobridge multihop=yes
```

On Router B:

```
/routing bgp instance set default as=65001
/routing bgp set peer1 remote-address=9.9.9.1 remote-as=65000 update-source=lobridge multihop=yes
```

Results

If we now print the route table on Router A, we see that the route from Router B is there, but it's not active:

```
...
2  Db  dst-address=4.4.4.0/24 gateway=9.9.9.2 interface="" gateway-state=unreachable
       distance=20 scope=40 target-scope=10 bgp-as-path="65001" bgp-origin=igp
       received-from=9.9.9.2
...
```

This is because eBGP routes are installed with lesser **target-scope** by default. To solve this, setup routing filter that sets larger target-scope:

```
/routing filter add chain=bgp-in set-target-scope=30
/routing bgp set peer1 in-filter=bgp-in
```

Or else, modify **scope** attribute of the static route:

```
/ip route set [find dst-address=9.9.9.2/32] scope=10
```

Either way, the route to 4.4.4.0/24 should be active now:

```
2 ADb  dst-address=4.4.4.0/24 gateway=9.9.9.2 interface=ether1,ether2
       gateway-state=recursive distance=20 scope=40 target-scope=10
       bgp-as-path="65001" bgp-origin=igp received-from=9.9.9.2
```

Notes

- BGP itself as protocol does not supports ECMP routes. When a recursively resolved BGP route is propagated further in the network, only one nexthop can be selected (as described here) and included in the BGP UPDATE message.
- Corresponding Cisco syntax can be found here: Load Sharing with BGP in Single and Multihomed Environments: Sample Configurations [1]

References

[1] http://www.cisco.com/en/US/tech/tk365/technologies_configuration_example09186a00800945bf.shtml

Manual:BGP nexthop selection and validation in RouterOS 3.x

The problem

Even though the BGP RFC (RFC 4271, 5.1.3) devotes several pages to the selection of the BGP nexthop that will be included in an UPDATE message, the specification is still vague at some places. Besides that, other router vendors tend to give better control over nexthop selection than the RFC describes. A particular example is XORP routing daemon. It has no nexthop selection logic on it's own at all, and requires configuration of set-nexthop routing map for each peer. On the other hand, RouterOS is trying to conform to the RFC. Quite complicated selection logic is used here by default; but if you wish, you can override this logic by using routing filters.

Introduction of IPv6 brings additional nexthop selection related problems, as the ubiquitous link-local addresses (fe80::/10) has no equivalent in IPv4 world.

Here we talk about the particular nexthop selection algorithm used in RouterOS 3.x. Most of the IPv4 related part also applies to 2.9 routing-test.

IPv4 BGP route output

- If a nexthop is configured with **set-out-nexthop** filter, always use this configured value (even if it's not valid!)
- If we are reflecting a BGP route to an iBGP router (**route-reflect=yes**), use the nexthop received in UPDATE message.
- If **nexthop-choice** is configured as **force-self**, go to the last step.
- If we are redistributing a BGP route, the nexthop we received in UPDATE message is considered.
  - If the peer is eBGP and not configured multihop -- go to the next step.
  - If the nexthop is the same as the remote peer's id or remote peer's address used to establish the connection, go to the next step.
  - Else use the received BGP nexthop.
- The nexthop from route table (FIB in BGP terms) is considered. If route has multiple nexthops, or is recursively resolved through multiple nexthops, only first of them is considered.
  - If the peer is iBGP and we are redistributing not locally originated route, go to the next step.
  - If the peer is eBGP and is multiple IP hops away go to the next step.
  - If the nexthop is the same as the remote peer's id or remote peer's address used to establish the connection, go to the next step.
  - Else use nexthop from route table (FIB).
- As the last fallback, use the address used to establish the connection. (In case of IPv6 connection between the peers, use a random IPv4 address of the connection's interface. Same applies to IPv6 nexthop with IPv4 connection.)

IPv4 BGP route input

- If the nexthop received in an UPDATE message is not a valid IPv4 unicast address, ignore this UPDATE message.
- If the nexthop is router's local address, ignore this UPDATE message.
- If the peer is eBGP (note that peer having different AS is considered eBGP, even if it's in the same confederation) and it's not configured as multihop, then the RFC requires to check that nexthops falls in a network shared with remote peer. In practice we use the network that is used to make connection with peer. For example, if connection is made with address 10.0.0.1/24 to address 10.0.0.2, the nexthop must fall in range 10.0.0.0 - 10.0.0.255.
  - (In case of IPv6 connection, all IPv4 networks belonging to the interface are tested. Same applies to IPv6 nexthop with IPv4 connection.)
- After these checks are passed, the user can modify the received nexthop with **set-in-nexthop** filter, without limitations. **set-in-nexthop-direct** filter also can be used; or they can combined. Both filters accepts multiple nexthop values.
- After the route are installed in RouterOS routing table with the selected nexthop, one last step remains. For this route to become active, the nexthop must be resolved.This can happen in two ways:
  1. When the nexthop falls in some connected route's range (i.e. gateway status is "reachable").
  2. When the nexthop falls in some other route's range with low enough scope attribute (i.e. gateway status is "recursive").

IPv6 BGP route output

For IPv6, everything is complicated with the introduction of link-local address nexthops (RFC 2545). In short, the are cases when two nexthops should be included in UPDATE message. The first nexthop always is present and is referred here as "global nexthop" (although it can be a link-local address). The second ("link-local nexthop"), when present, must be a link-local address. Note that link-local address always must be associated with a "link" (i.e. interface), otherwise it cannot be used for forwarding traffic. In BGP case, the interface index is deduced from the connection.

- If a nexthop is configured with **set-out-nexthop** filter, always use this configured value (even if it's not valid!)
- If we are reflecting a BGP route to an iBGP router (**route-reflect=yes**), use the nexthop from UPDATE message. Do not set link-local nexthop in this case.
- Select global nexthop in the same way we would select IPv4 nexthop.
- If the following holds:
  - peer is reachable directly (i.e. single IP hop away);
  - global nexthop falls in a network shared with peer;
  - global nexthop is not a link local address;

then also include link-local nexthop in the UPDATE message. Else terminate.

- Select the link-local nexthop.
  - First check the nexthop configured with **set-out-nexthop-linklocal** filter, if any. Use it if it's a link-local address.
  - Then try to use FIB nexthop as link-local nexthop. Use it if it's a link-local address.
  - Finally, take as nexthop the link-local address belonging to the interface used to establish the connection with remote peer.

IPv6 BGP route input

- Validate global nexthop exactly the same way as IPv4 nexthop would be validated. Multicast, reserved and loopback addresses are not acceptable as nexthops.
- If the link-local nexthop received is not a valid IPv6 link-local address, then ignore it.
- If the link-local nexthop is a router's local address, then ignore it.
- If the link-local nexthop is present in UPDATE message and should not be ignored, then use it for installing in route table (FIB). Else use global nexthop.
- The user can modify the received nexthop with **set-in-nexthop-ipv6** and **set-in-nexthop-linklocal** filters, without limitations. **set-in-nexthop-direct** filter also can be used; or they can be combined. All filters accepts multiple nexthop values.
- In routing table, non-link-local nexthops are resolved the same way as IPv4 nexthops. Link-local nexthops always are considered reachable, if nexthop's interface has IPv6 support. (Interface has IPv6 support if it has any IPv6 address.)

Other address families

For **l2vpn**, **l2vpn-cisco** and **vpnv4** address families nexthop is selected and validated in exactly the same way as for IPv4.

Currently there is no support for IPv6 nexthops for **l2vpn[-cisco]** address families.

References

- RFC 4271 [1] - A Border Gateway Protocol 4 (BGP-4) - section 5.1.3.
- RFC 2545 [2] - Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing

References

[1] http://www.ietf.org/rfc/rfc4271.txt#page-26
[2] http://www.ietf.org/rfc/rfc2545.txt

Manual:BGP soft reconfiguration alternatives in RouterOS

Applies to RouterOS: v3, v4

What is soft reconfiguration?

When a route is received from a dynamic routing protocol, it is passed through routing filters. These filters may change some attributes of the route or discard it altogether.

When the routing filters change, they must be reapplied to routes from BGP (and other protocols, but we are focusing on BGP here). One way to do is reset BGP session, that is, tear down the connection with peer and re-establish it again. The disadvantage of this approach are obvious.

Soft reconfiguration means that filtering policy can be reapplied after a change without session reset. For RouterOS, both dynamic and static variants are possible.

Static soft-reconfiguration

What could be the effect of routing filters to a route? There are two possible cases.

CASE 1: Filters only change some attributes of the route. The orginal received attributes always are stored with the route. They are use to calculate new routing table attributes if filters changes. This process is trigerred automatically.

CASE 2: The route is discarded by filters. If the route is discarded, original attributes are not saved and information about it is lost. To avoid that, use **action=reject** in filters instead of **action=discard**. Now the route is saved, but is not eligible to become active (that is, it will not be installed in kernel routing table or redistributed to protocols).

- + Router does not lose routing information, because session is not reset.
- - Memory overhead for storing rejected routes.

Example:

Original configuration (routes are rejected):

```
[admin@A] > routing filter add chain=bgp-in action=reject prefix=4.0.0.0/8 prefix-length=8-32
[admin@A] > routing bgp peer set peer1 in-filter=bgp-in
[admin@A] > ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE INTERFACE
 0 A S  0.0.0.0/0                           10.0.0.1        1        ether1
 1 ADb  3.0.0.0/8                           192.65.184.3    200      ether1
 2  Db  4.0.0.0/8                           192.65.184.3    20       ether1
 3  Db  4.21.104.0/24                       192.65.184.3    20       ether1
 4  Db  4.21.112.0/23                       192.65.184.3    20       ether1
 5  Db  4.21.130.0/23                       192.65.184.3    20       ether1
```

Change filters to less restrictive:

```
[admin@A] > routing filter disable 0
[admin@A] > ip route pr
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE INTERFACE
 0 A S  0.0.0.0/0                           10.0.0.1        1        ether1
 1 ADb  3.0.0.0/8                           192.65.184.3    200      ether1
 2 ADb  4.0.0.0/8                           192.65.184.3    200      ether1
 3 ADb  4.21.104.0/24                       192.65.184.3    200      ether1
 4 ADb  4.21.112.0/23                       192.65.184.3    200      ether1
 5 ADb  4.21.130.0/23                       192.65.184.3    200      ether1
```

Dynamic soft-reconfiguration

In this case, your BGP routing peer must support route refresh capability. Enter **/routing bgp peer print status** in CLI to check this.

- + No additional memory is used
- - Peer must support this capability.
- - It's not done automatically. You must issue **/routing bgp peer refresh** command after changes in filters are finished.

Example:

Original configuration (routes are discarded):

```
[admin@A] > routing filter add chain=bgp-in action=reject prefix=4.0.0.0/8 prefix-length=8-32
[admin@A] > ip route pr
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE INTERFACE
 0 A S  0.0.0.0/0                           10.0.0.1        1        ether1
 1 ADb  3.0.0.0/8                           192.65.184.3    200      ether1
```

Change filters to less restrictive and send refresh request:

```
[admin@A] > routing filter disable 0
[admin@A] > routing bgp peer refresh peer1
[admin@A] > ip route pr
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE INTERFACE
 0 A S  0.0.0.0/0                           10.0.0.1          1        ether1
 1 ADb  3.0.0.0/8                           192.65.184.3      200      ether1
 2 ADb  4.0.0.0/8                           192.65.184.3      200      ether1
 3 ADb  4.21.104.0/24                       192.65.184.3      200      ether1
 4 ADb  4.21.112.0/23                       192.65.184.3      200      ether1
```

Summary

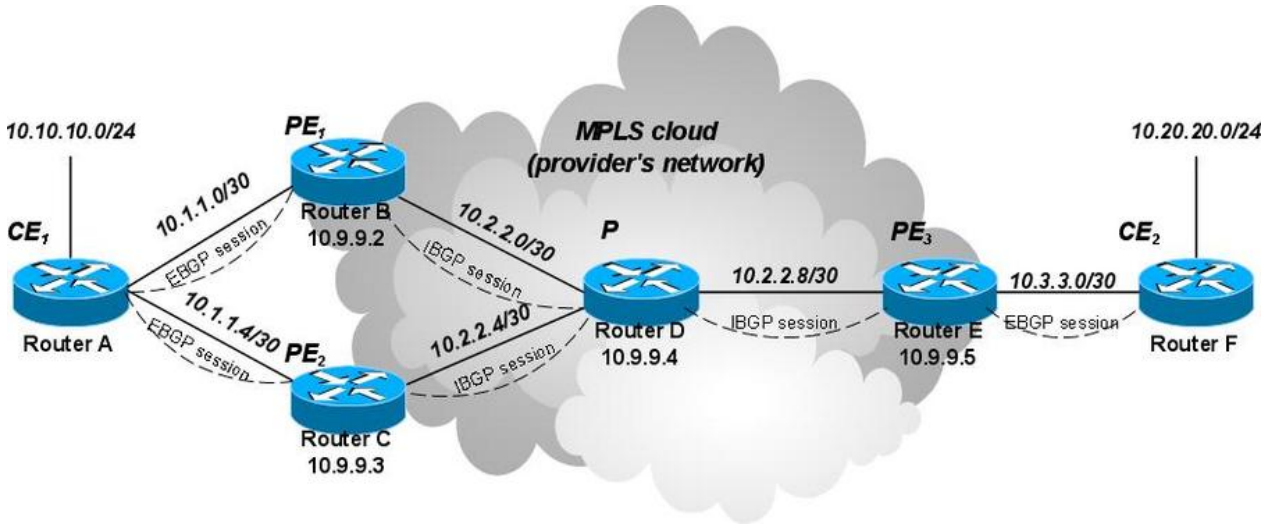
- Do nothing unless the filter change changes discard status for some prefixes.
- Use **routing bgp peer refresh** comand after filter change if peer supports this capability.
- Use **action=reject** in filters in other cases.

Manual:EBGP as PE-CE routing protocol

Applies to RouterOS: v4

- *Packages required:* **routing, mpls**
- *Software versions:* **4.3+**

Setup



In this setup we describe the use of EBGP as Provider Edge - Customer Edge (PE-CE) routing protocol.

Router A and Router F both belong to the same customer's VPN, but to different sites.

Router A is multihomed - is has connections to two PEs, router B and router C.

Routers B, C, and E are PE routers.

Router D is provider (P) router and functions as BGP route reflector.

All provider's routers belong to AS 100; all customer routers belong to private AS 65000.

Description

There are several tricky aspects about this setup.

First, it is not possible to use BGP built-in mechanism of routing loop prevention, that checks BGP AS path for presence of local AS path numbers and discards all routes that match. We want to distribute routes from A to F, and vice versa, but they belong to the same BGP AS. (One solution is to use different private AS numbers there, but that's not always possible or desirable.)

- One way to do work around this BGP AS path loop check is to configure BGP **as-override** option at exit point from provider's network.
- Another way is to configure **remove-private-as** at providers network entry point (it will work only if customer's AS numbers are private, of course!)
- Yet another way is to configure **allow-as-in=x** on customers edge router. "x" is the number of times local as number can be present in AS path.

In this configuration we use the **as-override** option on router E (to make router F accept routes from A), and **allow-as-in** option on router A, to make it accept routes from F.

Router A:

```
routing bgp peer add remote-address=10.1.1.2 remote-as=100 allow-as-in=1;
routing bgp peer add remote-address=10.1.1.6 remote-as=100 allow-as-in=1;
```

Router E:

```
routing bgp peer add instance=ebgp remote-address=10.3.3.2 remote-as=65000 as-override=yes;
```

The second tricky aspect is that since CE1 is multihomed (i.e. has links to multiple PEs) and BGP AS path loop prevention mechanism is disabled on router A because 'allow-as-in' option configured, the routes that A advertises to one PE router may be received back from the second PE. Installing those route in VRF table can also lead to suboptimal routing and even to BGP convergence failure. To avoid that, BGP Site of Origin (SOO) extended communities can be used. In this configuration we configure routing filter on PE routers that sets BGP SOO extended communities to routes received from CE router, and another filter, that filters out VPNv4 routes received from IBGP by the same SOO extended community attribute.

Routers B, C:

```
routing filter add chain=ibgp-in site-of-origin=1:100 action=discard;
routing filter add chain=ebgp-in set-site-of-origin=1:100;
```

We also use different BGP instances on PE routers: one for PE-CE (i.e. EBGP) peers and one for provider's network internal BGP peers.

Configuration

Router A:

```
ip address add address=10.1.1.1/30 interface=A_B;
ip address add address=10.1.1.5/30 interface=A_C;
interface bridge add name=somenet;
ip address add address=10.10.10.1/24 interface=somenet;
routing bgp instance set default as=65000 redistribute-connected=yes;
```

```
 routing bgp peer add remote-address=10.1.1.2 remote-as=100 allow-as-in=1;
 routing bgp peer add remote-address=10.1.1.6 remote-as=100 allow-as-in=1;
```

Router B:

```
ip address add address=10.1.1.2/30 interface=B_A;
ip address add address=10.2.2.1/30 interface=B_D;
interface bridge add name=lobridge;
ip address add address=10.9.9.2/32 interface=lobridge;
ip route add dst-address=10.9.9.3 gateway=10.2.2.2;
ip route add dst-address=10.9.9.4 gateway=10.2.2.2;
ip route add dst-address=10.9.9.5 gateway=10.2.2.2;
ip route vrf add routing-mark=vrf1 interfaces=B_A route-distinguisher=1:1 \
 import-route-targets=1:1 export-route-targets=1:1;
mpls ldp set enabled=yes transport-address=10.9.9.2;
mpls ldp interface add interface=B_D hello-interval=3;
routing bgp instance set default as=100;
routing bgp instance add name=ebgp router-id=0.0.0.2 as=100 routing-table=vrf1;
routing bgp instance vrf add instance=default routing-mark=vrf1 redistribute-connected=yes \
 redistribute-other-bgp=yes;
routing bgp peer add address-families=vpnv4 remote-address=10.9.9.4 remote-as=100 \
 in-filter=ibgp-in out-filter=ibgp-out update-source=10.9.9.2;
routing bgp peer add instance=ebgp remote-address=10.1.1.1 remote-as=65000 \
  in-filter=ebgp-in out-filter=ebgp-out;
routing filter add chain=ebgp-out site-of-origin=1:100 action=discard;
routing filter add chain=ebgp-in set-site-of-origin=1:100;
```

Router C:

```
ip address add address=10.1.1.6/30 interface=C_A;
ip address add address=10.2.2.5/30 interface=C_D;
interface bridge add name=lobridge;
ip address add address=10.9.9.3/32 interface=lobridge;
ip route add dst-address=10.9.9.2 gateway=10.2.2.6;
ip route add dst-address=10.9.9.4 gateway=10.2.2.6;
ip route add dst-address=10.9.9.5 gateway=10.2.2.6;
ip route vrf add routing-mark=vrf1 interfaces=C_A route-distinguisher=1:1 \
 import-route-targets=1:1 export-route-targets=1:1;
mpls ldp set enabled=yes transport-address=10.9.9.3;
mpls ldp interface add interface=C_D hello-interval=3;
routing bgp instance set default as=100;
routing bgp instance add name=ebgp router-id=0.0.0.3 as=100 routing-table=vrf1;
routing bgp instance vrf add instance=default routing-mark=vrf1 \
 redistribute-connected=yes redistribute-other-bgp=yes;
routing bgp peer add address-families=vpnv4 remote-address=10.9.9.4 remote-as=100 \
 in-filter=ibgp-in update-source=10.9.9.3;
routing bgp peer add instance=ebgp remote-address=10.1.1.5 remote-as=65000 \
 in-filter=ebgp-in out-filter=ebgp-out;
routing filter add chain=ibgp-in site-of-origin=1:100 action=discard;
```

```
routing filter add chain=ebgp-in set-site-of-origin=1:100;
```

Router D:

```
ip address add address=10.2.2.2/30 interface=D_B;
ip address add address=10.2.2.6/30 interface=D_C;
ip address add address=10.2.2.9/30 interface=D_E;
interface bridge add name=lobridge;
ip address add address=10.9.9.4/32 interface=lobridge;
ip route add dst-address=10.9.9.2 gateway=10.2.2.1;
ip route add dst-address=10.9.9.3 gateway=10.2.2.5;
ip route add dst-address=10.9.9.5 gateway=10.2.2.10;
mpls ldp set enabled=yes transport-address=10.9.9.4;
mpls ldp interface add interface=D_B hello-interval=3;
mpls ldp interface add interface=D_C hello-interval=3;
mpls ldp interface add interface=D_E hello-interval=3;
routing bgp instance set default as=100;
routing bgp peer add address-families=vpnv4 remote-address=10.9.9.2 remote-as=100 \
 update-source=10.9.9.4 route-reflect=yes;
routing bgp peer add address-families=vpnv4 remote-address=10.9.9.3 remote-as=100 \
 update-source=10.9.9.4 route-reflect=yes;
routing bgp peer add address-families=vpnv4 remote-address=10.9.9.5 remote-as=100 \
 update-source=10.9.9.4 route-reflect=yes;
```

Router E:

```
 ip address add address=10.3.3.1/30 interface=E_F;
 ip address add address=10.2.2.10/30 interface=E_D;
 interface bridge add name=lobridge;
 ip address add address=10.9.9.5/32 interface=lobridge;
 ip route add dst-address=10.9.9.2 gateway=10.2.2.9;
 ip route add dst-address=10.9.9.3 gateway=10.2.2.9;
 ip route add dst-address=10.9.9.4 gateway=10.2.2.9;
 ip route vrf add routing-mark=vrf1 interfaces=E_F route-distinguisher=1:1 \
  import-route-targets=1:1 export-route-targets=1:1;
 mpls ldp set enabled=yes transport-address=10.9.9.5;
 mpls ldp interface add interface=E_D hello-interval=3;
 routing bgp instance set default as=100;
 routing bgp instance add name=ebgp router-id=0.0.0.5 as=100 routing-table=vrf1;
 routing bgp instance vrf add instance=default routing-mark=vrf1 redistribute-connected=yes \
  redistribute-other-bgp=yes;
 routing bgp peer add address-families=vpnv4 remote-address=10.9.9.4 remote-as=100 \
  update-source=10.9.9.5;
 routing bgp peer add instance=ebgp remote-address=10.3.3.2 remote-as=65000 as-override=yes;
```

Router F:

```
ip address add address=10.3.3.2/30 interface=F_E;
interface bridge add name=somenet;
ip address add address=10.20.20.1/24 interface=somenet;
```

```
routing bgp instance set default as=65000 redistribute-connected=yes;
routing bgp peer add remote-address=10.3.3.1 remote-as=100;
```

Results

Routes on CE1 router A:

```
 [admin@A] > ip route print detail
 Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp,
 o - ospf, m - mme, B - blackhole, U - unreachable, P - prohibit

1 ADC dst-address=10.1.1.4/30 pref-src=10.1.1.5 gateway=A_C gateway-status=A_C reachable
 distance=0 scope=10

2 ADb dst-address=10.3.3.0/30 gateway=10.1.1.2 gateway-status=10.1.1.2 reachable A_B
 distance=20 scope=40 target-scope=10 bgp-as-path=100 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1 received-from=peer1

3  Db dst-address=10.3.3.0/30 gateway=10.1.1.6 gateway-status=10.1.1.6 reachable A_C
 distance=20 scope=40 target-scope=10 bgp-as-path=100 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1 received-from=peer2

4 ADC dst-address=10.10.10.1/30 pref-src=10.1.1.1 gateway=somenet
 gateway-status=somenet reachable distance=0 scope=10

5 ADb dst-address=10.20.20.0/24 gateway=10.1.1.2 gateway-status=10.1.1.2 reachable A_B
 distance=20 scope=40 target-scope=10 bgp-as-path=100,65000 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1 received-from=peer1

6  Db dst-address=10.20.20.0/24 gateway=10.1.1.6 gateway-status=10.1.1.6 reachable A_C
 distance=20 scope=40 target-scope=10 bgp-as-path=100,65000 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1 received-from=peer2
```

Routes on CE2 router F:

```
 [admin@F] > ip route print detail
 Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp,
 o - ospf, m - mme, B - blackhole, U - unreachable, P - prohibit

0 ADb dst-address=10.1.1.0/30 gateway=10.3.3.1 gateway-status=10.3.3.1 reachable F_E
 distance=20 scope=40 target-scope=10 bgp-as-path=100 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1 received-from=peer1

1 ADb dst-address=10.1.1.4/30 gateway=10.3.3.1 gateway-status=10.3.3.1 reachable F_E
 distance=20 scope=40 target-scope=10 bgp-as-path=100 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1 received-from=peer1

2 ADC dst-address=10.3.3.0/30 pref-src=10.3.3.2 gateway=F_E gateway-status=F_E reachable
 distance=0 scope=10
```

```
3 ADb dst-address=10.10.10.0/24 gateway=10.3.3.1 gateway-status=10.3.3.1 reachable F_E
 distance=20 scope=40 target-scope=10 bgp-as-path=100,100 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1,SOO:1:100 received-from=peer1

4 ADC dst-address=10.20.20.0/30 pref-src=10.20.20.1 gateway=somenet
 gateway-status=somenet reachable distance=0 scope=10
```

Routes on PE1 router B:

```
 [admin@B] > ip route print detail
 Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp,
 o - ospf, m - mme, B - blackhole, U - unreachable, P - prohibit

0 ADC dst-address=10.1.1.0/30 pref-src=10.1.1.2 gateway=B_A gateway-status=B_A reachable
 distance=0 scope=10 routing-mark=vrf1

1  Db dst-address=10.1.1.0/30 gateway=10.1.1.1 gateway-status=10.1.1.1 on vrf1 reachable A_B
 distance=20 scope=40 target-scope=10 routing-mark=vrf1 bgp-as-path=65000
 bgp-origin=incomplete bgp-ext-communities=SOO:1:100 received-from=peer2

2 ADb dst-address=10.1.1.4/30 =gateway=10.1.1.1 gateway-status=10.1.1.1 on vrf1 reachable B_A
 distance=20 scope=40 target-scope=10 routing-mark=vrf1 bgp-as-path=65000
 bgp-origin=incomplete bgp-ext-communities=SOO:1:100 received-from=peer2

3  Db dst-address=10.1.1.4/30 gateway=10.9.9.3 gateway-status=10.9.9.3 recursive via 10.2.2.2 B_D
 distance=20 scope=40 target-scope=30 routing-mark=vrf1 bgp-local-pref=100 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1

4 ADb dst-address=10.3.3.0/30 gateway=10.9.9.5 gateway-status=10.9.9.5 recursive via 10.2.2.2 B_D
 distance=20 scope=40 target-scope=30 routing-mark=vrf1 bgp-local-pref=100 bgp-origin=incomplete
 bgp-ext-communities=RT:1:1

5 ADb dst-address=10.10.10.0/24 gateway=10.1.1.1 gateway-status=10.1.1.1 on vrf1 reachable B_A
 distance=20 scope=40 target-scope=10 routing-mark=vrf1 bgp-as-path=65000 bgp-origin=incomplete
 bgp-ext-communities=SOO:1:100 received-from=peer2

6 ADb dst-address=10.20.20.0/24 gateway=10.9.9.5 gateway-status=10.9.9.5 recursive via 10.2.2.2 B_D
 distance=20 scope=40 target-scope=30 routing-mark=vrf1 bgp-as-path=65000 bgp-local-pref=100
 bgp-origin=incomplete bgp-ext-communities=RT:1:1

7 ADC dst-address=10.2.2.0/30 pref-src=10.2.2.1 gateway=B_D gateway-status=B_D reachable
 distance=0 scope=10

8 ADC dst-address=10.9.9.2/32 pref-src=10.9.9.2 gateway=lobridge gateway-status=lobridge reachable
 distance=0 scope=10

9 A S dst-address=10.9.9.3/32 gateway=10.2.2.2 gateway-status=10.2.2.2 reachable B_D
 distance=1 scope=30 target-scope=10
```

```
10 A S dst-address=10.9.9.4/32 gateway=10.2.2.2 gateway-status=10.2.2.2 reachable B_D
 distance=1 scope=30 target-scope=10

11 A S dst-address=10.9.9.5/32 gateway=10.2.2.2 gateway-status=10.2.2.2 reachable B_D
 distance=1 scope=30 target-scope=10
```

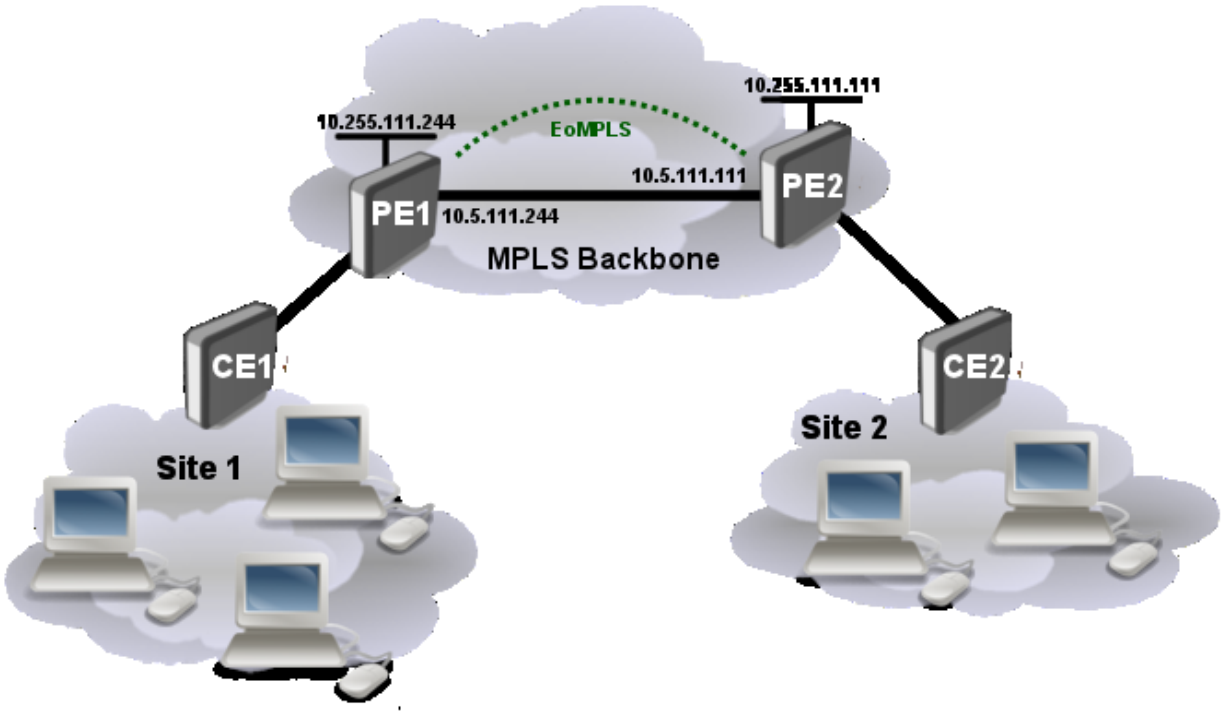
Manual:EoMPLS vs Cisco

Summary

This article describes the basic setup of Point-to-Point EoMPLS with Cisco routers. In this example IOS 15.1 was used, configuration in older versions may differ.

Configuration

Consider network setup as ilustrated below:



We will be setting up the layer 2 connection between the CE and PE routers as well as the MPLS and EoMPLS between PE routers. The layer 2 link between the CE and PE routers will be an Ethernet circuit.

Set up IP connectivity, OSPF and LDP

CE1:

```
/ip address add address=192.168.111.1/24 interface=ether1
```

CE2:

```
/ip address add address=192.168.111.2/24 interface=ether2
```

PE1 (RouterOS):

```
/interface bridge add name=loopback0

/ip address
add address=10.255.111.244/32 interface=loopback0
add address=10.5.111.244/24 interface=ether1

/mpls ldp
set enabled=yes lsr-id=10.255.111.244 transport-address=10.255.111.244

/mpls ldp interface
add interface=ether1

/routing ospf network
add area=backbone network=10.255.111.111/32
add area=backbone network=10.5.111.0/24
```

PE2 (IOS):

```
mpls label protocol ldp
!
interface Loopback0
 ip address 10.255.111.111 255.255.255.255
!
interface GigabitEthernet5/1
 ip address 10.5.111.111 255.255.255.0
 mpls ip
 mpls label protocol ldp
!
router ospf 1
 redistribute connected
 network 10.5.111.0 0.0.0.255 area 0
 network 10.255.111.111 0.0.0.0 area 0
!
```

Configure EoMPLS tunnel

PE1 (RouterOS):

```
/interface bridge
add name=vpn

/interface vpls
add cisco-style=yes cisco-style-id=111 disabled=no l2mtu=1500 name=cisco-eompls \
    remote-peer=10.255.111.111 use-control-word=yes

/interface bridge port
add interface=ether2 bridge=vpn
add interface=cisco-eompls bridge=vpn
```

PE2 (IOS):

```
pseudowire-class l2vpn
 encapsulation mpls
 control-word
!
interface GigabitEthernet5/2
 no ip address
 xconnect 10.255.111.244 111 pw-class l2vpn
  mtu 1500
!
```

Adjust MTUs

Because Ethernet over MPLS does not allow packets to be fragmented and reassembled on Cisco routers, ensure that the maximum transmission unit (MTU) of all intermediate links between endpoints is sufficient to carry the largest possible Layer 2 frame.

PE1 (RouterOS):

```
/mpls interface
set [find interface=all ] mpls-mtu=1526
```

PE2 (IOS):

```
interface GigabitEthernet5/1
 mtu 1526
 ip mtu 1500
 mpls mtu 1526
!
```

Verify Tunnel Operation

PE1 (RouterOS):

```
[admin@RouterOS] /interface vpls> monitor 0
       remote-label: 19
        local-label: 21
      remote-status:
          transport: 10.255.111.111/32
  transport-nexthop: 10.5.111.111
     imposed-labels: 19
```

PE2 (IOS):

```
C7606#show mpls l2transport vc

Local intf     Local circuit              Dest address    VC ID      Status
-------------  -------------------------- --------------- ---------- ----------
Gi5/2          Ethernet                   10.255.111.244  111        UP
```

For more detailed output:

```
C7606#show mpls l2transport vc detail
Local interface: Gi5/2 up, line protocol up, Ethernet up
  Destination address: 10.255.111.244, VC ID: 111, VC status: up
    Output interface: Gi5/1, imposed label stack {21}
    Preferred path: not configured
    Default path: active
    Next hop: 10.5.111.244
  Create time: 2d21h, last status change time: 01:36:12
  Signaling protocol: LDP, peer 10.255.111.244:0 up
    Targeted Hello: 10.255.111.111(LDP Id) -> 10.255.111.244, LDP is UP
    Status TLV support (local/remote)   : enabled/not supported
      LDP route watch                   : enabled
      Label/status state machine        : established, LruRru
      Last local dataplane   status rcvd: No fault
      Last local SSS circuit status rcvd: No fault
      Last local SSS circuit status sent: No fault
      Last local  LDP TLV    status sent: No fault
      Last remote LDP TLV    status rcvd: Not sent
      Last remote LDP ADJ    status rcvd: No fault
    MPLS VC labels: local 19, remote 21
    Group ID: local 0, remote 0
    MTU: local 1500, remote 1500
    Remote interface description:
  Sequencing: receive disabled, send disabled
  Control Word: On
  VC statistics:
    transit packet totals: receive 6073, send 5394
    transit byte totals:   receive 1309806, send 758021
    transit packet drops:  receive 0, seq error 0, send 18
```

See Also

- LDP Based VPLS
- BGP Based VPLS
- Cisco style VPLS
- L2VPN vs Juniper configuration example

Manual:Interface/Bonding

Applies to RouterOS: v3, v4

Summary

Bonding is a technology that allows to aggregate multiple ethernet-like interfaces into a single virtual link, thus getting higher data rates and providing failover.

Specifications

- **Packages required**: system
- **License required**: Level1
- **Submenu level**: `/interface bonding`
- **Standards and Technologies**: None
- **Hardware usage**: Not significant

Quick Setup Guide

Let us assume that we have 2 NICs in each router (Router1 and Router2) and want to get maximum data rate between 2 routers. To make this possible, follow these steps:

- Make sure that you do not have IP addresses on interfaces which will be enslaved for bonding interface!
- Add bonding interface on Router1:

```
[admin@Router1] interface bonding> add slaves=ether1,ether2
```

And on Router2:

```
[admin@Router2] interface bonding> add slaves=ether1,ether2
```

Add addresses to bonding interfaces:

```
[admin@Router1] ip address> add address=172.16.0.1/24 interface=bonding1
[admin@Router2] ip address> add address=172.16.0.2/24 interface=bonding1
```

Test the link from Router1:

```
[admin@Router1] interface bonding> /pi 172.16.0.2
172.16.0.2 ping timeout
172.16.0.2 ping timeout
172.16.0.2 ping timeout
```

```
172.16.0.2 64 byte ping: ttl=64 time=2 ms
172.16.0.2 64 byte ping: ttl=64 time=2 ms
```

Note: bonding interface needs a couple of seconds to get connectivity with its peer.

Link monitoring

It is critical that one of available link monitoring options are enabled. In example above if one of the bonded links fail, bonding driver will still continue to send packets over failed link which will lead to network degradation. Currently bonding in RouterOS supports two schemes for monitoring a link state of slave devices: MII and ARP monitoring. It is not possible to use both methods at a time due to restrictions in the bonding driver.

ARP Monitoring

ARP monitoring sends ARP queries and uses the response as an indication that the link is operational. This also gives assurance that traffic is actually flowing over the links. If balance-rr and balance-xor modes are set, then the switch should be configured to evenly distribute packets across all links. Otherwise all replies from the ARP targets will be received on the same link which could cause other links to fail. ARP monitoring is enabled by setting three properties `link-monitoring`, `arp-ip-targets` and `arp-interval`. Meaning of each option is described later in this article. It is possible to specify multiple ARP targets that can be useful in a High Availability setups. If only one target is set, the target itself may go down. Having an additional targets increases the reliability of the ARP monitoring.

Enable ARP monitoring

```
[admin@Router1] interface bonding> set 0 link-monitoring=arp arp-ip-targets=172.16.0.2
[admin@Router2] interface bonding> set 0 link-monitoring=arp arp-ip-targets=172.16.0.1
```

We will not change `arp-interval` value in our example, RouterOS sets `arp-interval` to 100ms by default.

Unplug one of the cables to test if link monitoring works correctly, you will notice some ping timeouts until arp monitoring detects link failure.

```
[admin@Router1] interface bonding> /pi 172.16.0.2
172.16.0.2 ping timeout
172.16.0.2 64 byte ping: ttl=64 time=2 ms
172.16.0.2 ping timeout
172.16.0.2 64 byte ping: ttl=64 time=2 ms
172.16.0.2 ping timeout
172.16.0.2 64 byte ping: ttl=64 time=2 ms
172.16.0.2 64 byte ping: ttl=64 time=2 ms
172.16.0.2 64 byte ping: ttl=64 time=2 ms
```

MII monitoring

MII monitoring monitors only the state of the local interface. In RouterOS it is possible to configure MII monitoring in two ways:

- **MII Type 1** - device driver determines whether link is up or down. If device driver does not support this option then link will appear as always up.
- **MII Type 2** - deprecated calling sequences within the kernel are used to determine if link is up. This method is less efficient but can be used on all devices. This mode should be set only if MII type 1 is not supported.

Main disadvantage is that MII monitoring can't tell if the link actually can pass the packets or not even if the link is detected as up.

MII monitoring is configured setting desired `link-monitoring` mode and `mii-interval`.

Enable MII Type2 monitoring:

```
[admin@Router1] interface bonding> set 0 link-monitoring=mii-type-2
[admin@Router2] interface bonding> set 0 link-monitoring=mii-type-2
```

We will leave mii-interval to it's default value (100ms)

When unplugging one of the cables, notice that failure was detected almost instantly compared to ARP link monitoring.

Bonding modes

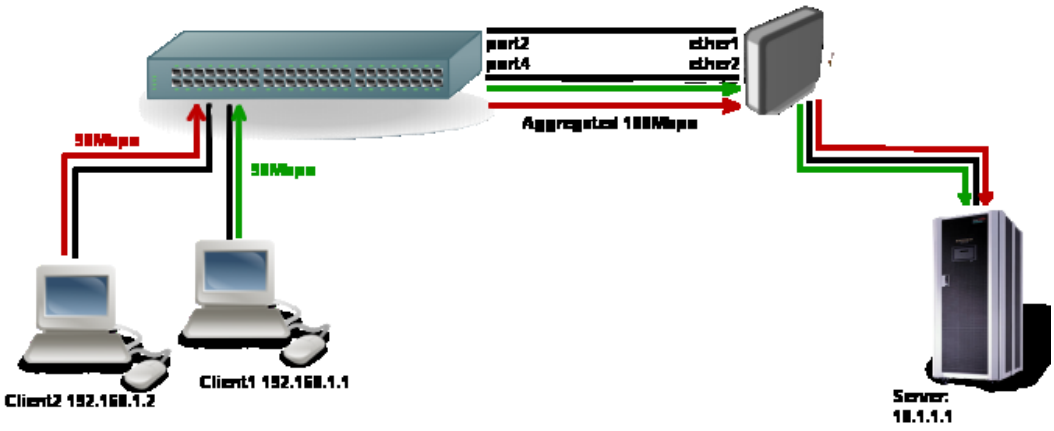
802.3ad

802.3ad mode is an IEEE standard also called LACP (Link Aggregation Control Protocol). It includes automatic configuration of the aggregates, so minimal configuration of the switch is needed. This standard also mandates that frames will be delivered in order and connections should not see mis-ordering of packets. Also standard mandates that all devices in the aggregate must operate at the same speed and duplex and works only with MII link monitoring.

LACP balances outgoing traffic across the active ports based on hashed protocol header information and accepts incoming traffic from any active port. The hash includes the Ethernet source and destination address, and, if available, the VLAN tag, and the IPv4/IPv6 source and destination address. How has is calculated depends on **transmit-hash-policy** parameter.

Note: layer-3-and-4 mode is not fully compatible with LACP.

Configuration example



Example connects two ethernet interfaces on a router to the Edimax switch as a single load balanced and fault tolerant link. More interfaces can be added to increase throughput and fault tolerance. Since frame ordering is mandatory on Ethernet links then any traffic between two devices always flows over the same physical link limiting the maximum speed to that of one interface. The transmit algorithm attempts to use as much information as it can to distinguish different traffic flows and balance across the available interfaces.

Router R1 configuration:

```
/inteface bonding add slaves=ether1,ether2 mode=802.3ad lacp-rate=30secs link-monitoring=mii-type1 \
  transmit-hash-policy=layer-2-and-3
```

Configuration on a switch:

```
                  Intelligent Switch : Trunk Configuration
                  ==================

 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 M1 M2
1 -  v  -  v  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
2 -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
3 -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
4 -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
5 -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
6 -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
7 -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -

TRK1   LACP
TRK2   Disable
TRK3   Disable
TRK4   Disable
TRK5   Disable
TRK6   Disable
TRK7   Disable
```

Notice that LACP is enabled on first trunk group (TRK1) and switch ports on first trunk group are bound with 'v' flag. In our case port 2 and port4 will run LACP.

Verify if LACP is working: On the switch at first we should verify if LACP protocol is enabled and running:

```
                  Intelligent Switch : LACP Port State Active Configuration
                  ==================


    Port          State Activity             Port          State Activity
    ---------------------------              ---------------------------
    2             Active
    4             Active
```

After that we can ensure that LACP negotiated with our router. If you don't see both ports on the list then something is wrong and LACP is not going to work.

```
                  Intelligent Switch : LACP Group Status
                  ==================
```

```
                              Group
               [Actor]                        [Partner]

  Priority:    1                              65535

  MAC     :    000E2E2206A9                   000C42409426

  Port_No   Key     Priority   Active       Port_No   Key     Priority
  2         513     1          selected     1         9       255
  4         513     1          selected     2         9       255
```

After we verified that switch successfully negotiated LACP with our router, we can start traffic from Client1 and Client2 to the Server and check how traffic is evenly forwarded through both bonding slaves:

```
[admin@test-host] /interface> monitor-traffic ether1,ether2,bonding1
    rx-packets-per-second: 8158     8120     16278
      rx-drops-per-second: 0        0        0
     rx-errors-per-second: 0        0        0
       rx-bits-per-second: 98.8Mbps 98.2Mbps 197.0Mbps
    tx-packets-per-second: 4833     4560     9394
      tx-drops-per-second: 0        0        0
     tx-errors-per-second: 0        0        0
       tx-bits-per-second: 2.7Mbps  3.0Mbps  5.8Mbps
```

balance-rr

If this mode is set, packets are transmitted in sequential order from the first available slave to the last.

Balance-rr is the only mode that will send packets across multiple interfaces that belong to the same TCP/IP connection.

When utilizing multiple sending and multiple receiving links, packets often are received out of order, which result in segment retransmission, for other protocols such as UDP it is not a problem if client software can tolerate out-of-order packets.

If switch is used to aggregate links together, then appropriate switch port configuration is required, however many switches do not support balance-rr.

Quick setup guide demonstrates use of the balance-rr bonding mode. As you can see, it is quite simple to set up. Balance-rr is also useful for bonding several wireless links, however it requires equal bandwidth for all bonded links. If bandwidth of one bonded link drops, then total bandwidth of bond will be equal to bandwidth of the slowest bonded link.

active-backup

This mode uses only one active slave to transmit packets. Different slave becomes active only if primary slave fails. Mac address of the bonding interface is visible only on active port to avoid confusing of the switch. Active-backup is best choice in high availability setups with multiple switches that are interconnected.

ARP monitoring in this mode will not work correctly if both routers are directly connected. In such setups `mii-type1` or `mii-type2` monitoring must be used or switch should be put between routers.

balance-xor

This mode balances outgoing traffic across the active ports based on hashed protocol header information and accepts incoming traffic from any active port. Mode is very similar to LACP except that it is not standardized and works with **layer-3-and-4** hash policy.

broadcast

When ports configured with broadcast mode, all slave ports transmits the same packets to the destination that way providing fault tolerance. This mode does not provide load balancing.

balance-tlb

This mode balances outgoing traffic by peer. Each link can be a different speed and duplex and no specific switch configuration is required as in other modes. Downside of this mode is that only MII link monitoring is supported and incoming traffic is not balanced. Incoming traffic will use the link that is configured as "primary".

Configuration example

Lets assume than router has two links - **ether1** max bandwidth is 10Mbps and **ether2** max bandwidth is 5Mbps. First link has more bandwidth so we set it as primary link

```
/interface bonding add mode=balance-tlb slaves=ether1,ether2 primary=ether1
```

No additional configuration is required for the switch.

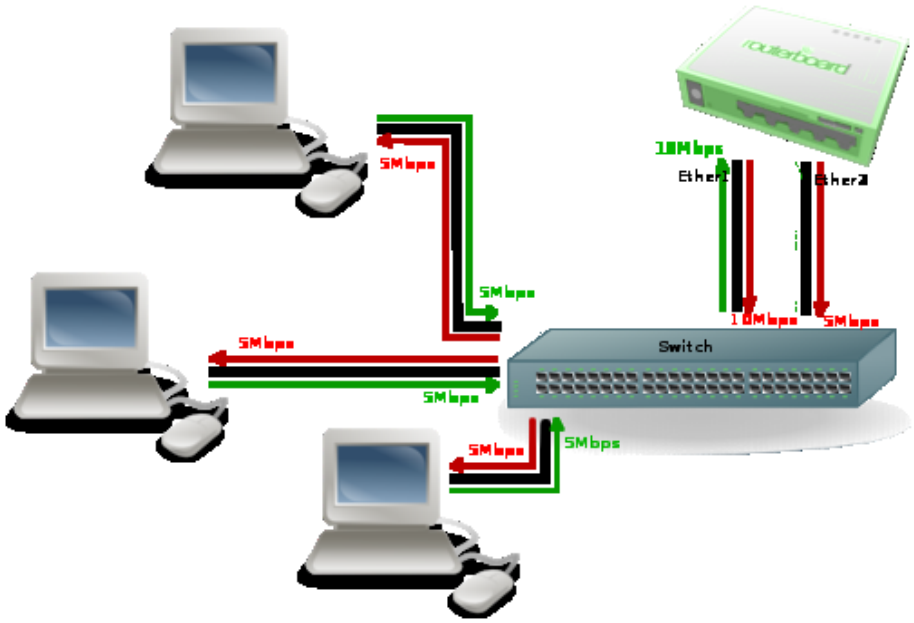


Image above illustrates how balance-tlb mode works. As you can see router can communicate to all the clients connected to switch with total bandwidth of both links (15Mbps). But as you already know, balance-tlb is not balancing incoming traffic. In our example clients can communicate to router with total bandwidth of primary link which is 10Mbps in our configuration.

balance-alb

Mode is basically the same as `balance-tlb` but incoming traffic is also balanced. Only additional downside of this mode is that it requires device driver capability to change mac address. Most of the cheap cards do not support this mode.

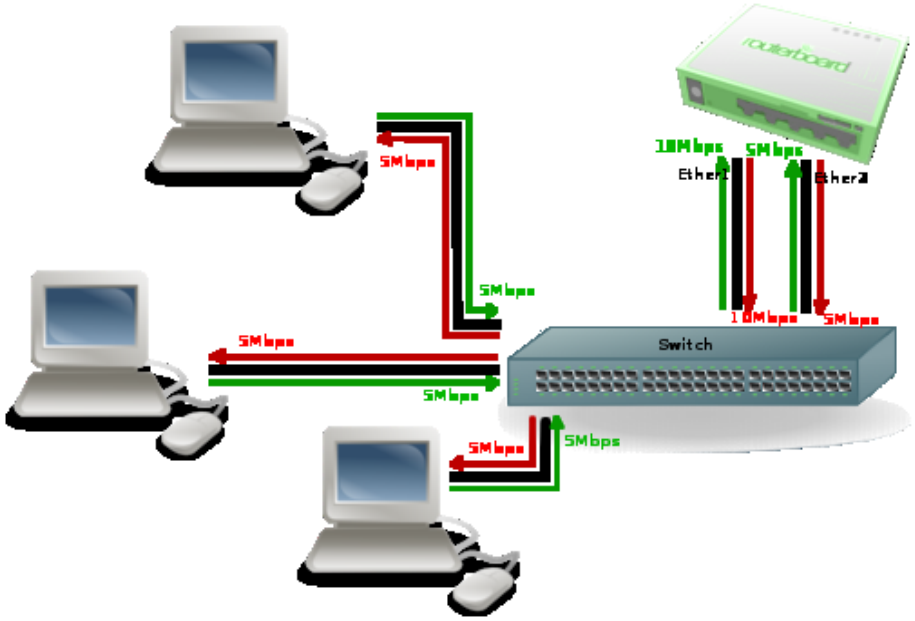


Image above illustrates how balance-alb mode works. Compared to balance-tlb traffic from clients also can use secondary link to communicate with router.

Property Description

Property	Description
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; Default: **enabled**)	Address Resolution Protocol for the interface. • `disabled` - the interface will not use ARP • `enabled` - the interface will use ARP • `proxy-arp` - the interface will use the ARP proxy feature • `reply-only` - the interface will only reply to the requests originated to its own IP addresses. Neighbour MAC addresses will be resolved using /ip arp statically set table only
arp-interval (*time*; Default: **00:00:00.100**)	time in milliseconds which defines how often to monitor ARP requests
arp-ip-targets (*IP addres*; Default:)	IP target address which will be monitored if `link-monitoring` is set to arp. You can specify multiple IP addresses, separated by comma
down-delay (*time*; Default: **00:00:00**)	if a link failure has been detected, bonding interface is disabled for down-delay time. Value should be a multiple of `mii-interval`
lacp-rate (*1sec* | *30secs*; Default: **30secs**)	Link Aggregation Control Protocol rate specifies how often to exchange with LACPDUs between bonding peer. Used to determine whether link is up or other changes have occurred in the network. LACP tries to adapt to these changes providing failover.

link-monitoring (*arp* | *mii-type1* | *mii-type2* | *none*; Default: **none**)

method to use for monitoring the link (whether it is up or down)

- `arp` - uses Address Resolution Protocol to determine whether the remote interface is reachable
- `mii-type1` - uses Media Independent Interface type1 to determine link status. Link status determenation relies on the device driver
- `mii-type2` - similar as mii-type1, but status determination does not rely on the device driver
- `none` - no method for link monitoring is used.

Note: some bonding modes require specific link monitoring to work properly.

mii-interval (*time*; Default: **00:00:00.100**)

how often to monitor the link for failures (parameter used only if `link-monitoring` is mii-type1 or mii-type2)

mode (*802.3ad* | *active-backup* | *balance-alb* | *balance-rr* | *balance-tlb* | *balance-xor* | *broadcast*; Default: **balance-rr**)

Specifies one of the bonding policies

- `802.3ad` - IEEE 802.3ad dynamic link aggregation. In this mode, the interfaces are aggregated in a group where each slave shares the same speed. Provides fault tolerance and load balancing. Slave selection for outgoing traffic is done according to the `transmit-hash-policy` more>
- `active-backup` - provides link backup. Only one slave can be active at a time. Another slave becomes active only, if first one fails. more>
- `balance-alb` - adaptive load balancing. The same as `balance-tlb` but received traffic is also balanced. Device driver should have support for changing the mac address. more>
- `balance-rr` - round-robin load balancing. Slaves in bonding interface will transmit and receive data in sequential order. Provides load balancing and fault tolerance. more>
- `balance-tlb` - Outgoing traffic is distributed according to the current load on each slave. Incoming traffic is not balanced and is received by the current slave. If receiving slave fails, then another slave takes the MAC address of the failed slave. more>
- `balance-xor` - Transmit based on the selected `transmit-hash-policy`. This mode provides load balancing and fault tolerance. more>
- `broadcast` - Broadcasts the same data on all interfaces at once. This provides fault tolerance but slows down traffic throughput on some slow machines. more>

mtu (*integer*; Default: **1500**)

Maximum Transmit Unit in bytes

name (*string*; Default:)

descriptive name of bonding interface

primary (*string*; Default:)

Interface is used as primary output interface. If primary interface fails, only then others slaves will be used. This value works only with active-backup mode

slaves (*string*; Default: **none**)

at least two ethernet-like interfaces separated by a comma, which will be used for bonding

up-delay (*time*; Default: **00:00:00**)

if a link has been brought up, bonding interface is disabled for up-delay time and after this time it is enabled. Value should be a multiple of `mii-interval`

transmit-hash-policy (*layer-2* | *layer-2-and-3* | *layer-3-and-4*; Default: **layer-2**)

Selects the transmit hash policy to use for slave selection in balance-xor and 802.3ad modes

- `layer-2` - Uses XOR of hardware MAC addresses to generate the hash. This algorithm will place all traffic to a particular network peer on the same slave. This algorithm is 802.3ad compliant.
- `layer-2-and-3` - This policy uses a combination of layer2 and layer3 protocol information to generate the hash. Uses XOR of hardware MAC addresses and IP addresses to generate the hash. This algorithm will place all traffic to a particular network peer on the same slave. For non-IP traffic, the formula is the same as for the layer2 transmit hash policy. This policy is intended to provide a more balanced distribution of traffic than layer2 alone, especially in environments where a layer3 gateway device is required to reach most destinations. This algorithm is 802.3ad compliant.
- `layer-3-and-4` - This policy uses upper layer protocol information, when available, to generate the hash. This allows for traffic to a particular network peer to span multiple slaves, although a single connection will not span multiple slaves. For fragmented TCP or UDP packets and all other IP protocol traffic, the source and destination port information is omitted. For non-IP traffic, the formula is the same as for the layer2 transmit hash policy. This algorithm is not fully 802.3ad compliant.

Notes

Link failure detection and failover is working significantly better with expensive network cards, for example, made by Intel, then with more cheap ones. For example, on Intel cards failover is taking place in less than a second after link loss, while on some other cards, it may require up to 20 seconds. Also, the Active load balancing (mode=balance-alb) does not work on some cheap cards.

Manual:Interface/Bridge

Applies to RouterOS: v3, v4+

Summary

Sub-menu: `/interface bridge`
Standards: `IEEE802.1D` [1]

Ethernet-like networks (Ethernet, Ethernet over IP, IEEE802.11 in ap-bridge or bridge mode, WDS, VLAN) can be connected together using MAC bridges. The bridge feature allows the interconnection of hosts connected to separate LANs (using EoIP, geographically distributed networks can be bridged as well if any kind of IP network interconnection exists between them) as if they were attached to a single LAN. As bridges are transparent, they do not appear in traceroute list, and no utility can make a distinction between a host working in one LAN and a host working in another LAN if these LANs are bridged (depending on the way the LANs are interconnected, latency and data rate between hosts may vary).

Network loops may emerge (intentionally or not) in complex topologies. Without any special treatment, loops would prevent network from functioning normally, as they would lead to avalanche-like packet multiplication. Each bridge runs an algorithm which calculates how the loop can be prevented. STP and RSTP allows bridges to communicate with each other, so they can negotiate a loop free topology. All other alternative connections that would otherwise form loops, are put to standby, so that should the main connection fail, another connection could take its place. This algorithm exchange configuration messages (BPDU - Bridge Protocol Data Unit) periodically, so that all bridges would be updated with the newest information about changes in network topology. (R)STP selects root bridge which is responosible for network reconfiguration, such as blocking and opening ports of the other bridges. The root bridge is the bridge with lowest bridge ID.

Bridge Interface Setup

Sub-menu: `/interface bridge`

To combine a number of networks into one bridge, a bridge interface should be created (later, all the desired interfaces should be set up as its ports). One MAC address will be assigned to all the bridged interfaces (the smallest MAC address will be chosen automatically).

Property	Description
admin-mac (*MAC address*; Default:)	Static MAC address of the bridge (takes effect if `auto-mac=no`)
ageing-time (*time*; Default: **00:05:00**)	How long a host information will be kept in the bridge database
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; Default: **enabled**)	Address Resolution Protocol setting
auto-mac (*yes* | *no*; Default: **yes**)	Automatically select the smallest MAC address of bridge ports as a bridge MAC address
forward-delay (*time*; Default: **00:00:15**)	Time which is spent during the initialization phase of the bridge interface (i.e., after router startup or enabling the interface) in listening/learning state before the bridge will start functioning normally
l2mtu (*integer*; read-only)	Layer2 Maximum transmission unit. read more»
max-message-age (*time*; Default: **00:00:20**)	How long to remember Hello messages received from other bridges
mtu (*integer*; Default: **1500**)	Maximum Transmission Unit
name (*text*; Default: **bridgeN**)	Name of the bridge interface
priority (*integer: 0..65535*; Default: **32768**)	Spanning tree protocol priority for bridge interface. Bridge with the smallest (lowest) bridge ID becomes a Root-Bridge. Bridge ID consists of two numbers - priority and MAC address of the bridge. To compare two bridge IDs, the priority is compared first. If two bridges have equal priority, then the MAC addresses are compared.
protocol-mode (*none* | *rstp* | *stp*; Default: **none**)	Select Spanning tree protocol (STP) or Rapid spanning tree protocol (RSTP) to ensure a loop-free topology for any bridged LAN. RSTP provides provides for faster spanning tree convergence after a topology change.
transmit-hold-count (*integer: 1..10*; Default: **6**)	The Transmit Hold Count used by the Port Transmit state machine to limit transmission rate

http://en.wikipedia.org/wiki/Spanning_Tree_Protocol [2]

To add and enable a bridge interface that will forward all the protocols:

```
[admin@MikroTik] /interface bridge> add
[admin@MikroTik] /interface bridge> print
Flags: X - disabled, R - running
 0  R name="bridge1" mtu=1500 l2mtu=65535 arp=enabled
      mac-address=00:00:00:00:00:00 protocol-mode=none priority=0x8000
      auto-mac=yes admin-mac=00:00:00:00:00:00 max-message-age=20s
      forward-delay=15s transmit-hold-count=6 ageing-time=5m
[admin@MikroTik] /interface bridge>
```

Bridge Settings

Sub-menu: `/interface bridge settings`

Property	Description
use-ip-firewall (*yes* | *no*; Default: **no**)	Makes bridged traffic to be processed through IP firewall
use-ip-firewall-for-pppoe (*yes* | *no*; Default: **no**)	Makes bridged unencrypted PPPoE traffic to be processed through IP firewall (requires `use-ip-firewall=yes` to work)
use-ip-firewall-for-vlan (*yes* | *no*; Default: **no**)	Makes bridged VLAN traffic to be processed through IP firewall (requires `use-ip-firewall=yes` to work)

Port Settings

Sub-menu: `/interface bridge port`

Port submenu is used to enslave interfaces in a particular bridge interface.

Property	Description
bridge (*name*; Default: **none**)	The bridge interface the respective interface is grouped in
edge (*auto* | *no* | *no-discover* | *yes* | *yes-discover*; Default: **auto**)	Set port as edge port or non-edge port, or enable automatic detection
external-fdb (*auto* | *no* | *yes*; Default: **auto**)	Whether to use wireless registration table to speed up bridge host learning
horizon (*none* | *integer 0..429496729*; Default: **none**)	Use split horizon bridging to prevent bridging loops. read more»
interface (*name*; Default: **none**)	Name of the interface
path-cost (*integer: 0..65535*; Default: **10**)	Path cost to the interface, used by STP to determine the "best" path
priority (*integer: 0..255*; Default: **128**)	The priority of the interface in comparison with other going to the same subnet

To group **ether1** and **ether2** in the already created **bridge1** bridge

```
[admin@MikroTik] /interface bridge port> add bridge=bridge1 interface=ether1
[admin@MikroTik] /interface bridge port> add bridge=bridge1 interface=ether2
[admin@MikroTik] /interface bridge port> print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE              BRIDGE              PRIORITY PATH-COST  HORIZON
 0    ether1                 bridge1             0x80     10         none
 1    ether2                 bridge1             0x80     10         none
[admin@MikroTik] /interface bridge port>
```

Bridge Monitoring

Sub-menu: `/interface bridge monitor`

Used to monitor the current status of a bridge.

Property	Description
current-mac-address (*MAC address*)	Current MAC address of the bridge
designated-port-count (*integer*)	Number of designated bridge ports
port-count (*integer*)	Number of the bridge ports
root-bridge (*yes* | *no*)	Shows whether bridge is the root bridge of the spanning tree
root-bridge-id (*text*)	The root bridge ID, which is in form of bridge-priority.bridge-MAC-address
root-path-cost (*integer*)	The total cost of the path to the root-bridge
root-port (*name*)	Port to which the root bridge is connected to
state (*enabled* | *disabled*)	State of the bridge

To monitor a bridge:

```
[admin@MikroTik] /interface bridge> monitor bridge1
                  state: enabled
    current-mac-address: 00:0C:42:52:2E:CE
            root-bridge: yes
         root-bridge-id: 0x8000.00:00:00:00:00:00
         root-path-cost: 0
              root-port: none
             port-count: 2
  designated-port-count: 0

[admin@MikroTik] /interface bridge>
```

Bridge Port Monitoring

Sub-menu: `/interface bridge port monitor`

Statistics of an interface that belongs to a bridge.

Property	Description
edge-port-discovery (*yes* | *no*)	Whether port to automatically detects edge ports
external-fdb (*yes* | *no*)	Shows whether registration table is used instead of forwarding data base
forwarding (*yes* | *no*)	Port state
learning (*yes* | *no*)	Port state
port-number (*integer 1..4095*)	Port identifier
role (*designated* | *root port* | *alternate* | *backup* | *disabled*)	(R)STP algorithm assigned role of the port: • **Disabled port** - not strictly part of STP, a network administrator can manually disable a port • **Root port –** a forwarding port that is the best port from Nonroot-bridge to Rootbridge • **Alternative port –** an alternate path to the root bridge. This path is different than using the root port • **Designated port –** a forwarding port for every LAN segment • **Backup port –** a backup/redundant path to a segment where another bridge port already connects.
sending-rstp (*yes* | *no*)	Whether the port is sending BPDU messages
status (*in-bridge* | *inactive*)	Port status

To monitor a bridge port:

```
[admin@MikroTik] /interface bridge port> monitor 0
               status: in-bridge
          port-number: 1
                 role: designated-port
            edge-port: no
  edge-port-discovery: yes
  point-to-point-port: no
         external-fdb: no
         sending-rstp: no
             learning: yes
           forwarding: yes

[admin@MikroTik] /interface bridge port>
```

Bridge Host Monitoring

Sub-menu: `/interface bridge host`

Property	Description
age (*read-only: time*)	The time since the last packet was received from the host
bridge (*read-only: name*)	The bridge the entry belongs to
external-fdb (*read-only: flag*)	Whether the host was learned using wireless registration table
local (*read-only: flag*)	Whether the host entry is of the bridge itself (that way all local interfaces are shown)
mac-address (*read-only: MAC address*)	Host's MAC address
on-interface (*read-only: name*)	Which of the bridged interfaces the host is connected to

To get the active host table:

```
[admin@MikroTik] /interface bridge host> print
Flags: L - local, E - external-fdb
  BRIDGE           MAC-ADDRESS       ON-INTERFACE          AGE
  bridge1          00:00:00:00:00:01 ether2                3s
  bridge1          00:01:29:FF:1D:CC ether2                0s
L bridge1          00:0C:42:52:2E:CF ether2                0s
  bridge1          00:0C:42:52:2E:D0 ether2                3s
  bridge1          00:0C:42:5C:A5:AE ether2                0s
[admin@MikroTik] /interface bridge host>
```

Bridge Firewall

Sub-menu: `/interface bridge filter, /interface bridge nat`

The bridge firewall implements packet filtering and thereby provides security functions that are used to manage data flow to, from and through bridge.

Packet flow diagram shows how packets are processed through router. It is possible to force bridge traffic to go through `/ip firewall filter` rules (see: Bridge Settings)

There are two bridge firewall tables:

- **filter** - bridge firewall with three predefined chains:
  - **input** - filters packets, which destination is the bridge (including those packets that will be routed, as they are anyway destined to the bridge MAC address)
  - **output** - filters packets, which come from the bridge (including those packets that has been routed normally)
  - **forward** - filters packets, which are to be bridged (note: this chain is not applied to the packets that should be routed through the router, just to those that are traversing between the ports of the same bridge)
- **nat** - bridge network address translation provides ways for changing source/destination MAC addresses of the packets traversing a bridge. Has two built-in chains:
  - **srcnat** - used for "hiding" a host or a network behind a different MAC address. This chain is applied to the packets leaving the router through a bridged interface
  - **dstnat** - used for redirecting some pakets to another destinations

You can put packet marks in bridge firewall (filter and NAT), which are the same as the packet marks in IP firewall put by mangle. So packet marks put by bridge firewall can be used in IP firewall, and vice versa.

General bridge firewall properties are described in this section. Some parameters that differ between nat and filter rules are described in further sections.

Property**802.3-sap** (*integer*)**802.3-type** (*integer*)**arp-dst-address** (*IP address; default:*)**arp-dst-mac-address** (*MAC address; default:*)**arp-gratuitous** (*yes | no; default:*)**arp-hardware-type** (*integer; default: 1*)**arp-opcode** (*arp-nak | drarp-error | drarp-reply | drarp-request | inarp-reply | inarp-request | reply | reply-reverse | request | request-reverse*)**arp-src-address** (*IP address; default:*)**arp-src-mac-address** (*MAC address; default:*)**chain** (*text*)**dst-address** (*IP address; default:*)**dst-mac-address** (*MAC address; default:*)**dst-port** (*integer 0..65535*)**in-bridge** (*name*)**in-interface** (*name*)**ingress-priority** (*integer 0..63*)**ip-protocol** (*ddp | ggp | icmp | igmp | ipsec-ah | ospf | rdp | tcp | vrrp | egp | gre | icmpv6 | ipencap | ipsec-esp | pim | rspf | udp | xns-idp | encap | hmp | idpr-cmtp | ipip | iso-tp4 | pup | st | vmtp | xtp*)**jump-target** (*name*)**limit** (*integer/time,integer*)**log-prefix** (*text*)**mac-protocol** (*arp | ip | ipv6 | ipx | length | pppoe | pppoe-discovery | rarp | vlan*)**out-bridge** (*name*)**out-interface** (*name*)**packet-mark** (*name*)**packet-type** (*broadcast | host | multicast | other-host*)**src-address** (*IP address; default:*)**src-mac-address** (*MAC address; default:*)**src-port** (*integer 0..65535*)**stp-flags** (*topology-change | topology-change-ack*)**stp-forward-delay** (*time 0..65535*)**stp-hello-time** (*time 0..65535*)**stp-max-age** (*time 0..65535*)**stp-msg-age** (*time 0..65535*)**stp-port** (*integer 0..65535*)**stp-root-address** (*MAC address*)**stp-root-cost** (*integer 0..65535*)**stp-root-priority** (*integer 0..65535*)**stp-sender-address** (*MAC address*)**stp-sender-priority** (*integer 0..65535*)**stp-type** (*config | tcn*)**vlan-encap** (*arp | ip | ipv6 | ipx | length | pppoe | pppoe-discovery | rarp | vlan*)**vlan-id** (*integer 0..4095*)**vlan-priority** (*integer 0..7*)DescriptionDSAP (Destination Service Access Point) and SSAP (Source Service Access Point) are 2 one byte fields, which identify the network protocol entities which use the link layer service. These bytes are always equal. Two hexadecimal digits may be specified here to match an SAP byteEthernet protocol type, placed after the IEEE 802.2 frame header. Works only if 802.3-sap is 0xAA (SNAP - Sub-Network Attachment Point header). For example, AppleTalk can be indicated by SAP code of

0xAA followed by a SNAP type code of 0x809BARP destination addressARP destination MAC addressMatches ARP gratuitous packetsARP hardware type. This normally Ethernet (Type 1) ARP opcode (packet type)

- **arp-nak** - negative ARP reply (rarely used, mostly in ATM networks)
- **drarp-error** - Dynamic RARP error code, saying that an IP address for the given MAC address can not be allocated
- **drarp-reply** - Dynamic RARP reply, with a temporaty IP address assignment for a host
- **drarp-request** - Dynamic RARP request to assign a temporary IP address for the given MAC address
- **inarp-reply** -
- **inarp-request** -
- **reply** - standard ARP reply with a MAC address
- **reply-reverse** - reverse ARP (RARP) reply with an IP address assigned
- **request** - standard ARP request to a known IP address to find out unknown MAC address
- **request-reverse** - reverse ARP (RARP) request to a known MAC address to find out unknown IP address (intended to be used by hosts to find out their own IP address, similarly to DHCP service)

ARP source addressARP source MAC addressBridge firewall chain, which the filter is functioning in (either a built-in one, or a user defined)Destination IP address (only if MAC protocol is set to IPv4)Destination MAC addressDestination port number or range (only for TCP or UDP protocols)Bridge interface through which the packet is coming inPhysical interface (i.e., bridge port) through which the packet is coming inMatches ingress priority of the packet. Priority may be derived from VLAN, WMM or MPLS EXP bit. read more» IP protocol (only if MAC protocol is set to IPv4)

- **ipsec-ah** - IPsec AH protocol
- **ipsec-esp** - IPsec ESP protocol
- **ddp** - datagram delivery protocol
- **egp** - exterior gateway protocol
- **ggp** - gateway-gateway protocol
- **gre** - general routing encapsulation
- **hmp** - host monitoring protocol
- **idpr-cmtp** - idpr control message transport
- **icmp** - internet control message protocol
- **icmpv6** -
- **igmp** - internet group management protocol
- **ipencap** - ip encapsulated in ip
- **encap** - ip encapsulation
- **ipip** - ip encapsulation
- **iso-tp4** - iso transport protocol class 4
- **ospf** - open shortest path first
- **pim** - protocol independent multicast
- **pup** - parc universal packet protocol
- **rspf** - radio shortest path first
- **rdp** - reliable datagram protocol
- **st** - st datagram mode
- **tcp** - transmission control protocol
- **udp** - user datagram protocol
- **vmtp** - versatile message transport
- **vrrp** -
- **xns-idp** - xerox ns idp
- **xtp** – xpress transfer protocol

If `action=jump` specified, then specifies the user-defined firewall chain to process the packet Restricts packet match rate to a given limit.

- **count** - maximum average packet rate, measured in packets per second (pps), unless followed by Time option
- **time** - specifies the time interval over which the packet rate is measured
- **burst** - number of packets to match in a burst

Defines the prefix to be printed before the logging informationEthernet payload type (MAC-level protocol)Outgoing bridge interfaceInterface via packet is leaving the bridgeMatch packets with certain packet mark MAC frame type:

- **broadcast** - broadcast MAC packet
- **host** - packet is destined to the bridge itself
- **multicast** - multicast MAC packet
- **other-host** - packet is destined to some other unicast address, not to the bridge itself

Source IP address (only if MAC protocol is set to IPv4)Source MAC addressSource port number or range (only for TCP or UDP protocols) The BPDU (Bridge Protocol Data Unit) flags. Bridge exchange configuration messages named BPDU peridiocally for preventing from loop

- **topology-change** - topology change flag is set when a bridge detects port state change, to force all other bridges to drop their host tables and recalculate network topology
- **topology-change-ack** - topology change acknowledgement flag is sen in replies to the notification packets

Forward delay timerSTP hello packets timeMaximal STP message ageSTP message ageSTP port identifierRoot bridge MAC addressRoot bridge costRoot bridge prioritySTP message sender MAC addressSTP sender priority The BPDU type:

- **config** - configuration BPDU
- **tcn** - topology change notification

the MAC protocol type encapsulated in the VLAN frameVLAN identifier fieldThe user priority field

- STP matchers are only valid if destination MAC address is 01:80:C2:00:00:00/FF:FF:FF:FF:FF:FF (Bridge Group address), also `stp` should be enabled.
- ARP matchers are only valid if `mac-protocol` is `arp` or `rarp`
- VLAN matchers are only valid for `vlan` ethernet protocol
- IP-related matchers are only valid if `mac-protocol` is set as `ipv4`
- 802.3 matchers are only consulted if the actual frame is compliant with IEEE 802.2 and IEEE 802.3 standards (**note**: it is not the industry-standard Ethernet frame format used in most networks worldwide!). These matchers are ignored for other packets.

Bridge Packet Filter

Sub-menu: `/interface bridge filter`

This section describes bridge packet filter specific filtering options, which were omitted in the general firewall description.

Property	Description

action (*accept* | *drop* | *jump* | *log* | *mark-packet* | *passthrough* | *return* | *set-priority*)	• **accept** - accept the packet. No action, i.e., the packet is passed through without undertaking any action, and no more rules are processed in the relevant list/chain • **drop** - silently drop the packet (without sending the ICMP reject message) • **jump** - jump to the chain specified by the value of the jump-target argument • **log** - log the packet • **mark** - mark the packet to use the mark later • **passthrough** - ignore this rule and go on to the next one. Acts the same way as a disabled rule, except for ability to count packets • **return** - return to the previous chain, from where the jump took place • **set-priority**

Bridge NAT

Sub-menu: `/interface bridge nat`

This section describes bridge NAT options, which were omitted in the general firewall description.

Property	Description
action (*accept* | *drop* | *jump* | *mark-packet* | *redirect* | *set-priority* | *arp-reply* | *dst-nat* | *log* | *passthrough* | *return* | *src-nat*)	• **accept** - accept the packet. No action, i.e., the packet is passed through without undertaking any action, and no more rules are processed in the relevant list/chain • **arp-reply** - send a reply to an ARP request (any other packets will be ignored by this rule) with the specified MAC address (only valid in dstnat chain) • **drop** - silently drop the packet (without sending the ICMP reject message) • **dst-nat** - change destination MAC address of a packet (only valid in dstnat chain) • **jump** - jump to the chain specified by the value of the jump-target argument • **log** - log the packet • **mark** - mark the packet to use the mark later • **passthrough** - ignore this rule and go on to the next one. Acts the same way as a disabled rule, except for ability to count packets • **redirect** - redirect the packet to the bridge itself (only valid in dstnat chain) • **return** - return to the previous chain, from where the jump took place • **set-priority** • **src-nat** - change source MAC address of a packet (only valid in srcnat chain)
to-arp-reply-mac-address (*MAC address*)	Source MAC address to put in Ethernet frame and ARP payload, when `action=arp-reply` is selected
to-dst-mac-address (*MAC address*)	Destination MAC address to put in Ethernet frames, when `action=dst-nat` is selected
to-src-mac-address (*MAC address*)	Source MAC address to put in Ethernet frames, when `action=src-nat` is selected

[Top | Back to Content]

References

[1] http://standards.ieee.org/getieee802/download/802.1D-2004.pdf
[2] http://en.wikipedia.org/wiki/Spanning_Tree_Protocol

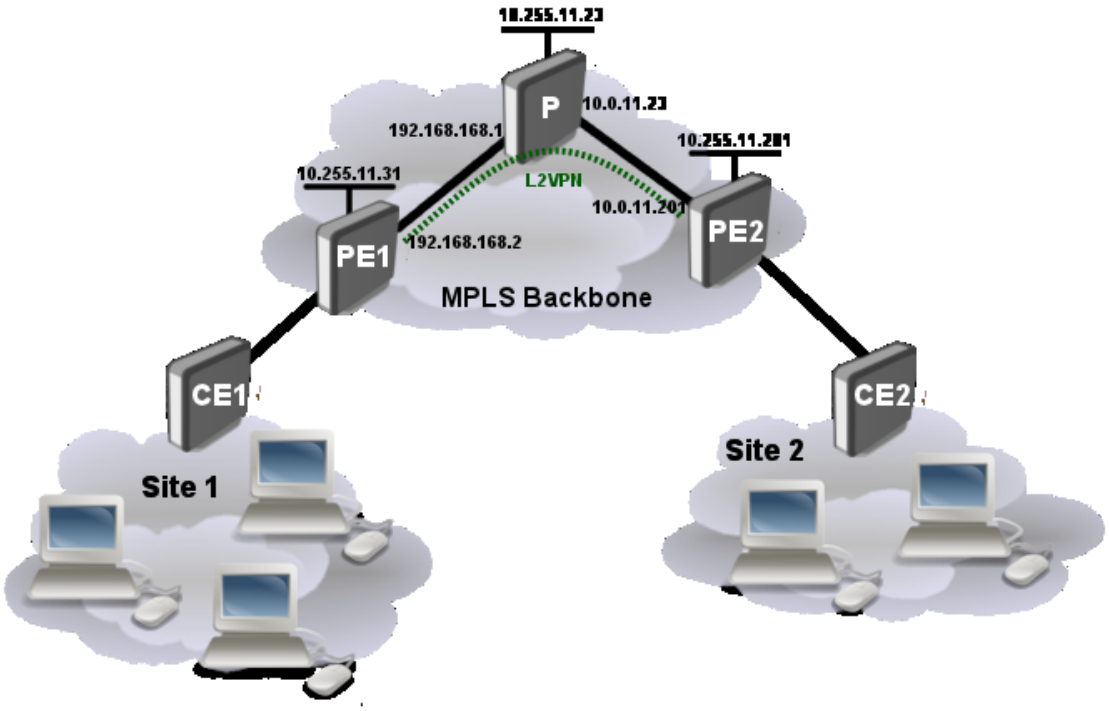
Manual:MPLS L2VPN vs Juniper

Summary

This article describes the basic setup of Point-to-Point L2VPN with Juniper J-series routers.

Configuration

Consider network setup as ilustrated below:



We will be setting up the layer 2 connection between the CE and PE routers as well as the MPLS and L2VPN between PE routers. The layer 2 link between the CE and PE routers will be an Ethernet VLAN circuit.

LDP based VPN

Set up VLANs

CE1 and **CE2** routers:

```
/interface vlan
  add vlan-id=600 name=vlan1 disabled=no interface=ether1
```

PE1 (RouterOS):

No configuration currently is needed, later we will bridge VPLS tunnel.

PE2 (JunOS):

```
interfaces {
    fe-0/0/1 {
        vlan-tagging;
        encapsulation vlan-ccc;
        unit 1 {
            encapsulation vlan-ccc;
            vlan-id 600;
        }
    }
}
```

Set up IP connection, OSPF and LDP

CE1:

```
/ip address add address=192.168.88.1/24 interface=vlan1
```

CE2:

```
/ip address add address=192.168.88.2/24 interface=vlan1
```

PE1 (RouterOS):

```
/interface bridge
  add name=loopback

/ip address
  add address=192.168.168.2/24 interface=ether3
  add address=10.255.11.31/32 interface=loopback

/routing ospf network
  add area=backbone disabled=no network=192.168.168.0/24
  add area=backbone disabled=no network=10.255.11.31/32

/mpls ldp
  set enabled=yes lsr-id=10.255.11.31 transport-address=10.255.11.31

/mpls ldp interface
  add interface=ether3
```

P (RouterOS):

```
/interface bridge
  add name=loopback

/ip address
  add address=10.0.11.23/24 interface=ether1
  add address=192.168.168.1/24 interface=ether2
  add address=10.255.11.23/32 interface=loopback

/routing ospf network
  add area=backbone disabled=no network=10.0.11.0/24
  add area=backbone disabled=no network=192.168.168.0/24
  add area=backbone disabled=no network=10.255.11.23/32

/mpls ldp
  set enabled=yes lsr-id=10.255.11.23 transport-address=10.255.11.23

/mpls ldp interface
  add interface=ether1
  add interface=ether2
```

PE2 (JunOS):

```
interfaces {
    fe-0/0/0 {
        unit 0 {
            family inet {
                address 10.0.11.201/24;
            }
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.255.11.201/32;
            }
        }
    }
}

protocols {
    mpls {
        interface fe-0/0/0.0;
        interface lo0.0;

    }

    ospf {
```

```
        export [ export-connected originate ];
        area 0.0.0.0 {
            interface fe-0/0/0.0;
            interface lo0.0 {
                passive;
            }
        }
    }
    ldp {
        egress-policy connected-only;
        transport-address 10.255.11.201;
        interface all;
    }
}
```

Finally we need to define policy options to advertise label binding for Loopback prefix:

```
policy-options {
    prefix-list loopback-prefix {
        10.255.11.201/32;
    }
    policy-statement connected-only {
        from {
            prefix-list loopback-prefix;
        }
        then accept;
    }
}
```

Set up L2VPN

PE1 (RouterOS):

```
/interface vpls
  add cisco-style=yes cisco-style-id=5 name=junos-l2circuit pw-type=tagged-ethernet \
    remote-peer=10.255.11.201

/interface bridge add name=vpn
/interface bridge port
  add interface=ether5 bridge=vpn
  add interface=junos-l2circuit bridge=vpn
```

We need to set pw-type**=tagged-ethernet** since on juniper encapsulation was set to **vlan-ccc**. Otherwise Juniper will throw an error */EM -- encapsulation mismatch /*

PE2 (JunOS):

```
protocol {
    l2circuit {
        neighbor 10.255.11.31 {
            interface fe-0/0/1.1 {
                virtual-circuit-id 5;
```

```
            }
        }
    }
}
```

Verify Operation

Verify if LDP neighbors are found and forwarding table is created:

PE1:

```
[admin@10.0.11.31] /mpls ldp neighbor> print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello,
V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                      SEN
 0 DO   10.255.11.23    10.255.11.31    10.255.11.23:0            no
 1 DOTV 10.255.11.201   10.255.11.31    10.255.11.201:0           yes
```

```
[admin@10.0.11.31] /mpls forwarding-table> print
Flags: L - ldp, V - vpls, T - traffic-eng
 #   IN-LABEL      OUT-LABELS  DESTINATION                   I NEXTHOP
 0   expl-null
 1 L 17            3396        10.255.11.201/32              e 192.168.168.1
 2 L 19                        10.255.11.23/32               e 192.168.168.1
 3 L 23            3390        10.5.101.0/24                 e 192.168.168.1
 4 V 29                        junos-l2circuit
```

PE2:

```
juniper@J4300> show ldp neighbor
Address            Interface          Label space ID         Hold time
10.255.11.31       lo0.0              10.255.11.31:0           42
10.0.11.23         fe-0/0/0.0         10.255.11.23:0           13
```

Verify traffic forwarding over LSP:

PE1:

```
[admin@10.0.11.31] /interface vpls> /tool traceroute 10.255.11.201
 # ADDRESS                                 RT1   RT2   RT3   STATUS
 1 192.168.168.1                           1ms   1ms   1ms   <MPLS:L=3396,E=0>
 2 10.255.11.201                           2ms   3ms   3ms
```

Verify if L2VPN tunnel is up and running:

PE1

```
[admin@10.0.11.31] /interface vpls> monitor junos-l2circuit once
       remote-label: 577168
        local-label: 29
      remote-status:
          transport: 10.255.11.201/32
  transport-nexthop: 192.168.168.1
     imposed-labels: 3396,577168
```

PE2

```
juniper@J4300> show l2circuit connections
Layer-2 Circuit Connections:

Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  XX -- unknown

Legend for interface status
Up -- operational
Dn -- down
Neighbor: 10.255.11.31
    Interface                 Type  St     Time last up          # Up trans
    fe-0/0/1.1(vc 5)          rmt   Up     Apr 19 12:28:30 2012           2
      Remote PE: 10.255.11.31, Negotiated control-word: No
      Incoming label: 577168, Outgoing label: 29
      Local interface: fe-0/0/1.1, Status: Up, Encapsulation: VLAN

juniper@J4300>
```

BGP Based VPN

Lets consider that we have the same network layout as in LDP based lab and lets assume that IP connectivity, OSPF and LDP are already set up.

In this case we will not use vlans.

First thing to do is to set up BGP peers and then we can add L2VPN configuration.

Adjust MTUs

On RouterOS we do not adjust L2MTU values since I am using RouterBoards that has L2MTU set to 1632 by default. Only MPLS MTU is adjusted.

PE1 (RouterOS):

```
/mpls interface set 0 mpls-mtu=1526
```

P (RouterOS):

```
/mpls interface set 0 mpls-mtu=1526
```

On Juniper router we will adjust L2MTU to 1600 and MPLS MTU to 1526 on interface running MPLS. We will also set up L2MTU to 1514 on cross circuit interface and set encapsulation to ethernet.

PE2 (JunOS):

```
interfaces {
    fe-0/0/0 {
        mtu 1600;
        unit 0 {
            family inet {
                mtu 1500;
                address 10.0.11.201/24;
            }
            family mpls {
                mtu 1526;
            }
        }
    }
    fe-0/0/1 {
        mtu 1514;
        encapsulation ethernet-ccc;
        unit 0 {
            family ccc;
        }
    }
}
```

Set up BGP

PE1 (RouterOS):

```
/routing bgp instance
  set default as=64201 router-id=10.255.11.31
/routing bgp peer
  add address-families=l2vpn name=juniper remote-address=10.255.11.201 \
    remote-as=64201 ttl=default
```

PE2 (JunOS):

```
routing-options {
    router-id 10.255.11.201;
    autonomous-system 64201;
}

protocol {
    bgp {
        log-updown;
        group int {
            type internal;
            local-address 10.255.11.201;
            import match-all;
            family l2vpn {
                signaling;
            }
            export match-all;
```

```
            neighbor 10.255.11.31;
        }
    }
}

policy-options {
    policy-statement match-all {
        term acceptable {
            then accept;
        }
    }
}
```

Set up L2VPN

PE1 (RouterOS):

```
/interface bridge
  add ame=vpn
/interface bridge port
  add interface=ether5 bridge=vpn

/interface vpls bgp-vpls
  add bridge=vpn bridge-cost=0 export-route-targets=1:1 \
    import-route-targets=1:1 name=juniper-l2vpn pw-type=tagged-ethernet \
    route-distinguisher=1:1 site-id=20 use-control-word=no
```

Note: Parameter **pw-type** is available starting from v5.16. It allows to choose advertised encapsulation in NLRI used only for comparison. Available options are raw-ethernet (5), tagged-ethernet (4) and vpls (19) which is default setting and was hard coded in previous versions.

PE2 (JunOS):

At first we define what is allowed to import and export by routing instance:

```
policy-options {
    policy-statement vpn-SPA-export {
        term a {
            then {
                community add SPA-com;
                accept;
            }
        }
        term b {
            then reject;
        }
    }
    policy-statement vpn-SPA-import {
        term a {
            from {
                protocol bgp;
```

```
                community SPA-com;
            }
            then accept;
        }
        term b {
            then reject;
        }
    }
    community SPA-com members target:1:1;
}
```

Now we can add L2VPN routing instance:

```
routing-instances {
    vpls1 {
        instance-type l2vpn;
        interface fe-0/0/1.0;
        route-distinguisher 1:1;
        vrf-import [ match-all vpn-SPA-import ];
        vrf-export vpn-SPA-export;
        protocols {
            l2vpn {
                traceoptions {
                    file VPLS-TEST size 100000 files 7;
                    flag all;
                }
                encapsulation-type ethernet;
                no-control-word;
                site c2 {
                    site-identifier 21;
                    interface fe-0/0/1.0 {
                        remote-site-id 20;
                    }
                }
            }
        }
    }
}
```

Note: By setting **encapsulation-type** (pw-type on RouterOS). Does not change actual encapsulation. It is also possible that configured encapsulation types do not match on both ends. In this case you can use **ignore-encapsulation-mismatch** on Juniper routers.

In this configuration we also have disabled Cotrol Word usage with **no-control-word** on JunOS and **use-control-word=no** on RouterOS.

Verify Operation

Verify if BGP peer is up

PE1 (RouterOS):

```
[admin@10.0.11.31] /routing bgp peer> print status
Flags: X - disabled, E - established
 0 E name="juniper" instance=default remote-address=10.255.11.201
     remote-as=64201 tcp-md5-key="" nexthop-choice=default multihop=no
     route-reflect=no hold-time=3m ttl=default in-filter="" out-filter=""
     address-families=l2vpn default-originate=never remove-private-as=no
     as-override=no passive=no use-bfd=no remote-id=10.255.11.201
     local-address=10.255.11.31 uptime=1h1m26s prefix-count=0 updates-sent=1
     updates-received=1 withdrawn-sent=0 withdrawn-received=0
     remote-hold-time=1m30s used-hold-time=1m30s used-keepalive-time=30s
     refresh-capability=yes as4-capability=yes state=established
```

PE2 (JunOS):

```
juniper@J4300> show bgp summary
Groups: 3 Peers: 4 Down peers: 3
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0                 0          0          0          0          0          0
bgp.l2vpn.0            1          1          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/Rec...
10.255.11.31          64201        148        152       0       2       59:56 Establ
  bgp.l2vpn.0: 1/1/1/0
  vpls1.l2vpn.0: 1/1/1/0
```

Verify if L2VPN tunnel is created

PE1 (RouterOS):

```
[admin@10.0.11.31] /interface vpls> print
Flags: X - disabled, R - running, D - dynamic,
B - bgp-signaled, C - cisco-bgp-signaled
 0 RDB name="vpls2" mtu=1500 l2mtu=1500 mac-address=02:04:3F:CD:06:97
       arp=enabled disable-running-check=no remote-peer=10.255.11.201
       cisco-style=no cisco-style-id=0 advertised-l2mtu=1500
       pw-type=raw-ethernet vpls=juniper-l2vpn


[admin@10.0.11.31] /interface vpls> monitor 0
       remote-label: 800021
        local-label: 27
      remote-status:
          transport: 10.255.11.201/32
  transport-nexthop: 10.0.11.201
     imposed-labels: 800021
```

PE2 (JunOS):

```
juniper@J4300> show l2vpn connections extensive
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not availble
BK -- Backup connection          ST -- Standby connection

Legend for interface status
Up -- operational
Dn -- down

Instance: vpls1
  Local site: c2 (21)
    Number of local interfaces: 1
    Number of local interfaces up: 1
    fe-0/0/1.1          20
    Label-base        Offset     Range     Preference
    800020            19         2         100
      status-vector: 80
    connection-site           Type  St     Time last up          # Up trans
    20                        rmt   Up     Apr 24 07:30:50 2012           1
      Remote PE: 10.255.11.31, Negotiated control-word: No
      Incoming label: 800021, Outgoing label: 27
      Local interface: fe-0/0/1.0, Status: Up, Encapsulation: ETHERNET
    Connection History:
        Apr 24 07:30:50 2012  status update timer
        Apr 24 07:30:50 2012  PE route changed
        Apr 24 07:30:50 2012  Out lbl Update                       27
        Apr 24 07:30:50 2012  In lbl Update                    800021
        Apr 24 07:30:50 2012  loc intf up                  fe-0/0/1.1

juniper@J4300>
```

See Also

- LDP Based VPLS
- BGP Based VPLS
- Cisco style VPLS
- EoMPLS vs Cisco configuration example

Manual:Routing/BFD

Summary

Bidirectional Forwarding Detection (BFD) is a low-overhead and short-duration protocol intended to detect faults in the bidirectional path between two forwarding engines, including physical interfaces, sub-interfaces, data link(s), and to the extent possible the forwarding engines themselves, with potentially very low latency. It operates independently of media, data protocols and routing protocols.

BFD is basically a hello protocol for checking bidirectional neighbor reachability. It provides sub-second link failure detection support. BFD is not routing protocol specific, unlike protocol hello timers or such.

BFD Control packets is transmitted in UDP packets with destination port 3784. Source port is in the range 49152 through 65535.

Standards and Technologies:

- RFC 5880 Bidirectional Forwarding Detection (BFD)
- RFC 5881 BFD for IPv4 and IPv6 (Single Hop)
- RFC 5882 Generic Application of BFD (Single Hop)
- RFC 5883 BFD for Multihop Paths
- RFC 5884 BFD for MPLS LSPs
- RFC 5885 BFD VCCV

Requirements

RouterOS 4.4 or newer with routing package installed.

Features supported

- asynchronous mode [1]
- BFD timer and detection multiplier configuration per interface;
- enabling BFD for OSPF interfaces
- enabling BFD for BGP peers
- single hop IPv4 and IPv6 transport [2]
- multihop IPv4 and IPv6 transport [3]

Features not yet supported

- echo function
- on-demand mode
- authentication

Configuration

BFD configuration should be added in different places as required

BFD timer configuration

Sub-menu: `/routing bfd interface`

Properties

Property	Description
interface (*string*; Default:)	Interface name to which BFD timers will be applied
interval (*decimal [0.01 .. 10]sec*; Default: **0.2sec**)	Desired rate at which BFD Control packets should be transmitted to the remote system.
min-rx (*decimal [0.01..10]sec*; Default: **0.2sec**)	Min interval desired between received BFD packets
multiplier (*integer [1..100]*; Default: **5**)	The negotiated Control packet transmission interval, multiplied by this variable, will be the Detection Time for the session.

BFD neighbor status

Sub-menu: `/routing bfd neighbor`

Read-only properties

Property	Description
actual-tx-interval (*decimal*)	Actual rate at which BFD Control packets are transmitted
address (*IP* | *IPv6*)	IP/IPv6 address of the neighbor
desired-tx-interval (*decimal*)	The minimum interval between transmitted BFD Control packets that this system would like to use.
hold-time (*time*)	
interface (*string*)	interface name on which BFD neighbor is reachable
multihop (*yes* | *no*)	Whether neighbor is multiple hops away
multiplier (*integer*)	Desired Detection Time multiplier for BFD Control packets on the local system
packets-rx (*integer*)	Number of received packets
protocols (*ospf* | *bgp*)	For which protocols BFD is used, currently only OSPF and BGP are possible.
remote-min-rx (*decimal*)	The last value of Required Min RX Interval received from the remote system in a BFD Control packet
required-min-rx (*decimal*)	The minimum interval between received BFD Control packets that this system requires.
state (*up* | *down*)	Shows the current BFD session state
state-changes (*integer*)	Number of state changes occurred between the neighbors
up (*yes* | *no*)	Whether link to the neighbor is up
uptime (*time*)	Link uptime

Example of BFD neighbor. Neighbor is used by OSPF and is directly connected.

```
[admin@R3-493G] /routing bfd neighbor> print detail
Flags: U - up
 0 U state=up address=10.5.101.1 interface=ether1 protocols=ospf multihop=no
     state-changes=1 uptime=12s desired-tx-interval=0.2sec
     actual-tx-interval=0.2sec required-min-rx=0.2sec remote-min-rx=0.2sec
     multiplier=5 hold-time=1sec packets-rx=76 packets-tx=77
```

OSPF

There is only one parameter per OSPF interface to enable BFD

```
 /routing ospf interface add interface=all use-bfd=yes
```

BGP

Similar to OSPF, only one option per BGP peer to enable BFD

```
 /routing bgp peer add remote-address=x.x.x.x remote-as=xxxxx use-bfd=yes
```

Interoperability

For interoperability with Cisco make sure to disable echo mode (it is enabled on Cisco by default), since it's not supported on MT.

To do that, on Cisco in interface configuration mode type:

```
no bfd echo
```

[Top | Back to Content **]**

References

[1] http://tools.ietf.org/html/draft-ietf-bfd-base-09/draft-ietf-bfd-base-09.txt
[2] http://tools.ietf.org/html/draft-ietf-bfd-base-09/draft-ietf-bfd-v4v6-1hop-10.txt
[3] http://tools.ietf.org/html/draft-ietf-bfd-base-09/draft-ietf-bfd-multihop-08.txt

Manual:Routing/BGP

Applies to RouterOS: v3, v4 +

Summary

The Border Gateway Protocol (BGP) allows setting up an interdomain dynamic routing system that automatically updates routing tables of devices running BGP in case of network topology changes.
MikroTik RouterOS supports BGP Version 4, as defined in RFC 4271
Standards and Technologies:

- RFC 4271 Border Gateway Protocol 4
- RFC 4456 BGP Route Reflection
- RFC 5065 Autonomous System Confederations for BGP
- RFC 1997 BGP Communities Attribute
- RFC 2385 TCP MD5 Authentication for BGPv4
- RFC 5492 Capabilities Advertisement with BGP-4
- RFC 2918 Route Refresh Capability
- RFC 4760 Multiprotocol Extensions for BGP-4
- RFC 2545 Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing
- RFC 4893 BGP Support for Four-octet AS Number Space

Instance

Sub-menu: `/routing bgp instance`

Property	Description
as (*integer [0..4294967295]*; Default:)	32-bit BGP autonomous system number. Value can be entered in AS-Plain and AS-Dot formats.
client-to-client-reflection (*yes* | *no*; Default: **yes**)	In case this instance is a route reflector: whether to redistribute routes learned from one routing reflection client to other clients.
cluster-id (*IP address*; Default:)	In case this instance is a route reflector: cluster ID of the router reflector cluster this instance belongs to. This attribute helps to recognize routing updates that comes from another route reflector in this cluster and avoid routing information looping. Note that normally there is only one route reflector in a cluster; this case 'cluster-id' does not need to be configured and BGP router ID is used instead
comment (*string*; Default:)	Short description of the instance.
confederation (*integer [0..4294967295]*; Default:)	In case of BGP confederations: autonomous system number that identifies the [local] confederation as a whole.
confederation-peers (*list/range of integer[0..4294967295]*; Default:)	In case of BGP confederations: list of AS numbers internal to the [local] confederation. Range of as numbers are also supported. For example 10,20,30-50.
disabled (*yes* | *no*; Default: **no**)	Whether instance is disabled.
ignore-as-path-len (*yes* | *no*; Default: **no**)	Whether to ignore **AS_PATH** attribute in BGP route selection algorithm
name (*string*; Default:)	BGP instance name
out-filter (*string*; Default:)	Output routing filter chain used by all BGP peers belonging to this instance

redistribute-connected (*yes* | *no*; Default: **no**)	If enabled, this BGP instance will redistribute the information about connected routes, i.e., routes to the networks that can be directly reached.
redistribute-ospf (*yes* | *no*; Default: **no**)	If enabled, this BGP instance will redistribute the information about routes learned by OSPF
redistribute-other-bgp (*yes* | *no*; Default: **no**)	If enabled, this BGP instance will redistribute the information about routes learned by other BGP instances
redistribute-rip (*yes* | *no*; Default: **no**)	If enabled, this BGP instance will redistribute the information about routes learned by RIP
redistribute-static (*yes* | *no*; Default: **no**)	If enabled, the router will redistribute the information about static routes added to its routing database, i.e., routes that have been created using the '/ip route add' command on the router
router-id (*IP*; Default: **0.0.0.0**)	BGP Router ID (for this instance). If set to 0.0.0.0, BGP will use one of router's IP addresses.
routing-table (*string*; Default:)	Name of routing table this BGP instance operates on. Non-default routing-table and list of VRFs cannot be configured for the same instance at the same time. Available starting from v4.3

VRF

Sub-menu: `/routing bgp instance vrf`

Instance related VRF configuration

Property	Description
comment (*string*; Default:)	Short description of the VRF.
disabled (*yes* | *no*; Default: **no**)	
in-filter (*string*; Default:)	Name of the routing filter chain that is applied to the incoming routing information
instance (*string*; Default:)	Name of the instance this configuration applies to.
out-filter (*string*; Default:)	Name of the routing filter chain that is applied to the outgoing routing information
redistribute-connected (*yes* | *no*; Default: **no**)	Redistribute connected routes that belongs to VRF.
redistribute-ospf (*yes* | *no*; Default: **no**)	Redistribute OSPF routes that belongs to VRF.
redistribute-other-bgp (*yes* | *no*; Default: **no**)	Redistribute BGP routes that belongs to VRF received from other BGP instance.
redistribute-rip (*yes* | *no*; Default: **no**)	Redistribute RIP routes that belongs to VRF.
redistribute-static (*yes* | *no*; Default: **no**)	Redistribute static routes that belongs to VRF.
routing-mark (*string*; Default:)	Name of the routing-mark used by VRF configured in */ip route vrf* menu.

Peer

Sub-menu: `/routing bgp peer`

Property	Description
address-families (*ip* | *ipv6* | *l2vpn* | *l2vpn-cisco* | *vpnv4*; Default: **ip**)	List of address families about which this peer will exchange routing information. The remote peer must support (they usually do) BGP capabilities optional parameter to negotiate any other families than IP.
allow-as-in (*integer [0..10]*; Default:)	How many times to allow own AS number in AS-PATH, before discarding a prefix.
as-override (*yes* | *no*; Default: **no**)	If set, then all instances of remote peer's AS number in BGP AS PATH attribute are replaced with local AS number before sending route update to that peer. Happens before routing filters and prepending.
cisco-vpls-nlri-len-fmt (*auto-bits* | *auto-bytes* | *bits* | *bytes*; Default:)	VPLS NLRI length format type. Used for compatibility with Cisco VPLS. Read more>>.
comment (*string*; Default:)	Description of the peer.
default-originate (*always* | *if-installed* | *never*; Default: **never**)	Specifies how to distribute default route
disabled (*yes* | *no*; Default: **no**)	Whether peer is disabled.
hold-time (*time[3s..1h]* | *infinity*; Default: **3m**)	Specifies the BGP Hold Time value to use when negotiating with peers. According to the BGP specification, if router does not receive successive **KEEPALIVE** and/or **UPDATE** and/or **NOTIFICATION** messages within the period specified in the Hold Time field of the **OPEN** message, then the BGP connection to the peer will be closed. The minimal hold-time value of both peers will be actually used (note that the special value 0 or 'infinity' is lower than any other values) • `infinity` - never expire the connection and never send keepalive messages.
in-filter (*string*; Default:)	Name of the routing filter chain that is applied to the incoming routing information
instance (*string*; Default: **default**)	Name of the instance this peer belongs to.
keepalive-time (*time [1s..30m]*; Default:)	
max-prefix-limit (*integer [0..4294967295]*; Default:)	Maximum number of prefixes to accept from a specific peer. When this limit is exceeded, TCP connection between peers is closed.
max-prefix-restart-time (*time [1m..1w3d]* | *infinity*; Default:)	Minimum time interval after which peers can reestablish BGP session. • `infinity` - session is not reestablished until administrator's intervention.
multihop (*yes* | *no*; Default: **no**)	Specifies whether the remote peer is more than one hop away. This option affects outgoing nexthop selection as described in RFC 4271 (for EBGP only, excluding EBGP peers local to the confederation). It also affects: • whether to accept connections from peers that are not in the same network (the remote address of the connection is used for this check); • whether to accept incoming routes with NEXT_HOP attribute that is not in the same network as the address used to establish the connection; • the target-scope of the routes installed from this peer; routes from multi-hop or IBGP peers resolve their nexthops through IGP routes by default.
name (*string*; Default:)	Descriptive name of the peer
nexthop-choice (*default* | *force-self* | *propagate*; Default: **default**)	Affects the outgoing **NEXT_HOP** attribute selection. Note that nexthops set in filters always takes precedence. Also note that nexthop is not changed on route reflection, expect when it's set in filter. • `default` - select the nexthop as described in RFC 4271 • `force-self` - always use a local address of the interface that used to connect to the peer as the nexthop; • `propagate` - try to propagate further the nexthop received; i.e. if the route has BGP **NEXT_HOP** attribute, then use it as the nexthop, otherwise fall back to the default case

out-filter (*string*; Default:)	Name of the routing filter chain that is applied to the outgoing routing information. If instance has also configured out-filter, then instance filters are applied firs and only then peer's filters.
passive (*yes* | *no*; Default: **no**)	If set to **yes**, then connection attempts to remote peer are not made. The remote peer must initialize connection in this case. Available starting from v4.3
remote-address (*IP/IPv6 address*; Default:)	Address of the remote peer. If remote address is IPv6 link-local address then interface must be specified after '%', for example, fe80::21a:4dff:fe5d:8e56**%ether1**
remote-as (*integer [0..4294967295]*; Default:)	32-bit AS number of the remote peer. AS number can be specified in AS-Plain and AS-Dot formats.
remote-port (*integer [0..65535]*; Default:)	Remote peers port to establish tcp session. If not set, then default 179 port will be used.
remove-private-as (*yes* | *no*; Default: **no**)	If set, then BGP AS-PATH attribute is removed before sending out route update if attribute contains only private AS numbers. removal process happens before routing filters are applied and before local AS number is prepended to the AS path. Option is available starting from v4.3.
route-reflect (*yes* | *no*; Default: **no**)	Specifies whether this peer is route reflection client.
tcp-md5-key (*string*; Default:)	Key used to authenticate the connection with TCP MD5 signature as described in RFC 2385. If not specified, authentication is not used.
ttl (*integer [1..255]* | *default*; Default: **default**)	Time To Live, the hop limit for TCP connection. For example, if 'ttl=1' then only single hop neighbors will be able to establish the connection. This property only affects EBGP peers. • `default` - system's default TTL value is used
update-source (*IPv4* | *IPv6* | *Interface* | *none*; Default:)	If address is specified, this address is used as the source address of the outgoing TCP connection. If interface name is specified, an address belonging to the interface is used as described. This property is ignored, if the value specified is not a valid address of the router or name an interface with active addresses. Do not specify name of interface that is added as a bridge port here!
use-bfd (*yes* | *no*; Default: **no**)	Whether to use BFD protocol for fast state detection.

Read only status properties:

Property	Description
as4-capability (*yes* | *no*)	Shows whether peer has 4-byte AS support
established (*yes* | *no*)	Set to yes if BGP peering is established.
local-address (*IP* | *IPv6*)	Address that is used as source address of BGP packets.
prefix-count (*integer*)	Number of routing prefixes received from this peer currently in routing table.
refresh-capability (*yes* | *no*)	Whether **route refresh** is supported by the peer
remote-hold-time (*time*)	Hold time set on remote peer.
remote-id (*IP*)	Remote peer's instance ID.
state (*idle* | *connect* | *active* | *opensent* | *openconfirm* | *established*)	BGP protocol state.
updates-received (*integer*)	Total number of reachable routing prefixes received
updates-sent (*integer*)	Total number of reachable routing prefixes sent
uptime (*time*)	Shows how long BGP has established state.
used-hold-time (*time*)	Negotiated and used hold time on both peers
used-keepalive-time (*time*)	Negotiated and used keepalive time on both peers (used-hold-time / 3)
withdraws-received (*integer*)	Total number of withdrawn routing prefixes received.
withdraws-sent (*integer*)	Total number of withdrawn routing prefixes advertised

Advertisements

Sub-menu: `/routing bgp advertisements`

Read only information about outgoing routing information currently advertised.

This information is calculated dynamically after 'print' command is issued. As a result, it may not correspond to the information that at the exact moment has been sent out. Especially if in case of slow connection, routing information prepared for output will spend long time in buffers. 'advertisements print' will show as things should be, not as they are!

Note: At the moment AS-PATH attribute for advertised routes is shown without prepends.

Property	Description
aggregator (*IP*)	Advertised **AGGREGATOR** attribute value
as-path (*string*)	Advertised **AS_PATH** attribute value
atomic-aggregate (*yes* | *no*)	Advertised **ATOMIC_AGGREGATE** attribute value
bgp-ext-communities ()	
cluster-list (*string*)	Advertised **CLUSTER_LIST** attribute value
communities ()	
local-pref (*integer*)	Advertised **LOCAL_PREF** attribute value
med (*integer*)	Advertised **MULTI_EXIT_DISC** attribute value
nexthop (*IP* | *IPv6*)	Advertised **NEXT_HOP** attribute value
origin (*igp* | *egp* | *incomplete*)	Advertised **ORIGIN** attribute value
originator-id (*IP*)	Advertised **ORIGINATOR_ID** attribute value
peer (*string*)	Name of the peer this information is advertised to
prefix (*IPv4* | *IPv6 prefix*)	Advertised NLRI prefix

Network

Sub-menu: `/routing bgp network`

BGP network configuration. BGP Networks is a list of IP prefixes to be advertised.

Property	Description
network (*IP prefix*;)	the aggregate prefix
synchronize (*yes* | *no*; Default: **no**)	install a route for this network only when there is an active IGP route matching this network

Aggregate

Sub-menu: `/routing bgp aggregate`

BGP allows the aggregation of specific routes into one route with. This menu ('/routing bgp aggregate') allows to specify which routes you want to aggregate, and what attributes to use for the route created by aggregation.

Property	Description
advertise-filter (*string*;)	name of the filter chain used to select the routes from which to inherit attributes
attribute-filter (*string*;)	name of the filter chain used to set the attributes of the aggregate route
include-igp (*yes* | *no*; Default:)	By default, BGP aggregate takes into account only BGP routes. Use this option to take IGP and connected routes into consideration.
inherit-attributes (*yes* | *no*; Default: **yes**)	whether to inherit BGP attributes from aggregated routes
instance (*string*;)	the instance this network belongs to
prefix (*IP prefix*;)	the aggregate prefix
summary-only (*yes* | *no*; Default: **yes**)	whether to suppress advertisements of all routes that fall within the range of this aggregate
suppress-filter (*string*;)	name of the filter chain used to select the routes to be suppressed

Read only status property:

routes-used (*integer*) aggregated route statistics.

- in console- list of route console IDs used;
- in winbox- number of routes used.

Terminology

- `aggregated routes` - all routes, that fall within the range of this aggregate; they possibly are suppressed;
- `aggregate route` - route created by aggregation.

Note: Each aggregate will only affect routes coming from peers that belong to it's instance. *suppress-filter* is useful only if *summary-only=no*; *advertise-filter* is useful only if *inherit-attributes=yes*.

If result *attribute-filter* match **reject** or **discard**, the aggregate route is not created.

Vpnv4 route

Sub-menu: `/routing bgp vpnv4-route`

Read only information about vpnv4 routing information currently advertised.

Property	Description
bgp-as-path (*string*;)	the **AS_PATH** attribute value
bgp-atomic-aggregate (*string*;)	the **ATOMIC_AGGREGATE** attribute value
bgp-communities (;)	
bgp-ext-communities (*string*;)	
bgp-local-pref (*string*;)	the **LOCAL_PREF** attribute value
bgp-med (*string*;)	the **MULTI_EXIT_DISC** attribute value
bgp-origin (*igp|egp|incomplete*;)	the **ORIGIN** attribute value
bgp-prepend (*string*;)	
bgp-weight (*string*;)	
dst-address (*string*;)	
gateway (*string*;)	
in-label (*integer*;)	assigned MPLS in label
interface (*string*;)	
out-label (*integer*;)	assigned MPLS out label
route-distinguisher (*string*;)	

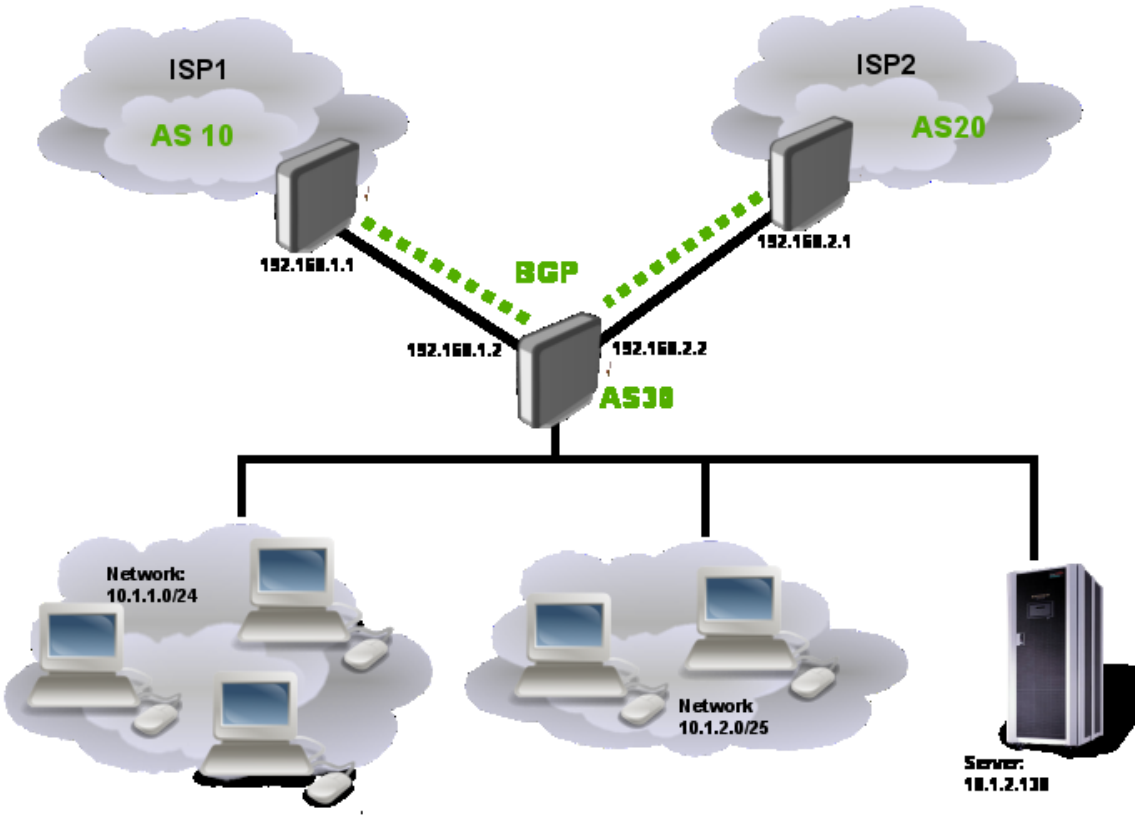
[Top | Back to Content]

Manual:Simple BGP Multihoming

Applies to RouterOS: all

Setup

Ilustration below shows simple multihomed BGP setup. This setup can be used for load sharing between ISPs or one ISP as main and other ISP as backup link.



Lets say that local Internet registry assigned to us two /24 networks: **10.1.1.0/24** and **10.1.2.0/24** and our AS is 30 (Private AS cannot be used in such setups). First network entirely is used for workstations in our corporate network. Part of the other network is also used for workstation and another part is reserved for server. At this point our company has only one server with address 10.1.2.130

The goal is advertise our assigned networks to BGP peers and use only one provider as main link, ISP2 link is for backup only.

Note: This example does not show how to provide connectivity between core router, local networks and servers

BGP Peering

Consider that IP connectivity between ISPs edge routers and Our Core router is already set up and working properly. So we can start to establish BGP peering to both ISPs.

```
#set our AS number
/routing bgp instance
set default as=30
```

```
#add BGP peers
/routing bgp peer
add name=toISP1 remote-address=192.168.1.1 remote-as=10
add name=toISP2 remote-address=192.168.2.1 remote-as=20
```

If everything is set up properly, peer should have E (established) flag and router should receive bunch of BGP routes from both ISPs

```
[admin@RB1100test] /routing bgp peer> print
Flags: X - disabled, E - established
 #   INSTANCE        REMOTE-ADDRESS                                  REMOTE-AS
 0 E default         192.168.1.1                                     10
 1 E default         192.168.1.2                                     20
```

Network Advertisements and Routing Filters

Now we can start to advertise our networks and filter out all other unnecessary advertisements.

First step is to advertise our networks

```
/routing bgp network
add network=10.1.1.0/24 synchronize=no
add network=10.1.2.0/24 synchronize=no
```

Next step is to specify which routing filter chains will be used

```
/routing bgp peer
set isp1 in-filter=isp1-in out-filter=isp1-out
set isp2 in-filter=isp2-in out-filter=isp2-out
```

in-filter is for incoming (received) prefixes, *out-filter* is for advertised prefixes.

Main/Backup link setup

After chains are specified we can accept our networks and drop everything else as we are not **transit provider**. As you know one of the BGP attributes that influence best path selection is AS Path length (shorter AS Path is more preferred). So as we want ISP2 to be backup only, we will use *BGP AS prepend* (increase length of AS path) to force incoming traffic through ISP1.

Outgoing filters to ISP1:

```
/routing filter
#accept our networks
add chain=isp1-out prefix=10.1.1.0/24 action=accept
add chain=isp1-out prefix=10.1.2.0/24 action=accept
#discard the rest
add chain=isp1-out action=discard
```

Outgoing filters to ISP2:

```
/routing filter
#accept our networks and prepend AS path three times
add chain=isp2-out prefix=10.1.1.0/24 action=accept set-bgp-prepend=3
add chain=isp2-out prefix=10.1.2.0/24 action=accept set-bgp-prepend=3
#discard the rest
```

```
add chain=isp2-out action=discard
```

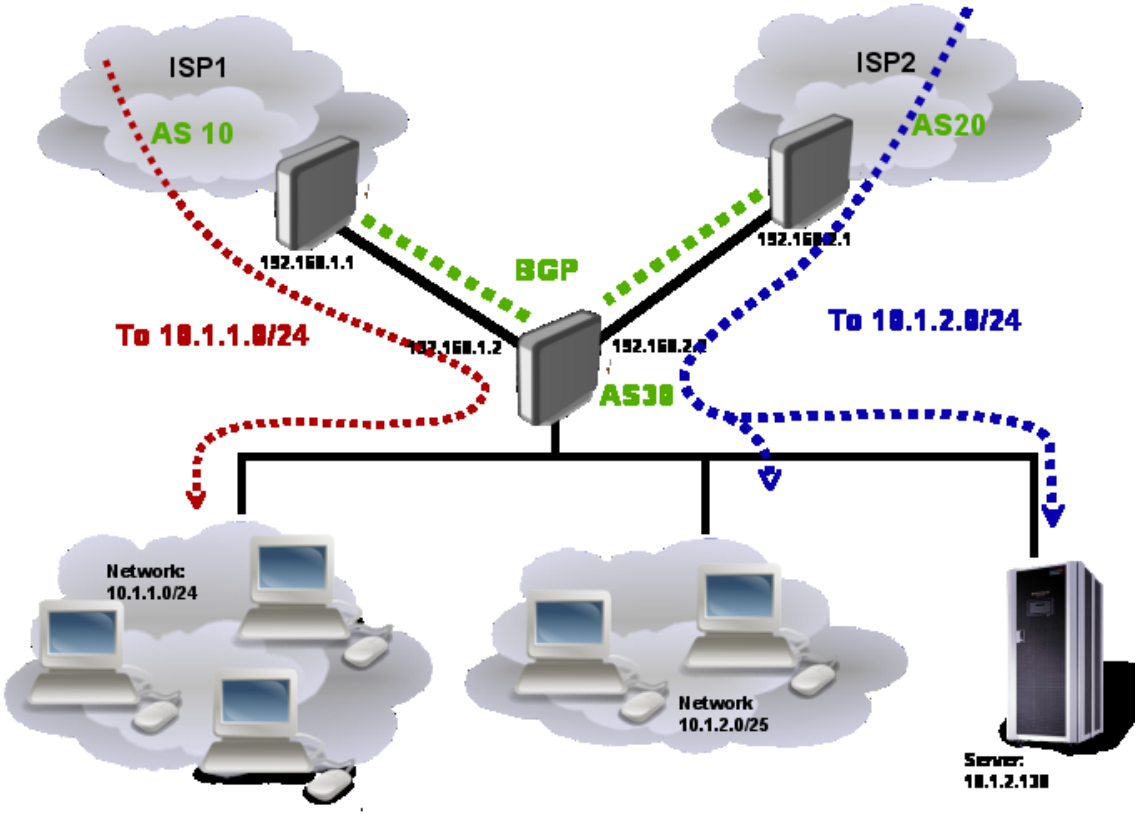
We also do not need any routes from both ISPs, because default route is used to force outgoing traffic through ISP1 and leave ISP2 as backup.

```
/routing filter
add chain=isp1-in action=discard
add chain=isp2-in action=discard
```

```
/ip route
add gateway=192.168.1.1 check-gateway=ping
add gateway=192.168.2.1 distance=30 check-gateway=ping
```

Load sharing setup

Using previous setup we are kind of wasting one link. So it is possible to redesign our setup as illustrated below to utilize both links.



The same as in previous setup BGP AS prepend will be used to achieve our goal. This time we will advertise one of the netowrks to ISP1 without prepend and another network prepended three times. The opposite for ISP2.

Outgoing filters to ISP1:

```
/routing filter
#accept our networks and prepend second network
add chain=isp1-out prefix=10.1.1.0/24 action=accept
add chain=isp1-out prefix=10.1.2.0/24 action=accept  set-bgp-prepend=3
#discard the rest
add chain=isp1-out action=discard
```

Outgoing filters to ISP2:

```
/routing filter
#accept our networks and prepend first network
add chain=isp2-out prefix=10.1.1.0/24 action=accept set-bgp-prepend=3
add chain=isp2-out prefix=10.1.2.0/24 action=accept
#discard the rest
add chain=isp2-out action=discard
```

Configuration above is only for packets going to our network. There are several options how to deal with packets going from our network:

- leave gateways as in main/backup configuration - this will result in only one link utilized and asymmetric routing
- use policy routing to force outgoing packets over the same link as incoming
- use BGP to receive full routing tables from both peers and using BGP attributes make part of the routes available through one link and other part through another link. For example, traffic local to your country is sent over ISP1 the rest is sent over ISP2.

All those methods are covered in other articles and will not be shown here.

[Top | Back to Content]

Manual:BCP bridging (PPP tunnel bridging)

Applies to RouterOS: v3, v4

Summary

RouterOS supports BCP (Bridge Control Protocol) for PPP, PPTP, L2TP and PPPoE interfaces. BCP allows to bridge Ethernet packets through the PPP link. Established BCP is independent part of the PPP tunnel, it is not related to any IP address of PPP interface, bridging and routing can happen at the same time independently. BCP can be used instead of EoIP + used VPN Tunnel or WDS link over the wireless network.

Requirements

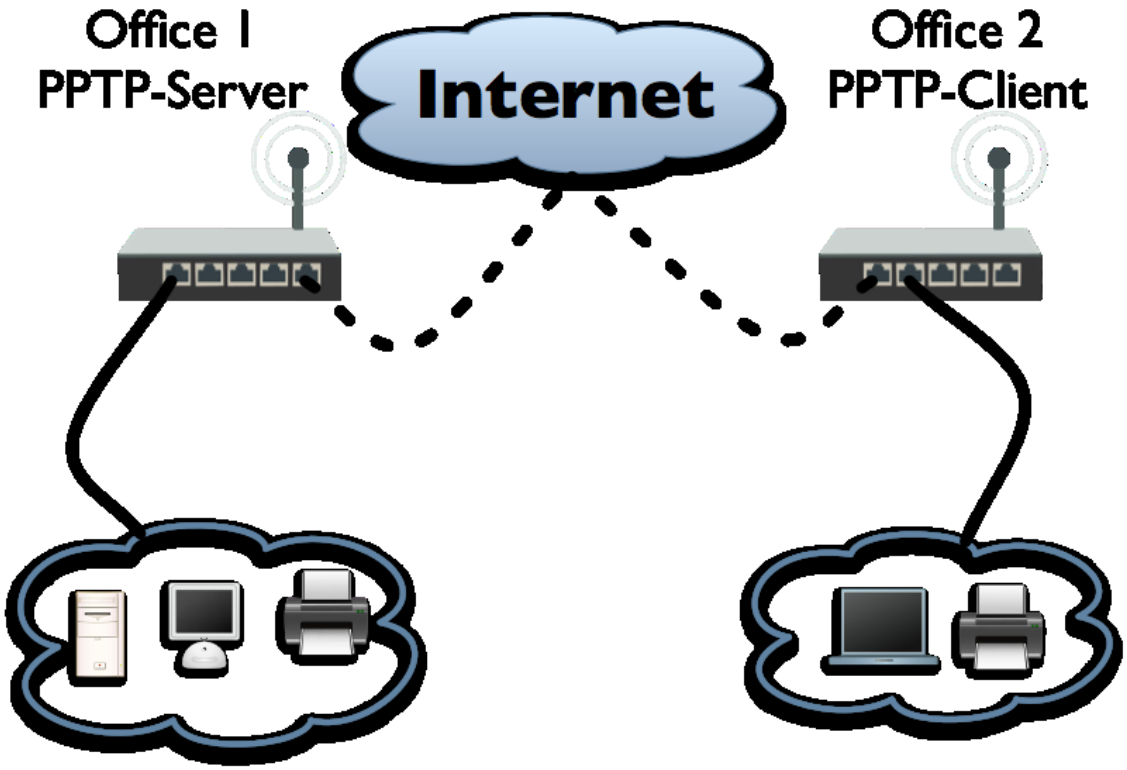
BCP (Bridge Control Protocol) should be enabled on both sides (PPP server and PPP client) to make it work. MikroTik RouterOS can be used with other PPP device, that supports BCP accordingly to the standards, but BCP enabled is necessary.

Configuration Example

We need to interconnect two remote offices and make them in one Ethernet network. We have requirement to use encryption to protect data exchange between two offices. Let's see, how it is possible with PPTP tunnel and BCP protocol usage

Configuration Diagramm

Simple configuration is like this. We have two offices, which are remotely located. Office I is going to be used as PPTP server, Office 2 is going to be used PPTP client. Below you will see how to set configuration using Winbox and CLI.



BCP Configuration (CLI)

Office 1 configuration

First we need to create bridge interface and make sure that bridge will always have MAC address of existing interface. Reason for that is simple - when BCP is used PPP bridge port do not have any MAC address.

```
/interface bridge add name=bridge_local protocol-mode=rstp
/interface bridge port add bridge=bridge_local interface=ether1_local
/interface bridge set bridge_local admin-mac=xx:xx:xx:xx:xx:xx
//// where xx:xx:xx:xx:xx:xx is MAC address of the ether1_local interface
```

Now we can assign local and public addresses to proper interfaces.

```
/ip address add address=192.168.88.1/24 interface=bridge_local
/ip address add address=1.1.1.1/24 interface=ether2_public
```

In case you use PPP only for bridging, configuration of the ppp profile and secret is very easy - just assign user name and password in secret) and specify bridge option in the profile. Even if PPP is bridged local and remote addresses on server side must be specified.

```
/ppp profile add name=ppp_bridging bridge=bridge_local use-encryption=yes
/ppp secret add profile=ppp_bridging name=ppp1 password=ppp1
```

When bridging packets PPP tunnel need to pass packets with Layer-2 (MAC) header included , so default interface MTU (in case of pptp it is 1460) is not sufficient for this task. To ensure proper operation itis suggested to override the value by specifying MRRU option in server settings to a higher value.

MRRU allows to enable multi-link support over single link, it divides the packet to multiple channels therefore increasing possible MTU and MRU (up to 65535 bytes)

```
/interface pptp-server server set enabled=yes mrru=1600
```

Office 2 configuration

First we need to create bridge interface and make sure that bridge will always have MAC address of existing interface. Reason for that is simple - when BCP is used PPP bridge port do not have any MAC address.

```
/interface bridge add name=bridge_local protocol-mode=rstp
/interface bridge port add bridge=bridge_local interface=ether1_local
/interface bridge set bridge_local admin-mac=xx:xx:xx:xx:xx:xx
//// where xx:xx:xx:xx:xx:xx is MAC address of the ether1_local interface
```

```
Assign local and public addresses to proper interfaces.
```

```
/ip address add address=192.168.88.254/24 interface=bridge_local
/ip address add address=2.2.2.2/24 interface=ether2_public
```

Configure ppp profile so it will corespond to the profile used on the server side.

```
/ppp profile add name=ppp_bridging bridge=bridge_local use-encryption=yes
```

Create an pptp-client interface. Do not forget to specify MRRU option to ensure that bridged frames get trough the ppp tunnel.

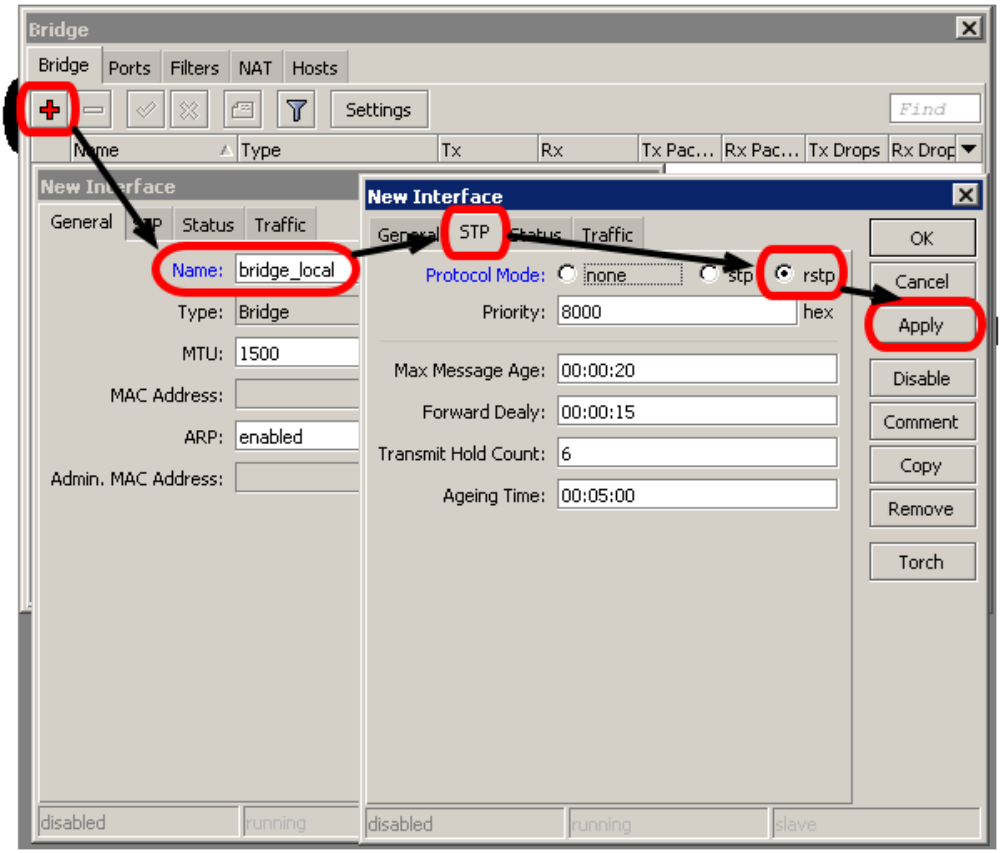
```
/interface pptp-client
add profile=ppp_bridging mrru=1600 connect-to=1.1.1.1 user=ppp1 password=ppp1 disabled=no
```

BCP Configuration (Winbox)

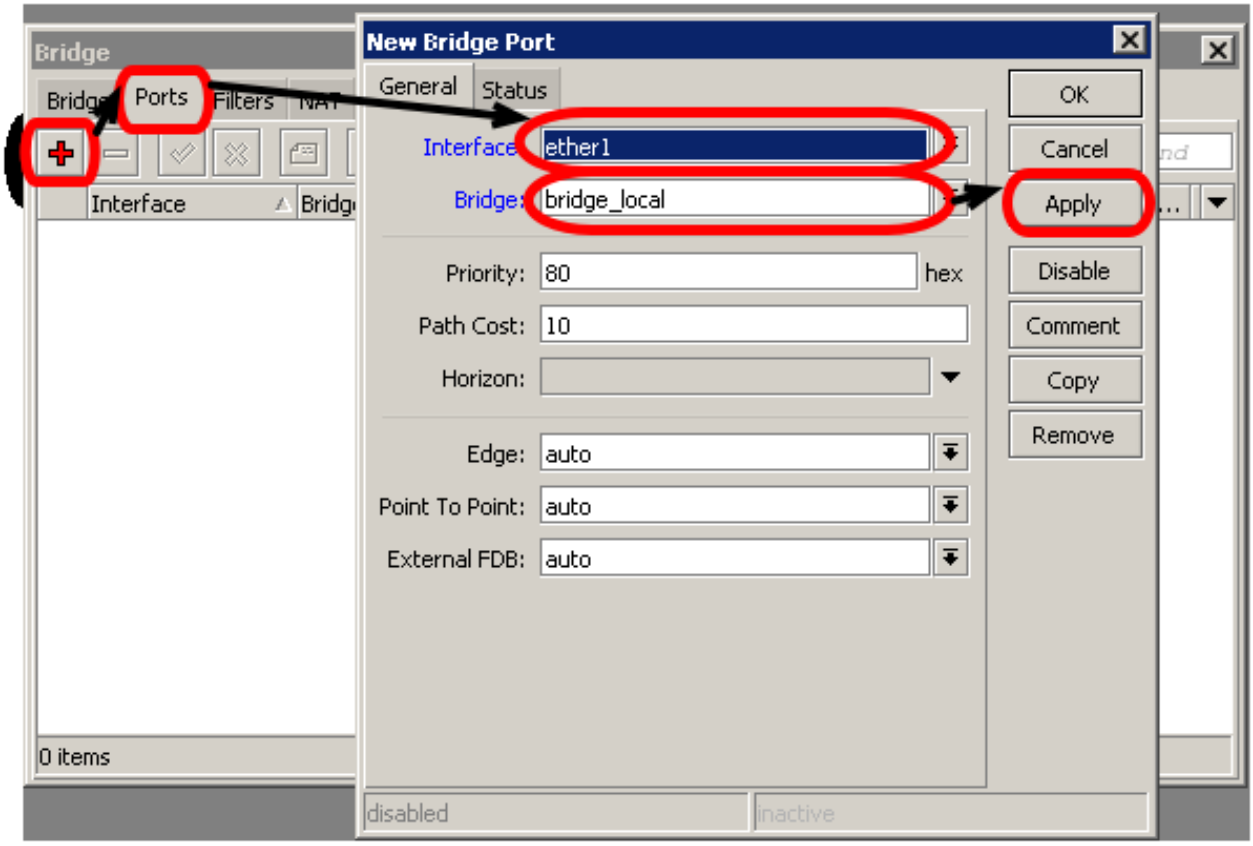
Office 1 Configuration

Bridge Configuration:

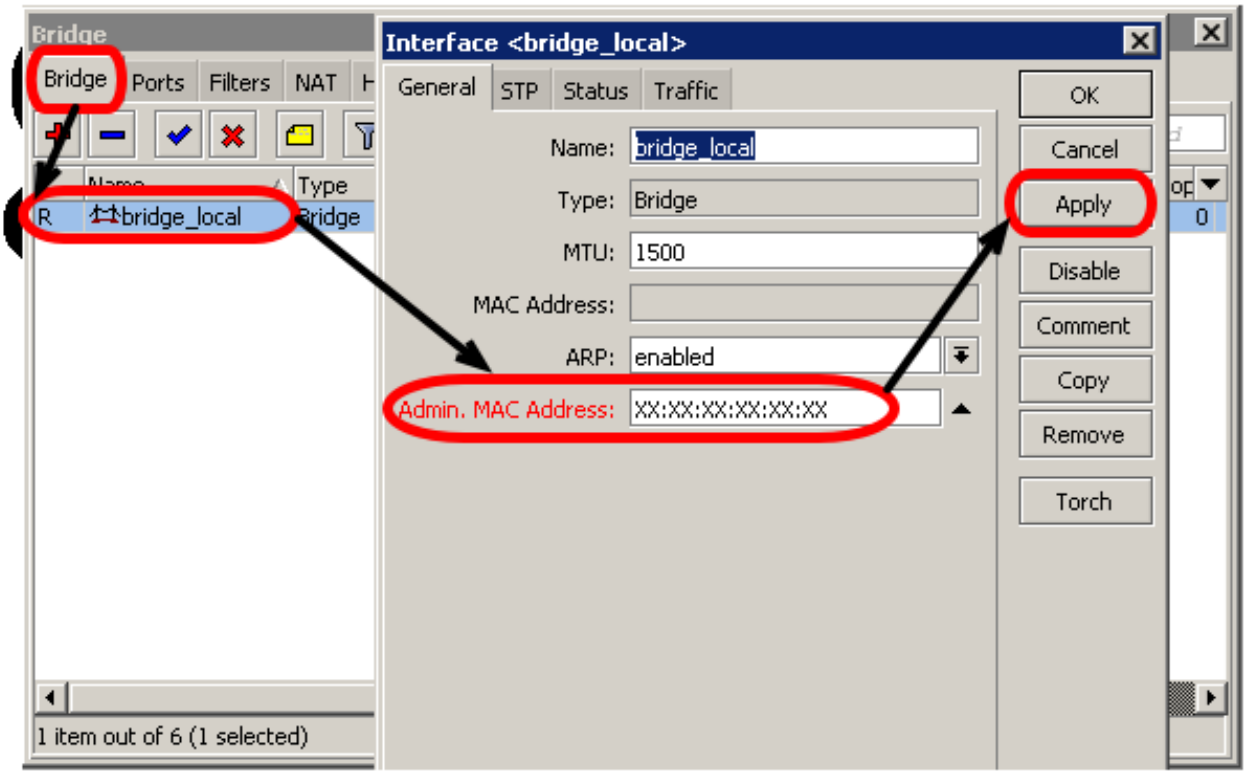
- Add **Bridge**,



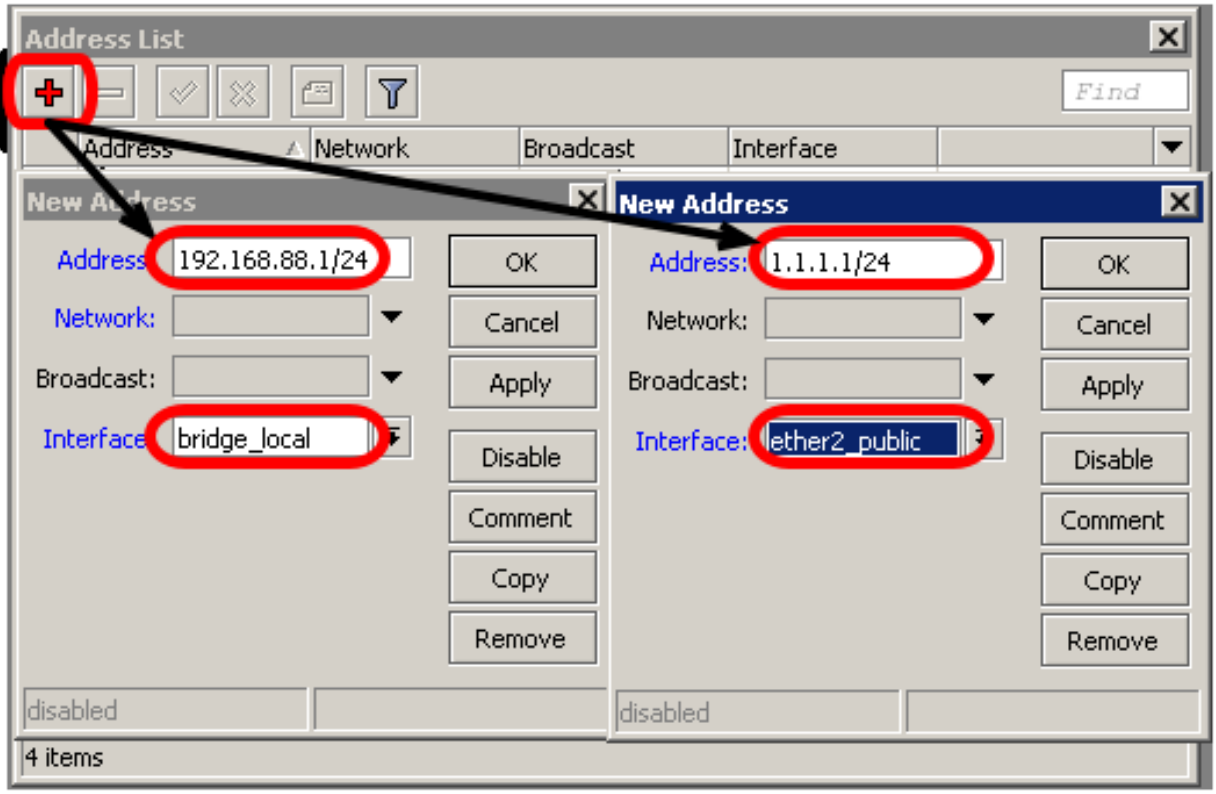
- Add Bridge **Port**,



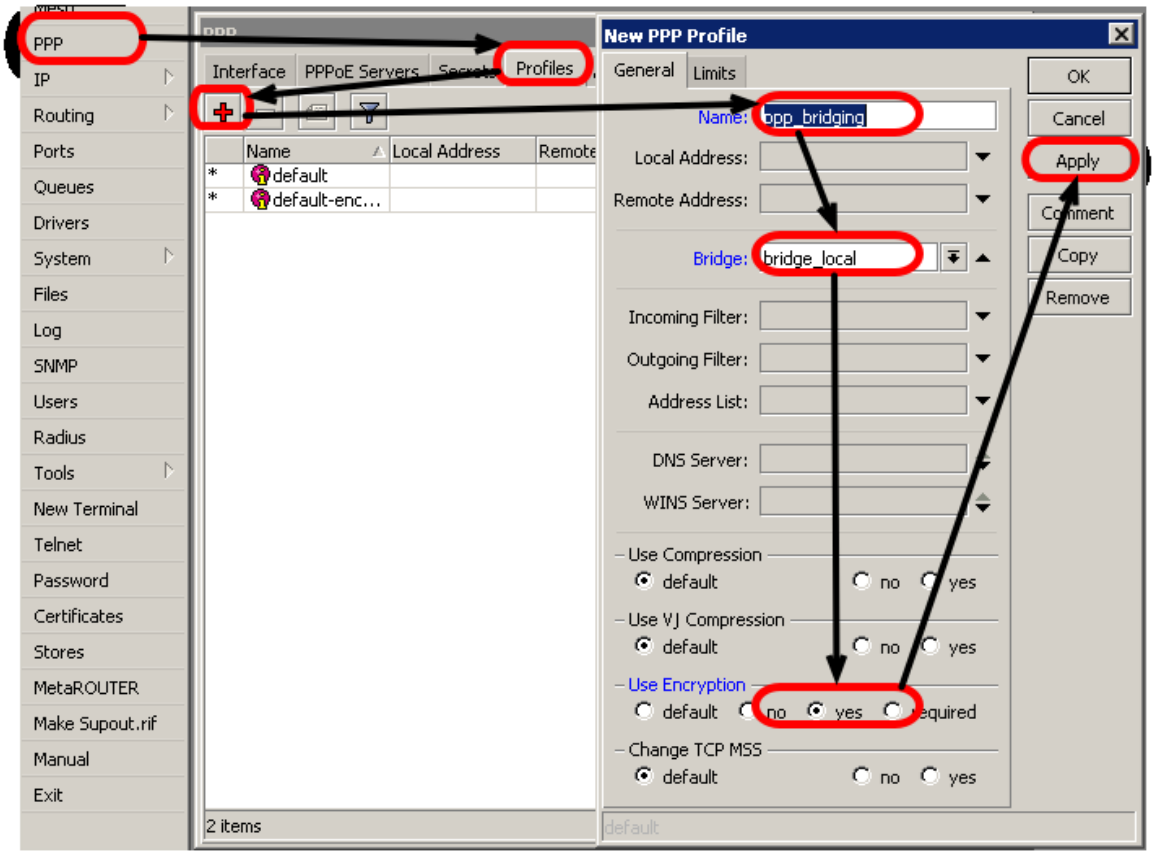
- Add Bridge **MAC-address**,



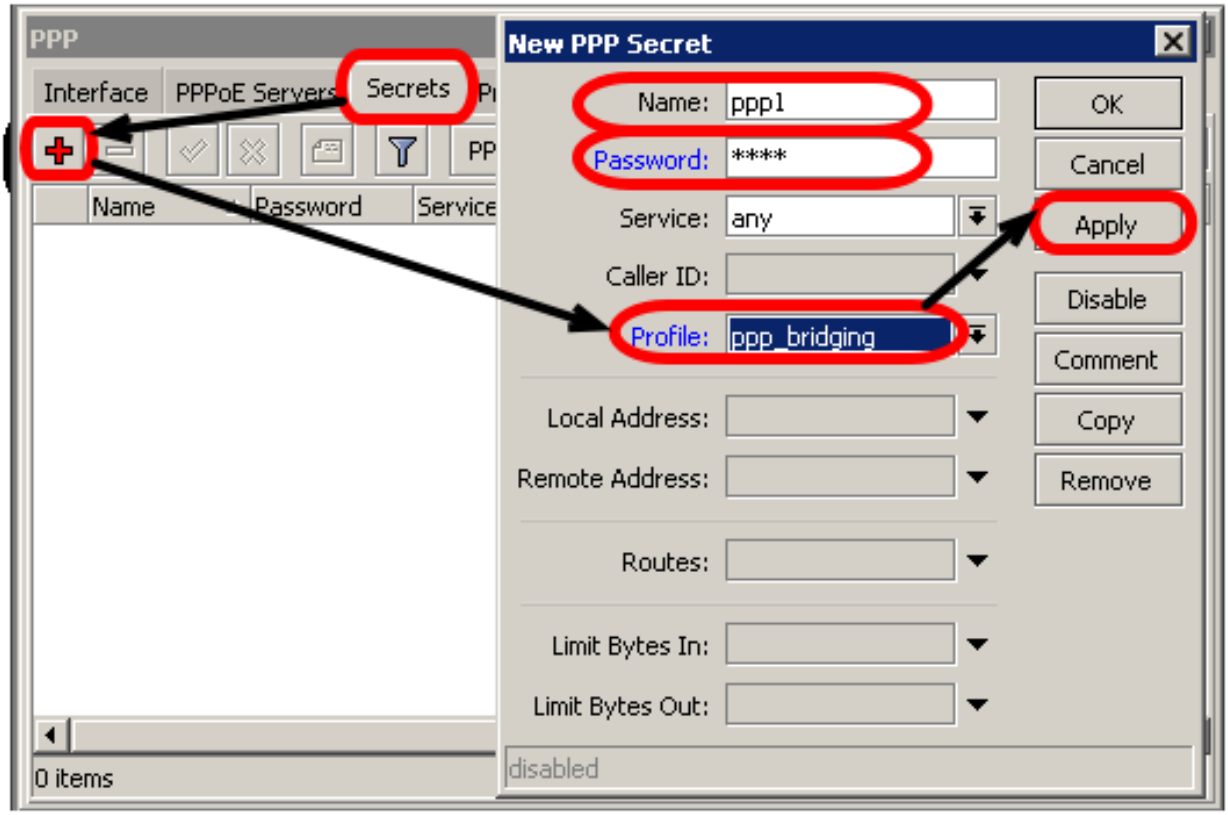
- Assign **IP addresses**,



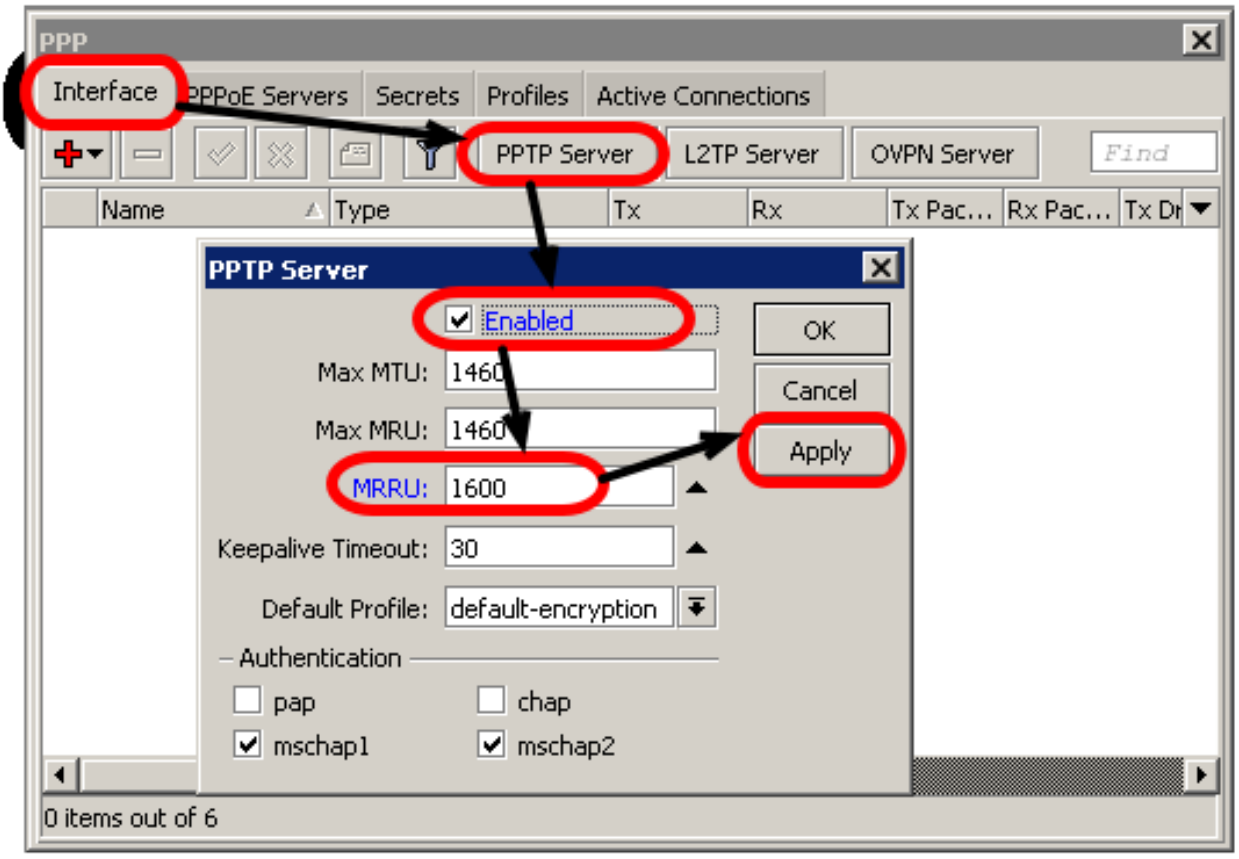
- Create PPP **profile** for bridging,



- Add PPP client,



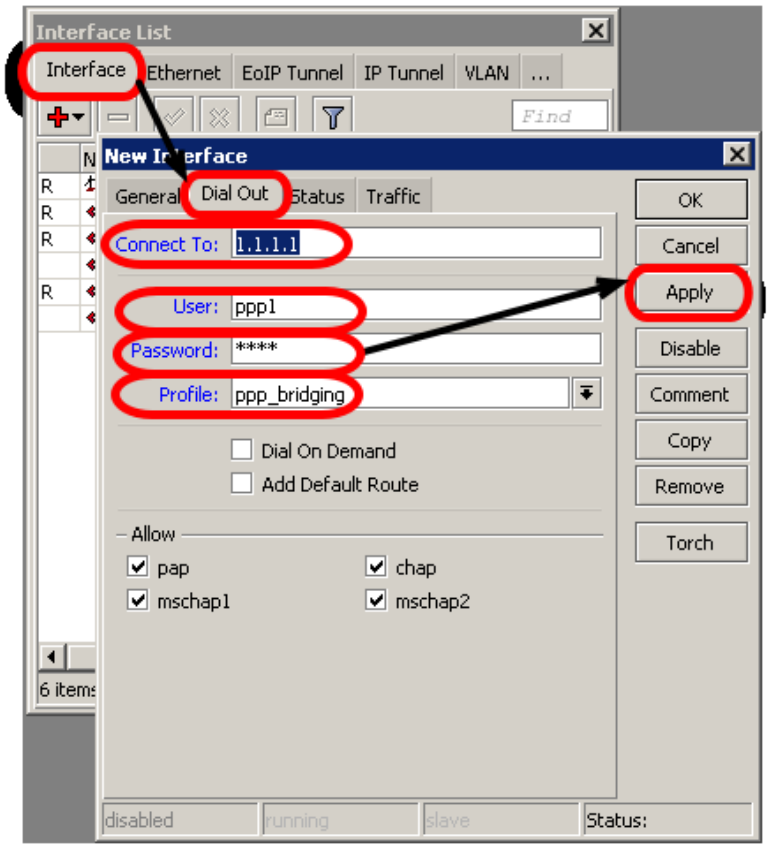
- Enable **PPTP-server**,



Office 2 Configuration

The client router configuration is the same, except that you need to configure and enable PPTP client,

- Add PPTP client,



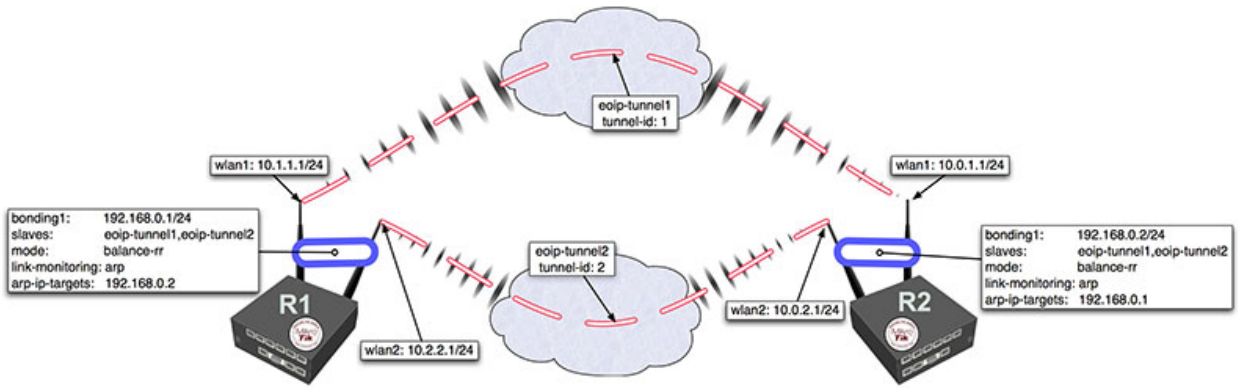
Manual:Bonding Examples

Bonding EoIP tunnels over two wireless links

This is an example of aggregating multiple network interfaces into a single pipe. In particular, it is shown how to aggregate multiple virtual (EoIP) interfaces to get maximum throughput (MT) with emphasis on availability.

Network Diagram

Two routers R1 and R2 are interconnected via multihop wireless links. Wireless interfaces on both sides have assigned IP addresses.



Getting started

Bonding could be used only on OSI layer 2 (Ethernet level) connections. Thus we need to create EoIP interfaces on each of the wireless links. This is done as follows:

- on router R1:

```
[admin@MikroTik] > /interface eoip add remote-address=10.0.1.1/24 tunnel-id=1
[admin@MikroTik] > /interface eoip add remote-address=10.0.2.1/24 tunnel-id=2
```

- and on router R2

```
[admin@MikroTik] > /interface eoip add remote-address=10.1.1.1/24 tunnel-id=1
[admin@MikroTik] > /interface eoip add remote-address=10.2.2.1/24 tunnel-id=2
```

The second step is to add bonding interface and specify EoIP interfaces as slaves:

- R1:

```
[admin@MikroTik] > / interface bonding add slaves=eoip-tunnel1,eoip-tunnel2 mode=balance-rr
```

- R2

```
[admin@MikroTik] > / interface bonding add slaves=eoip-tunnel1,eoip-tunnel2 mode=balance-rr
```

The last step is to add IP addresses to the bonding interfaces:

- R1:

```
[admin@MikroTik] > / ip address add address 192.168.0.1/24 interface=bonding1
```

- R2

```
[admin@MikroTik] > / ip address add address 192.168.0.2/24 interface=bonding1
```

Test the configuration

Now two routers are able to reach each other using addresses from the 192.168.0.0/24 network. To verify bonding interface functionality, do the following:

- R1:

```
[admin@MikroTik] > /interface monitor-traffic eoip-tunnel1,eoip-tunnel2
```

- R2

```
[admin@MikroTik] > /tool bandwidth-test 192.168.0.1 direction=transmit
```

You should see that traffic is distributed equally across both EoIP interfaces:

```
[admin@MikroTik] > /int monitor-traffic eoip-tunnel1,eoip-tunnel2
    received-packets-per-second: 685      685
       received-bits-per-second: 8.0Mbps  8.0Mbps
        sent-packets-per-second: 21       20
           sent-bits-per-second: 11.9kbps 11.0kbps
    received-packets-per-second: 898      899
       received-bits-per-second: 10.6Mbps 10.6Mbps
        sent-packets-per-second: 20       21
           sent-bits-per-second: 11.0kbps 11.9kbps
    received-packets-per-second: 975      975
       received-bits-per-second: 11.5Mbps 11.5Mbps
        sent-packets-per-second: 22       22
           sent-bits-per-second: 12.4kbps 12.3kbps
    received-packets-per-second: 980      980
       received-bits-per-second: 11.6Mbps 11.6Mbps
        sent-packets-per-second: 21       21
           sent-bits-per-second: 11.9kbps 11.8kbps
    received-packets-per-second: 977      977
       received-bits-per-second: 11.6Mbps 11.5Mbps
        sent-packets-per-second: 21       21
           sent-bits-per-second: 11.9kbps 11.8kbps
-- [Q quit|D dump|C-z pause]
[admin@MikroTik] >
```

Link Monitoring

It is easy to notice that with the configuration above as soon as any of individual link fails, the bonding interface throughput collapses. That's because no link monitoring is performed, consequently, the bonding driver is unaware of problems with the underlying links. Enabling link monitoring is a must in most bonding configurations. To enable ARP link monitoring, do the following:

- R1:

```
[admin@MikroTik] > / interface bonding set bonding1 link-monitoring=arp arp-ip-targets=192.168.0.2
```

- R2

```
[admin@MikroTik] > / interface bonding set bonding1 link-monitoring=arp arp-ip-targets=192.168.0.1
```

Bonding Multiple P2P wireless links

Consider following setup:

Manual:Bootloader upgrade

This page shows how to upgrade the Bootloader firmware of a RouterBOARD device.

Simple Upgrade

- Run command */system routerboard upgrade*
- Reboot your router to apply the upgrade (*/system reboot*)]

Note! If you need to install a different version than included in your "routerboard.npk - Upload the latest RouterBOOT firmware to your router's FTP, the latest firmware is available on routerboard.com [1] and then follow above steps.

Checking RouterBOOT version

This command shows the current RouterBOOT version of your device, and available upgrade which is *either included in routerboard.npk package, or if you uploaded a FWF file* corresponding to device model:

```
[admin@MikroTik] > system routerboard print
       routerboard: yes
             model: "750"
     serial-number: "1FC201DD513B"
  current-firmware: "2.18"
  upgrade-firmware: "2.20"
[admin@MikroTik] >
```

In this case you see, that there is **a newer version** of the Bootloader firmware available already inside your current RouterOS version.

Xmodem Method

If there is no IP connectivity with your RouterBOARD, you can also use the Serial Console XMODEM transfer to send the FWF file to the router, while connected via Serial Console. From the Bootloader menu it's possible to upgrade the firmware with this method. This method is the last resort, and should be used only if the first two methods are not available.

References

[1] http://www.routerboard.com

Manual:Console

Applies to RouterOS: 2.9, v3, v4

Overview

The console is used for accessing the MikroTik Router's configuration and management features using text terminals, either remotely using serial port, telnet, SSH or console screen within Winbox, or directly using monitor and keyboard. The console is also used for writing scripts. This manual describes the general console operation principles. Please consult the Scripting Manual on some advanced console commands and on how to write scripts.

Hierarchy

The console allows configuration of the router's settings using text commands. Since there is a lot of available commands, they are split into groups organized in a way of hierarchical menu levels. The name of a menu level reflects the configuration information accessible in the relevant section, eg. **/ip hotspot**.

Example

For example, you can issue the **/ip route print** command:

```
[admin@MikroTik] > ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DIS INTE...
 0 A S  0.0.0.0/0                          r 10.0.3.1        1   bridge1
 1 ADC  1.0.1.0/24         1.0.1.1                           0   bridge1
 2 ADC  1.0.2.0/24         1.0.2.1                           0   ether3
 3 ADC  10.0.3.0/24        10.0.3.144                        0   bridge1
 4 ADC  10.10.10.0/24      10.10.10.1                        0   wlan1
[admin@MikroTik] >
```

Instead of typing **ip route** path before each command, the path can be typed only once to move into this particular branch of menu hierarchy. Thus, the example above could also be executed like this:

```
[admin@MikroTik] > ip route
[admin@MikroTik] ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DIS INTE...
 0 A S  0.0.0.0/0                          r 10.0.3.1        1   bridge1
 1 ADC  1.0.1.0/24         1.0.1.1                           0   bridge1
 2 ADC  1.0.2.0/24         1.0.2.1                           0   ether3
 3 ADC  10.0.3.0/24        10.0.3.144                        0   bridge1
 4 ADC  10.10.10.0/24      10.10.10.1                        0   wlan1
```

```
[admin@MikroTik] ip route>
```

Notice that the prompt changes in order to reflect where you are located in the menu hierarchy at the moment. To move to the top level again, type " **/** "

```
[admin@MikroTik] > ip route
[admin@MikroTik] ip route> /
[admin@MikroTik] >
```

To move up one command level, type " **..** "

```
[admin@MikroTik] ip route> ..
[admin@MikroTik] ip>
```

You can also use **/** and **..** to execute commands from other menu levels without changing the current level:

```
[admin@MikroTik] ip route> /ping 10.0.0.1
10.0.0.1 ping timeout
2 packets transmitted, 0 packets received, 100% packet loss
[admin@MikroTik] ip firewall nat> .. service-port print
Flags: X - disabled, I - invalid
 #   NAME                                                                 PORTS
 0   ftp                                                                  21
 1   tftp                                                                 69
 2   irc                                                                  6667
 3   h323
 4   sip
 5   pptp
[admin@MikroTik] ip firewall nat>
```

Item Names and Numbers

Many of the command levels operate with arrays of items: interfaces, routes, users etc. Such arrays are displayed in similarly looking lists. All items in the list have an item number followed by flags and parameter values.

To change properties of an item, you have to use **set** command and specify name or number of the item.

Item Names

Some lists have items with specific names assigned to each of them. Examples are **interface** or **user** levels. There you can use item names instead of item numbers.

You do not have to use the **print** command before accessing items by their names, which, as opposed to numbers, are not assigned by the console internally, but are properties of the items. Thus, they would not change on their own. However, there are all kinds of obscure situations possible when several users are changing router's configuration at the same time. Generally, item names are more "stable" than the numbers, and also more informative, so you should prefer them to numbers when writing console scripts.

Item Numbers

Item numbers are assigned by the print command and are not constant - it is possible that two successive print commands will order items differently. But the results of last print commands are memorized and, thus, once assigned, item numbers can be used even after **add**, **remove** and **move** operations (since version 3, **move** operation does not renumber items). Item numbers are assigned on a per session basis, they will remain the same until you quit the console or until the next print command is executed. Also, numbers are assigned separately for every item list, so **ip address print** will not change numbering of the interface list.

Since version 3 it is possible to use item numbers without running **print** command. Numbers will be assigned just as if the **print** command was executed.

You can specify multiple items as targets to some commands. Almost everywhere, where you can write the number of item, you can also write a list of numbers.

```
[admin@MikroTik] > interface print
Flags: X - disabled, D - dynamic, R - running
  #    NAME                TYPE             MTU
  0  R ether1              ether            1500
  1  R ether2              ether            1500
  2  R ether3              ether            1500
  3  R ether4              ether            1500
[admin@MikroTik] > interface set 0,1,2 mtu=1460
[admin@MikroTik] > interface print
Flags: X - disabled, D - dynamic, R - running
  #    NAME                TYPE             MTU
  0  R ether1              ether            1460
  1  R ether2              ether            1460
  2  R ether3              ether            1460
  3  R ether4              ether            1500
[admin@MikroTik] >
```

Quick Typing

There are two features in the console that help entering commands much quicker and easier - the [Tab] key completions, and abbreviations of command names. Completions work similarly to the bash shell in UNIX. If you press the [Tab] key after a part of a word, console tries to find the command within the current context that begins with this word. If there is only one match, it is automatically appended, followed by a space:

*/inte***[Tab]**_ becomes **/interface** _

If there is more than one match, but they all have a common beginning, which is longer than that what you have typed, then the word is completed to this common part, and no space is appended:

*/interface set e***[Tab]**_ becomes **/interface set ether**_

If you've typed just the common part, pressing the tab key once has no effect. However, pressing it for the second time shows all possible completions in compact form:

```
[admin@MikroTik] > interface set e[Tab]_
[admin@MikroTik] > interface set ether[Tab]_
[admin@MikroTik] > interface set ether[Tab]_
ether1 ether5
[admin@MikroTik] > interface set ether_
```

The **[Tab]** key can be used almost in any context where the console might have a clue about possible values - command names, argument names, arguments that have only several possible values (like names of items in some lists or name of protocol in firewall and NAT rules). You cannot complete numbers, IP addresses and similar values.

Another way to press fewer keys while typing is to abbreviate command and argument names. You can type only beginning of command name, and, if it is not ambiguous, console will accept it as a full name. So typing:

```
[admin@MikroTik] > pi 10.1 c 3 si 100
```

equals to:

```
[admin@MikroTik] > ping 10.0.0.1 count 3 size 100
```

It is possible to complete not only beginning, but also any distinctive substring of a name: if there is no exact match, console starts looking for words that have string being completed as first letters of a multiple word name, or that simply contain letters of this string in the same order. If single such word is found, it is completed at cursor position. For example:

```
[admin@MikroTik] > interface x[TAB]_
[admin@MikroTik] > interface export _

[admin@MikroTik] > interface mt[TAB]_
[admin@MikroTik] > interface monitor-traffic _
```

General Commands

There are some commands that are common to nearly all menu levels, namely: **print, set, remove, add, find, get, export, enable, disable, comment, move**. These commands have similar behavior throughout different menu levels.

- **add** - this command usually has all the same arguments as **set**, except the item number argument. It adds a new item with the values you have specified, usually at the end of the item list, in places where the order of items is relevant. There are some required properties that you have to supply, such as the interface for a new address, while other properties are set to defaults unless you explicitly specify them.
  - Common Parameters
    - *copy-from* - Copies an existing item. It takes default values of new item's properties from another item. If you do not want to make exact copy, you can specify new values for some properties. When copying items that have names, you will usually have to give a new name to a copy
    - *place-before* - places a new item before an existing item with specified position. Thus, you do not need to use the move command after adding an item to the list
    - *disabled* - controls disabled/enabled state of the newly added item(-s)
    - *comment* - holds the description of a newly created item
  - Return Values
    - add command returns internal number of item it has added
- **edit** - this command is associated with the **set** command. It can be used to edit values of properties that contain large amount of text, such as scripts, but it works with all editable properties. Depending on the capabilities of the terminal, either a fullscreen editor, or a single line editor is launched to edit the value of the specified property.
- **find** - The find command has the same arguments as set, plus the flag arguments like *disabled* or *active* that take values *yes* or *no* depending on the value of respective flag. To see all flags and their names, look at the top of **print** command's output. The **find** command returns internal numbers of all items that have the same values of arguments as specified.
- **move** - changes the order of items in list.

  - Parameters
    - first argument specifies the item(-s) being moved.
    - second argument specifies the item before which to place all items being moved (they are placed at the end of the list if the second argument is omitted).
- **print** - shows all information that's accessible from particular command level. Thus, **/system clock print** shows system date and time, **/ip route print** shows all routes etc. If there's a list of items in current level and they are not read-only, i.e. you can change/remove them (example of read-only item list is /system history, which shows history of executed actions), then print command also assigns numbers that are used by all commands that operate with items in this list.
  - Common Parameters
    - *from* - show only specified items, in the same order in which they are given.
    - *where* - show only items that match specified criteria. The syntax of *where* property is similar to the **find** command.
    - *brief* - forces the print command to use tabular output form
    - *detail* - forces the print command to use property=value output form
    - *count-only* - shows the number of items
    - *file* - prints the contents of the specific submenu into a file on the router.
    - *interval* - updates the output from the *print* command for every interval seconds.
    - *oid* - prints the OID value for properties that are accessible from SNMP
    - *without-paging* - prints the output without stopping after each screenful.
- **remove** - removes specified item(-s) from a list.
- **set** - allows you to change values of general parameters or item parameters. The set command has arguments with names corresponding to values you can change. Use ? or double [Tab] to see list of all arguments. If there is a list of items in this command level, then set has one action argument that accepts the number of item (or list of numbers) you wish to set up. This command does not return anything.

Modes

Console line editor works either in multiline mode or in single line mode. In multiline mode line editor displays complete input line, even if it is longer than single terminal line. It also uses full screen editor for editing large text values, such as scripts. In single line mode only one terminal line is used for line editing, and long lines are shown truncated around the cursor. Full screen editor is not used in this mode.

Choice of modes depends on detected terminal capabilities.

List of keys

Control-C

: keyboard interrupt.

Control-D

: log out (if input line is empty)

Control-K

: clear from cursor to the end of line

Control-X

: toggle safe mode

Control-V

: toggle hotlock mode mode

F6

toggle cellar

F1 or ?

show context sensitive help. If the previous character is \, then inserts literal ?.

Tab

perform line completion. When pressed second time, show possible completions.

Delete

remove character at cursor

Control-H or Backspace

remove character before cursor and move cursor back one position.

Control-\

split line at cursor. Insert newline at cursor position. Display second of the two resulting lines.

Control-B or Left

move cursor backwards one character

Control-F or Right

move cursor forward one character

Control-P or Up

go to previous line. If this is the first line of input then recall previous input from history.

Control-N or Down

go to next line. If this is the last line of input then recall next input from history.

Control-A or Home

move cursor to the beginning of the line. If cursor is already at the beginning of the line, then go to the beginning of the first line of current input.

Control-E or End

move cursor to the end of line. If cursor is already at the end of line, then move it to the end of the last line of current input.

Control-L or F5

reset terminal and repaint screen.

up, **down** and **split** keys leave cursor at the end of line.

Built-in Help

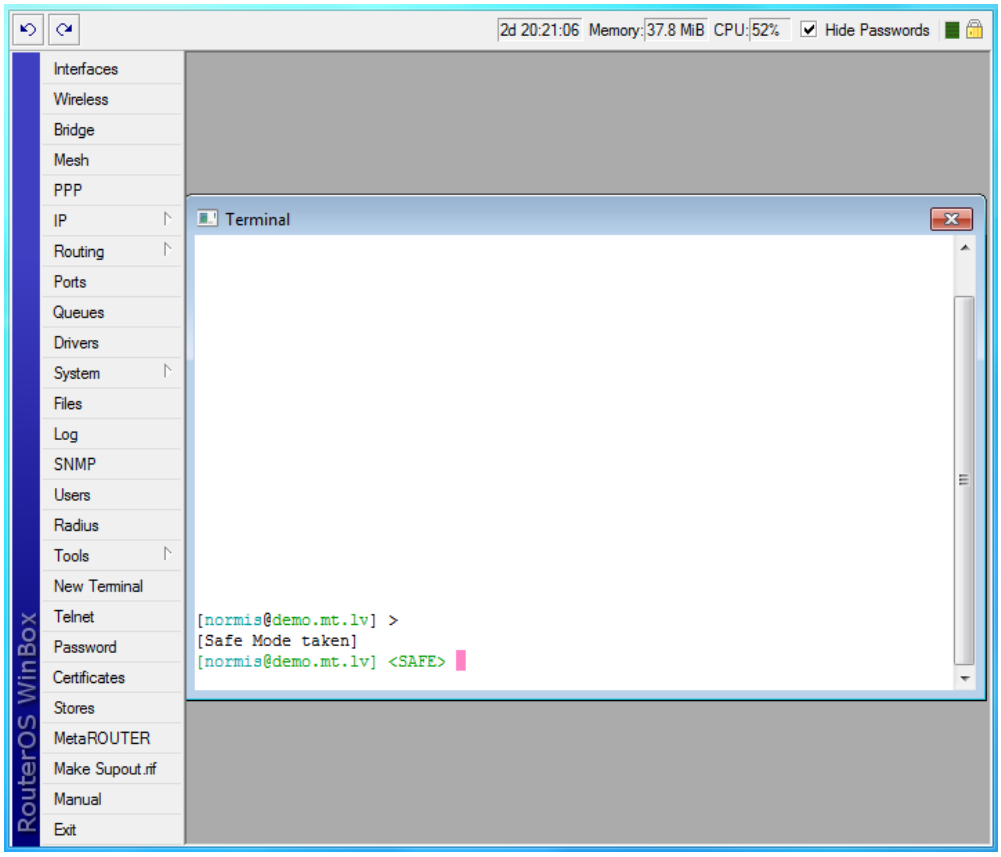
The console has a built-in help, which can be accessed by typing **?**. General rule is that help shows what you can type in position where the **?** was pressed (similarly to pressing **[Tab]** key twice, but in verbose form and with explanations).

Safe Mode

It is sometimes possible to change router configuration in a way that will make the router inaccessible (except from local console). Usually this is done by accident, but there is no way to undo last change when connection to router is already cut. Safe mode can be used to minimize such risk.

Safe mode is entered by pressing **[CTRL]+[X]**. To save changes and quit safe mode, press **[CTRL]+[X]** again. To exit without saving the made changes, hit **[CTRL]+[D]**

```
[admin@MikroTik] ip route>[CTRL]+[X]
[Safe Mode taken]

[admin@MikroTik] ip route<SAFE>
```



Message **Safe Mode taken** is displayed and prompt changes to reflect that session is now in safe mode. All configuration changes that are made (also from other login sessions), while router is in safe mode, are automatically undone if safe mode session terminates abnormally. You can see all such changes that will be automatically undone tagged with an **F** flag in system history:

```
[admin@MikroTik] ip route>
[Safe Mode taken]

[admin@MikroTik] ip route<SAFE> add
[admin@MikroTik] ip route<SAFE> /system history print
Flags: U - undoable, R - redoable, F - floating-undo
  ACTION                                   BY                  POLICY
F route added                              admin               write
```

Now, if telnet connection (or winbox terminal) is cut, then after a while (TCP timeout is **9** minutes) all changes that were made while in safe mode will be undone. Exiting session by **[Ctrl]+[D]** also undoes all safe mode changes, while **/quit** does not.

If another user tries to enter safe mode, he's given following message:

```
[admin@MikroTik] >
Hijacking Safe Mode from someone - unroll/release/don't take it [u/r/d]:
```

- [u] - undoes all safe mode changes, and puts the current session in safe mode.

- [r] - keeps all current safe mode changes, and puts current session in a safe mode. Previous owner of safe mode is notified about this:

```
    [admin@MikroTik] ip firewall rule input
    [Safe mode released by another user]
```

- [d] - leaves everything as-is.

If too many changes are made while in safe mode, and there's no room in history to hold them all (currently history keeps up to 100 most recent actions), then session is automatically put out of the safe mode, no changes are automatically undone. Thus, it is best to change configuration in small steps, while in safe mode. Pressing [Ctrl]+[X] twice is an easy way to empty safe mode action list.

HotLock Mode

When HotLock mode is enabled commands will be auto completed.

To enter/exit HotLock mode press **[CTRL]+[V]**.

```
[admin@MikroTik] /ip address> [CTRL]+[V]
[admin@MikroTik] /ip address>>
```

Double `>>` is indication that HotLock mode is enabled. For example if you type `/in e`, it will be auto completed to

```
[admin@MikroTik] /ip address>> /interface ethernet
```

Quick Help menu

F6 key enables menu at the bottom of the terminal which shows common key combinations and their usage.

```
[admin@RB493G] >

tab compl ? F1 help ^V hotlk ^X safe ^C brk ^D quit
```

Manual:Create Certificates

Following is a step-by-step guide to creating your own CA (Certificate Authority) with openssl on Linux.

Generate certificates

Note: Starting from v5.15 RouterOS supports **pkcs8** key format. If you are using older versions, to import keys in pkcs8 format run command:
`openssl rsa -in myKey.key -text` and write key output to new file. Upload new file to RouterOS and import

- First step is to build the CA private key and CA certificate pair.

```
openssl genrsa -des3 -out ca.key 4096
openssl req -new -x509 -days 3650 -key ca.key -out ca.crt
```

During the process you will have to fill few entries (Common Name (CN), Organization, State or province .. etc). Created CA certificate/key pair will be valid for 10 years (3650 days).

- Now create private-key/certificate pair for the server

```
openssl genrsa -des3 -out server.key 4096
openssl req -new -key server.key -out server.csr

openssl x509 -req -days 3650 -in server.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out server.crt
```

Warning: RSA Key length must be at least 472 bits if certificate is used by SSTP. Shorter keys are considered as security threats.

And again during the process you will have to fill some entries. When filling CN remember that it must not match on CA and server certificate otherwise later naming collision will occur.

Note: Common Name (CN) in server certificate should match the the IP address of your server otherwise you will get **"domain mismatch"** message and for example Windows SSTP client will not be able to connect to the server. If clients are only Windows machines then CN can be a DNS name, too.

- Client key/certificate pair creation steps are very similar to server. Remember to Specify unique CN.

```
openssl genrsa -des3 -out client.key 4096
openssl req -new -key client.key -out client.csr

openssl x509 -req -days 3650 -in client.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out client.crt
```

To examine certificate run following command:

```
openssl x509 -noout -text -in server.crt -purpose
```

Import certificates

To import newly created certificates to your router, first you have to upload server.crt and server.key files to the router via FTP. Now go to `/certificate` submenu and run following commands:

```
[admin@test_host] /certificate> import file-name=server.crt
passphrase:
     certificates-imported: 1
     private-keys-imported: 0
            files-imported: 1
       decryption-failures: 0
  keys-with-no-certificate: 0
[admin@test_host] /certificate> import file-name=server.key
passphrase:
     certificates-imported: 0
     private-keys-imported: 1
            files-imported: 1
       decryption-failures: 0
  keys-with-no-certificate: 0
```

If everything is imported properly then certificate should show up with **KR** flag.

```
[admin@test_host] /certificate> print
Flags: K - decrypted-private-key, Q - private-key, R - rsa, D - dsa
 0 KR name="cert1" subject=C=LV,ST=RI,L=Riga,O=MT,CN=server,emailAddress=xxx@mt.lv
      issuer=C=LV,ST=RI,L=Riga,O=MT,CN=MT CA,emailAddress=xxx@mt.lv serial-number="01"
      email=xxx@mt.lv invalid-before=jun/25/2008 07:24:33
      invalid-after=jun/23/2018 07:24:33 ca=yes
```

Note: If you want to use server certificates for OVPN or SSTP and use client certificate verification, then CA certificate must be imported, too.

[Top | Back to Content]

Manual:System/Certificates

Applies to RouterOS: v6.0 +

Summary

Sub-menu: `/certificate`
Package required: `security`
Standards: `RFC 5280, draft-nourse-scep-22`

Certificate manager is used to collect all certificates inside router, to manage and create serlf-signed certificates and to control and set SCEP related configuration.

Note: Starting from v6 certificate validity is shown using local time zone offset. In previous versions it was UTF.

Warning: RSA Key length must be at least 472 bits if certificate is used by SSTP. Shorter keys are considered as security threats.

General Menu

Sub-menu: `/certificate`

Properties

Property	Description
alias ()	
ca (*yes* | *no*)	
decrypted-private-key (*yes* | *no*)	Whether private key is decrypted
dsa (*yes* | *no*)	
email (*string*)	
invalid-after (*date*)	The date after which certificate wil be invalid.
invalid-before (*date*)	The date before which certificate is invalid.
issuer (*string*)	
name (*string*)	Name of the certificate. Name can be edited.
private-key (*yes* | *no*)	
rsa (*yes* | *no*)	
serial-number (*string*)	
subject (*string*)	

Commands

Command	Description
create-certificate-request ()	Creates certificate request file and key.
decrypt ()	Decrypt private key.
import (*file-name*)	File name of certificate or key to be imported.
reset-certificate-cache ()	Resets certificate cache after this private keys must be decrypted.

Self-Signed CA Management

Sub-menu: `/certificate ca`

Starting from RouterOS version 6 it is possible to manage and create self-signed CAs. It is not possible to import self signed CAs here. Implementation was made based on RFC 5280 and all certificates are X.509 v3.

All certificate fingerprints are SHA1. All private keys and CA export passphrase are stored encrypted with hardware ID. CRL renewal happens at every certificate revocation and after 24hours.

Note: Time and date on routers MUST be correct

Properties

Property	Description
alias ()	
common-name (*string*)	
country (*string*)	
crl-host (*string*)	
email (*string*)	
expired (*yes* | *no*)	Whether CA is expired.
fingerprint (*string*)	
invalid-after (*date*)	The date after which CA wil be invalid.
invalid-before (*date*)	The date before which CA is invalid.
issuer (*string*)	
locality (*string*)	
name (*string*)	Name of the certificate. Name can be edited.
organization (*string*)	
self-signed (*yes* | *no*)	Whether CA is self signed
serial-number (*string*)	
state (*string*)	
unit (*string*)	

Commands

Command	Description
create-self-signed-ca ()	Creates self signed CA and generates key. Required extensions are export passphrase (which is used to protect private key when user tries to export it), validity period and IP address.
export (*name or number of cert*)	Exports certificate and private key which is encrypted with provided passphrase.
remove (*name or number of cert*)	Remove specified CA and all linked certificates.

Issued Certificates

Sub-menu: `/certificate ca certificate`

Properties

Property	Description
ca (*string*)	Name of the CA certificate stored in Self-Signed CAs menu
common-name (*string*)	
country (*string*)	
email (*string*)	
expired (*yes* | *no*)	Whether certificate is expired
fingerprint (*string*)	
invalid-after (*date*)	Date after which certificate will be invalid
invalid-before (*date*)	Date before which certificate is invalid
locality (*string*)	
name (*string*)	
organization (*string*)	
revoked (*date*)	Date and time when certificate was revoked
serial-number (*string*)	
state (*string*)	
unit (*string*)	

Commands

Command	Description
create-certificate ()	Generate certificate and key assigned from specified CA. User manually provides standard certificate parameters.
sign-certificate-request (*ca, days-valid, file-name, key-bits*)	Generates certificate and key, except that standard parameters are taken from certificate request. Command takes four parameters: • **ca** - name of the CA certificate • **days-valid** - validity period • **file-name** - certificate request filename • **key-bits** - RSA key bits
revoke (*name or number of cert*)	Certificate can't be deleted. You can only revoke it. After revoke is executed certificate is added to CRL and CRL is renewed.

export (*name or number of cert*)	Export certificate and private key. Difference from CA export is that private key is protected with passphrase specified during the export process. Everyone ho has rights to export can access private keys.

SCEP

Sub-menu: `/certificate`
Standards: `draft-nourse-scep-22`

Simple Certificate Enrollment protocol (SCEP) was developed based on draft-nourse-scep-22.

The protocol is designed so that any user can request certificate as simple as possible. The protocol allows to issue and revoke certificates.

How SCEP works

Topology: CL ---- RA ---- CA

- CL - client
- RA - registration authority (proxy)
- CA - certification authority (server)

SCEP is using HTTP protocol and base64 encoded **GET** requests. Most of requests are without authentication and cipher, however important ones can be protected if necessary (ciphered or signed using received public key).

SCEP client in RouterOS will:

- get CA certificate from CA server or RA (if used);
- user should compare fingerprint of the CA certificate or if it comes from the right server;
- generate self-signed certificate with temporary key;
- sends certificate request to the server;
- if server respond with status **x**, then client keeps requesting until server sends an error or approval.

SCEP server supports issue of one certificate only. RouterOS supports also renew and next-ca options:

- **renew** - possibility to renew old certificate automatically with the same CA.
- **next-ca** - possibility to change current CA certificate to the new one. Client polls the server for any changes, if server advertise that next-ca is available, then client may request next CA or wait until CA almost expires and then request next-ca.

RouterOS Server also supports *POST* ***operation,*** *3DES* ***cipher and*** **'SHA1** hashing. If client does not support these features then http **GET**, **DES** cipher and **MD5** hashing is used.

RouterOS client by default will try to use POST, 3DES and SHA1 if server advertises that.

Client

Sub-menu: `/certificate scep client`

Properties

Property	Description
ca-identity (*string*; Default: **DummyCAIdentity**)	
challenge-password (*string*; Default: **""**)	OTP password on the server used to grant certificate automatically after request.
common-name (*string*; Default:)	
country (*string*; Default:)	
disabled (*yes* | *no*; Default: **no**)	
email (*string*; Default:)	
fingerprint-algorithm (*md5* | *sha1*; Default: **sha1**)	
key-bits (*1024* | *2048* | *4096*; Default: **1024**)	
locality (*string*; Default:)	
name (*string*; Default:)	Short descriptive name of an item
organziation (*string*; Default:)	
path (*string*; Default:)	Path of certificate located on the server. If server is RouterOS then you should add "scep/"+path since certificates on server are stored in "scep" dir.
serial-number (*string*; Default:)	
server (*IP* | *IPv6*; Default:)	IP or IPv6 address of the SCEP server
state (*string*; Default:)	
store-name (*string*; Default:)	Name of the certificate which will be used after importing into certificate store.
unit (*string*; Default:)	

Status Properties

Property	Description
ca-fingerprint (*string*)	
req-fingerprint (*string*)	
status (*string*)	Shows the current status of the client. Idle, pending, requesting etc.

Commands

Command	Description
renew (*ca_client_name*)	Renew Ca certificate of specified CA client Name.

Server

Sub-menu: `/certificate scep server`

OTP

Sub-menu: `/certificate scep server otp`

Transactions

Sub-menu: `/certificate scep server transactions`

[Top | Back to Content]

Manual:CD Install

Applies to RouterOS: 2.9, v3, v4

CD Install Description

CD-Install allows to install MikroTik RouterOS to x86 boxes, which do not support Netinstall (all the RouterBOARDs should be reinstalled with Netinstall).

Note: RouterOS installation will erase all data on your HDD, it will only work as the only operating system in your PC. Remove any drives that you don't want to be erased

CD Install Requirements

Router

- x86 box with hard drive
- CD-ROM

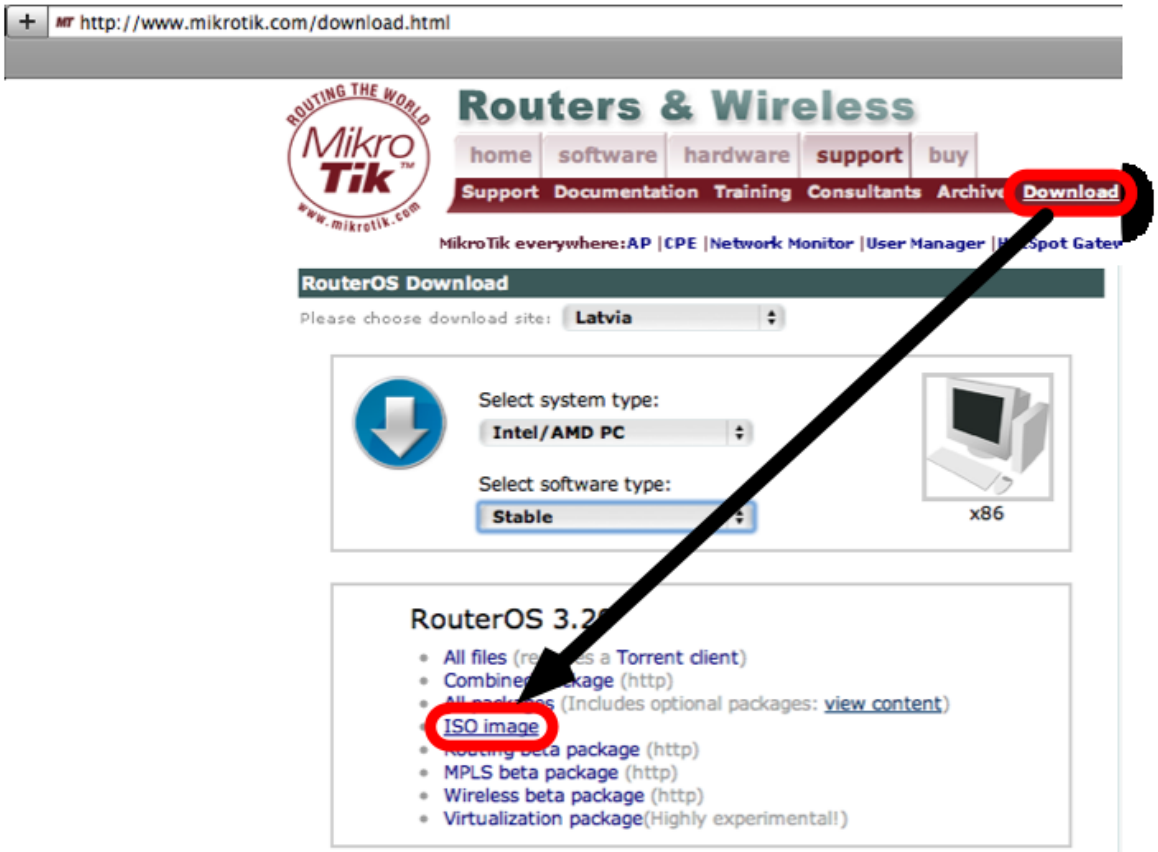
Additional PC

- CD-ROM
- CD burning application
- MikroTik RouterOS CD installation ISO image

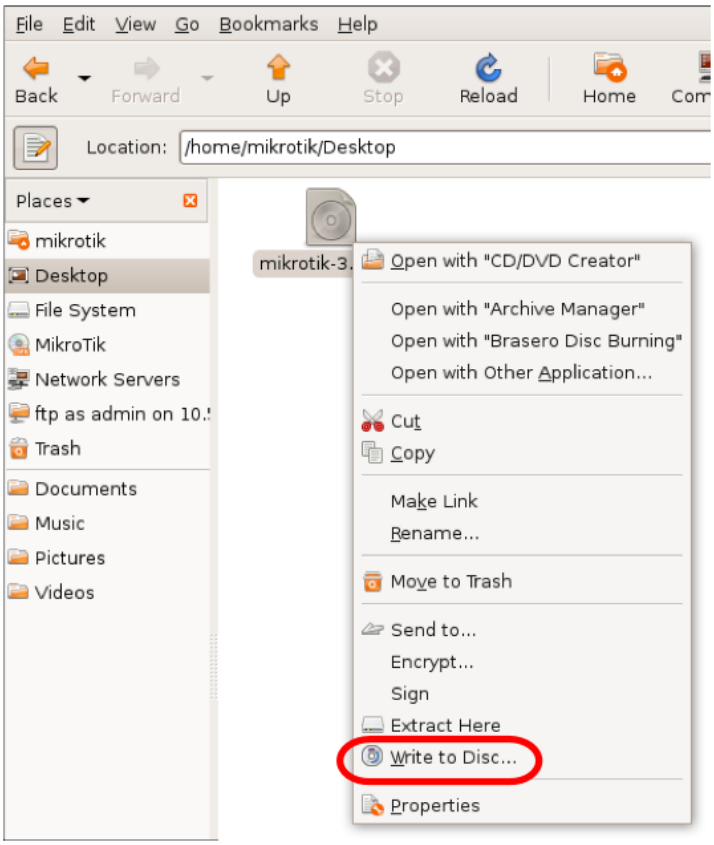
CD Install Example

Prepare MikroTik RouterOS CD Installation Disk

1. Download CD installation Image from MikroTik download page [1],

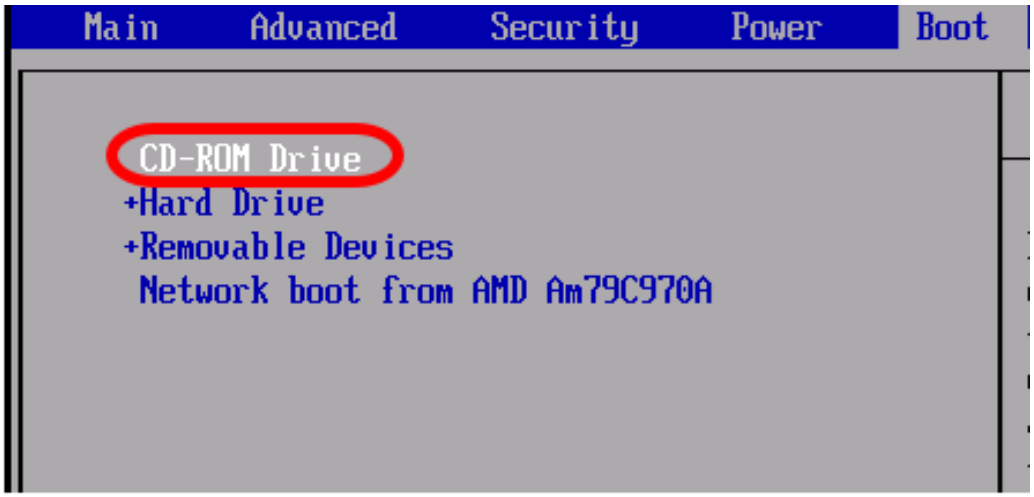


2. Burn ISO image to disk, you need PC with CD-ROM and application to write ISO files to CD. For Linux (the latest Ubuntu release) you can use built-in application. Mouse right-click on the **.iso** file and specify 'Write to Disk'. You got MikroTik RouterOS installation disk after process is finished.

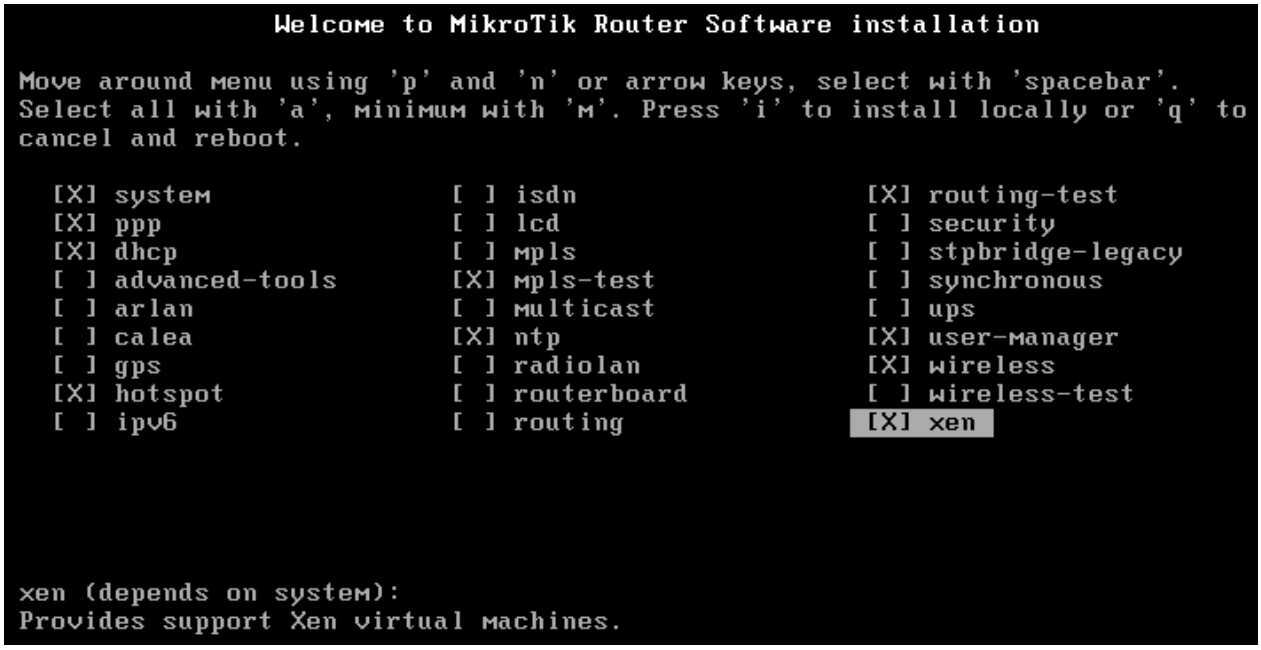


Router Preconfiguration

3. Switch on the x86 box, where you want to install MikroTik RouterOS, it should be with CD-ROM as well. Put MikroTik RouterOS installation disk to CD-ROM and set to boot from CD-ROM in BIOS settings,



4. x86 will boot from MikroTik RouterOS installation disk and should offer you to select the RouterOS Packages to install,

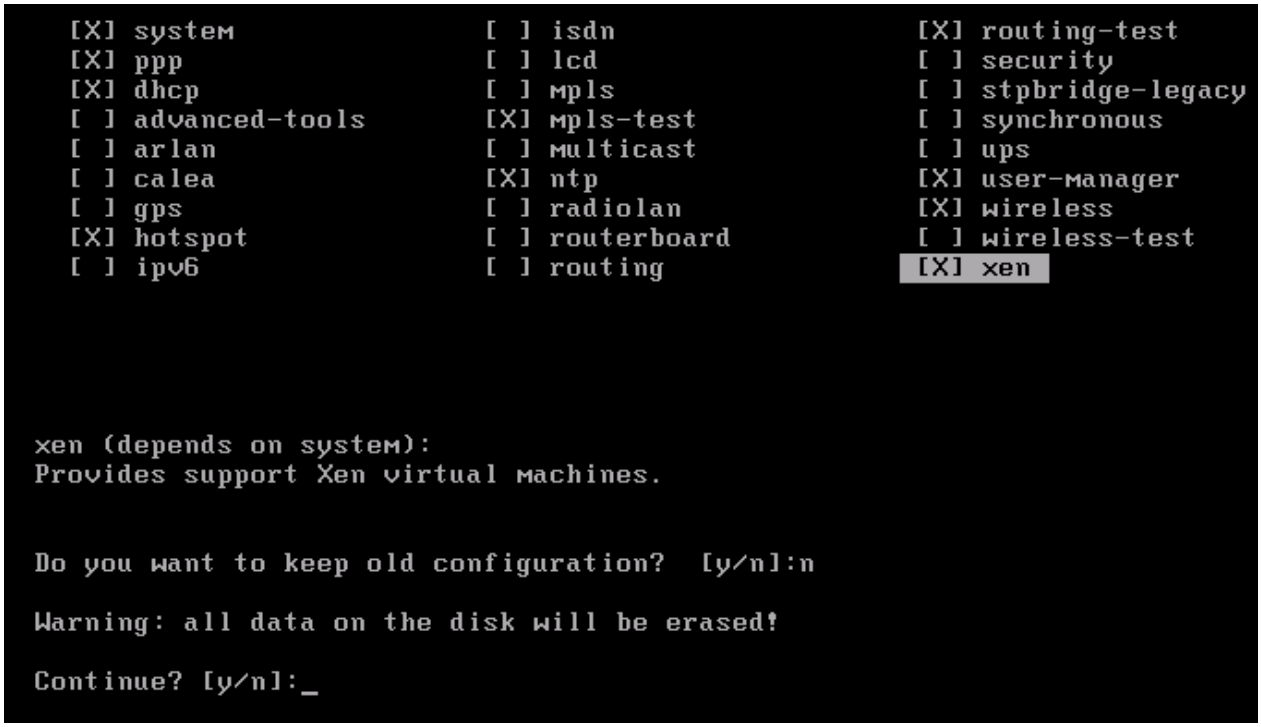


Package Selection

5. Select the packages you want to install, it is possible to select all packages with **a** or minimum with **m**, then Press **i** to install the RouterOS.

Installation

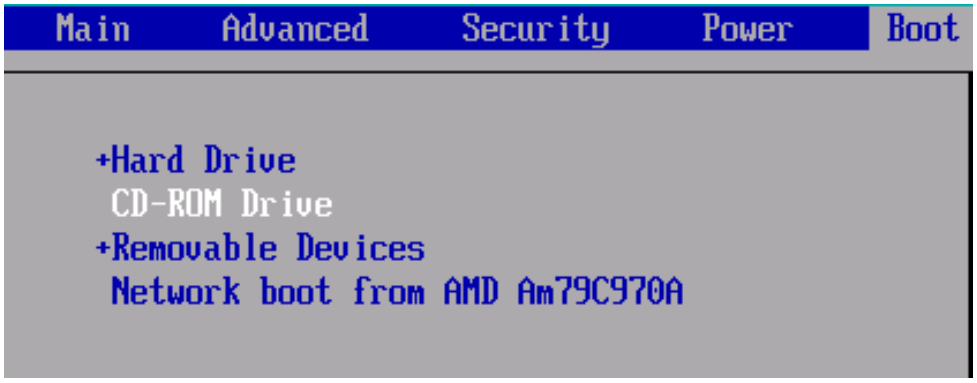
6. If you have previous installation of the RouterOS and want to reset the configuration, then answer **no** for the question 'Do you want to keep old configuration ?' and click **y** to proceed,



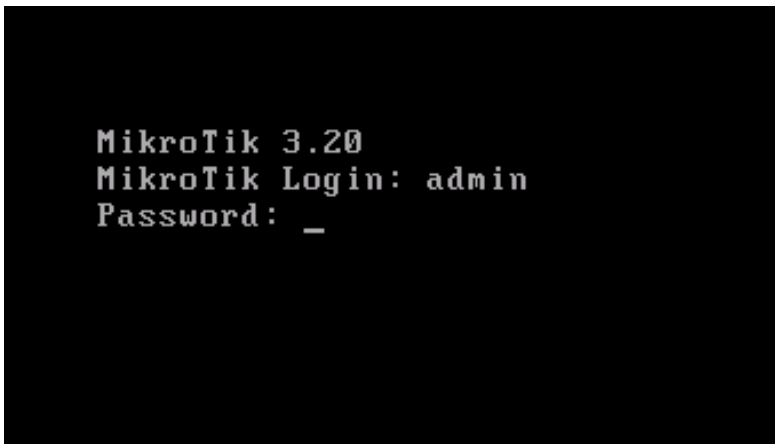
7. You will the process of the packages installation. Router will ask for the reboot after installation is finished,

Post Installation procedures

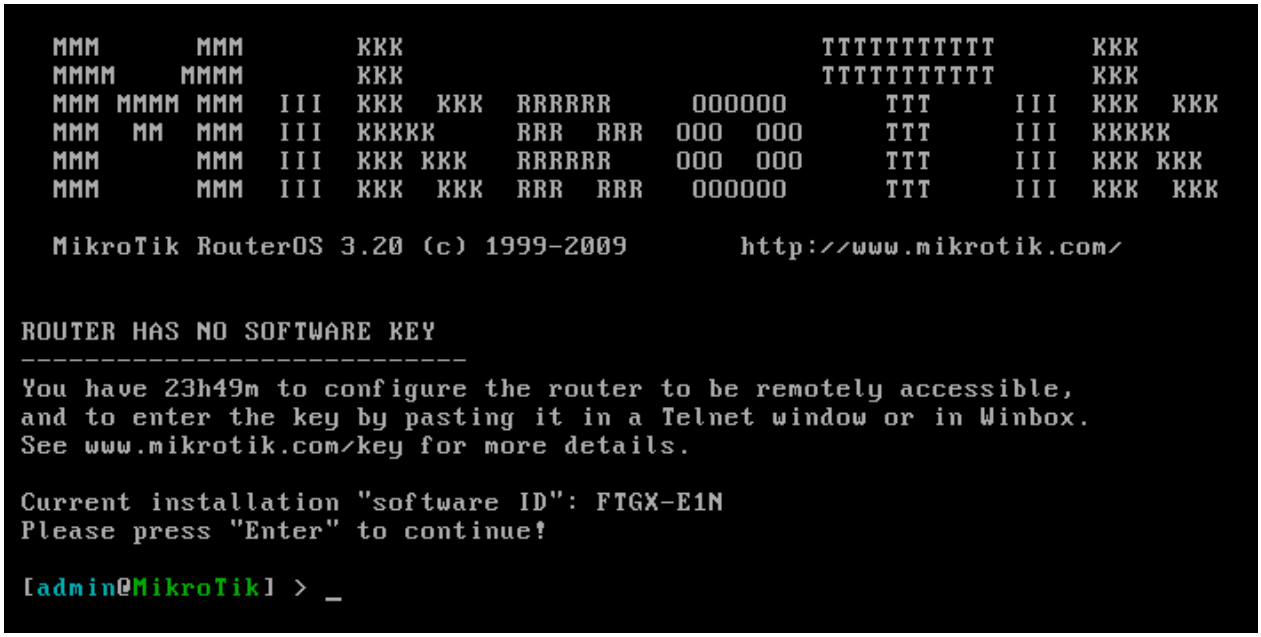
8. MikroTik RouterOS is successfully **installed**, do not forget to eject CD installation disk and set PC to boot from Hard Drive,



9. MikroTik RouterOS is booted and you are ready to login. Default login is **admin** without any password,



10. The last of the installation to license the router, use the **software-id** to purchase the license,



Reset RouterOS configuration with CD Intstall

To reset the RouterOS configuration with CD Install, follow the procedure and on the step 6, set **no** for the answer 'Do you want to keep old configuration ?'.

References

[1] http://www.mikrotik.com/download.html

Manual:Configuration Management

Applies to RouterOS: 2.9, v3, v4

Summary

This manual introduces you with commands which are used to perform the following functions:

- system backup;
- system restore from a backup;
- configuration export;
- configuration import;
- system configuration reset.

Description

The configuration backup can be used for backing up MikroTik RouterOS configuration to a binary file, which can be stored on the router or downloaded from it using FTP for future use. The configuration restore can be used for restoring the router's configuration, exactly as it was at the backup creation moment, from a backup file. The restoration procedure assumes the cofiguration is restored on the same router, where the backup file was originally created, so it will create partially broken configuration if the hardware has been changed.

The configuration export can be used for dumping out complete or partial MikroTik RouterOS configuration to the console screen or to a text (script) file, which can be downloaded from the router using FTP protocol. The configuration dumped is actually a batch of commands that add (without removing the existing configuration) the selected configuration to a router. The configuration import facility executes a batch of console commands from a script file.

System reset command is used to erase all configuration on the router. Before doing that, it might be useful to backup the router's configuration.

System Backup

Submenu level: */system backup*

Description

The **backup save** command is used to store the entire router configuration in a backup file. The file is shown in the /file submenu. It can be downloaded via ftp to keep it as a backup for your configuration.

Important! The backup file contains sensitive information, do not store your backup files inside the router's Files directory, instead, download them, and keep them in a secure location.

To restore the system configuration, for example, after a */system reset-configuration*, it is possible to upload that file via ftp and load that backup file using load command in */system backup* submenu. Command Description

- **load name=[filename]** - Load configuration backup from a file
- **save name=[filename]** - Save configuration backup to a file

Warning: If TheDude and user-manager is installed on the router then backup will not take care of configuration used by these tools. Therefore additional care should be taken to save configuration from these. Use provided tool mechanisms to save/export configuration if you want to save it.

Example

To save the router configuration to file test:

```
[admin@MikroTik] system backup> save name=test
Configuration backup saved
[admin@MikroTik] system backup>
```

To see the files stored on the router:

```
[admin@MikroTik] > file print
  # NAME                          TYPE         SIZE       CREATION-TIME
  0 test.backup                   backup       12567      sep/08/2004 21:07:50
[admin@MikroTik] >
```

To load the saved backup file test:

```
[admin@MikroTik] > system backup load name=test
Restore and reboot? [y/N]:
y
Restoring system configuration
System configuration restored, rebooting now
```

Exporting Configuration

Command name: */export*

The **export** command prints a script that can be used to restore configuration. The command can be invoked at any menu level, and it acts for that menu level and all menu levels below it. The output can be saved into a file, available for download using FTP.

Command Description

- **file=[filename]** - saves the export to a file

Example

```
[admin@MikroTik] > ip address print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   10.1.0.172/24      10.1.0.0        10.1.0.255      bridge1
 1   10.5.1.1/24        10.5.1.0        10.5.1.255      ether1
[admin@MikroTik] >
```

To make an export file:

```
[admin@MikroTik] ip address> export file=address
[admin@MikroTik] ip address>
```

To see the files stored on the router:

```
[admin@MikroTik] > file print
 # NAME                          TYPE         SIZE       CREATION-TIME
0  address.rsc                   script       315        dec/23/2003 13:21:48
[admin@MikroTik] >
```

Compact Export

Starting from v5.12 compact export was added. It allows to export only part of configuration that is not default RouterOS config.

Note: Starting from v6rc1 "export compact" is default behavior. To do old style export use *export verbose*

For example compact OSPF export:

```
[admin@SXT-ST] /routing ospf> export compact
# jan/02/1970 20:16:32 by RouterOS 5.12
# software id = JRB7-9UGC
#
/routing ospf instance
set [ find default=yes ] redistribute-connected=as-type-1
/routing ospf interface
add disabled=yes interface=wlan1 network-type=point-to-point
/routing ospf network
add area=backbone network=10.255.255.36/32
add area=backbone disabled=yes network=10.5.101.0/24
add area=backbone network=10.10.10.0/24
[admin@SXT-ST] /routing ospf>
```

Compact export introduces another feature that indicates which part of config is default on RouterOS and cannot be deleted. As in example below '*' indicates that this OSPF instance is part of default configuration.

```
[admin@SXT-ST] /routing ospf instance> print
Flags: X - disabled, * - default
 0  * name="default" router-id=0.0.0.0 distribute-default=never
      redistribute-connected=as-type-1 redistribute-static=no
      redistribute-rip=no redistribute-bgp=no redistribute-other-ospf=no
      metric-default=1 metric-connected=20 metric-static=20 metric-rip=20
      metric-bgp=auto metric-other-ospf=auto in-filter=ospf-in
      out-filter=ospf-out
```

List of default config by menus that cannot be removed:

Menu	Entries
/interface wireless security-profiles	default
/ppp profile	"default", "default-encryption"
/ip hotspot profile	"default"
/ip hotspot user profile	"default"
/ip ipsec proposal	"default"
/ip smb shares	"pub"
/ip smb users	"guest"
/ipv6 nd	"all"

/mpls interface	"all"
/routing bfd interface	"all"
/routing bgp instance	"default"
/routing ospf instance	"default"
/routing ospf area	"backbone"
/routing ospf-v3 instance	"default"
/routing ospf-v3 area	"backbone"
/snmp community	"public"
/tool mac-server mac-winbox	"all"
/tool mac-server	"all"
/system logging	"info", "error", "warning", "critical"
/system logging action	"memory", "disk", "echo", "remote"
/queue type	"default", "ethernet-default", "wireless-default", "synchronous-default", "hotspot-default", "only-hardware-queue", "multi-queue-ethernet-default", "default-small"

Importing Configuration

Command name: */import*

The root level command **/import [file_name]** executes a script, stored in the specified file adds the configuration from the specified file to the existing setup. This file may contain any console comands, including scripts. is used to restore configuration or part of it after a /system reset event or anything that causes configuration data loss.

Note that it is impossible to import the whole router configuration using this feature. It can only be used to import a part of configuration (for example, firewall rules) in order to spare you some typing.

Command Description

- **file=[filename]** - loads the exported configuration from a file to router

Automatic Import

Since RouterOS v3rc it is possible to automatically execute scripts - your script file has to be called anything**.auto.rsc** - once this file is uploaded with FTP to the router, it will automatically be executed, just like with the Import command.

Example

To load the saved export file use the following command:

```
[admin@MikroTik] > import address.rsc
Opening script file address.rsc

Script file loaded and executed successfully
[admin@MikroTik] >
```

Configuration Reset

Command name: */system reset-configuration*

Description

The command clears all configuration of the router and sets it to the default including the login name and password ('admin' and no password), IP addresses and other configuration is erased, interfaces will become disabled. After the reset command router will reboot.

Command Description

- **keep-users**: *keeps router users and passwords*
- **no-defaults**: *doesn't load any default cofigurations, just clears everything*
- **skip-backup**: *automatic backup is not created before reset, when yes is specified*
- **run-after-reset**: *specify export file name to run after reset*

Warning: If the router has been installed using netinstall and had a script specified as the initial configuration, the reset command executes this script after purging the configuration. To stop it doing so, you will have to reinstall the router.

Example

```
[admin@MikroTik] > system reset-configuration
Dangerous! Reset anyway? [y/N]: n
action cancelled
[admin@MikroTik] >
```

Manual:Conformance Testing Mode

This mode allows you to test wireless channels outside the default scan-list and/or regulatory domain. This mode should only be used in controlled environments, or if you have a special permission to use it.

Applies to RouterOS: v4.3+

Before v4.3 this was called *Custom Frequency Upgrade*, or *Superchannel*. Since RouterOS v4.3 it is called Conformance Testing Mode and is available *without* special key upgrades for all installations.

Please note that the Conformance Testing Mode is available free of charge since v4.3.

License upgrade purchase is only needed if you intend to use older versions, where this mode was called Superchannel.

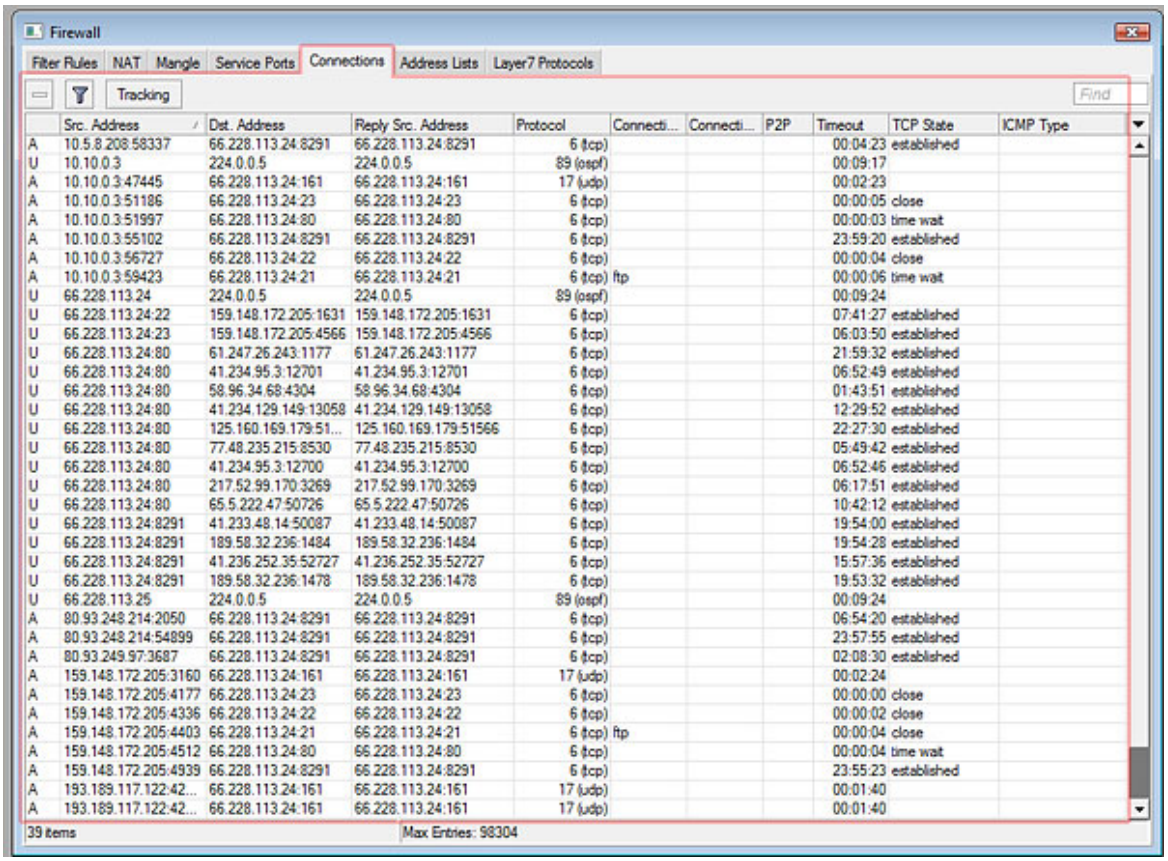
Manual:IP/Firewall/Connection tracking

Connection tracking entries

Sub-menu: `/ip firewall connection`

There are several ways to see what connections are making their way though the router.

In the Winbox Firewall window, you can switch to the Connections tab, to see current connections to/from/through your router. It looks like this:



Properties

All properties in connection list are read-only

Property	Description
seen reply (*yes* | *no*)	
assured (*yes* | *no*)	"assured" flag indicates that this connection is assured and that it will not be erased if maximum possible tracked connection count is reached.
connection-mark (*string*)	connection mark set by mangle rule.
connection-type (*pptp* | *ftp* | *p2p*)	Type of connection, property is empty if connection tracking is unable to determine predefined connection type.
dst-address (*ip[:port]*)	Destination address and port (if protocol is port based).
gre-key (*integer*)	
gre-version (*string*)	
icmp-code (*string*)	
icmp-id (*string*)	
icmp-type (*string*)	
p2p (*yes* | *no*)	Shows if connection is identified as p2p by firewall p2p matcher.
protocol (*string*)	IP protocol type
reply-dst-address (*ip[:port]*)	Destination address (and port) expected of return packets. Usually the same as "src-address:port"
reply-src-address (*ip[:port]*)	Source address (and port) expected of return packets. Usually the same as "dst-address:port"
src-address (*ip[:port]*)	Source address and port (if protocol is port based).
tcp-state (*string*)	Current state of TCP connection : • "established" • "time-wait" • "close" • "syn-sent" • "syn-received"
timeout (*time*)	Time after connection will be removed from connection list.

Connection tracking settings

Sub-menu: `/ip firewall connection tracking`

Properties

Property	Description
enabled (*yes* | *no* | *auto*; Default: **auto**)	Allows to disable or enable connection tracking. Disabling connection tracking will cause several firewall features to stop working. See the list of affected features. Starting from v6.0rc2 default value is auto. Which means that connection tracing is disabled until at least one firewall rule is added.
tcp-syn-sent-timeout (*time*; Default: **5s**)	TCP SYN timeout.
tcp-syn-received-timeout (*time*; Default: **5s**)	TCP SYN timeout.
tcp-established-timeout (*time*; Default: **1d**)	Time when established TCP connection times out.
tcp-fin-wait-timeout (*time*; Default: **10s**)	
tcp-close-wait-timeout (*time*; Default: **10s**)	
tcp-last-ack-timeout (*time*; Default: **10s**)	
tcp-time-wait-timeout (*time*; Default: **10s**)	
tcp-close-timeout (*time*; Default: **10s**)	
udp-timeout (*time*; Default: **10s**)	
udp-stream-timeout (*time*; Default: **3m**)	
icmp-timeout (*time*; Default: **10s**)	
generic-timeout (*time*; Default: **10m**)	Timeout for all other connection entries
tcp-syncookie (*yes* | *no*; Default: **no**)	

Read-only properties

Property	Description
max-entries (*integer*)	Max amount of entries that connection tracking table can hold. This value depends on installed amount of RAM.
total-entries (*integer*)	Amount of connections that currently connection table holds.

Features affected by connection tracking

- NAT
- firewall:
  - connection-bytes
  - connection-mark
  - connection-type
  - connection-state
  - connection-limit
  - connection-rate
  - layer7-protocol

- p2p
- new-connection-mark
- tarpit

- p2p matching in simple queues

Manual:Connection Rate

Applies to RouterOS: 3, v4

Introduction

Connection Rate is a firewall matcher that allow to capture traffic based on present speed of the connection.

Theory

Each entry in connection tracking table represents bidirectional communication. Every time packet gets associated to particular entry, packet size value (including IP header) is added to "connection-bytes" value for this entry. (in another words "connection-bytes" includes both - upload and download)

Connection Rate calculates speed of connection based on change of "connection-bytes". Connection Rate is recalculated every second and does not have any averages.

Both options "connection-bytes" and "connection-rate" work only with TCP and UDP traffic. (you need to specify protocol to activate these options)

In "connection-rate" you can specify range of speed that you like to capture.

```
ConnectionRate ::= [!]From-To
  From,To ::= 0..4294967295    (integer number)
```

Example

These rules will capture TCP/UDP traffic that was going trough the router when connection speed was below 100kbps

```
/ip firewall filter
add action=accept chain=forward connection-rate=0-100k protocol=tcp
add action=accept chain=forward connection-rate=0-100k protocol=udp
```

Notes

Connection Rate is available in RouterOS since v3.30. This option was introduced to allow capture traffic intensive connections.

Application Example - Traffic Prioritization

Connection-rate can be used in various different ways, that still need to be realized, but most common setup will be to detect and set lower priorities to the "heavy connections" (connections that maintain fast rate for long periods of time (such as P2P,HTTP,FTP downloads). By doing this you can prioritize all other traffic that usually includes VOIP and HTTP browsing and online gaming.

Method described in this example can be used together with other ways to detect and prioritize traffic

As connection-rate option does not have any averages we need to determine what will be the margin that identifies "heavy connections". If we assume that normal HTTP browsing connection is less than 500kB (4Mb) long and VOIP requires no more than 200kbps speed, then every connection that after first 500kB still have more than 200kbps speed can be assumed as "heavy".

(You might have different "connection-bytes" for HTTP browsing and differenet "connection-rate" for VOIP in your network - so, please, do your own research before applying this example)

For this example lets assume that we have 6Mbps upload and download connection to ISP.

Quick Start for Impatient

```
/ip firewall mangle
add chain=forward action=mark-connection connection-mark=!heavy_traffic_conn \
    new-connection-mark=all_conn
add chain=forward action=mark-connection connection-bytes=500000-0 \
    connection-mark=all_conn connection-rate=200k-100M \
    new-connection-mark=heavy_traffic_conn protocol=tcp
add chain=forward action=mark-connection connection-bytes=500000-0 \
    connection-mark=all_conn connection-rate=200k-100M \
    new-connection-mark=heavy_traffic_conn protocol=udp
add chain=forward action=mark-packet connection-mark=heavy_traffic_conn \
    new-packet-mark=heavy_traffic passthrough=no
add chain=forward action=mark-packet connection-mark=all_conn \
    new-packet-mark=other_traffic passthrough=no

/queue tree
add name=upload parent=public max-limit=6M
add name=other_upload parent=upload limit-at=4M max-limit=6M \
    packet-mark=other_traffic priority=1
add name=heavy_upload parent=upload limit-at=2M max-limit=6M \
    packet-mark=heavy_traffic priority=8
add name=download parent=local max-limit=6M
add name=other_download parent=download limit-at=4M max-limit=6M \
    packet-mark=other_traffic priority=1
add name=heavy_download parent=download limit-at=2M max-limit=6M \
    packet-mark=heavy_traffic priority=8
```

Explanation

In mangle we need to separate all connections into two groups, then mark packets from there 2 groups. As we are talking about client's traffic most logical place for marking would be mangle chain forward.

Keep in mind that as soon as "heavy" connection will have lower priority and queue will hit max-limit - heavy connection will drop speed, and connection-rate will be lower. This will result in a change to higher priority and connection will be able to get more traffic for a short while, when again connection-rate will raise and that again will result in change to lower priority). To avoid this we must make sure that once detected "heavy connections" will remain marked as "heavy connections" for all times.

IP Firewall mangle

```
/ip firewall mangle
add chain=forward action=mark-connection connection-mark=!heavy_traffic_conn \
    new-connection-mark=all_conn
```

This rule will ensure that that "heavy" connections will remain heavy". and mark rest of the connections with default connection mark.

```
add chain=forward action=mark-connection connection-bytes=500000-0 \
    connection-mark=all_conn connection-rate=200k-100M \
    new-connection-mark=heavy_traffic_conn protocol=tcp
add chain=forward action=mark-connection connection-bytes=500000-0 \
    connection-mark=all_conn connection-rate=200k-100M \
    new-connection-mark=heavy_traffic_conn protocol=udp
```

These two rules will mark all heavy connections based on our standarts, that every connection that after first 500kB still have more than 200kbps speed can be assumed as "heavy"

```
add chain=forward action=mark-packet connection-mark=heavy_traffic_conn \
    new-packet-mark=heavy_traffic passthrough=no
add chain=forward action=mark-packet connection-mark=all_conn \
    new-packet-mark=other_traffic passthrough=no
```

Last two rules in mangle will simple mark all traffic from corresponding connections.

Queue

This is a simple queue tree that is placed on the Interface HTB - "public" is interface where your ISP is connected, "local" where are your clients. If you have more than 1 "public" or more than 1 "local" you will need to mangle upload and download separately and place queue tree in global-out.

```
/queue tree
add name=upload parent=public max-limit=6M
add name=other_upload parent=upload limit-at=4M max-limit=6M \
    packet-mark=other_traffic priority=1
add name=heavy_upload parent=upload limit-at=2M max-limit=6M \
    packet-mark=heavy_traffic priority=8
add name=download parent=local max-limit=6M
add name=other_download parent=download limit-at=4M max-limit=6M \
    packet-mark=other_traffic priority=1
add name=heavy_download parent=download limit-at=2M max-limit=6M \
    packet-mark=heavy_traffic priority=8
```

Manual:Console login process

Applies to RouterOS: 2.9, v3, v4

Description

There are different ways to log into console:

- serial port
- console (screen and keyboard)
- telnet
- ssh
- mac-telnet
- winbox terminal

Input and validation of user name and password is done by login process. Login process can also show different informative screens (license, demo version upgrade reminder, software key information, default configuration).

At the end of successful login sequence login process prints banner and hands over control to the console process.

Console process displays system note, last critical log entries, auto-detects terminal size and capabilities and then displays command prompt]. After that you can start writing commands.

Use up arrow to recall previous commands from command history, TAB key to automatically complete words in the command you are typing, ENTER key to execute command, and Control-C to interrupt currently running command and return to prompt.

Easiest way to log out of console is to press Control-D at the command prompt while command line is empty (You can cancel current command and get an empty line with Control-C, so Control-C followed by Control-D will log you out in most cases).

Console login options

Starting from v3.14 it is possible to specify console options during login process. These options enables or disables various console features like color, terminal detection and many other.

Additional login parameters can be appended to login name after '+' sign.

```
login_name ::= user_name [ '+' parameters ]
parameters ::= parameter [ parameters ]
parameter ::= [ number ] 'a'..'z'
number ::= '0'..'9' [ number ]
```

If parameter is not present, then default value is used. If number is not present then implicit value of parameter is used.

example: admin+c80w - will disable console colors and set terminal width to 80.

Param	Default	Implicit	Description
"w"	auto	auto	Set terminal width
"h"	auto	auto	Set terminal height
"c"	on	off	disable/enable console colors
"t"	on	off	Do auto detection of terminal capabilities
"e"	on	off	Enables "dumb" terminal mode

Different information shown by login process

Banner

Login process will display MikroTik banner after validating user name and password.

```
  MMM       MMM        KKK                          TTTTTTTTTTT      KKK
  MMMM     MMMM        KKK                          TTTTTTTTTTT      KKK
  MMM MMMM MMM  III   KKK  KKK   RRRRRR     OOOOOO      TTT     III  KKK  KKK
  MMM  MM  MMM  III   KKKKK     RRR  RRR  OOO  OOO     TTT     III  KKKKK
  MMM       MMM  III   KKK KKK   RRRRRR    OOO  OOO     TTT     III  KKK KKK
  MMM       MMM  III   KKK  KKK  RRR  RRR   OOOOOO      TTT     III  KKK  KKK

  MikroTik RouterOS 3.0rc (c) 1999-2007       http://www.mikrotik.com/
```

Actual banner can be different from the one shown here if it is replaced by distributor. See also: branding.

License

After logging in for the first time after installation you are asked to read software licenses.

```
Do you want to see the software license? [Y/n]:
```

Answer y to read licenses, n if you do not wish to read licenses (question will not be shown again). Pressing SPACE will skip this step and the same question will be asked after next login.

Demo version upgrade reminder

After logging into router that has demo key, following remonder is shown:

```
UPGRADE NOW FOR FULL SUPPORT
----------------------------
FULL SUPPORT benefits:
- receive technical support
- one year feature support
- one year online upgrades
    (avoid re-installation and re-configuring your router)
To upgrade, register your license "software ID"
 on our account server www.mikrotik.com

Current installation "software ID": ABCD-456

Please press "Enter" to continue!
```

Software key information

If router does not have software key, it is running in the time limited trial mode. After logging in following information is shown:

```
ROUTER HAS NO SOFTWARE KEY
----------------------------
You have 16h58m to configure the router to be remotely accessible,
and to enter the key by pasting it in a Telnet window or in Winbox.
See www.mikrotik.com/key for more details.

Current installation "software ID": ABCD-456
Please press "Enter" to continue!
```

After entering valid software key, following information is shown after login:

```
ROUTER HAS NEW SOFTWARE KEY
----------------------------
Your router has a valid key, but it will become active
only after reboot. Router will automatically reboot in a day.

=== Automatic configuration ===

Usually after [[netinstall|installation]] or configuration [[reset]] RouterOS will apply [[default
settings]], such as an IP address.
First login into will show summary of these settings and offer to undo them.
This is an example:
<pre>
The following default configuration has been installed on your router:
-------------------------------------------------------------------------------
IP address 192.168.88.1/24 is on ether1
ether1 is enabled

-------------------------------------------------------------------------------
You can type "v" to see the exact commands that are used to add and remove
this default configuration, or you can view them later with
'/system default-configuration print' command.
To remove this default configuration type "r" or hit any other key to continue.
If you are connected using the above IP and you remove it, you will be disconnected.
```

Applying and removing of the default configuration is done using console script (you can press 'v' to review it).

Different information shown by console process after logging in

System Note

It is possible to always display some fixed text message after logging into console.

Critical log messages

Console will display last critical error messages that this user has not seen yet. See log for more details on configuration. During console session these messages are printed on screen.

```
dec/10/2007 10:40:06 system,error,critical login failure for user root from 10.0.0.1 via telnet
dec/10/2007 10:40:07 system,error,critical login failure for user root from 10.0.0.1 via telnet
dec/10/2007 10:40:09 system,error,critical login failure for user test from 10.0.0.1 via telnet
```

Prompt

- `[admin@MikroTik] /interface>` - Default command prompt, shows user name, system identity, and current command path.
- `[admin@MikroTik] /interface<SAFE>` - Prompt indicates that console session is in Safe Mode.
- `[admin@MikroTik] >>` - Prompt indicates that HotLock is turned on.
- `{(\...` - While entering multiple line command continuation prompt shows open parentheses.
- `line 2 of 3>` - While editing multiple line command prompt shows current line number and line count.
- `address:` - Command requests additional input. Prompt shows name of requested value.

Console can show different prompts depending on enabled modes and data that is being edited. Default command prompt looks like this:

```
[admin@MikroTik] /interface>
```

Default command prompt shows name of user, '@' sign and system name in brackets, followed by space, followed by current command path (if it is not '/'), followed by '>' and space. When console is in safe mode, it shows word SAFE in the command prompt.

```
[admin@MikroTik] /interface<SAFE>
```

Hotlock mode is indicated by an additional yellow '>' character at the end of the prompt.

```
[admin@MikroTik] >>
```

It is possible to write commands that consist of multiple lines. When entered line is not a complete command and more input is expected, console shows continuation prompt that lists all open parentheses, braces, brackets and quotes, and also trailing backslash if previous line ended with backslash-whitespace.

```
[admin@MikroTik] > {
{... :put (\
{(\... 1+2)}
3
```

When you are editing such multiple line entry, prompt shows number of current line and total line count instead of usual username and system name.

```
line 2 of 3> :put (\
```

Sometimes commands ask for additional input from user. For example, command '/password' asks for old and new passwords. In such cases prompt shows name of requested value, followed by colon and space.

```
[admin@MikroTik] > /password
old password: ******
new password: **********
retype new password: **********
```

FAQ

Q: ***How do I turn off colors in console?***
A: Add '+c' after login name.

Q: ***After logging in console prints rubbish on the screen, what to do?***
Q: ***My expect script does not work with newer 3.0 releases, it receives some strange characters. What are those?***
A: These sequences are used to automatically detect terminal size and capabilities. Add '+t' after login name to turn them off.

Q: ***Thank you, now terminal width is not right. How do I set terminal width?***
A: Add '+t80w' after login name, where 80 is your terminal width.

[Top | Back to Content]

Manual:CPU Usage

Applies to RouterOS: v2,v3,v4

RouterOS is capable of showing the status of your hardware device and it's available resources. This includes CPU load.

Above zero CPU usage usually means that your machine is doing something and that it is not in standby state. This in no way indicates a problem.

A higher than average CPU usage that stays for a long time usually indicates much traffic which is being processed by RouterOS, this includes Queues, Mangle, Firewall etc. Dynamic routing protocols also can take CPU resources in heavy traffic conditions. Still, this does not mean that your router is having trouble handling it. The number 100 does not indicate any kind of limit in your hardware power.

```
[normis@demo.mt.lv] > system resource monitor
     cpu-used: 41
  free-memory: 31488
```

If your router does stay on cpu usage 100 for a lot of time, you should try the following:

1. See what kind of traffic is going through your router. You can use Torch for this. An attack to the router can also cause heavy CPU load.
2. Disable the interfaces and see if the problem goes away, you can also unplug the Ethernet cables to be sure the traffic is not causing it.
3. Disable some or all of your Queues/Filter Rules to see if you have too many of them. You can optimize your ruleset, or use PCQ to drastically reduce the number of Queues.
4. See if the cpu load numbers actually affect anything apart from the number displayed. The fact that the router is doing something does not imply any kind of problem, you should only investigate if there are visible problems with the operation of the router.

Manual:Default Configurations

Applies to RouterOS: v5

List of Default Configs

Integrated Indoors

	Wan port	Lan port	Wireless mode	ht chain	ht extension	dhcp-server	dhcp-client	Firewall	NAT	Default IP	Mac Server
RB750 RB750G	ether1	Switched ether2-ether5	-	-	-	on lan port	on wan port	blocked access to wan port	Masquerade wan port	192.168.88.1/24 on lan port	Disabled on wan port
RB751	ether1	Switched ether2-ether5, bridged wlan1 with switch	AP b/g/n 2412MHz	0,1	above-control	on lan port	on wan port	blocked access to wan port	Masquerade wan port	192.168.88.1/24 on lan port	Disabled on wan port
RB951	ether1	Switched ether2-ether5, bridged wlan1 with switch	AP b/g/n 2412MHz	0	above-control	on lan port	on wan port	blocked access to wan port	Masquerade wan port	192.168.88.1/24 on lan port	Disabled on wan port
RB1100 AH/AHx2	-	-	-	-	-	-	-	-	-	192.168.88.1/24 on ether1	-
RB1200	-	-	-	-	-	-	-	-	-	192.168.88.1/24 on ether1	-
RB2011	sfp1,ether1	two switch gropups bridged (ether2-ether10, wlan1 if present)	-	-	-	on lan port	on wan port	blocked access to wan port	Masquerade wan port	192.168.88.1/24 on ether1	Disabled on wan port

Integrated Outdoors

	Wan port	Lan port	Wireless mode	ht chain	ht extension	dhcp-server	dhcp-client	Firewall	NAT	Default IP	Mac Server
Groove 2Hn	wlan1	ether1	station a/n 2.4GHz	0	above control	on lan port	on wan port	blocked access to wan port	Masquerade wan port	192.168.88.1/24 on lan port	Disabled on wan port
Groove 5Hn	wlan1	ether1	station a/n 5GHz	0	above control	on lan port	on wan port	blocked access to wan port	Masquerade wan port	192.168.88.1/24 on lan port	Disabled on wan port
Groove A-5Hn	-	bridged wlan1,ether1	AP a/n 5300MHz	0	-	-	-	-	-	192.168.88.1/24 on lan port	-
Metal 5	wlan1	ether1	station a/n 5GHz	0	above control	on lan port	on wan port	blocked access to wan port	Masquerade wan port	192.168.88.1/24 on lan port	Disabled on wan port
SXT 5xx, SXT G-5xx	wlan1	ether1	station a/n 5GHz	0,1	above control	on lan port	on wan port	blocked access to wan port	Masquerade wan port	192.168.88.1/24 on lan port	Disabled on wan port
OmniTik	ether1	Switched ether2-ether5, bridged wlan1 with switch	AP a/n 5300MHz	0,1	-	on lan port	on wan port	-	Masquerade wan port	192.168.88.1/24 on lan port	-
SEXTANT	wlan1	ether1	station a/n 5GHz	0,1	above control	on lan port	on wan port	blocked access to wan port	Masquerade wan port	192.168.88.1/24 on lan port	Disabled on wan port

Engineered

	Wan port	Lan port	Wireless mode	ht chain	ht extension	dhcp-server	dhcp-client	Firewall	NAT	Default IP	Mac Server
RB411xx, RB435G, RB433xx, RB495xx, RB800	-	-	-	-	-	-	-	-	-	192.168.88.1/24 on ether1	-
RB450xx	ether1	Switched ether2-ether5	-	-	-	on lan port	on wan port	blocked access to wan port	Masquerade wan port	192.168.88.1/24 on lan port	Disabled on wan port
RB711-5xx, RB711G-5xx	wlan1	ether1	station a/n 5GHz	0	above control	on lan port	on wan port	blocked access to wan port	Masquerade wan port	192.168.88.1/24 on lan port	Disabled on wan port
RB711UA-5xx, RB711GA-5xx	-	bridged wlan1,ether1	AP a/n 5300MHz	0	-	-	-	-	-	192.168.88.1/24 on lan port	-

RB711-2xx	wlan1	ether1	station b/g/n 2.4GHz	0	above control	on lan port	on wan port	blocked access to wan port	Masquerade wan port	192.168.88.1/24 on lan port	Disabled on wan port
RB711UA-2xx	-	bridged wlan1,ether1	AP a/n 2412MHz	0	-	-	-	-	-	192.168.88.1/24 on lan port	-

Note: To see exact configuration script that will be applied after system reset use following command **`/system default-configuration print`**

Wan Port

When applying configuration WAN port is renamed to "<wan port>-gateway", for example, if wan port is ether1, it will be renamed to "ether1-gateway".

Local Port

Local port can be:

- single interface
- ethernets configured in switch group
- bridged all interfaces that are not WAN and switch slaves.

If ports are switched then master port is renamed to "<ethernet name>-master-local" and slaves to "<ethernet name>-slave-local".

Lets take **RB751** as an example. Board has ether1 configured as WAN port, it has switch chip and one pre-configured wireless interface. So in this case all ethernets except ether1 are grouped in switch group and bridged with wireless interface.

Generated config will be:

```
/interface set ether2 name=ether2-master-local;
/interface set ether3 name=ether3-slave-local;
/interface set ether4 name=ether4-slave-local;
/interface set ether5 name=ether5-slave-local;
/interface ethernet set ether3-slave-local master-port=ether2-master-local;
/interface ethernet set ether4-slave-local master-port=ether2-master-local;
/interface ethernet set ether5-slave-local master-port=ether2-master-local;

/interface bridge add name=bridge-local disabled=no auto-mac=no protocol-mode=rstp;

:local bMACIsSet 0;
:foreach k in=[/interface find] do={
        :local tmpPort [/interface get $k name];
        :if ($bMACIsSet = 0) do={
                :if ([/interface get $k type] = "ether") do={
                        /interface bridge set "bridge-local" admin-mac=[/interface ethernet get $tmpPort mac-address];
                        :set bMACIsSet 1;
                  }
        }
        :if (!($tmpPort~"bridge" || $tmpPort~"ether1" || $tmpPort~"slave")) do={
```

```
                /interface bridge port add bridge=bridge-local interface=$tmpPort;
        }
}
```

Wireless Config

Wireless configuration depends on market segment for which board is designed. It can be configured as AP or station in 2GHz and 5GHz frequencies. Default 2GHz frequency is 2412 and default 5GHz frequency is 5300. SSID is "Mikrotik".

If board has two chains (letter D in the naming of the board), then both chains are enabled. HT Extension is enabled on all CPEs.

For example generated config on RB751:

```
:if ( $wirelessEnabled = 1) do={
# wait for wireless
        :while ([/interface wireless find] = "") do={ :delay 1s; };

        /interface wireless set wlan1 mode=ap-bridge band=2ghz-b/g/n ht-txchains=0,1 ht-rxchains=0,1 \
                disabled=no country=no_country_set wireless-protocol=any
        /interface wireless set wlan1 channel-width=20/40mhz-ht-above ;
}
```

Default IP and DHCP Config

Default IP address on all boards is 192.168.88.1/24. Boards without specific configuration has IP address set on ether1, other boards has IP address on LAN interface.

All boards that has WAN port configured, DHCP client is set on WAN port.

Typically on all CPEs DHCP server is set on LAN port, giving out addresses in range from 192.168.88.2-192.168.88.254

As an example **RB751** applied DHCP config.

```
/ip dhcp-client add interface=ether1-gateway disabled=no

/ip pool add name="default-dhcp" ranges=192.168.88.10-192.168.88.254;
/ip dhcp-server
  add name=default address-pool="default-dhcp" interface=bridge-local disabled=no;

/ip dhcp-server network
  add address=192.168.88.0/24 gateway=192.168.88.1 dns-server=192.168.88.1 comment="default configuration";
```

Firewall, NAT and MAC server

All boards with configured WAN port has configured protection on that port. Any traffic leaving WAN port is masqueraded.

Config example:

```
/ip firewall {
      filter add chain=input action=accept protocol=icmp comment="default configuration"
      filter add chain=input action=accept connection-state=established in-interface=ether1-gateway comment="default configuration"
      filter add chain=input action=accept connection-state=related in-interface=ether1-gateway comment="default configuration"
      filter add chain=input action=drop in-interface=ether1-gateway comment="default configuration"
      nat add chain=srcnat out-interface=ether1-gateway action=masquerade comment="default configuration"
}



/tool mac-server remove [find];
/tool mac-server mac-winbox disable [find];
:foreach k in=[/interface find] do={
       :local tmpName [/interface get $k name];
       :if (!($tmpName~"ether1")) do={
              /tool mac-server add interface=$tmpName disabled=no;
              /tool mac-server mac-winbox add interface=$tmpName disabled=no;
        }
}
/ip neighbor discovery set [find name="ether1-gateway"] discover=no
```

DNS

Every board allows remote DNS requests and static DNS name is pre-configured.

```
       /ip dns {
              set allow-remote-requests=yes
              static add name=router address=192.168.88.1
       }
```

[Top | Back to Content]

Manual:IP/DHCP Client

Applies to RouterOS: v3, v4 +

Summary

The MikroTik RouterOS DHCP client may be enabled on any Ethernet-like interface at a time. The client will accept an address, netmask, default gateway, and two dns server addresses. The received IP address will be added to the interface with the respective netmask. The default gateway will be added to the routing table as a dynamic entry. Should the DHCP client be disabled or not renew an address, the dynamic default route will be removed. If there is already a default route installed prior the DHCP client obtains one, the route obtained by the DHCP client would be shown as invalid.

RouterOS DHCP cilent asks for following options:

- option 1 - SUBNET_MASK,
- option 3 - GATEWAY_LIST,
- option 6 - TAG_DNS_LIST,
- option 33 - STATIC_ROUTE,
- option 42 - NTP_LIST,
- option 121 - CLASSLESS_ROUTE,

IPv6

Starting from v5.8 DHCP Client can receive delegated prefixes from DHCPv6 server. Currently received prefix is added to IPv6 pool, which later can be used for example in pppoe server configuration. Starting from v5.9, DHCPv6 client configuration was moved to /ipv6 sub-menu. `Read-more >>`

Quick setup example

Add a DHCP client on ether1 interface:

```
/ip dhcp-client add interface=ether1 disabled=no
```

After interface is added, you can use rint" or "print detail" command to see what parameters DHCP client acquired:

```
[admin@MikroTik] ip dhcp-client> print detail
Flags: X - disabled, I - invalid
 0   interface=ether1 add-default-route=yes use-peer-dns=yes use-peer-ntp=yes
     status=bound address=192.168.0.65/24 gateway=192.168.0.1
     dhcp-server=192.168.0.1 primary-dns=192.168.0.1 primary-ntp=192.168.0.1
     expires-after=9m44s
[admin@MikroTik] ip dhcp-client>
```

Note: If interface used by DHCP client is part of VRF configuration, then default route and other received routes from DHCP server will be added to VRF routing table.

Properties

Sub-menu: `/ip dhcp-client`

Property	Description
add-default-route (*yes* | *no*; Default: **yes**)	Whether to install default route in routing table received from dhcp server.
client-id (*string*; Default:)	Corresponds to the settings suggested by the network administrator or ISP. If not specified, client's MAC address will be sent
comment (*string*; Default:)	Short description of the client
default-route-distance (*integer:0..255*; Default:)	Distance of default route. Applicable if `add-default-route` is set to `yes`.
disabled (*yes* | *no*; Default: **yes**)	
host-name (*string*; Default:)	Host name of the client sent to a DHCP server. If not specified, client's system identity will be used.
interface (*string*; Default:)	Interface on which DHCP client will be running.
use-peer-dns (*yes* | *no*; Default: **yes**)	Whether to accept the DNS settings advertised by DHCP Server. (Will override the settings put in the `/ip dns` submenu.
use-peer-ntp (*yes* | *no*; Default: **yes**)	Whether to accept the NTP settings advertised by DHCP Server. (Will override the settings put in the `/system ntp client` submenu)

Status

Command `/ip dhcp-client print detail` will show current status of dhcp client and **read-only** properties listed in table below:

Property	Description
address (*IP/Netmask*)	IP address and netmask, which is assigned to DHCP Client from the Server
dhcp-server (*IP*)	IP address of the DHCP server.
expires-after (*time*)	Time when the lease expires (specified by the DHCP server).
gateway (*IP*)	IP address of the gateway which is assigned by DHCP server
invalid (*yes* | *no*)	Shows whether configuration is invalid.
netmask (*IP*)	
primary-dns (*IP*)	IP address of the primary DNS server, assigned by the DHCP server
primary-ntp (*IP*)	IP address of the primary NTP server, assigned by the DHCP server
secondary-dns (*IP*)	IP address of the secondary DNS server, assigned by the DHCP server
secondary-ntp (*IP*)	IP address of the secondary NTP server, assigned by the DHCP server
status (*bound* | *error* | *rebinding...* | *requesting...* | *searching...* | *stopped*)	Shows the status of DHCP Client

Menu specific commands

Property	Description
release (*numbers*)	Release current binding and restart DHCP client
renew (*numbers*)	Renew current leases. If the renew operation was not successful, client tries to reinitialize lease (i.e. it starts lease request procedure (rebind) as if it had not received an IP address yet)

[Top | Back to Content]

Manual:IP/DHCP Relay

Applies to RouterOS: v3, v4 +

Summary

DHCP Relay is just a proxy that is able to receive a DHCP request and resend it to the real DHCP server.

Properties

Sub-menu: `/ip dhcp-client`

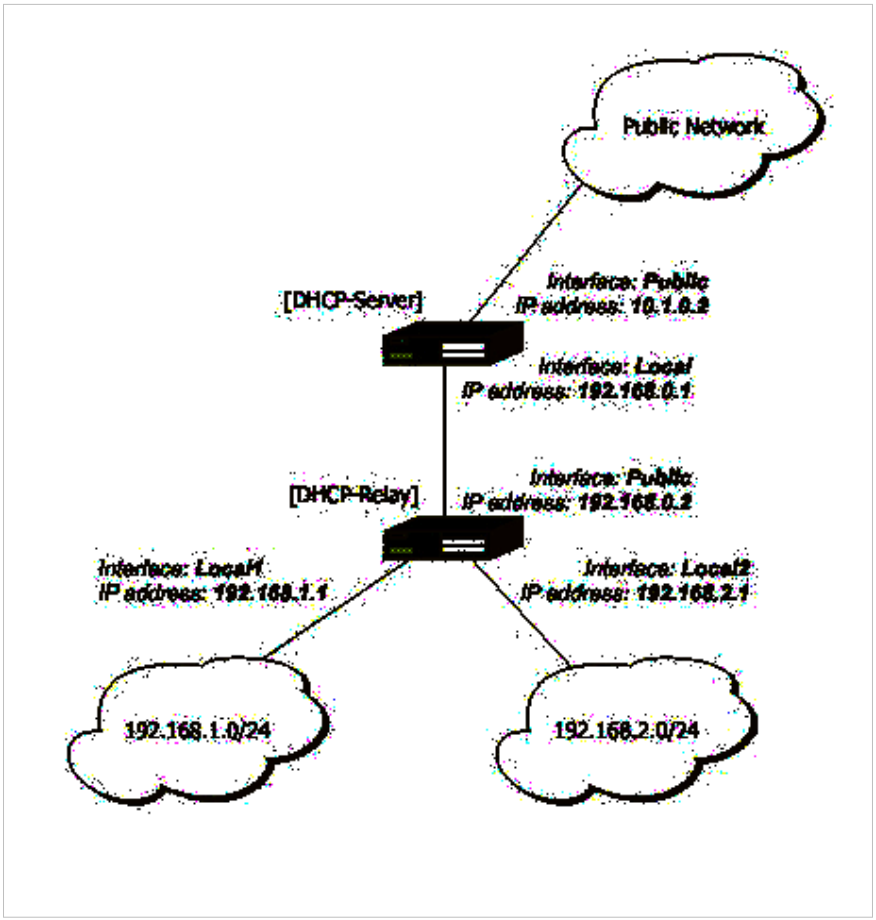
Property	Description
delay-threshold (*time*; Default: **none**)	If secs field in DHCP packet is smaller than delay-threshold, then this packet is ignored
dhcp-server (*string*; Default:)	List of DHCP servers' IP addresses which should the DHCP requests be forwarded to
interface (*string*; Default:)	Interface name the DHCP relay will be working on.
local-address (*IP*; Default: **0.0.0.0**)	The unique IP address of this DHCP relay needed for DHCP server to distinguish relays. If set to **0.0.0.0** - the IP address will be chosen automatically
name (*string*; Default:)	Descriptive name for relay

DHCP relay does not choose the particular DHCP server in the dhcp-server list, it just send the incoming request to all the listed servers.

Example setup

Let us consider that you have several IP networks 'behind' other routers, but you want to keep all DHCP servers on a single router. To do this, you need a DHCP relay on your network which relies DHCP requests from clients to DHCP server.

This example will show you how to configure a DHCP server and a DHCP relay which serve 2 IP networks - 192.168.1.0/24 and 192.168.2.0/24 that are behind a router DHCP-Relay.



IP Address Configuration

IP addresses of DHCP-Server:

```
[admin@DHCP-Server] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   192.168.0.1/24     192.168.0.0     192.168.0.255   To-DHCP-Relay
 1   10.1.0.2/24      10.1.0.0      10.1.0.255      Public
[admin@DHCP-Server] ip address>
```

IP addresses of DHCP-Relay:

```
[admin@DHCP-Relay] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   192.168.0.2/24     192.168.0.0     192.168.0.255   To-DHCP-Server
 1   192.168.1.1/24     192.168.1.0     192.168.1.255   Local1
 2   192.168.2.1/24     192.168.2.0     192.168.2.255   Local2
[admin@DHCP-Relay] ip address>
```

DHCP Server Setup

To setup 2 DHCP Servers on DHCP-Server router add 2 pools. For networks 192.168.1.0/24 and 192.168.2.0:

```
/ip pool add name=Local1-Pool ranges=192.168.1.11-192.168.1.100
/ip pool add name=Local1-Pool ranges=192.168.2.11-192.168.2.100
[admin@DHCP-Server] ip pool> print
```

```
 # NAME                                            RANGES
 0 Local1-Pool                                     192.168.1.11-192.168.1.100
 1 Local2-Pool                                     192.168.2.11-192.168.2.100
[admin@DHCP-Server] ip pool>
```

Create DHCP Servers:

```
/ip dhcp-server add interface=To-DHCP-Relay relay=192.168.1.1 \
   address-pool=Local1-Pool name=DHCP-1 disabled=no
/ip dhcp-server add interface=To-DHCP-Relay relay=192.168.2.1 \
   address-pool=Local2-Pool name=DHCP-2 disabled=no
[admin@DHCP-Server] ip dhcp-server> print
Flags: X - disabled, I - invalid
 #   NAME         INTERFACE     RELAY           ADDRESS-POOL LEASE-TIME ADD-ARP
 0   DHCP-1       To-DHCP-Relay 192.168.1.1     Local1-Pool  3d00:00:00
 1   DHCP-2       To-DHCP-Relay 192.168.2.1     Local2-Pool  3d00:00:00
[admin@DHCP-Server] ip dhcp-server>
```

Configure respective networks:

```
/ip dhcp-server network add address=192.168.1.0/24 gateway=192.168.1.1 \
   dns-server=159.148.60.20
/ip dhcp-server network add address=192.168.2.0/24 gateway=192.168.2.1 \
   dns-server 159.148.60.20
[admin@DHCP-Server] ip dhcp-server network> print
 # ADDRESS            GATEWAY         DNS-SERVER      WINS-SERVER     DOMAIN
 0 192.168.1.0/24     192.168.1.1     159.148.60.20
 1 192.168.2.0/24     192.168.2.1     159.148.60.20
[admin@DHCP-Server] ip dhcp-server network>
```

DHCP Relay Config

Configuration of DHCP-Server is done. Now let's configure DHCP-Relay:

```
/ip dhcp-relay add name=Local1-Relay interface=Local1 \
   dhcp-server=192.168.0.1 local-address=192.168.1.1 disabled=no
/ip dhcp-relay add name=Local2-Relay interface=Local2 \
   dhcp-server=192.168.0.1 local-address=192.168.2.1 disabled=no
[admin@DHCP-Relay] ip dhcp-relay> print
Flags: X - disabled, I - invalid
 #   NAME                        INTERFACE      DHCP-SERVER     LOCAL-ADDRESS
 0   Local1-Relay                Local1         192.168.0.1     192.168.1.1
 1   Local2-Relay                Local2         192.168.0.1     192.168.2.1
[admin@DHCP-Relay] ip dhcp-relay>
```

[Top | Back to Content **]**

Manual:IP/DHCP Server

Applies to RouterOS: v3, v4, v5+

Summary

Standards: `RFC 2131, RFC 3315, RFC 3633`
Package: `dhcp`

The DHCP (Dynamic Host Configuration Protocol) is needed for easy distribution of IP addresses in a network. The MikroTik RouterOS implementation includes both server and client parts and is compliant with RFC 2131.

The router supports an individual server for each Ethernet-like interface. The MikroTik RouterOS DHCP server supports the basic functions of giving each requesting client an IP address/netmask lease, default gateway, domain name, DNS-server(s) and WINS-server(s) (for Windows clients) information (set up in the DHCP networks submenu)

In order DHCP server to work, you must set up also IP pools (do not include the DHCP server's own IP address into the pool range) and DHCP networks.

It is also possible to hand out leases for DHCP clients using the RADIUS server, here are listed the parameters for used in RADIUS server.

Access-Request:

- NAS-Identifier - router identity
- NAS-IP-Address - IP address of the router itself
- NAS-Port - unique session ID
- NAS-Port-Type - Ethernet
- Calling-Station-Id - client identifier (active-client-id)
- Framed-IP-Address - IP address of the client (active-address)
- Called-Station-Id - name of DHCP server
- User-Name - MAC address of the client (active-mac-address)
- Password - ""

Access-Accept:

- Framed-IP-Address - IP address that will be assigned to client
- Framed-Pool - ip pool from which to assign ip address to client
- Rate-Limit - Datarate limitation for DHCP clients. Format is: rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time][priority] [rx-rate-min[/tx-rate-min]]]]. All rates should be numbers with optional 'k' (1,000s) or 'M' (1,000,000s). If tx-rate is not specified, rx-rate is as tx-rate too. Same goes for tx-burst-rate and tx-burst-threshold and tx-burst-time. If both rx-burst-threshold and tx-burst-threshold are not specified (but burst-rate is specified), rx-rate and tx-rate are used as burst thresholds. If both rx-burst-time and tx-burst-time are not specified, 1s is used as default. Priority takes values 1..8, where 1 implies the highest priority, but 8 - the lowest. If rx-rate-min and tx-rate-min are not specified rx-rate and tx-rate values are used. The rx-rate-min and tx-rate-min values can not exceed rx-rate and tx-rate values.
- Ascend-Data-Rate - tx/rx data rate limitation if multiple attributes are provided, first limits tx data rate, second - rx data rate. If used together with Ascend-Xmit-Rate, specifies rx rate. 0 if unlimited

- Ascend-Xmit-Rate - tx data rate limitation. It may be used to specify tx limit only instead of sending two sequential Ascend-Data-Rate attributes (in that case Ascend-Data-Rate will specify the receive rate). 0 if unlimited
- Session-Timeout - max lease time (lease-time)

Note: Currently DHCP server requires real interface to receive raw ethernet packets. It cannot function correctly on dummy (empty bridge) interface.

Quick Setup Guide

RouterOS has built in command that lets you easily set up DHCP server. Lets say we want to configure DHCP server on ether1 interface to lend addresses from 192.168.0.2 to 192.168.0.254 which belong to the 192.168.0.0/24 network. The gateway and DNS server is 192.168.0.1.

From `/ip dhcp-server` menu run **setup** command and follow instructions:

```
[admin@MikroTik] ip dhcp-server> setup
Select interface to run DHCP server on

dhcp server interface: ether1
Select network for DHCP addresses

dhcp address space: 192.168.0.0/24
Select gateway for given network

gateway for dhcp network: 192.168.0.1
Select pool of ip addresses given out by DHCP server

addresses to give out: 192.168.0.2-192.168.0254
Select DNS servers

dns servers: 192.168.0.1
Select lease time

lease time: 3d
[admin@MikroTik] ip dhcp-server>
```

The wizard has made the following configuration based on the answers above:

```
[admin@MikroTik] ip dhcp-server> print
Flags: X - disabled, I - invalid
  #   NAME            INTERFACE RELAY           ADDRESS-POOL LEASE-TIME ADD-ARP
  0   dhcp1           ether1    0.0.0.0         dhcp_pool1   3d         no

[admin@MikroTik] ip dhcp-server> network print
  # ADDRESS            GATEWAY         DNS-SERVER      WINS-SERVER     DOMAIN
  0 192.168.0.0/24       192.168.0.1       192.168.0.1

[admin@MikroTik] ip dhcp-server> /ip pool print
  # NAME                                      RANGES
  0 dhcp_pool1                                192.168.0.2-192.168.0.254
```

```
[admin@MikroTik] ip dhcp-server>
```

IPv6

Starting from v5.8 RouterOS supports IPv6 prefix delegation according to RFC 3315 and RFC 3633.

Starting from v5.9, DHCPv6 server configuration was moved to /ipv6 sub-menu. `Read-more >>`

General

Sub-menu: `/ip dhcp-server`

Property	Description
add-arp (*yes* | *no*; Default: **no**)	Whether to add dynamic ARP entry. If set to `no` either ARP mode should be enabled on that interface or static ARP entries should be administratively defined in `/ip arp` submenu.
address-pool (*string* | *static-only*; Default: **static-only**)	IP pool, from which to take IP addresses for the clients. If set to **static-only**, then only the clients that have a static lease (added in lease submenu) will be allowed.
always-broadcast (*yes* | *no*; Default: **no**)	Always send replies as broadcasts.
authoritative (*after-10sec-delay* | *after-2sec-delay* | *yes* | *no*; Default: **after-2sec-delay**)	Option changes the way how server responds to DHCP requests: • `yes` - replies to clients request for an address that is not available from this server, dhcp server will send negative acknowledgment (DHCPNAK) • `no` - dhcp server ignores clients requests for addresses that are not available from this server • `after-10sec-delay` - requests with "secs < 10" will be processed as in "no" setting case and requests with "secs >= 10" will be processed as in "yes" case. • `after-2sec-delay` - requests with "secs < 2" will be processed as in "no" setting case and requests with "secs >= 2" will be processed as in "yes" case. If all requests with "secs < x" should be ignored, then **delay-thershold=x** setting should be used.
boot-support (*none* | *static* | *dynamic*; Default: **static**)	Support for BOOTP clients: • `none` - do not respond to BOOTP requests • `static` - offer only static leases to BOOTP clients • `dynamic` - offer static and dynamic leases for BOOTP clients
delay-threshold (*time* | *none*; Default: **none**)	If secs field in DHCP packet is smaller than delay-threshold, then this packet is ignored. If set to **none** - there is no threshold (all DHCP packets are processed)
interface (*string*; Default:)	Interface on which server will be running.
lease-time (*time*; Default: **72h**)	The time that a client may use the assigned address. The client will try to renew this address after a half of this time and will request a new address after time limit expires.
name (*string*; Default:)	Reference name
relay (*IP*; Default: **0.0.0.0**)	The IP address of the relay this DHCP server should process requests from: • `0.0.0.0` - the DHCP server will be used only for direct requests from clients (no DHCP really allowed) • `255.255.255.255` - the DHCP server should be used for any incomming request from a DHCP relay except for those, which are processed by another DHCP server that exists in the `/ip dhcp-server` submenu.
src-address (*IP*; Default: **0.0.0.0**)	The address which the DHCP client must send requests to in order to renew an IP address lease. If there is only one static address on the DHCP server interface and the source-address is left as 0.0.0.0, then the static address will be used. If there are multiple addresses on the interface, an address in the same subnet as the range of given addresses should be used.
use-radius (*yes* | *no*; Default: **no**)	Whether to use RADIUS server for dynamic leases

Menu specific commands

Property	Description
setup ()	Start DHCP server setup wizard, which guides you through the steps to easily create all necessary configuration. Read more>>

Lease Store Configuration

Sub-menu: /ip dhcp-server config

This sub-menu allows to configure how often DHCP leases will be stored on disk. If they would be saved on disk on every lease change, a lot of disk writes would happen which is very bad for Compact Flash (especially, if lease times are very short). To minimize writes on disk, all changes are saved on disk every store-leases-disk seconds. Additionally leases are always stored on disk on graceful shutdown and reboot.

Note: Manual changes to leases - addition/removal of static lease, removal of dynamic lease will cause changes to be pushed for this lease to storage.

This sub-menu has only one configurable property:

Property	Description
store-leases-disk (*time* | *immediately* | *never*; Default: **5m**)	How frequently lease changes should be stored on disk

Networks

Sub-menu: /ip dhcp-server network

Property	Description
address (*IP/netmask*; Default:)	the network DHCP server(s) will lend addresses from
boot-file-name (*string*; Default:)	Boot file name
dhcp-option (*string*; Default:)	Add additional DHCP options from option list.
dns-server (*string*; Default:)	the DHCP client will use these as the default DNS servers. Two comma-separated DNS servers can be specified to be used by DHCP client as primary and secondary DNS servers
domain (*string*; Default:)	The DHCP client will use this as the 'DNS domain' setting for the network adapter.
gateway (*IP*; Default: **0.0.0.0**)	The default gateway to be used by DHCP Client.
netmask (*integer: 0..32*; Default: **0**)	The actual network mask to be used by DHCP client. If set to '0' - netmask from network address will be used.
next-server (*IP*; Default:)	IP address of next server to use in bootstrap.
ntp-server (*IP*; Default:)	the DHCP client will use these as the default NTP servers. Two comma-separated NTP servers can be specified to be used by DHCP client as primary and secondary NTP servers
wins-server (*IP*; Default:)	The Windows DHCP client will use these as the default WINS servers. Two comma-separated WINS servers can be specified to be used by DHCP client as primary and secondary WINS servers

Leases

Sub-menu: `/ip dhcp-server lease`

DHCP server lease submenu is used to monitor and manage server's leases. The issued leases are showed here as dynamic entries. You can also add static leases to issue a particular client (identified by MAC address) the desired IP address.

Generally, the DHCP lease it allocated as follows:

- an unused lease is in waiting state
- if a client asks for an IP address, the server chooses one
- if the client will receive statically assigned address, the lease becomes offered, and then bound with the respective lease time
- if the client will receive a dynamic address (taken from an IP address pool), the router sends a ping packet and waits for answer for 0.5 seconds. During this time, the lease is marked testing
- in case, the address does not respond, the lease becomes offered, and then bound with the respective lease time
- in other case, the lease becomes busy for the lease time (there is a command to retest all busy addresses), and the client's request remains unanswered (the client will try again shortly)

A client may free the leased address. The dynamic lease is removed, and the allocated address is returned to the address pool. But the static lease becomes busy until the client will reacquire the address.

Note: that the IP addresses assigned statically are not probed.

Properties

Property	Description
address (*IP*; Default:)	Specify ip address (or ip pool) for static lease. If set to **0.0.0.0** - pool from server will be used
address-list (*string*; Default:)	Address list to which address will be added if lease is bound.
always-broadcast (*yes* | *no*; Default:)	Send all repies as broadcasts
block-access (*yes* | *no*; Default: **no**)	Block access for this client
client-id (*string*; Default:)	If specified, must match DHCP 'client identifier' option of the request
lease-time (*time*; Default: **0s**)	Time that the client may use the address. If set to **0s** lease will never expire.
mac-address (*MAC*; Default: **00:00:00:00:00:00**)	If specified, must match the MAC address of the client
src-mac-address (*MAC*; Default:)	Source MAC address
use-src-mac (*MAC*; Default:)	Use this source MAC address instead

Read only properties

Property	Description
active-address (*IP*)	Actual IP address for this lease
active-client-id (*string*)	Actual client-id of the client
active-mac-address (*MAC*)	Actual MAC address of the client
active-server (*list*)	Actual dhcp server, which serves this client
agent-circuit-id (*string*)	Circuit ID of DHCP relay agent
agent-remote-id (*string*)	Remote ID, set by DHCP relay agent
blocked (*flag*)	Whether the lease is blocked
expires-after (*time*)	Time until lease expires
host-name (*text*)	Shows host name option from last received DHCP request
radius (*yes* | *no*)	Shows, whether this dynamic lease is authenticated by RADIUS or not
rate-limit (*string*)	Sets rate limit for active lease. Format is: rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time]]]]. All rates should be numbers with optional 'k' (1,000s) or 'M' (1,000,000s). If tx-rate is not specified, rx-rate is as tx-rate too. Same goes for tx-burst-rate and tx-burst-threshold and tx-burst-time. If both rx-burst-threshold and tx-burst-threshold are not specified (but burst-rate is specified), rx-rate and tx-rate is used as burst thresholds. If both rx-burst-time and tx-burst-time are not specified, 1s is used as default
server (*string*)	Server name which serves this client
status (*waiting* | *testing* | *authorizing* | *busy* | *offered* | *bound*)	Lease status: • **waiting** - not used static lease • **testing** - testing whether this address is used or not (only for dynamic leases) by pinging it with timeout of 0.5s • **authorizing** - waiting for response from radius server • **busy** - this address is assigned statically to a client or already exists in the network, so it can not be leased • **offered** - server has offered this lease to a client, but did not receive confirmation from the client • **bound** - server has received client's confirmation that it accepts offered address, it is using it now and will free the address not later, than the lease time will be over

Menu specific commands

Property	Description
check-status (*id*)	Check status of a given busy dynamic lease, and free it in case of no response
make-static (*id*)	Convert a dynamic lease to a static one

Alerts

Sub-menu: `/ip dhcp-server alert`

To find any rogue DHCP servers as soon as they appear in your network, DHCP Alert tool can be used. It will monitor ethernet for all DHCP replies and check, whether this reply comes from a valid DHCP server. If reply from unknown DHCP server is detected, alert gets triggered:

```
[admin@MikroTik] ip dhcp-server alert>/log print
00:34:23 dhcp,critical,error,warning,info,debug dhcp alert on Public:
    discovered unknown dhcp server, mac 00:02:29:60:36:E7, ip 10.5.8.236
[admin@MikroTik] ip dhcp-server alert>
```

When the system alerts about a rogue DHCP server, it can execute a custom script.

As DHCP replies can be unicast, rogue dhcp detector may not receive any offer to other dhcp clients at all. To deal with this, rogue dhcp detector acts as a dhcp client as well - it sends out dhcp discover requests once a minute

Properties

Property	Description
alert-timeout (*none* | *time*; Default: **none**)	Time, after which alert will be forgotten. If after that time the same server will be detected, new alert will be generated. If set to **none** timeout will never expire.
interface (*string*; Default:)	Interface, on which to run rogue DHCP server finder.
on-alert (*string*; Default:)	Script to run, when an unknown DHCP server is detected.
valid-server (*string*; Default:)	List of MAC addresses of valid DHCP servers.

Read only properties

Property	Description
unknown-server (*string*)	List of MAC addresses of detected unknown DHCP servers. Server is removed from this list after alert-timeout

Menu specific commands

Property	Description
reset-alert (*id*)	Clear all alerts on an interface

DHCP Options

Sub-menu: `/ip dhcp-server option`

With help of DHCP Option list, it is possible to define additional custom options for DHCP Server to advertise.

According to the DHCP protocol, a parameter is returned to the DHCP client only if it requests this parameter, specifying the respective code in DHCP request Parameter-List (code 55) attribute. If the code is not included in Parameter-List attribute, DHCP server will not send it to the DHCP client.

Properties

Property	Description
code (*integer:1..254*; Default:)	dhcp option code. All codes are available at [1]
name (*string*; Default:)	Descriptive name of the option
value (*string*; Default:)	Parameter's value in form of a string. If the string begins with "0x", it is assumed as a hexadecimal value

Example

Classless route adds specified route in clients routing table. In our example it will add dst-address=160.0.0.0/24 gateway=10.1.101.1

```
/ip dhcp-server option
add code=121 name=classless value=0x18A000000A016501000A016501
/ip dhcp-server network
set 0 dhcp-option=classless
```

Result:

```
[admin@MikroTik] /ip route> print
Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp, o - ospf,
m - mme, B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 0 ADS  0.0.0.0/0                          10.1.101.1         0
 1 ADS  160.0.0.0/24                       10.1.101.1         0
```

Configuration Examples

[Top | Back to Content]

References

[1] http://www.iana.org/assignments/bootp-dhcp-parameters

Manual:IP/DNS

Applies to RouterOS: v4.6

DNS cache is used to minimize DNS requests to an external DNS server as well as to minimize DNS resolution time. This is a simple recursive DNS server with local items.

Specifications

- Packages required: system
- License required: Level1
- Submenu level: /ip dns
- Standards and Technologies: DNS
- Hardware usage: Not significant

Description

A MikroTik router with DNS feature enabled can be set as a DNS server for any DNS-compliant client. Moreover, MikroTik router can be specified as a primary DNS server under its dhcp-server settings. When the remote requests are enabled, the MikroTik router responds to TCP and UDP DNS requests on port 53.

DNS Cache Setup

- Submenu level: **/ip dns**

Description

DNS facility is used to provide domain name resolution for router itself as well as for the clients connected to it.

Property Description

Property	Desciption
allow-remote-requests (*yes* | *no; default: no*)	specifies whether to allow network requests
cache-max-ttl (*time; default: 1w*)	specifies maximum time-to-live for cache records. In other words, cache records will expire unconditionally after cache-max-ttl time. Shorter TTL received from DNS servers are respected
cache-size (*integer: 512..10240; default: 2048KiB*)	specifies the size of DNS cache in KiB
cache-used (*read-only: integer*)	displays the current cache size in KiB
servers (*IPv4/IPv6 address list; default: 0.0.0.0*)	comma seperated list of DNS server IP addresses

Note: Prior RouterOS v4.6 DNS servers in CLI was set up using fields *primary-dns* and *secondary-dns* starting from mentioned version these two fields are replaced with one field *servers* where all DNS server IP addresses should be listed

Note: If the property use-peer-dns under /ip dhcp-client is set to yes then primary-dns under /ip dns will change to a DNS address given by DHCP Server.

Example

To set 159.148.60.2 as the primary DNS server and allow the router to be used as a DNS server, do the following:

```
[admin@MikroTik] ip dns> set servers=159.148.60.2 \
\... allow-remote-requests=yes
[admin@MikroTik] ip dns> print
                servers: 159.148.60.2
  allow-remote-requests: yes
             cache-size: 2048KiB
          cache-max-ttl: 1w
             cache-used: 7KiB
[admin@MikroTik] ip dns>
```

Cache Monitoring

- Submenu level: **/ip dns cache**

Description

This menu provides a list with all address (DNS type "A") records stored on the server

Property Description

Property	Desciption
address (read-only: IP address)	IP address of the host
name (read-only: name)	DNS name of the host
ttl (read-only: time)	remaining time-to-live for the record

All DNS Entries

- Submenu level: **/ip dns cache all**

Description

This menu provides a complete list with all DNS records stored on the server

Property Description

Property	Desciption
data (read-only: text)	DNS data field. IP address for type "A" records. Other record types may have different contents of the data field (like hostname or arbitrary text)
name (read-only: name)	DNS name of the host
ttl (read-only: time)	remaining time-to-live for the record
type (read-only: text)	DNS record type

Static DNS Entries

- Submenu level: **/ip dns static**

Description

The MikroTik RouterOS has an embedded DNS server feature in DNS cache. It allows you to link the particular domain names with the respective IP addresses and advertize these links to the DNS clients using the router as their DNS server. This feature can also be used to provide fake DNS information to your network clients. For example, resolving any DNS request for a certain set of domains (or for the whole Internet) to your own page.

The server is capable of resolving DNS requests based on POSIX basic regular expressions, so that multiple requets can be matched with the same entry. In case an entry does not conform with DNS naming standards, it is considered a regular expression and marked with ‘R’ flag. The list is ordered and is checked from top to bottom. Regular expressions are checked first, then the plain records.

Property Description

Property	Desciption
address (IP address)	IP address to resolve domain name with
name (text)	DNS name to be resolved to a given IP address. May be a regular expression
ttl (time)	time-to-live of the DNS record

Notes

Reverse DNS lookup (Address to Name) of the regular expression entries is not possible. You can, however, add an additional plain record with the same IP address and specify some name for it.

Remember that the meaning of a dot (.) in regular expressions is any character, so the expression should be escaped properly. For example, if you need to match anything within example.com domain but not all the domains that just end with *example.com*, like *www.another-example.com*, use *name=".*\\.example\\.com"*

Regular expression matching is significantly slower than of the plain entries, so it is advised to minimize the number of regular expression rules and optimize the expressions themselves. Example

To add a static DNS entry for www.example.com to be resolved to 10.0.0.1 IP address:

```
[admin@MikroTik] ip dns static> add name www.example.com address=10.0.0.1
[admin@MikroTik] ip dns static> print
Flags: D - dynamic, X - disabled, R - regexp
 #     NAME               ADDRESS                                  TTL
 0     www.example.com    10.0.0.1                                 1d
[admin@MikroTik] ip dns static>
```

Flushing DNS cache

- Command name: **/ip dns cache flush**

Command Description

Command	Desciption
flush	clears internal DNS cache

Example

```
[admin@MikroTik] ip dns> cache flush
[admin@MikroTik] ip dns> print
              servers: 159.148.60.2
    allow-remote-requests: yes
               cache-size: 2048 KiB
            cache-max-ttl: 1w
               cache-used: 10 KiB
[admin@MikroTik] ip dns>
```

See Also

- http://www.freesoft.org/CIE/Course/Section2/3.htm
- http://www.networksorcery.com/enp/protocol/dns.htm
- RFC1035 [1]

References

[1] http://www.ietf.org/rfc/rfc1035.txt?number=1035

Manual:Tools/Dynamic DNS

Applies to RouterOS: 2.9, v3, v4 +

Summary

Sub-menu: `/tool dns-update`
Standards: `RFC 2136, RFC 3007`

Dynamic DNS Update Tool gives a way to keep domain name pointing to dynamic IP address. It works by sending domain name system update request to name server, which has a zone to be updated. Secure DNS updates are also supported.

The DNS update tool supports only one algorithm - **hmac-md5**. It's the only proposed algorithm for signing DNS messages.

Note: DNS update tool works only with BIND server, it will not work with DynDNS, EveryDNS or any other similar service. For these services other methods should be used. `Read more >>`

Properties

Property	Description
address (*IP*; Default:)	Defines IP address associated with the domain name.
dns-server (*IP*; Default:)	DNS server to send update to.
key (*string*; Default:)	Authorization key to access the server.
key-name (*string*; Default:)	Authorization key name (like a username) to access the server.
name (*string*; Default:)	Name to attach with the IP address.
ttl (*integer*; Default:)	Time to live for the item (in seconds).
zone (*string*; Default:)	DNS zone where to update the domain name in.

Note: that the system clock time on your router can't differ from the DNS server's time more than 5 minutes. Otherwise the DNS server will ignore this request.

Example

To tell 23.34.45.56 DNS server to (re)associate mydomain name in the myzone.com zone with 68.42.14.4 IP address specifying that the name of the key is dns-update-key and the actual key is update:

```
[admin@MikroTik] tool> dns-update dns-server=23.34.45.56 name=mydomain \
\... zone=myzone.com address=68.42.14.4 key-name=dns-update-key key=update
```

[Top | Back to Content]

Manual:IPv6/DHCP Client

Applies to RouterOS: v5.9 +

Summary

Currently DHCPv6 client can receive only delegated prefix from DHCPv6-PD server.

Quick setup example

This simple example demonstrates how to enable dhcp client to receive IPv6 prefix and add it to the pool.

```
/ipv6 dhcp-client add pool-name=test-ipv6 pool-prefix-length=64 interface=ether13
```

Detailed print should show status of the client and we can verify if prefix is received

```
[admin@x86-test] /ipv6 dhcp-client> print detail
Flags: D - dynamic, X - disabled, I - invalid
 0    interface=bypass pool-name="test-ipv6" pool-prefix-length=64 status=bound
      prefix=2001:db8:7501:ff04::/62  expires-after=2d23h11m53s
```

Notice that server gave us prefix 2a02:610:7501:ff04::/62 . And it should be also added to ipv6 pools

```
[admin@MikroTik] /ipv6 pool> print
Flags: D - dynamic
 #   NAME                        PREFIX                                  PREFIX-LENGTH
 0 D test-ipv6                   2001:db8:7501:ff04::/62                            64
```

It works! Now you can use this pool, for example, for pppoe clients.

Properties

Sub-menu: `/ipv6 dhcp-client`

Property	Description
add-default-route (*yes* | *no*; Default: **no**)	Whether to add default IPv6 route after client connects.
comment (*string*; Default:)	Short description of the client
disabled (*yes* | *no*; Default: **no**)	
interface (*string*; Default:)	Interface on which DHCPv6 client will be running.
pool-name (*string*; Default:)	Name of the IPv6 pool in which received IPv6 prefix will be added
pool-prefix-length (*string*; Default:)	Prefix length parameter that will be set for IPv6 pool in which received IPv6 prefix is added. Prefix length must be greater than the length of received prefix, otherwise prefix-length will be set to received prefix length + 8 bits.

Status

Command `/ipv6 dhcp-client print detail` will show current status of dhcp client and **read-only** properties listed in table below:

Property	Description
duid (*string*)	Auto generated DUID that is sent to the server. DUID is generated using one of the MAC addresses available on the router.
dynamic (*yes* | *no*)	
expires-after (*time*)	Time when the IPv6 prefix expires (specified by the DHCPv6 server).
invalid (*yes* | *no*)	Shows whether configuration is invalid.
prefix (*IPv6 prefix*)	Shows received IPv6 prefix from DHCPv6-PD server
status (*stopped* | *searching* | *requesting...* | *bound* | *renewing* | *rebinding* | *error* | *stopping*)	Shows the status of DHCPv6 Client: • **stopped** - dhcpv6 client is stopped • **searching** - sending "solicit" and trying to get "advertise" • **requesting** - sent "request" waiting for "reply" • **bound** - received "reply". Prefix assigned. • **renewing** - sent "renew", waiting for "reply" • **rebinding** - sent "rebind", waiting for "reply" • **error** - reply was not received in time or some other error ocurred. • **stopping** - sent "release"

To determine what IAID will be used, convert internal ID of an interface on which DHCP client is running from hex to decimal.

For example, DHCP client is running on interface pppoe-out1. To get internal ID use following command

```
[admin@t36] /interface> :put [find name="pppoe-out1"]
*15
```

Now convert hex value 15 to decimal and you get IAID=21

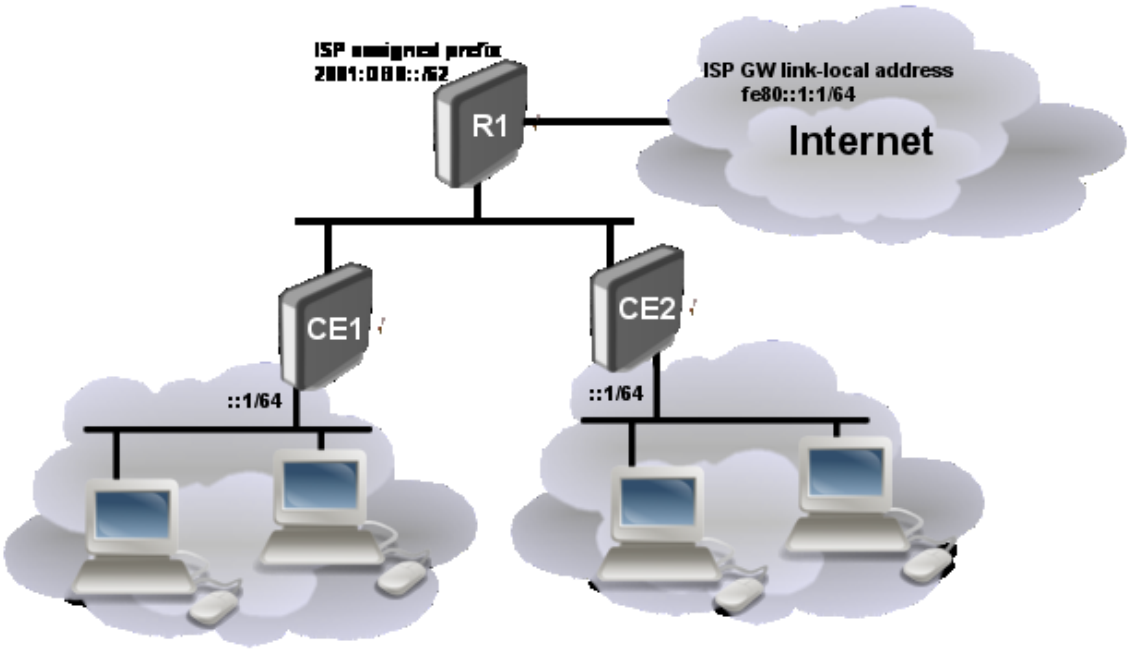
Menu specific commands

Property	Description
release (*numbers*)	Release current binding and restart DHCPv6 client
renew (*numbers*)	Renew current leases. If the renew operation was not successful, client tries to reinitialize lease (i.e. it starts lease request procedure (rebind) as if it had not received an IP address yet)

Application Examples

Use received prefix for local RA

Consider following setup:



- ISP is routing prefix 2001:DB8::/62 to the router R1
- Router R1 runs DHCPv6 server to delegate /64 prefixes to the customer routers CE1 CE2
- DHCP client on routers CE1 and CE2 receives delegated /64 prefix from the DHCP server (R1).
- Client routers uses received prefix to set up RA on the local interface

Configuration

R1

```
/ipv6 route
add gateway=fe80::1:1%to-ISP

/ipv6 pool
add name=myPool prefix=2001:db8::/62 prefix-length=64

/ipv6 dhcp-server
add address-pool=myPool disabled=no interface=to-CE-routers lease-time=3m name=server1
```

CE1

```
/ipv6 dhcp-client
add interface=to-R1 pool-name=my-ipv6

/ipv6 address
add address=::1/64 from-pool=my-ipv6 interface=to-clients advertise=yes
```

CE2

```
/ipv6 dhcp-client
add interface=to-R1 pool-name=my-ipv6

/ipv6 address
add address=::1/64 from-pool=my-ipv6 interface=to-clients advertise=yes
```

Check the status

After configuration is complete we can verify that each CE router received its own prefix

On server:

```
[admin@R1] /ipv6 dhcp-server binding> print
Flags: X - disabled, D - dynamic
 #   ADDRESS                                 DUID               IAID SERVER         STATUS
 1 D 2001:db8:1::/64                         0019d1393536        566 server1        bound
 2 D 2001:db8:2::/64                         0019d1393535        565 server1        bound
```

On client:

```
[admin@CE1] /ipv6 dhcp-client> print
Flags: D - dynamic, X - disabled, I - invalid
 #    INTERFACE           STATUS        PREFIX
 0    to-R1               bound         2001:db8:1::/64

[admin@CE1] /ipv6 dhcp-client> /ipv6 pool print
Flags: D - dynamic
 #   NAME                PREFIX                                PREFIX-LENGTH
 0 D my-ipv6             2001:db8:1::/64                                  64
```

We can also see that IPv6 address was automatically added from the prefix pool:

```
[admin@CE1] /ipv6 address> print
Flags: X - disabled, I - invalid, D - dynamic, G - global, L - link-local
 #    ADDRESS                                 FROM-POOL INTERFACE     ADVERTISE
 0  G 2001:db8:1::1/64                                  to-clients         yes
..
```

And pool usage shows that 'Address' is allocating the pool

```
[admin@CE1] /ipv6 pool used> print
POOL         PREFIX                                  OWNER          INFO
my-ipv6      2001:db8:1::/64                         Address        to-clients
```

[Top | Back to Content]

Manual:IPv6/DHCP Server

Applies to RouterOS: v5.9+

Summary

Standards: RFC 3315, RFC 3633
Package: dhcp,ipv6

Starting from v5.9 DHCPv6 server is moved to /ipv6 sub menu

Single DUID is used for client and server identification, only IAID will vary between cients corresponding to their assigned interface.

Client binding creates dynamic pool with timeout set to binding's expiration time (note that now dynamic pools can have a timeout), which will be updated every time binding gets renewed.

When client is bound to prefix, DHCP server adds routing information to know how to reach assigned prefix.

Client bindings in server does not show MAC address anymore (as it was in v5.8), DUID (hex) and IAID are used instead. After upgrade MAC addresses will be converted to DUIDs automatically, but due to unknown DUID type and unknown IAID, they should be further updated by user;

General

Sub-menu: /ipv6 dhcp-server

This sub menu lists and allows to configure DHCPv6 servers.

Properties

Property	Description
authoritative (*after-10sec-delay* | *after-2sec-delay* | *yes* | *no*; Default: **after-2sec-delay**)	Whether the DHCP server is the only one DHCP server for the network: • after-10sec-delay - to clients request for an address, dhcp server will wait 10 seconds and if there is another request from the client after this period of time, then dhcp server will offer the address to the client or will send DHCPNAK, if the requested address is not available from this server • after-2sec-delay - to clients request for an address, dhcp server will wait 2 seconds and if there is another request from the client after this period of time, then dhcp server will offer the address to the client or will send DHCPNAK, if the requested address is not available from this server • yes - to clients request for an address that is not available from this server, dhcp server will send negative acknowledgment (DHCPNAK) • no - dhcp server ignores clients requests for addresses that are not available from this server
delay-threshold (*time* | *none*; Default: **none**)	If secs field in DHCP packet is smaller than delay-threshold, then this packet is ignored. If set to **none** - there is no threshold (all DHCP packets are processed)
disabled (*yes* | *no*; Default: **no**)	Whether DHCPv6 server participate in prefix assignment process.
lease-time (*time*; Default: **3d**)	The time that a client may use the assigned address. The client will try to renew this address after a half of this time and will request a new address after time limit expires.
address-pool (*string* | *static-only*; Default: **static-only**)	IPv6 pool, from which to take IPv6 prefix for the clients. If set to **static-only**, then only the clients that have a static binding (added in bindings submenu) will be allowed.

interface (*string*; Default:)	Interface on which server will be running.
name (*string*; Default:)	Reference name

Read-only Properties

Property	Description
dynamic (*yes* | *no*)	
invalid (*yes* | *no*)	

Bindings

Sub-menu: `/ipv6 dhcp-server binding`

DUID is used only for dynamic bindings, so if it changes then client will receive different prefix than previously.

Property	Description
address (*IPv6 prefix*; Default:)	IPv6 prefix that will be assigned to the client
comment (*string*; Default:)	Short description of an item.
disabled (*yes* | *no*; Default: **no**)	Whether item is disabled
life-time (*time*; Default: **3d**)	Time period after which binding expires/
duid (*string*; Default:)	
iaid (*integer [0..4294967295]*; Default:)	
server (*string* | *all*; Default: **all**)	Name of the server. If set to **all**, then binding applies to all created DHCPv6 servers.

Read-only properties

Property	Description
dynamic (*yes* | *no*)	Whether item is dynamically created.
expires-after (*time*)	Time period after which binding expires.
last-seen (*time*)	Time period since client was last seen.
status (*waiting* | *offered* | *bound*)	Three status vales are possible: • **waiting** - Shown for static bindings if it is not used. For dynamic bindings this status is shown if it was used previously, server will wait 10 minutes to allow old client to get this binding, otherwise binding will be cleared and prefix willbe offered to other clients. • **offered** - if **solicit** message was received, and server responded with **advertise** message, but **request** was not received. During this state client have 2 minutes to get this binding, otherwise it is freed or changed status to **waiting** for static bindings. • **bound** - currently bound.

For example, dynamically assigned /62 prefix

```
[admin@RB493G] /ipv6 dhcp-server binding> print detail
Flags: X - disabled, D - dynamic
 0 D address=2a02:610:7501:ff00::/62 duid="1605fcb400241d1781f7" iaid=0
     server=local-dhcp life-time=3d status=bound expires-after=2d23h40m10s
     last-seen=19m50s

 1 D address=2a02:610:7501:ff04::/62 duid="0019d1393535" iaid=2
     server=local-dhcp life-time=3d status=bound expires-after=2d23h43m47s
```

```
     last-seen=16m13s
```

Menu specific commands

Property	Description
make-static ()	Set dynamic binding as static.

Configuration Examples

Enabling IPv6 Prefix delegation

Lets consider that we already have running DHCP server.

To enable IPv6 prefix delegation, first we need to create address pool

```
/ipv6 pool add name=myPool prefix=2001:db8:7501::/60  prefix-length=62
```

Notice that prefix-length is 62 bits, it means that clients will receive /62 prefixes from the /60 pool.

Next step is to enable DHCPv6.

```
/ipv6 dhcp-server add name=myServer address-pool=myPool interface=local
```

To test our server we will set up wide-dhcpv6 on ubuntu machine:

- install wide-dhcpv6-client
- edit "/etc/wide-dhcpv6/dhcp6c.conf" as above

Note: You can use also RouterOS as DHCPv6-PD client. Read more >>

```
interface eth2{
  send ia-pd 0;
};

id-assoc pd {
   prefix-interface eth3{
   sla-id 1;
   sla-len 2;
   };
};
```

- Run DHCPv6 client

```
sudo dhcp6c -d -D -f eth2
```

- Verify that prefix was added to eth3

```
mrz@bumba:/media/aaa$ ip -6 addr
..
2: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
```

```
    inet6 2001:db8:7501:1:200:ff:fe00:0/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::224:1dff:fe17:81f7/64 scope link
       valid_lft forever preferred_lft forever
```

- You can make binding to specific client static, so that it always receives the same prefix

```
[admin@RB493G] /ipv6 dhcp-server binding> print
Flags: X - disabled, D - dynamic
 #   ADDRESS                                      DU       IAID SER.. STATUS
 0 D 2001:db8:7501:1::/62                      16          0 loc.. bound
[admin@RB493G] /ipv6 dhcp-server binding> make-static 0
```

- DHCPv6 also installs route to assigned prefix into IPv6 routing table

```
[admin@RB493G] /ipv6 route> print
Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, o - ospf, b - bgp, U - unreachable
 #      DST-ADDRESS             GATEWAY                 DISTANCE
...
 2 ADS  2001:db8:7501:1::/62    fe80::224:1dff:fe17:8...       1
```

[Top | Back to Content]

Manual:Interface/EoIP

Applies to RouterOS: 2.9, v3, v4+

Summary

Sub-menu: `/interface eoip`
Standards: `GRE RFC 1701`

Ethernet over IP (EoIP) Tunneling is a MikroTik RouterOS protocol that creates an Ethernet tunnel between two routers on top of an IP connection. The EoIP tunnel may run over IPIP tunnel, PPTP tunne or any other connection capable of transporting IP.
When the bridging function of the router is enabled, all Ethernet traffic (all Ethernet protocols) will be bridged just as if there where a physical Ethernet interface and cable between the two routers (with bridging enabled). This protocol makes multiple network schemes possible.
Network setups with EoIP interfaces:

- Possibility to bridge LANs over the Internet
- Possibility to bridge LANs over encrypted tunnels
- Possibility to bridge LANs over 802.11b 'ad-hoc' wireless networks

The EoIP protocol encapsulates Ethernet frames in GRE (IP protocol number 47) packets (just like PPTP) and sends them to the remote side of the EoIP tunnel.

Properties

Property	Description
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; Default: **enabled**)	Address Resolution Protocol mode
keepalive (*integer*; Default: **not set**)	keep-alive timer, sets time interval (seconds) in what keep-alive messages should be received. If 3 messages are missed, interface running flag is removed. For this to work, keepalive has to be set to same value on both ends of the tunnel, since one end is expecting messages from the other one and is sending keepalive messages in that direction.
l2mtu (*integer*; Default:)	Layer2 Maximum transmission unit. Not configurable for EoIP. Read more>>
mac-address (*MAC*; Default:)	Media Access Control number of an interface. The address numeration authority allows to use MAC addresses in the range from **00:00:5E:80:00:00 - 00:00:5E:FF:FF:FF** freely
mtu (*integer*; Default: **1500**)	Layer3 Maximum transmission unit
name (*string*; Default:)	Interface name
remote-address (*IP*; Default:)	IP address of remote end of EoIP tunnel
tunnel-id (*integer: 65536*; Default:)	Unique tunnel identifier, which must match other side of the tunnel

Notes

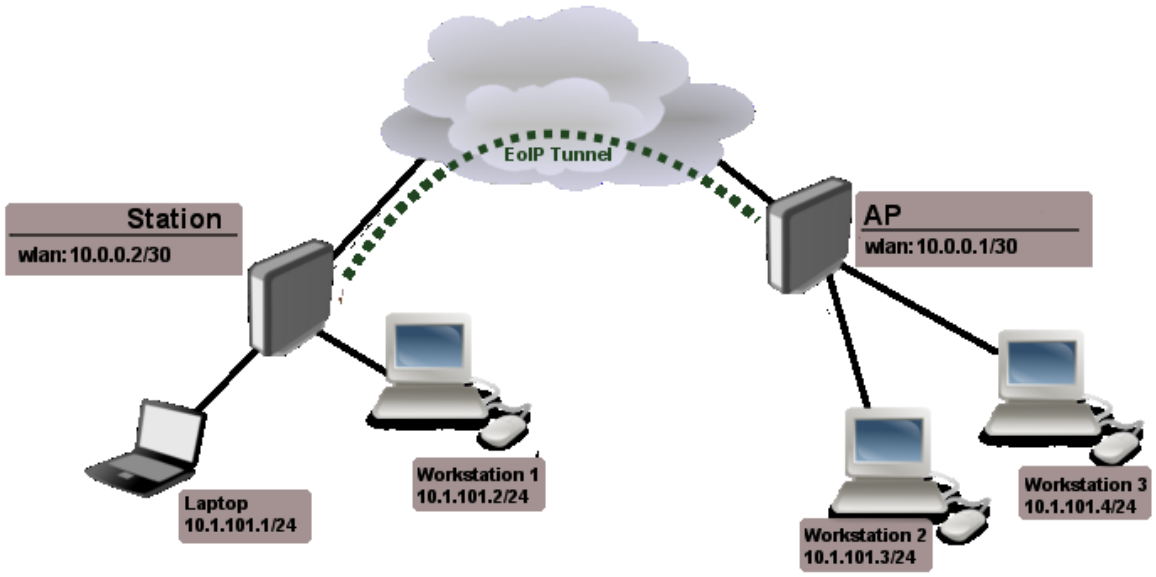
tunnel-id is method of identifying tunnel. It must be unique for each EoIP tunnel.
mtu should be set to 1500 to eliminate packet refragmentation inside the tunnel (that allows transparent bridging of Ethernet-like networks, so that it would be possible to transport full-sized Ethernet frame over the tunnel).
When bridging EoIP tunnels, it is highly recommended to set unique MAC addresses for each tunnel for the bridge algorithms to work correctly. For EoIP interfaces you can use MAC addresses that are in the range from **00:00:5E:80:00:00 - 00:00:5E:FF:FF:FF** , which IANA has reserved for such cases. Alternatively, you can set the second bit of the first byte to mark the address as locally administered address, assigned by network administrator, and use any MAC address, you just need to ensure they are unique between the hosts connected to one bridge.

Note: EoIP tunnel adds at least 42 byte overhead (8byte GRE + 14 byte Ethernet + 20 byte IP)

Setup examples

Let us assume we want to bridge two networks: 'Office LAN' and 'Remote LAN'. By using EoIP setup can be made so that Office and Remote LANs are in the same Layer2 broadcast domain.

Consider following setup:



As you know wireless station cannot be bridged, to overcome this limitation (not involving WDS) we will create EoIP tunnel over the wireless link and bridge it with interfaces connected to local networks.

We will not cower wireless configuration in this example, lets assume that wireless link is already established

At first we create EoIP tunnel on our gateway ...

```
[admin@Our_GW] interface eoip> add name="eoip-remote" tunnel-id=0 \
\... remote-address=10.0.0.2
[admin@Our_GW] interface eoip> enable eoip-remote
[admin@Our_GW] interface eoip> print
Flags: X - disabled, R - running
  0    name=eoip-remote mtu=1500 arp=enabled remote-address=10.0.0.2 tunnel-id=0
[admin@Our_GW] interface eoip>
```

... and on Remote router

```
[admin@Remote] interface eoip> add name="eoip" tunnel-id=0 \
\... remote-address=10.0.0.1
[admin@Remote] interface eoip> enable eoip-main
[admin@Remote] interface eoip> print
Flags: X - disabled, R - running
  0   name=eoip mtu=1500 arp=enabled remote-address=10.0.0.1 tunnel-id=0

[admin@Remote] interface eoip>
```

Next step is to bridge local interfaces with EoIP tunnel On Our GW ...

```
[admin@Our_GW] interface bridge> add
[admin@Our_GW] interface bridge> print
Flags: X - disabled, R - running
 0  R name="bridge1" mtu=1500 arp=enabled mac-address=00:00:00:00:00:00
      protocol-mode=none priority=0x8000 auto-mac=yes
      admin-mac=00:00:00:00:00:00 max-message-age=20s forward-delay=15s
```

```
      transmit-hold-count=6 ageing-time=5m
[admin@Our_GW] interface bridge> port add bridge=bridge1 interface=eoip-remote
[admin@Our_GW] interface bridge> port add bridge=bridge1 interface=office-eth
[admin@Our_GW] interface bridge> port print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE      BRIDGE  PRIORITY PATH-COST
 0    eoip-remote    bridge1 128      10
 1    office-eth     bridge1 128      10
[admin@Our_GW] interface bridge>
```

... and Remote router:

```
[admin@Remote] interface bridge> add
[admin@Remote] interface bridge> print
Flags: X - disabled, R - running
 0  R name="bridge1" mtu=1500 arp=enabled mac-address=00:00:00:00:00:00
      protocol-mode=none priority=0x8000 auto-mac=yes
      admin-mac=00:00:00:00:00:00 max-message-age=20s forward-delay=15s
      transmit-hold-count=6 ageing-time=5m
[admin@Remote] interface bridge> port add bridge=bridge1 interface=ether
[admin@Remote] interface bridge> port add bridge=bridge1 interface=eoip-main
[admin@Remote] interface bridge> port print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE      BRIDGE  PRIORITY PATH-COST
 0    ether          bridge1 128      10
 1    eoip-main      bridge1 128      10
[admin@Remote] interface bridge>
```

Now both sites are in the same Layer2 broadcast domain. You can set up IP addresses from the same network on both sites.

[Top | Back to Content **]**

Manual:Interface/Ethernet

Applies to RouterOS: v3, v4+

Summary

Sub-menu: `/interface ethernet`
Standards: `IEEE 802.3` [1]

MikroTik RouterOS supports various types of Ethernet interfaces.

Properties

Property	Description
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; Default: **enabled**)	Address Resolution Protocol mode
auto-negotiation (*yes* | *no*; Default: **yes**)	When enabled, the interface "advertises" its maximum capabilities to achieve the best connection possible. **Note:** Auto-negotiation must be disabled on both ends, otherwise Ethernets may not work properly. **Note2:** Gigabit link cannot work with auto-negotiation disabled.
bandwidth (*integer/integer*; Default: **unlimited/unlimited**)	Sets max rx/tx bandwidth that will be handled by an interface.
cable-setting (*default* | *short* | *standard*; Default: **default**)	changes the cable length setting (only applicable to NS DP83815/6 cards)
disable-running-check (*yes* | *no*; Default: **yes**)	Disable running check. If this value is set to 'no', the router automatically detects whether the NIC is connected with a device in the network or not.
full-duplex (*yes* | *no*; Default: **yes**)	Defines whether the transmission of data appears in two directions simultaneously
l2mtu (*integer*; Default:)	Layer2 Maximum transmission unit. Read more>>
mac-address (*MAC*; Default:)	Media Access Control number of an interface.
master-port (*name* | *none*; Default: **none**)	Sets switch group master interface
mdix-enable (*yes* | *no*; Default:)	Whether the MDI/X auto crosscable correction feature is enabled for the port
mtu (*integer*; Default: **1500**)	Layer3 Maximum transmission unit
name (*string*; Default:)	Name of an interface
speed (*10Mbps* | *100Mbps* | *1Gbps*; Default: **max available**)	Sets the data transmission speed of the interface. By default, this value is the maximal data rate supported by the interface
poe-out (*auto-on* | *forced-on* | *off*; Default: **off**)	Poe Out settings. `Read more >>`

Property	Description
running (*yes* | *no*)	Whether interface is running. Note that some interface does not have running check and they are always reported as "running"
rx-1024-1518 (*integer*)	Total count of received 1024 to 1518 byte packets
rx-128-255 (*integer*)	Total count of received 128 to 255 byte packets
rx-1519-max (*integer*)	Total count of received packets larger than 1519 bytes
rx-256-511 (*integer*)	Total count of received 256 to 511 byte packets
rx-512-1023 (*integer*)	Total count of received 512 to 1023 byte packets
rx-64 (*integer*)	Total count of received 64 byte packets
rx-65-127 (*integer*)	Total count of received 65 to 127 byte packets
rx-align-error (*integer*)	Total count of received align error messages
rx-broadcast (*integer*)	Total count of received broadcast packets
rx-bytes (*integer*)	Total count of received bytes
rx-fcs-error (*integer*)	Total count of received frames with incorrect checksum
rx-fragment (*integer*)	Total count of received fragmented frames
rx-multicast (*integer*)	Total count of received multicast packets
rx-overflow (*integer*)	
rx-pause (*integer*)	Amount of received pause frames
rx-runt (*integer*)	Amount of received frames shorter than the minimum 64 bytes but with a valid CRC
rx-too-long (*integer*)	
slave (*yes* | *no*)	Whether interface is configured as a slave of another interface (for example Bonding)
switch (*integer*)	ID to which switch chip interface belongs to.
tx-1024-1518 (*integer*)	
tx-128-255 (*integer*)	
tx-1519-max (*integer*)	
tx-256-511 (*integer*)	
tx-512-1023 (*integer*)	
tx-64 (*integer*)	
tx-65-127 (*integer*)	
tx-align-error (*integer*)	
tx-broadcast (*integer*)	
tx-bytes (*integer*)	
tx-fcs-error (*integer*)	
tx-fragment (*integer*)	
tx-multicast (*integer*)	
tx-overflow (*integer*)	
tx-pause (*integer*)	
tx-runt (*integer*)	

tx-too-long (*integer*)

Menu specific commands

Property	Description
blink (*[id, name]*)	Blink Ethernet leds
monitor (*[id, name]*)	Monitor ethernet status. Read more>>
reset-counters (*[id, name]*)	Reset stats counters. Read more>>
reset-mac (*[id, name]*)	Reset MAC address to manufacturers default.

Monitor

`/interface ethernet monitor` command prints out current link, rate and duplex status of an interface.
Properties:

Property	Description
auto-negotiation (*done* | *incomplete*)	Current auto negotiation status. • `done`-negotiation completed • `incomplete`-negotiation failed or not yet completed
default-cable-settings (*short* | *standard*)	default cable length setting (only applicable to NS DP83815/6 cards) • `short`-support short cables • `standard`-support standard cables
full-duplex (*yes* | *no*)	Whether transmission of data occurs in two directions simultaneously
rate (*10Mbps* | *100Mbps* | *1Gbps*)	Actual data rate of the connection
status (*link-ok* | *no-link* | *unknown*)	Current link status of an interface • `link-ok`-the card is connected to the network • `no-link`-the card is not connected to the network • `unknown`-the connection is not recognized (if the card does not report connection status)

Example output of ethernet status:

```
[admin@MikroTik] /interface ethernet> monitor ether1
            status: link-ok
  auto-negotiation: done
              rate: 1Gbps
       full-duplex: yes
```

Stats

RouterOS v3.22 introduces a new command:

```
/interface ethernet print stats
```

This command will display all kinds of other statistics if the interface is supporting them (currently only RB450G ether2-ether5, RB750 ether2-ether5, RB750G ether1-ether5 and also RB1100 ether1-ether10). Complete list of properties can be found in section above
For example, output of ethernet stats on RB450G:

```
[admin@MikroTik] /interface ethernet> print stats
                      name: ether1-gateway ether2-local ether3-local ether4-local ether5-local
               rx-broadcast:                22           31           3666         11
                   rx-pause:                0            0            0            0
               rx-multicast:                4            7            1423         5
               rx-fcs-error:                0            0            2            0
             rx-align-error:                0            0            0            0
                    rx-runt:                0            0            0            0
               rx-fragment:                 0            0            1            0
                     rx-64:                 0            0            0            0
                  rx-65-127:                8            14           21598        10
                 rx-128-255:                0            0            0            0
                 rx-256-511:                18           24           2245         6
                rx-512-1023:                28926        7649         371938       24476
               rx-1024-1518:                0            0            0            0
                rx-1519-max:                0            0            0            0
                rx-too-long:                0            0            0            0
                rx-overflow:                0            0            0            0
                   rx-bytes:                15337844     4063737      199738064    12975401
               tx-broadcast:                13           13           1496         8
                   tx-pause:                0            0            0            0
               tx-multicast:                13           13           1496         8
                tx-underrun:                0            0            0            0
                     tx-64:                 0            0            0            0
                  tx-65-127:                26           26           2992         16
                 tx-128-255:                0            0            0            0
                 tx-256-511:                0            0            0            0
                tx-512-1023:                0            0            0            0
               tx-1024-1518:                0            0            0            0
                tx-1519-max:                0            0            0            0
                tx-too-long:                0            0            0            0
               tx-collision:                0            0            0            0
    tx-excessive-collision:                 0            0            0            0
     tx-multiple-collision:                 0            0            0            0
       tx-single-collision:                 0            0            0            0
     tx-excessive-deferred:                 0            0            0            0
                tx-deferred:                0            0            0            0
           tx-late-collision:               0            0            0            0
                   tx-bytes:                2561         2561         294712       1576
```

Switch

Sub-menu: `/interface ethernet switch`

This submenu allows to configure certain RouterBoard switch chip feature. Read more >>.

PoE out

PoE out settings are only available on RouterBOARD devices that have this hardware feature present.

See more here: PoE-Out

[Top | Back to Content **]**

References

[1] http://grouper.ieee.org/groups/802/3/

Manual:Interface/Gre

Applies to RouterOS: v5+

Summary

Sub-menu: `/interface gre`
Standards: `GRE RFC 1701`

GRE (generic routing encapsulation) is a tunneling protocol that was originally developed by Cisco. It can encapsulate wide variety of protocols creating virtual point-to-point link.

GRE the same as IPIP and EoIP were originally developed as stateless tunnels. Meaning that if remote end of the tunnels goes down all traffic that was routed over the tunnels gets blackholed. To solve this problem RouterOS have added keepalive feature for GRE tunnels.

GRE tunnel adds 24 byte overhead (4-byte gre header + 20-byte IP header).

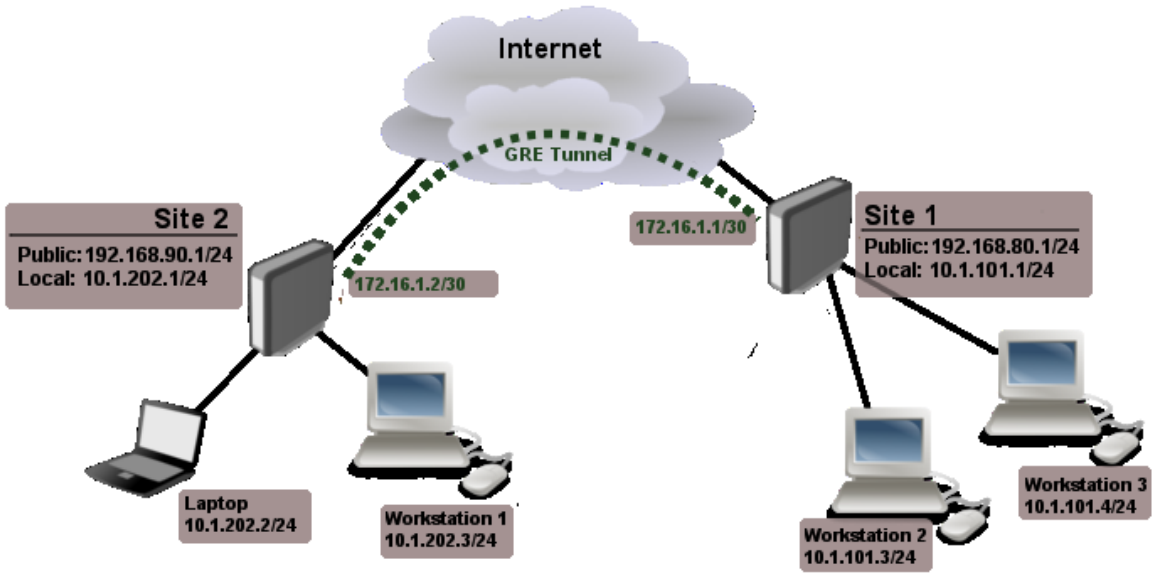
Properties

Property	Description
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; Default:)	Address Resolution Protocol mode
comment (*string*; Default:)	Short description of the tunnel.
disabled (*yes* | *no*; Default: **no**)	Whether tunnel is enabled.
keepalive (*integer [1..4294967295]*; Default:)	Tunnel keepalive timeout in seconds. By default keepalive is disabled.
l2mtu (*integer [0..65536]*; Default: **65535**)	Layer2 Maximum transmission unit.
local-address (*IP*; Default: **0.0.0.0**)	Ip addres that will be used as local tunnel end. If set to 0.0.0.0 then ip address of outgoing interface will be taken.
mtu (*integer [0..65536]*; Default: **1476**)	Layer3 Maximum transmission unit.

name (*string*; Default:)	Name of the tunnel.
remote-address (*IP*; Default:)	IP address of remote tunnel end.

Setup examples

The goal of example is to get Layer 3 connectivity between two remote sites over the internet.



We two sites **Site1** with local network range 10.1.101.0/24 and **Site2** with local network range 10.1.202.0/24.

First step is to create GRE tunnels. Router on site 1:

```
/interface gre add name=myGre remote-address=192.168.90.1 local-address=192.168.80.1
```

Router on site 2:

```
/interface gre add name=myGre remote-address=192.168.80.1 local-address=192.168.90.1
```

As you can see tunnel configuration is quite simple.

Note: In this example keepalive is not configured, so tunnel interface will have **running** flag even if remote tunnel end is not reachable

Now we just need to set up tunnel addresses and proper routing. Router on site 1:

```
/ip address
  add address=172.16.1.1/30 interface=myGre

/ip route
  add dst-address=10.1.202.0/24 gateway=172.16.1.2
```

Router on site 2:

```
/ip address
  add address=172.16.1.2/30 interface=myGre

/ip route
  add dst-address=10.1.101.0/24 gateway=172.16.1.1
```

At this point sites have Layer 3 connectivity over GRE tunnel.

[Top | Back to Content]

Manual:Interface/Gre6

Applies to RouterOS: v5.11+

Summary

Sub-menu: `/interface gre6`

Interface *gre6* is a Generic Routing Encapsulation (GRE) tunnel over IPv6 network. It can encapsulate IPv4 or IPv6 network packets and transfer them over the tunnel.

Properties

Property	Description
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; Default:)	Address Resolution Protocol mode
comment (*string*; Default:)	Short description of the tunnel.
disabled (*yes* | *no*; Default: **no**)	Whether tunnel is enabled.
keepalive (*integer [1..4294967295]*; Default:)	Tunnel keepalive timeout in seconds. By default keepalive is disabled.
l2mtu (*integer [0..65536]*; Default: **65535**)	Layer2 Maximum transmission unit.
local-address (*IP*; Default:)	IPv6 addres that will be used as local tunnel end.
mtu (*integer [0..65536]*; Default: **1476**)	Layer3 Maximum transmission unit.
name (*string*; Default:)	Name of the tunnel.
remote-address (*IP*; Default:)	IP address of remote tunnel end.

Example

on router 1:

```
/interface gre6 add local-address=2001:db8:bad:ee1::a remote-address=2001:db8:f00:baa::b keepalive=5
```

on router 2:

```
/interface gre6 add local-address=2001:db8:f00:baa::b remote-address=2001:db8:bad:ee1::a keepalive=5
```

Manual:Tools/email

Applies to RouterOS: v3, v4, v5 +

Summary

E-mail tool is the utility that allows to send e-mails from the router. Tool can be used to send regular configuration backups and exports to network administrator.

Email tool uses only plain authentication and tls encryption. Other methods are not supported.

Properties

Sub-menu: `/tool e-mail`

This submenu allows to set smtp server that will be used.

Property	Description
from (*string*; Default: **<>**)	Name or email address that will be shown as receiver.
password (*string*; Default: "")	Password used for authenticate to SMTP server.
address (*IP/IPv6 address*; Default: **0.0.0.0**)	SMTP server's IP address.
port (*integer[0..65535]*; Default: **25**)	SMTP server's port.
username (*string*; Default: "")	Username used for authenticate to SMTP server.

Note: All server's configuration (if specified) can be overridden by send command.

Sending Email

Command: `/tool e-mail send`

Send command takes following parameters:

Property	Description
body (*string*; Default:)	Actual body of the email message
cc (*string*; Default:)	Send a copy to listed recipients.
file (*string*; Default:)	Name of the file that will be attached to the email. Only one file can be attached.
from (*string*; Default:)	Name or email address which will appear as sender. If not specified value from server's configuration is used.
password (*string*; Default:)	Password used to authenticate to SMTP server. If not specified value from server's configuration is used.
port (*integer[0..65535]*; Default:)	Port of SMTP server. If not specified, value from server's configuration is used.
server (*IP/IPv6 address*; Default:)	Ip or IPv6 address of SMTP server. If not specified, value from server's configuration is used.
subject (*string*; Default:)	Subject of the message.
tls (*yes|no*; Default: **yes**)	Whether to use tls encryption or not.
to (*string*; Default:)	Destination email address
user (*string*; Default:)	Username used to authenticate to SMTP server. If not specified, value from server's configuration is used.

Basic examples

This example will show how to send email with configuration export every 24hours.

1. Configure SMTP server

```
[admin@MikroTik] /tool e-mail> set server=10.1.1.1 port=25 from="router@mydomain.com"
```

2. Add new script named "export-send"

```
/export file=export
/tool e-mail send to="config@mydomain.com" subject="$[/system identity get name]  export) \
body="$[/system clock get date] configuration file" file=export.rsc
```

3. Add scheduler to run our script

```
/system scheduler
add on-event="export-send" start-time=00:00:00 interval=24h
```

Send e-mail to server using TLS/SSL encryption. For example, Google mail requires that.

1. configure client to connect to correct server

```
/tool e-mail
set address=173.194.77.108
set port=587
set from=myuser@gmail.com
set user=myuser
set password=mypassword
```

2. send e-mail using send command with tls attribute set to yes

```
send to=myuser@anotherdomain.com subject="email test" body="email test" tls=yes
```

[Top | Back to Content]

Manual:Tools/Ping

Applies to RouterOS: v3, v4, v5 +

Summary

Ping uses Internet Control Message Protocol (ICMP) Echo messages to determine if a remote host is active or inactive and to determine the round-trip delay when communicating with it. Ping tool sends ICMP (type 8) message to the host and waits for the ICMP echo-reply (type 0). The interval between these events is called round trip. If the response (that is called pong) has not come until the end of the interval, we assume it has timed out. The second significant parameter reported is ttl (Time to Live). Is is decremented at each machine in which the packet is processed. The packet will reach its destination only when the ttl is greater than the number of routers between the source and the destination.

Properties

Command: `/ping [address] [properties]`

Ping tool can be used to ping IP address and mac address. Mac ping works only to devices that has mac ping server configured. `Read more>>`

Property	Description
arp-ping (*yes* | *no*; Default:)	
count (*integer [0..4294967295]*; Default: **0**)	Total number of packets to send (defult is to send forever until interrupted).
do-not-fragment (; Default:)	If do-not-fragment flag is set packets will not be fragmented if size exceeds interface mtu.
interface (*string*; Default:)	Which interface to use (required when pinging IPv6 address)
interval (*time [10ms..5s]*; Default: **1s**)	how long to wait for response. If no response is received within 1000ms, ping will show as "timed out", but if you will receive a response after 3ms, still the ping program will wait the rest of 997ms until it sends next ping.
routing-table (*string*; Default: **main**)	Which routing table to use to resolve destination. Used in VRF setups.
size (*integer*; Default: **64**)	Packet size to be used in bytes (includes payload and IP header)
src-address (*IPv4,IPv6*; Default:)	IPv4/IPv6 address to be set as packets source. Useful if replies must be sent to specific address.
ttl (*integer [1..255]*; Default:)	Time to live parameter adjustment

Note: If DNS is configured, then DNS name can be used to ping destination

Examples

Ping IP address

```
[admin@dzeltenais_burkaans] > /ping 10.1.101.3
HOST                                     SIZE  TTL TIME  STATUS
```

```
10.1.101.3                              56    64  3ms
10.1.101.3                              56    64  10ms
10.1.101.3                              56    64  7ms
    sent=3 received=3 packet-loss=0% min-rtt=3ms avg-rtt=6ms max-rtt=10ms
```

```
[admin@dzeltenais_burkaans] > /ping 10.1.101.9
HOST                                    SIZE  TTL TIME  STATUS
                                                        timeout
                                                        timeout
                                                        timeout
    sent=3 received=0 packet-loss=100%
```

It is also possible to ping multicast address to discover all hosts belongign to multicast group:

```
[admin@dzeltenais_burkaans] > /ping ff02::1
HOST                                    SIZE  TTL TIME  STATUS
fe80::20c:42ff:fe49:fceb                56    64  1ms   echo reply
fe80::20c:42ff:fe72:a1b0                56    64  1ms   echo reply
fe80::20c:42ff:fe28:7945                56    64  1ms   echo reply
fe80::21a:4dff:fe5d:8e56                56    64  3ms   echo reply
    sent=1 received=4 packet-loss=-300% min-rtt=1ms avg-rtt=1ms max-rtt=3ms
```

Ping large packets:

```
[admin@dzeltenais_burkaans] > /ping 10.1.101.3 size=1600 do-not-fragment
HOST                                    SIZE  TTL TIME  STATUS
                                        576   64  3ms   fragmentation needed and DF set
                                        576   64  6ms   fragmentation needed and DF set
    sent=2 received=2 packet-loss=0% min-rtt=3ms avg-rtt=4ms max-rtt=6ms
```

Ping by DNS name

```
[admin@dzeltenais_burkaans] > /ping www.google.lv
HOST                                    SIZE  TTL TIME  STATUS
74.125.77.99                            56    47  59ms
74.125.77.99                            56    47  85ms
    sent=2 received=2 packet-loss=0% min-rtt=59ms avg-rtt=72ms max-rtt=85ms
```

Ping MAC address

```
[admin@dzeltenais_burkaans] > /ping 00:0C:42:72:A1:B0
HOST                                    SIZE  TTL TIME  STATUS
00:0C:42:72:A1:B0                       56        0ms
00:0C:42:72:A1:B0                       56        0ms
    sent=2 received=2 packet-loss=0% min-rtt=0ms avg-rtt=0ms max-rtt=0ms
```

Mac Ping

Sub-menu: `/mac-server ping`

This submenu allows to enable mac ping server.

When mac ping is enabled, other hosts on the same broadcast domain can use ping tool to ping mac address.

```
[admin@dzeltenais_burkaans] > /tool mac-server ping set enabled=yes
```

[Top | Back to Content]

Manual:Routing/Routing filters

Applies to RouterOS: v3, v4 +

Properties

Sub-menu: `/routing filter`

Note: Values from "set-..." properties are set no matter what action is specified in "action" property.

Property	Description
`action` (*accept* | *discard* | *jump* | *log* | *passthrough* | *reject* | *return*; Default: **passthrough**)	action to perform on route matching the rule. • `accept` - accept the routing information • `discard` - completely exclude matching prefix from further processing. For incoming filters, 'discard' means that information about this route is completely lost. For outgoing filters it's the same as 'reject' • `jump` - pass control to another filter list that should be specified as 'jump-target' parameter • `log` - log message about this match in system log and continue with the next rule in chain • `passthrough` - continue to the next rule in chain • `reject` - reject the routing information for matching prefix. For incoming filters, 'reject' means that information about this route stored in memory, but the route will not become active. For outgoing filters it's the same as 'discard' • `return` - return to the previous chain from which a jump to the current chain took place
`address-family` (*ip|ipv6|l2vpn|l2vpn-cisco|vpnv4*;)	match by BGP address family
`append-bgp-communities` (*integer:integer* | *internet* | *local-as* | *no-advertise* | *no-export*;)	similar to 'set-bgp-communities', but does not delete any existing information about communities
`append-route-targets` (*AsIP|AsNum*;)	Append value to route target EXTENDED_COMMUNITIES path attribute
`bgp-as-path` (*string*;)	unanchored pattern to be searched inside **AS_PATH** attribute of the route. POSIX regular expressions are supported.

bgp-as-path-length (*integer-integer*;)	match length of **AS_PATH BGP** attribute, representing the number of ASes that have been traversed. Read how the **AS_PATH** length is calculated before using this matcher
bgp-atomic-aggregate (*absent* | *present*;)	match **ATOMIC_AGGREGATE** BGP attribute
bgp-communities (*integer:integer* | *internet* | *local-as* | *no-advertise* | *no-export*;)	match the **COMMUNITIES** BGP attribute. Match is done when communities attribute in a route contains all entries from this configured list. But note that if communities list contains 'internet', the whole list always matched.
bgp-local-pref (*integer[-integer]*;)	match **LOCAL_PREF** BGP attribute. If the **LOCAL_PREF** for a route is not set, value 0 is used instead
bgp-med (*integer[-integer]*;)	match **MULTI_EXIT_DISC** BGP attribute. If the **MULTI_EXIT_DISC** for a route is not set, value 0 is used instead
bgp-origin (*igp* | *egp* | *incomplete*;)	match **ORIGIN** BGP attribute. If the **ORIGIN** for a route is not set, value 'incomplete' is used instead
bgp-weight (*signed integer[-signed integer]*;)	match BGP weight property. If this property for a route is not set, value 0 is used instead
chain (*string*;)	chain name to place this rule in. If a chain with the specified name does not exist it will be automatically created • `ospf-in` - predefined filter chain for routes received via OSPF; • `ospf-out` - predefined filter chain for external routes redistributed via OSPF; • `rip-in` - predefined filter chain for routes received via RIP; • `rip-out` - predefined filter chain for external routes redistributed via RIP; • `mme-in` - predefined filter chain for routes received via MME; • `connected-in` - predefined filter chain for all connected routes; • `dynamic-in` - predefined filter chain for all other dynamic routes, i.e. all dynamic routes except (1) those added by routing protocols and (2) connected routes. In this category falls routes added by some external program, for example PPP daemon. **Note** that internal RIP filtering is done using prefix lists [and internal (intra-area) OSPF filtering is not supported yet]
distance (*integer: 0..255[- integer:0..255]*;)	match routes with specific administrative distance
invert-math (*yes* | *no*; Default: **no**)	invert this match, i.e. apply the rule to routes that would fail to match it and vice versa
jump-target (*string*;)	name of the target chain to jump to, if the 'action=jump' is used
locally-originated-bgp (*yes|no*;)	
match-chain (*string*;)	the name of the chain which is used to evaluate the route. If the chain accepts the route, 'match-chain' property produces a true match
ospf-type (*string*;)	OSPF route type matcher
pref-src (*IP address range*;)	match routes with a specific preferred source value
prefix (*IP prefix*; Default: **0.0.0.0/0**)	network prefix to match. If prefix-length is not set, only exact match is done. For example, 0.0.0.0/0 then matches only the default route and nothing else. If network mask is not set, /32 is assumed

prefix-length (*integer*; Default: **0-32**)	network prefix mask length to match. If prefix-length is set, for a route to match the prefix and prefix-length of a rule, the following should hold:
	• the network prefix of the route falls within the range of the prefix of the rule, (i.e.
	  • the network mask of the route is greater than or equal to the network mask of the prefix;
	  • the network address of the route masked out by the network mask of the prefix is equal to the network address of the prefix;)
	• the length of the network mask of the route falls within the range of the prefix-length

protocol (*connect* | *static* | *rip* | *ospf* | *bgp*;)	match routes coming from a specific protocol (the values are self-explanatory)
route-comment (*string*;)	match routes with a specific comment
route-tag (*integer*;)	match routes with a specific route-tag property value

route-target (*[integer|IP]:integer*;)	Match value against route target EXTENDED_COMMUNITIES path attribute
routing-mark (*string*;)	match routes with a specific routing mark
scope (*integer 0..255[-integer 0..255]*;)	match routes with a specific scope property value
set-bgp-communities (*integer:integer | internet | local-as | no-advertise | no-export*;)	set **COMMUNITIES** BGP attribut
set-bgp-local-pref (*integer*;)	set **LOCAL_PREF** BGP attribute
set-bgp-med (*integer*;)	set **MULTI_EXIT_DISC** BGP attribute
set-bgp-prepend (*integer: 0..16 | default*;)	how many times to prepend router's own AS number to **AS_PATH** attribute For incoming filters, it affects the AS_PATH attribute length, which is used in BGP route selection process. For outgoing filters, the prepending is done when announcing route via BGP and affects only routes sent to **EBGP** peers (for **IBGP** value 1 is always used) If value is set to **0** then peer's own AS is removed from AS_PATH (Similar to CISCO feature "no bgp enforce-first-as")
set-bgp-prepend-path (*AS list*;)	add specified list of AS numbers to **AS_PATH** attribute If both **set-bgp-prepend** and **set-bgp-prepend-path** are used then **set-bgp-prepend** will have highest priority.
set-bgp-weight (*signed integer*;)	set BGP weight property to be used in BGP route selection process. Valid only in incoming filters and for BGP routes
set-check-gateway (*arp | none | ping*;)	set which protocol to use for gateway reachability, if any. Valid only in incoming filters
set-disabled (*yes | no*;)	if set, the route will not become active. Valid only in incoming filters
set-distance (*integer: 0..255*;)	set the administrative distance of the route. If set to value 255, the route will not become active. Valid only in incoming filters
set-in-nexthop (*IP address*;)	set gateway value to the specific IP address[es]. Valid only in incoming filters
set-in-nexthop-direct (*interface name*;)	set gateway value to the specific interface. Valid only in incoming filters
set-in-nexthop-ipv6 (*IPv6 address*;)	set gateway value to the specific IPv6 address[es]. Valid only in incoming filters
set-in-nexthop-linklocal (*IPv6 link-local address % interface name*;)	set gateway value to the specific IPv6 link-local address[es] on specific interfaces. The syntax separates address and interface by '%'. Valid only in incoming filters
set-out-nexthop (*IP address*;)	set gateway to be announced to the specific IP address[es]. Valid only in outgoing filters
set-out-nexthop-ipv6 (*IPv6 address*;)	set gateway to be announced to the specific IPv6 address[es]. Valid only in outgoing filters
set-out-nexthop-linklocal (*IPv6 link-local address*;)	set gateway value to be announced using BGP link-local nexthop feature. Valid only in outgoing filters and for BGP routes
set-pref-src (*IP address*;)	set the preferred source address for packets leaving via this route. Valid only in incoming filters
set-route-comment (*string*;)	set comment text. Valid only in incoming filters
set-route-tag (*integer*;)	set OSPF or RIP route tag property value. For RIP only values 0..65535 are valid
set-route-targets (*AsNum|AsIP*;)	Set route target EXTENDED_COMMUNITIES path attribute
set-routing-mark (*string*;)	set routing mark for the route. Valid only in incoming filters
set-scope (*integer: 0..255*;)	set scope property, used in recursive nexthop resolving. Valid only in incoming filters
set-target-scope (*integer: 0..255*;)	set target-scope property, used in recursive nexthop resolving. Valid only in incoming filters
set-type (*blackhole | prohibit | unicast | unreachable*;)	set route type. Valid only in incoming filters • `unicast` - standard route • `blackhole` - silently discard packets • `prohibit` - reply to sender with ICMP Communication Administratively Prohibited messages • `unreachable` - reply to sender with ICMP Network Unreachable messages

set-use-te-nexthop (*yes|no*;)	
site-of-origin (*string*;)	Match BGP Site of Origin extended community. Available starting from v4.3
set-site-of-origin (*string*;)	Set BGP Site of Origin extended community. Available starting from v4.3
target-scope (*integer 0..255[-integer 0..255]*;)	match routes with a specific 'target-scope' value

Examples

- Routing filter usage in BGP Simple Multihoming

[Top | Back to Content]

Manual:Tools/Fetch

Applies to RouterOS: v3, v4 +

Summary

Sub-menu: `/tool fetch`
Standards:

Fetch is one of the console tools in Mikrotik RouterOS. It is used to copy files from any network device to a Mikrotik router via HTTP or FTP. In latest v5 versions it is possible also to upload files to remote locations.

Properties

Property	Description
address (*string*; Default:)	IP address of the device to copy file from.
ascii (*yes* | *no*; Default: **no**)	
dst-path (*string*; Default:)	Destination filename and path
host (*string*; Default:)	Domain name or virtual domain name (if used on web-site, from which you want to copy information). For example, `address=wiki.mikrotik.com host=forum.mikrotik.com` In this example the resolved ip address is the same (66.228.113.27), but hosts are different.
keep-result (*yes* | *no*; Default: **yes**)	If yes, creates an input file.
mode (*ftp|http|tftp*; Default: **http**)	Choose the protocol of connection - http, ftp or tftp.
password (*string*; Default: **anonymous**)	Password, which is needed for authentication to the remote device.
port (*integer*; Default:)	Connection port.
src-path (*string*; Default:)	Title of the remote file you need to copy.
upload (*yes* | *no*; Default: **no**)	If enabled then fetch will be used to upload file to remote server. Requires *src-path* and *dst-path* parameters to be set.

url (*string*; Default:) URL pointing to file. Can be used instead of **address** and **src-path** parameters.

user (*string*; Default: **anonymous**) User name, which is needed for authentication to the remote device.

Examples

The following example shows how to copy the file with filename "conf.rsc" from device with ip address 192.168.88.2 by FTP protocol and save it as file with filename "123.rsc". User and password are needed to login into the device.

```
[admin@mt-test] /tool> fetch address=192.168.88.2 src-path=conf.rsc \
user=admin mode=ftp password=123 dst-path=123.rsc port=21 \
host="" keep-result=yes
```

Example to upload file to other router:

```
[admin@mt-test] /tool> fetch address=192.168.88.2 src-path=conf.rsc \
user=admin mode=ftp password=123 dst-path=123.rsc upload=yes
```

Another example that demonstrates the usage of `url` property.

```
[admin@test_host] /> /tool fetch url="http://www.mikrotik.com/img/netaddresses2.pdf" mode=http
  status: finished

[admin@test_host] /> /file print
 # NAME                    TYPE                SIZE                CREATION-TIME
 ...
 5 netaddresses2.pdf       .pdf file           11547               jun/01/2010 11:59:51
```

[Top | Back to Content]

Manual:Fast Path

Applies to RouterOS: v6.0rc2 +

Summary

Fast path is RouterOS new feature which allows to forward packets without additional processing in the Linux kernel. It improves forwarding speeds significantly.

For fast path to work, interface support and specific configuration conditions are required.

List of RouterBoards with FastPath support

RouterBoard	Interfaces
RB3xx series	ether1,2
RB6xx series	ether1,2
RB7xx series	all ethernets
RB8xx series	ether1,2
RB9xx series	all ethernets
RB1000	all ethernets
RB1100 series	ether1-10,11
RB2011 series	all ethernets and sfp
CCR series routers	all ethernets and sfps

FastPath Handlers

Currently RouterOS has following fast path handlers:

- ipv4
- traffic generator
- mpls
- bridge

Note: Packet will be forwarded in fast path only if source and destination interfaces support fast path. See the list of supported interfaces.

IPv4 handler

IPv4 fast path is automatically used if following conditions are met:

- firewal rules are not configured,
- Traffic flow is disabled `/ip traffic-flow enabled=no`
- Simple and queue trees with parent=global are not configured.
- source interface is not bridge port or bonding slave
- destination interface queue is set to only-hw-queue and no queue tree entries with parent="dst interface"
- no vrrp, mesh, metarouter interface configuration,

- sniffer, torch and traffic generator is not running
- connection tracking is not active

`/ip firewall connection tracking set enabled` parameter has new `auto` value Which means that connection tracking is disabled by default until firewall rules are added.

Traffic Generator handler

Traffic Generator fast path is automatically used for interfaces that support this feature.

MPLS handler

MPLS fast path is automatically used for interfaces that support this feature.

Currently MPLS fast-path applies only to MPLS switched traffic (frames that enter router as MPLS and must leave router as MPLS) - MPLS ingress and egress (including VPLS tunnel endpoints that do VPLS encap/decap) will operate as before.

Bridge handler

Bridge fast path is automatically used if following conditions are met:

- no bridge firewall rules (`/interface bridge filter, /interface bridge nat`) are configured,
- `/interface bridge settings use-ip-firwall=no`,
- destination interface queue is set to only-hw-queue,
- no vrrp, mesh, metarouter interface configuration,
- sniffer, torch and traffic generator is not running

[Top | Back to Content]

Fast Path spanish

Applies to RouterOS: v6.0rc2 +

Resumen

Fast path es una nueva característica de RouterOS la cual permite el reenvío de paquetes sin procesamiento adicional en el Linux Kernel. Esto mejora la velocidad de reenvío significativamente.

Para que fast path funcione, es necesario que la interface lo soporte y una configuración especifica.

Lista de RouterBOARDs con soporte de FastPath

RouterBoard	Interfaces
RB3xx series	ether1,2
RB6xx series	ether1,2
RB7xx series	all ethernets
RB8xx series	ether1,2
RB9xx series	all ethernets
RB1000	all ethernets
RB1100 series	ether1-10,11
RB2011 series	all ethernets and sfp
CCR series routers	all ethernets and sfps

FastPath Handlers

Actualmente RouterOS tiene los siguientes fast path handlers:

- ipv4
- traffic generator
- mpls
- bridge

Note: Los paquetes serán reenviados en fast path únicamente si la interface origen y destino soportan fast path. Mire la lista de intefaces soportadas.

IPv4 handler

IPv4 fast path es automáticamente usado si las siguientes condiciones se encuentran:

- firewal rules no está configurado,
- Traffic flow está deshabilitado `/ip traffic-flow enabled=no`
- Simple y queue trees con parent=global no está configurado.
- la interface origen no está en un puerto de bridge o bonding slave
- la queue de interface destino es seteada a only-hw-queue y no hay entradas de quee tree con parent="dst interface"
- no hay configuración en la interface de vrrp, mesh, metarouter ,
- sniffer, torch y traffic generator no está corriendo
- connection tracking no está activo

`/ip firewall connection tracking set enabled` tiene un nuevo valor `auto`. El cual significa que el connection tracking está deshabilitado por defecto hasta que una regla de firewall es agregada

Traffic Generator handler

Traffic Generator fast path es automáticamente usado para interfaces que soporten esta capacidad.

MPLS handler

MPLS fast path es automáticamente usado para interfaces que soporten esta capacidad.

Bridge handler

Bridge fast path es automáticamente usado si las siguientes condiciones se encuentran:

- no hay reglas de bridge firewall (`/interface bridge filter, /interface bridge nat`) configuradas,
- `/interface bridge settings use-ip-firwall=no`,

- la queue de la interface destino es seteada a only-hw-queue,
- no hay configuración en la interface de vrrp, mesh, metarouter ,
- sniffer, torch y traffic generator no está corriendo
- connection tracking no está activo

Traducción: [Maximiliano Dobladez [1]] - [MKE Solutions [1]]

[Top | Back to Content]

References

[1] http://mikrotikexpert.com

Manual:IP/Firewall

List of reference sub-pages **Case studies** **List of examples**

<splist showparent=yes />

Manual:IPv6/Firewall

List of reference sub-pages **Case studies** **List of examples**

<splist showparent=yes />

Manual:IP/Firewall/Address list

Applies to RouterOS: 2.9, v3, v4 +

Summary

Sub-menu: `/ip firewall address-list`

Firewall address lists allow user to create lists of IP addresses grouped together. Firewall filter, mangle and NAT facilities can use address lists to match packets against them.

The address list records could be updated dynamically via the `action=add-src-to-address-list` or `action=add-dst-to-address-list` items found in NAT, mangle and filter facilities.

Properties

Property	Description
address (*IP address/netmask* | *IP-IP*; Default:)	IP address or range to add to address list
list (*string*; Default:)	Name of the address list where to add IP address

Example

The following example creates an address list of people thet are connecting to port 23 (telnet) on the router and drops all further traffic from them. Additionaly, the address list will contain one static entry of address=192.0.34.166/32 (www.example.com):

```
[admin@MikroTik] > /ip firewall address-list add list=drop_traffic address=192.0.34.166/32
[admin@MikroTik] > /ip firewall address-list print
Flags: X - disabled, D - dynamic
 #   LIST         ADDRESS
 0   drop_traffic 192.0.34.166
[admin@MikroTik] > /ip firewall mangle add chain=prerouting protocol=tcp dst-port=23 \
\... action=add-src-to-address-list address-list=drop_traffic
[admin@MikroTik] > /ip firewall filter add action=drop chain=input src-address-list=drop_traffic
[admin@MikroTik] > /ip firewall address-list print
Flags: X - disabled, D - dynamic
 #   LIST         ADDRESS
 0   drop_traffic 192.0.34.166
 1 D drop_traffic 1.1.1.1
 2 D drop_traffic 10.5.11.8
[admin@MikroTik] >
```

As seen in the output of the last print command, two new dynamic entries appeared in the address list. Hosts with these IP addresses tried to initialize a telnet session to the router.

[Top | Back to Content **]**

Manual:IP/Firewall/Filter

Applies to RouterOS: v3, v4

Summary

Sub-menu: `/ip firewall filter`

The firewall implements packet filtering and thereby provides security functions that are used to manage data flow to, from and through the router. Along with the Network Address Translation it serves as a tool for preventing unauthorized access to directly attached networks and the router itself as well as a filter for outgoing traffic.

Network firewalls keep outside threats away from sensitive data available inside the network. Whenever different networks are joined together, there is always a threat that someone from outside of your network will break into your LAN. Such break-ins may result in private data being stolen and distributed, valuable data being altered or destroyed, or entire hard drives being erased. Firewalls are used as a means of preventing or minimizing the security risks inherent in connecting to other networks. Properly configured firewall plays a key role in efficient and secure network infrastrure deployment.

MikroTik RouterOS has very powerful firewall implementation with features including:

- stateful packet inspection
- Layer-7 protocol detection
- peer-to-peer protocols filtering
- traffic classification by:
- source MAC address
- IP addresses (network or list) and address types (broadcast, local, multicast, unicast)
- port or port range
- IP protocols
- protocol options (ICMP type and code fields, TCP flags, IP options and MSS)
- interface the packet arrived from or left through
- internal flow and connection marks
- DSCP byte
- packet content
- rate at which packets arrive and sequence numbers
- packet size
- packet arrival time
- and much more!

Chains

The firewall operates by means of firewall rules. Each rule consists of two parts - the matcher which matches traffic flow against given conditions and the action which defines what to do with the matched packet.

Firewall filtering rules are grouped together in chains. It allows a packet to be matched against one common criterion in one chain, and then passed over for processing against some other common criteria to another chain. For example a packet should be matched against the IP address:port pair. Of course, it could be achieved by adding as many rules with IP address:port match as required to the forward chain, but a better way could be to add one rule that matches traffic from a particular IP address, e.g.: /ip firewall filter add src-address=1.1.1.2/32 jump-target="mychain" and in case of successfull match passes control over the IP packet to some other chain, id est mychain in this example. Then rules that perform matching against separate ports can be added to mychain chain without specifying the IP addresses.

There are three predefined chains, which cannot be deleted:

- **input** - used to process packets entering the router through one of the interfaces with the destination IP address which is one of the router's addresses. Packets passing through the router are not processed against the rules of the input chain
- **forward** - used to process packets passing through the router
- **output** - used to process packets originated from the router and leaving it through one of the interfaces. Packets passing through the router are not processed against the rules of the output chain

Packet flow diagrams illustrate how packets are processed in RouterOS.

When processing a chain, rules are taken from the chain in the order they are listed there from top to bottom. If a packet matches the criteria of the rule, then the specified action is performed on it, and no more rules are processed in that chain (the exception is the passthrough action). If a packet has not matched any rule within the chain, then it is accepted.

Properties

Property	Description
action (*action name*; Default: **accept**)	Action to take if packet is matched by the rule: • accept - accept the packet. Packet is not passed to next firewall rule. • add-dst-to-address-list - add destination address to address list specified by address-list parameter • add-src-to-address-list - add source address to address list specified by address-list parameter • drop - silently drop the packet • jump - jump to the user defined chain specified by the value of jump-target parameter • log - add a message to the system log containing following data: in-interface, out-interface, src-mac, protocol, src-ip:port->dst-ip:port and length of the packet. After packet is matched it is passed to next rule in the list, similar as passthrough • passthrough - ignore this rule and go to next one (useful for statistics). • reject - drop the packet and send an ICMP reject message • return - passes control back to the chain from where the jump took place • tarpit - captures and holds TCP connections (replies with SYN/ACK to the inbound TCP SYN packet)
address-list (*string*; Default:)	Name of the address list to be used. Applicable if action is add-dst-to-address-list or add-src-to-address-list

address-list-timeout (*time*; Default: **00:00:00**)
Time interval after which the address will be removed from the address list specified by `address-list` parameter. Used in conjunction with `add-dst-to-address-list` or `add-src-to-address-list` actions
Value of `00:00:00` will leave the address in the address list forever

chain (*name*; Default:)
Specifies to which chain rule will be added. If the input does not match the name of an already defined chain, a new chain will be created.

comment (*string*; Default:)
Descriptive comment for the rule.

connection-bytes (*integer-integer*; Default:)
Matches packets only if a given amount of bytes has been transfered through the particular connection. 0 - means infinity, for example `connection-bytes=2000000-0` means that the rule matches if more than 2MB has been transfered through the relevant connection

connection-limit (*integer,netmask*; Default:)
Restrict connection limit per address or address block

connection-mark (*no-mark* | *string*; Default:)
Matches packets marked via mangle facility with particular connection mark. If **no-mark** is set, rule will match any unmarked connection.

connection-rate (*Integer 0..4294967295*; Default:)
Connection Rate is a firewall matcher that allow to capture traffic based on present speed of the connection. Read more >>

connection-state (*estabilished* | *invalid* | *new* | *related*; Default:)
Interprets the connection tracking analysis data for a particular packet:

- `established` - a packet which belongs to an existing connection
- `invalid` - a packet which could not be identified for some reason
- `new` - the packet has started a new connection, or otherwise associated with a connection which has not seen packets in both directions.
- `related` - a packet which is related to, but not part of an existing connection, such as ICMP errors or a packet which begins FTP data connection

connection-type (*ftp* | *h323* | *irc* | *pptp* | *quake3* | *sip* | *tftp*; Default:)
Matches packets from related connections based on information from their connection tracking helpers. A relevant connection helper must be enabled under /ip firewall service-port

content (*string*; Default:)
Match packets that contain specified text

dscp (*integer: 0..63*; Default:)
Matches DSCP IP header field.

dst-address (*IP/netmask* | *IP range*; Default:)
Matches packets which destination is equal to specified IP or falls into specified IP range.

dst-address-list (*name*; Default:)
Matches destination address of a packet against user-defined address list

dst-address-type (*unicast* | *local* | *broadcast* | *multicast*; Default:)
Matches destination address type:

- `unicast` - IP address used for point to point transmission
- `local` - if dst-address is assigned to one of router's interfaces
- `broadcast` - packet is sent to all devices in subnet
- `multicast` - packet is forwarded to defined group of devices

dst-limit (*integer,time,integer,dst-address* | *dst-port* | *src-address, time*; Default:)
Matches packets within given pps limit. As opposed to the `limit` matcher, every destination IP address / destination port has it's own limit. Parameters are written in following format: `count,time,burst,mode,expire`.

- **count** - maximum average packet rate measured in packets per `time` interval
- **time** - specifies the time interval in which the packet rate is measured
- **burst** - number of packets which are not counted by packet rate
- **mode** - the classifier for packet rate limiting
- **expire** - specifies interval after which recored ip address /port will be deleted

dst-port (*integer[-integer]: 0..65535*; Default:)
List of destination port numbers or port number ranges

fragment (*yes|no*; Default:)	Matches fragmented packets. First (starting) fragment does not count. If connection tracking is enabled there will be no fragments as system automatically assembles every packet
hotspot (*auth* | *from-client* | *http* | *local-dst* | *to-client*; Default:)	
icmp-options (*integer:integer*; Default:)	Matches ICMP type:code fileds
in-bridge-port (*name*; Default:)	Actual interface the packet has entered the router, if incoming interface is bridge
in-interface (*name*; Default:)	Interface the packet has entered the router
ingress-priority (*integer: 0..63*; Default:)	Matches ingress priority of the packet. Priority may be derived from VLAN, WMM or MPLS EXP bit. Read more>>
ipv4-options (*any* | *loose-source-routing* | *no-record-route* | *no-router-alert* | *no-source-routing* | *no-timestamp* | *none* | *record-route* | *router-alert* | *strict-source-routing* | *timestamp*; Default:)	Matches IPv4 header options. • `any` - match packet with at least one of the ipv4 options • `loose-source-routing` - match packets with loose source routing option. This option is used to route the internet datagram based on information supplied by the source • `no-record-route` - match packets with no record route option. This option is used to route the internet datagram based on information supplied by the source • `no-router-alert` - match packets with no router alter option • `no-source-routing` - match packets with no source routing option • `no-timestamp` - match packets with no timestamp option • `record-route` - match packets with record route option • `router-alert` - match packets with router alter option • `strict-source-routing` - match packets with strict source routing option • `timestamp` - match packets with timestamp
jump-target (*name*; Default:)	Name of the target chain to jump to. Applicable only if `action=jump`
layer7-protocol (*name*; Default:)	Layer7 filter name defined in layer7 protocol menu.
limit (*integer,time,integer*; Default:)	Matches packets within given pps limit. Parameters are written in following format: `count,time,burst`. • **count** - maximum average packet rate measured in packets per `time` interval • **time** - specifies the time interval in which the packet rate is measured • **burst** - number of packets which are not counted by packet rate
log-prefix (*string*; Default:)	Adds specified text at the beginning of every log message. Applicable if `action=log`
nth (*integer,integer*; Default:)	Matches every nth packet. Read more >>
out-bridge-port (*name*; Default:)	Actual interface the packet is leaving the router, if outgoing interface is bridge
out-interface (; Default:)	Interface the packet is leaving the router
p2p (*all-p2p* | *bit-torrent* | *blubster* | *direct-connect* | *edonkey* | *fasttrack* | *gnutella* | *soulseek* | *warez* | *winmx*; Default:)	Matches packets from various peer-to-peer (P2P) protocols. Does not work on encrypted p2p packets.
packet-mark (*no-mark* | *string*; Default:)	Matches packets marked via mangle facility with particular packet mark. If **no-mark** is set, rule will match any unmarked packet.
packet-size (*integer[-integer]:0..65535*; Default:)	Matches packets of specified size or size range in bytes.
per-connection-classifier (*ValuesToHash:Denominator/Remainder*; Default:)	PCC matcher allows to divide traffic into equal streams with ability to keep packets with specific set of options in one particular stream. Read more >>
port (*integer[-integer]: 0..65535*; Default:)	Matches if any (source or destination) port matches the specified list of ports or port ranges. Applicable only if `protocol` is TCP or UDP

protocol (*name or protocol ID*; Default: **tcp**) — Matches particular IP protocol specified by protocol name or number

psd (*integer,time,integer,integer*; Default:) — Attempts to detect TCP and UDP scans. Parameters are in following format `WeightThreshold, DelayThreshold, LopPortWeight, HighPortWeight`

- **WeightThreshold** - total weight of the latest TCP/UDP packets with different destination ports coming from the same host to be treated as port scan sequence
- **DelayThreshold** - delay for the packets with different destination ports coming from the same host to be treated as possible port scan subsequence
- **LowPortWeight** - weight of the packets with privileged (<=1024) destination port
- **HighPortWeight** - weight of the packet with non-priviliged destination port

random (*integer: 1..99*; Default:) — Matches packets randomly with given probability.

reject-with (; Default:) — Specifies error to be sent back if packet is rejected. Applicable if `action=reject`

routing-mark (*string*; Default:) — Matches packets marked by mangle facility with particular routing mark

src-address (*Ip/Netmaks, Ip range*; Default:) — Matches packets which source is equal to specified IP or falls into specified IP range.

src-address-list (*name*; Default:) — Matches source address of a packet against user-defined address list

src-address-type (*unicast* | *local* | *broadcast* | *multicast*; Default:) — Matches source address type:

- `unicast` - IP address used for point to point transmission
- `local` - if address is assigned to one of router's interfaces
- `broadcast` - packet is sent to all devices in subnet
- `multicast` - packet is forwarded to defined group of devices

src-port (*integer[-integer]: 0..65535*; Default:) — List of source ports and ranges of source ports. Applicable only if protocol is TCP or UDP.

src-mac-address (*MAC address*; Default:) — Matches source MAC address of the packet

tcp-flags (*ack* | *cwr* | *ece* | *fin* | *psh* | *rst* | *syn* | *urg*; Default:) — Matches specified TCP flags

- `ack` - acknowledging data
- `cwr` - congestion window reduced
- `ece` - ECN-echo flag (explicit congestion notification)
- `fin` - close connection
- `psh` - push function
- `rst` - drop connection
- `syn` - new connection
- `urg` - urgent data

tcp-mss (*integer: 0..65535*; Default:) — Matches TCP MSS value of an IP packet

time (*time-time,sat* | *fri* | *thu* | *wed* | *tue* | *mon* | *sun*; Default:) — Allows to create filter based on the packets' arrival time and date or, for locally generated packets, departure time and date

ttl (*integer: 0..255*; Default:) — Matches packets TTL value

Stats

`/ip firewall filter print stats` will show additional read-only properties

Property	Description
bytes (*integer*)	Total amount of bytes matched by the rule
packets (*integer*)	Total amount of packets matched by the rule

By default **print** is equivalent to **print static** and shows only static rules.

```
[admin@dzeltenais_burkaans] /ip firewall mangle> print stats
Flags: X - disabled, I - invalid, D - dynamic
 #   CHAIN              ACTION                   BYTES           PACKETS
 0   prerouting         mark-routing             17478158        127631
 1   prerouting         mark-routing             782505          4506
```

To print also dynamic rules use **print all**.

```
[admin@dzeltenais_burkaans] /ip firewall mangle> print all stats
Flags: X - disabled, I - invalid, D - dynamic
 #   CHAIN              ACTION                   BYTES           PACKETS
 0   prerouting         mark-routing             17478158        127631
 1   prerouting         mark-routing             782505          4506
 2 D forward            change-mss               0               0
 3 D forward            change-mss               0               0
 4 D forward            change-mss               0               0
 5 D forward            change-mss               129372          2031
```

Or to print only dynamic rules use **print dynamic**

```
[admin@dzeltenais_burkaans] /ip firewall mangle> print stats dynamic
Flags: X - disabled, I - invalid, D - dynamic
 #   CHAIN              ACTION                   BYTES           PACKETS
 0 D forward            change-mss               0               0
 1 D forward            change-mss               0               0
 2 D forward            change-mss               0               0
 3 D forward            change-mss               132444          2079
```

Menu specific commands

Property	Description
reset-counters (*id*)	Reset statistics counters for specified firewall rules.
reset-counters-all ()	Reset statistics counters for all firewall rules.

Basic examples

Router protection

Lets say our private network is 192.168.0.0/24 and public (WAN) interface is ether1. We will set up firewall to allow connections to router itself only from our local network and drop the rest. Also we will allow ICMP protocol on any interface so that anyone can ping your router from internet.

```
/ip firewall filter
add chain=input connection-state=invalid action=drop \
      comment="Drop Invalid connections"
add chain=input connection-state=established action=accept \
      comment="Allow Established connections"
add chain=input protocol=icmp action=accept \
      comment="Allow ICMP"
add chain=input src-address=192.168.0.0/24 action=accept \
      in-interface=!ether1
add chain=input action=drop comment="Drop everything else"
```

Customer protection

To protect the customer's network, we should check all traffic which goes through router and block unwanted. For icmp, tcp, udp traffic we will create chains, where will be droped all unwanted packets:

```
/ip firewall filter
add chain=forward protocol=tcp connection-state=invalid \
       action=drop comment="drop invalid connections"
add chain=forward connection-state=established action=accept \
       comment="allow already established connections"
add chain=forward connection-state=related action=accept \
       comment="allow related connections"
```

Block "bogon" IP addresses

```
add chain=forward src-address=0.0.0.0/8 action=drop
add chain=forward dst-address=0.0.0.0/8 action=drop
add chain=forward src-address=127.0.0.0/8 action=drop
add chain=forward dst-address=127.0.0.0/8 action=drop
add chain=forward src-address=224.0.0.0/3 action=drop
add chain=forward dst-address=224.0.0.0/3 action=drop
```

Make jumps to new chains:

```
add chain=forward protocol=tcp action=jump jump-target=tcp
add chain=forward protocol=udp action=jump jump-target=udp
```

```
add chain=forward protocol=icmp action=jump jump-target=icmp
```

Create tcp chain and deny some tcp ports in it:

```
add chain=tcp protocol=tcp dst-port=69 action=drop \
       comment="deny TFTP"
add chain=tcp protocol=tcp dst-port=111 action=drop \
       comment="deny RPC portmapper"
add chain=tcp protocol=tcp dst-port=135 action=drop \
       comment="deny RPC portmapper"
add chain=tcp protocol=tcp dst-port=137-139 action=drop \
       comment="deny NBT"
add chain=tcp protocol=tcp dst-port=445 action=drop \
       comment="deny cifs"
add chain=tcp protocol=tcp dst-port=2049 action=drop comment="deny NFS"
add chain=tcp protocol=tcp dst-port=12345-12346 action=drop comment="deny NetBus"
add chain=tcp protocol=tcp dst-port=20034 action=drop comment="deny NetBus"
add chain=tcp protocol=tcp dst-port=3133 action=drop comment="deny BackOriffice"
add chain=tcp protocol=tcp dst-port=67-68 action=drop comment="deny DHCP"
```

Deny udp ports in udp chain:

```
add chain=udp protocol=udp dst-port=69 action=drop comment="deny TFTP"
add chain=udp protocol=udp dst-port=111 action=drop comment="deny PRC portmapper"
add chain=udp protocol=udp dst-port=135 action=drop comment="deny PRC portmapper"
add chain=udp protocol=udp dst-port=137-139 action=drop comment="deny NBT"
add chain=udp protocol=udp dst-port=2049 action=drop comment="deny NFS"
add chain=udp protocol=udp dst-port=3133 action=drop comment="deny BackOriffice"
```

Allow only needed icmp codes in icmp chain:

```
add chain=icmp protocol=icmp icmp-options=0:0 action=accept \
       comment="echo reply"
add chain=icmp protocol=icmp icmp-options=3:0 action=accept \
       comment="net unreachable"
add chain=icmp protocol=icmp icmp-options=3:1 action=accept \
       comment="host unreachable"
add chain=icmp protocol=icmp icmp-options=3:4 action=accept \
       comment="host unreachable fragmentation required"
add chain=icmp protocol=icmp icmp-options=4:0 action=accept \
       comment="allow source quench"
add chain=icmp protocol=icmp icmp-options=8:0 action=accept \
       comment="allow echo request"
add chain=icmp protocol=icmp icmp-options=11:0 action=accept \
       comment="allow time exceed"
add chain=icmp protocol=icmp icmp-options=12:0 action=accept \
       comment="allow parameter bad"
add chain=icmp action=drop comment="deny all other types"
```

other ICMP codes are found here [1].

Brute force protection

Bruteforce_login_prevention_(FTP_&_SSH)

[Top | Back to Content]

References

[1] http://www.iana.org/assignments/icmp-parameters

Manual:IPv6/Firewall/Filter

Applies to RouterOS: v5

Summary

Sub-menu: `/ipv6 firewall filter`

Properties

Property	Description
`action` (*accept* | *add-dst-to-address-list* | ...; Default: **accept**)	Action to take if packet is matched by the rule: • `accept` - Accept the packet. It is not passed to the next firewall rule. • `add-dst-to-address-list` - Add destination address to address list specified by `address-list` parameter • `add-src-to-address-list` - Add source address to address list specified by `address-list` parameter • `drop` -Silently drop the packet. • `jump` - Jump to the user defined chain specified by the value of `jump-target` parameter • `log` - Add a message to the system log containing following data: in-interface, out-interface, src-mac, protocol, src-ip:port->dst-ip:port and length of the packet. After packet is matched it is passed to the next rule in the list, similar as `passthrough` • `passthrough` - Ignore this rule and go to next one (useful for statistics). • `reject` - Drop the packet and send an ICMP reject message • `return` - Passes control back to the chain from where the jump took place.
`address-list` (; Default:)	
`time` (; Default:)	

[Top | Back to Content]

Manual:IP/Firewall/L7

Applies to RouterOS: v3, v4 +

Summary

layer7-protocol is a method of searching for patterns in ICMP/TCP/UDP streams.

L7 matcher is collecting first **10 packets** of connection or first **2KB** of connection and searches for pattern in collected data. If pattern is not found in collected data, matcher is not inspecting further. Allocated memory is freed and protocol is considered as **unknown**. You should take into account that a lot of connections will significantly increase memory usage. To avoid it add regular firewall matchers to reduce amount of data passed to layer-7 filters.

Additional requirement is that layer7 matcher must see both directions of traffic (incoming and outgoing). To satisfy this requirement l7 rules should be set in **forward** chain. If rule is set in **input/prerouting** chain then the same rule **must** be set also in **output/postrouting** chain, otherwise collected data may not be complete resulting in incorrectly matched pattern.

L7 patterns found in l7-filter project page [1] and in [2] are compatible with RouterOS.
You can also download a script with a list of common protocols here [3] (only for RouterOS v3), just run Import command with this file.

Warning: In some cases when layer 7 regular expression cannot be performed, RotuerOS will log *topic=firewall, warning* with error message stating exactly what is te problem in the message

Properties

Sub-menu: `/ip firewall layer7-protocol`

Property	Description
name (*string*; Default:)	Descriptive name of l7 pattern used by configuration in firewall rules. See example >>.
regexp (*string*; Default:)	POSIX compliant regular expression used to match pattern.

Examples

Simple L7 usage example

First, add Regexp strings to the protocols menu, to define strings you will be looking for. In this example we will use pattern to match bittorent packets.

```
/ip firewall layer7-protocol
add comment="" name=bittorrent regexp="^(\\x13bittorrent
protocol|azver\\x01\$\
    |get /scrape\\\?info_hash=|get /announce\\\?info_hash=|get
/client/bitcomet\
    /|GET /data\\\?fid=)|d1:ad2:id20:|\\x08'7P\\)[RP]"
```

Then, use the defined protocols in firewall.

```
/ip firewall filter

# add few known protocols to reduce mem usage
add action=accept chain=forward comment="" disabled=no port=80 protocol=tcp
add action=accept chain=forward comment="" disabled=no port=443 protocol=tcp

# add l7 matcher
add action=accept chain=forward comment="" disabled=no layer7-protocol=\
    bittorrent protocol=tcp
```

As you can see before l7 rule we added several regular rules that will match known traffic thus reducing memory usage.

L7 in input chain

In this example we will try to match telnet protocol connecting to our router.

```
/ip firewall layer7-protocol
add comment="" name=telnet regexp=\
    "^\\xff[\\xfb-\\xfe].\\xff[\\xfb-\\xfe].\\xff[\\xfb-\\xfe]"
```

Note that we need both directions that is why we need also l7 rule in output chain that sees outgoing packets.

```
/ip firewall filter

add action=accept chain=input comment="" disabled=no layer7-protocol=telnet \
    protocol=tcp

add action=passthrough chain=output comment="" disabled=no layer7-protocol=telnet \
    protocol=tcp
```

[Top | Back to Content **]**

References

[1] http://l7-filter.sourceforge.net/protocols
[2] http://protocolinfo.org/wiki/Main_Page
[3] http://www.mikrotik.com/download/l7-protos.rsc

Manual:IP/Firewall/Mangle

Applies to RouterOS: v3, v4

Summary

Sub-menu: `/ip firewall mangle`

Mangle is a kind of 'marker' that marks packets for future processing with special marks. Many other facilities in RouterOS make use of these marks, e.g. queue trees, NAT, routing. They identify a packet based on its mark and process it accordingly. The mangle marks exist only within the router, they are not transmitted across the network.

Additionally, the mangle facility is used to modify some fields in the IP header, like TOS (DSCP) and TTL fields.

Properties

Property	Description
`action` (*action name*; Default: **accept**)	Action to take if packet is matched by the rule: • `accept` - accept the packet. Packet is not passed to next firewall rule. • `add-dst-to-address-list` - add destination address to Address list specified by `address-list` parameter • `add-src-to-address-list` - add source address to Address list specified by `address-list` parameter • `change-dscp` - change Differentiated Services Code Point (DSCP) field value specified by the new-dscp parameter • `change-mss` - change Maximum Segment Size field value of the packet to a value specified by the new-mss parameter • `change-ttl` - change Time to Live field value of the packet to a value specified by the new-ttl parameter • `jump` - jump to the user defined chain specified by the value of `jump-target` parameter • `log` - add a message to the system log containing following data: in-interface, out-interface, src-mac, protocol, src-ip:port->dst-ip:port and length of the packet. After packet is matched it is passed to next rule in the list, similar as `passthrough` • `mark-connection` - place a mark specified by the new-connection-mark parameter on the entire connection that matches the rule • `mark-packet` - place a mark specified by the new-packet-mark parameter on a packet that matches the rule • `mark-routing` - place a mark specified by the new-routing-mark parameter on a packet. This kind of marks is used for policy routing purposes only • `passthrough` - ignore this rule and go to next one (useful for statistics). • `return` - pass control back to the chain from where the jump took place • `set-priority` - set priority speciefied by the new-priority parameter on the packets sent out through a link that is capable of transporting priority (VLAN or WMM-enabled wireless interface). Read more> • `strip-ipv4-options` - strip IPv4 option fields from IP header.
`address-list` (*string*; Default:)	Name of the address list to be used. Applicable if action is `add-dst-to-address-list` or `add-src-to-address-list`

address-list-timeout (*time*; Default: **00:00:00**)
Time interval after which the address will be removed from the address list specified by `address-list` parameter. Used in conjunction with `add-dst-to-address-list` or `add-src-to-address-list` actions
Value of `00:00:00` will leave the address in the address list forever

chain (*name*; Default:)
Specifies to which chain rule will be added. If the input does not match the name of an already defined chain, a new chain will be created.

comment (*string*; Default:)
Descriptive comment for the rule.

connection-bytes (*integer-integer*; Default:)
Matches packets only if a given amount of bytes has been transfered through the particular connection. 0 - means infinity, for example `connection-bytes=2000000-0` means that the rule matches if more than 2MB has been transfered through the relevant connection

connection-limit (*integer,netmaks*; Default:)
Restrict connection limit per address or address block/td>

connection-mark (*no-mark* | *string*; Default:)
Matches packets marked via mangle facility with particular connection mark. If **no-mark** is set, rule will match any unmarked connection.

connection-rate (*Integer 0..4294967295*; Default:)
Connection Rate is a firewall matcher that allow to capture traffic based on present speed of the connection. `Read more >>`

connection-state (*estabilished* | *invalid* | *new* | *related*; Default:)
Interprets the connection tracking analysis data for a particular packet:

- `established` - a packet which belongs to an existing connection
- `invalid` - a packet which could not be identified for some reason
- `new` - the packet has started a new connection, or otherwise associated with a connection which has not seen packets in both directions
- `related` - a packet which is related to, but not part of an existing connection, such as ICMP errors or a packet which begins FTP data connection

connection-type (*ftp* | *h323* | *irc* | *pptp* | *quake3* | *sip* | *tftp*; Default:)
Matches packets from related connections based on information from their connection tracking helpers. A relevant connection helper must be enabled under /ip firewall service-port

content (*string*; Default:)
Match packets that contain specified text

dscp (*integer: 0..63*; Default:)
Matches DSCP IP header field.

dst-address (*IP/netmask* | *IP range*; Default:)
Matches packets which destination is equal to specified IP or falls into specified IP range.

dst-address-list (*name*; Default:)
Matches destination address of a packet against user-defined address list

dst-address-type (*unicast* | *local* | *broadcast* | *multicast*; Default:)
Matches destination address type:

- `unicast` - IP address used for point to point transmission
- `local` - if dst-address is assigned to one of router's interfaces
- `broadcast` - packet is sent to all devices in subnet
- `multicast` - packet is forwarded to defined group of devices

dst-limit (*integer,time,integer,dst-address* | *dst-port* | *src-address, time*; Default:)
Matches packets within given pps limit. As opposed to the `limit` matcher, every destination IP address / destination port has it's own limit. Parameters are written in following format: `count,time,burst,mode,expire`.

- **count** - maximum average packet rate measured in packets per `time` interval
- **time** - specifies the time interval in which the packet rate is measured
- **burst** - number of packets which are not counted by packet rate
- **mode** - the classifier for packet rate limiting
- **expire** - specifies interval after which recored ip address /port will be deleted

dst-port (*integer[-integer]: 0..65535*; Default:)
List of destination port numbers or port number ranges

fragment (*yes|no*; Default:)
Matches fragmented packets. First (starting) fragment does not count. If connection tracking is enabled there will be no fragments as system automatically assembles every packet

hotspot (*auth | from-client | http | local-dst | to-client*; Default:)

icmp-options (*integer:integer*; Default:)
Matches ICMP type:code fileds

in-bridge-port (*name*; Default:)
Actual interface the packet has entered the router, if incoming interface is bridge

in-interface (*name*; Default:)
Interface the packet has entered the router

ingress-priority (*integer: 0..63*; Default:)
Matches ingress priority of the packet. Priority may be derived from VLAN, WMM or MPLS EXP bit. `Read more >>`

ipv4-options (*any | loose-source-routing | no-record-route | no-router-alert | no-source-routing | no-timestamp | none | record-route | router-alert | strict-source-routing | timestamp*; Default:)
Matches IPv4 header options.

- `any` - match packet with at least one of the ipv4 options
- `loose-source-routing` - match packets with loose source routing option. This option is used to route the internet datagram based on information supplied by the source
- `no-record-route` - match packets with no record route option. This option is used to route the internet datagram based on information supplied by the source
- `no-router-alert` - match packets with no router alter option
- `no-source-routing` - match packets with no source routing option
- `no-timestamp` - match packets with no timestamp option
- `record-route` - match packets with record route option
- `router-alert` - match packets with router alter option
- `strict-source-routing` - match packets with strict source routing option
- `timestamp` - match packets with timestamp

jump-target (*name*; Default:)
Name of the target chain to jump to. Applicable only if `action=jump`

layer7-protocol (*name*; Default:)
Layer7 filter name defined in layer7 protocol menu.

limit (*integer,time,integer*; Default:)
Matches packets if given pps limit is exceeded. Parameters are written in following format: `count,time,burst`.

- **count** - maximum average packet rate measured in packets per `time` interval
- **time** - specifies the time interval in which the packet rate is measured
- **burst** - number of packets which are not counted by packet rate

log-prefix (*string*; Default:)
Adds specified text at the beginning of every log message. Applicable if `action=log`

new-connection-mark (*string*; Default:)

new-dscp (*integer: 0..63*; Default:)

new-mss (*integer*; Default:)

new-packet-mark (*string*; Default:)

new-priority (*integer*; Default:)

new-routing-mark (*string*; Default:)

new-ttl (*decrement | increment | set:integer*; Default:)

nth (*integer,integer*; Default:)
Matches every nth packet. `Read more >>`

out-bridge-port (*name*; Default:)
Actual interface the packet is leaving the router, if outgoing interface is bridge

out-interface (; Default:)
Interface the packet is leaving the router

p2p (*all-p2p* | *bit-torrent* | *blubster* | *direct-connect* | *edonkey* | *fasttrack* | *gnutella* | *soulseek* | *warez* | *winmx*; Default:)	Matches packets from various peer-to-peer (P2P) protocols. Does not work on encrypted p2p packets.
packet-mark (*no-mark* | *string*; Default:)	Matches packets marked via mangle facility with particular packet mark. If **no-mark** is set, rule will match any unmarked packet.
packet-size (*integer[-integer]:0..65535*; Default:)	Matches packets of specified size or size range in bytes.
per-connection-classifier (*ValuesToHash:Denominator/Remainder*; Default:)	PCC matcher allows to divide traffic into equal streams with ability to keep packets with specific set of options in one particular stream. `Read more >>`
port (*integer[-integer]: 0..65535*; Default:)	Matches if any (source or destination) port matches the specified list of ports or port ranges. Applicable only if `protocol` is TCP or UDP
protocol (*name or protocol ID*; Default: **tcp**)	Matches particular IP protocol specified by protocol name or number
psd (*integer,time,integer,integer*; Default:)	Attempts to detect TCP and UDP scans. Parameters are in following format `WeightThreshold, DelayThreshold, LopPortWeight, HighPortWeight` • **WeightThreshold** - total weight of the latest TCP/UDP packets with different destination ports coming from the same host to be treated as port scan sequence • **DelayThreshold** - delay for the packets with different destination ports coming from the same host to be treated as possible port scan subsequence • **LowPortWeight** - weight of the packets with privileged (<=1024) destination port • **HighPortWeight** - weight of the packet with non-priviliged destination port
random (*integer: 1..99*; Default:)	Matches packets randomly with given probability.
routing-mark (*string*; Default:)	Matches packets marked by mangle facility with particular routing mark
src-address (*Ip/Netmaks, Ip range*; Default:)	Matches packets which source is equal to specified IP or falls into specified IP range.
src-address-list (*name*; Default:)	Matches source address of a packet against user-defined address list
src-address-type (*unicast* | *local* | *broadcast* | *multicast*; Default:)	Matches source address type: • `unicast` - IP address used for point to point transmission • `local` - if address is assigned to one of router's interfaces • `broadcast` - packet is sent to all devices in subnet • `multicast` - packet is forwarded to defined group of devices
src-port (*integer[-integer]: 0..65535*; Default:)	List of source ports and ranges of source ports. Applicable only if protocol is TCP or UDP.
src-mac-address (*MAC address*; Default:)	Matches source MAC address of the packet
tcp-flags (*ack* | *cwr* | *ece* | *fin* | *psh* | *rst* | *syn* | *urg*; Default:)	Matches specified TCP flags • `ack` - acknowledging data • `cwr` - congestion window reduced • `ece` - ECN-echo flag (explicit congestion notification) • `fin` - close connection • `psh` - push function • `rst` - drop connection • `syn` - new connection • `urg` - urgent data
tcp-mss (*integer: 0..65535*; Default:)	Matches TCP MSS value of an IP packet
time (*time-time,sat* | *fri* | *thu* | *wed* | *tue* | *mon* | *sun*; Default:)	Allows to create filter based on the packets' arrival time and date or, for locally generated packets, departure time and date
ttl (*equal* | *greater-than* | *less-than* | *not-equal : integer(0..255)*; Default:)	Matches packets TTL value.

Stats

`/ip firewall filter print stats` will show additional read-only properties

Property	Description
bytes (*integer*)	Total amount of bytes matched by the rule
packets (*integer*)	Total amount of packets matched by the rule

By default **print** is equivalent to **print static** and shows only static rules.

```
[admin@dzeltenais_burkaans] /ip firewall mangle> print stats
Flags: X - disabled, I - invalid, D - dynamic
 #   CHAIN              ACTION                   BYTES           PACKETS
 0   prerouting         mark-routing             17478158        127631
 1   prerouting         mark-routing             782505          4506
```

To print also dynamic rules use **print all**.

```
[admin@dzeltenais_burkaans] /ip firewall mangle> print all stats
Flags: X - disabled, I - invalid, D - dynamic
 #   CHAIN              ACTION                   BYTES           PACKETS
 0   prerouting         mark-routing             17478158        127631
 1   prerouting         mark-routing             782505          4506
 2 D forward            change-mss               0               0
 3 D forward            change-mss               0               0
 4 D forward            change-mss               0               0
 5 D forward            change-mss               129372          2031
```

Or to print only dynamic rules use **print dynamic**

```
[admin@dzeltenais_burkaans] /ip firewall mangle> print stats dynamic
Flags: X - disabled, I - invalid, D - dynamic
 #   CHAIN              ACTION                   BYTES           PACKETS
 0 D forward            change-mss               0               0
 1 D forward            change-mss               0               0
 2 D forward            change-mss               0               0
 3 D forward            change-mss               132444          2079
```

Menu specific commands

Property	Description
reset-counters (*id*)	Reset statistics counters for specified firewall rules.
reset-counters-all ()	Reset statistics counters for all firewall rules.

Basic examples

It is a well known fact that VPN links have smaller packet size due to incapsulation overhead. A large packet with MSS that exceeds the MSS of the VPN link should be fragmented prior to sending it via that kind of connection. However, if the packet has DF flag set, it cannot be fragmented and should be discarded. On links that have broken path MTU discovery (PMTUD) it may lead to a number of problems, including problems with FTP and HTTP data transfer and e-mail services.

In case of link with broken PMTUD, a decrease of the MSS of the packets coming through the VPN link solves the problem. The following example demonstrates how to decrease the MSS value via mangle:

```
/ip firewall mangle
add out-interface=pppoe-out protocol=tcp tcp-flags=syn action=change-mss new-mss=1300 chain=forward
```

Marking each packet is quite resource expensive especially if rule has to match against many parameters from IP header or address list containing hundreds of entries.

Lets say we want to

- mark all **tcp** packets except tcp/80 and match these packets against **first** address list
- mark all **udp** packets and match them against **second** address list.

```
/ip firewall mangle
  add chain=forward protocol=tcp port=!80 dst-address-list=first action=mark-packet new-packet-mark=first
  add chain=forward protocol=udp dst-address-list=second action=mark-packet new-packet-mark=second
```

Setup looks quite simple and probably will work without problems in small networks. Now multiply count of rules by 10, add few hundred entries in address list, run 100Mbit of traffic over this router and you will see how rapidly CPU usage is increasing. The reason for such behavior is that each rule reads IP header of every packet and tries to match collected data against parameters specified in firewall rule.

Fortunately if connection tracking is enabled, we can use connection marks to optimize our setup.

```
/ip firewall mangle
  add chain=forward protocol=tcp port=!80 dst-address-list=first connection-state=new action=mark-connection \
new-connection-mark=first
  add chain=forward connection-mark=first action=mark-packet new-packet-mark=first passthrough=no


  add chain=forward protocol=udp dst-address-list=second connection-state=new action=mark-connection \
new-connection-mark=second
  add chain=forward connection-mark=second action=mark-packet new-packet-mark=second passthrough=no
```

Now first rule will try to match data from IP header only from first packet of new connection and add connection mark. Next rule will no longer check IP header for each packet, it will just compare connection marks resulting in lower CPU consumption. Additionally `passthrough=no` was added that helps to reduce CPU consumption even more.

[Top | Back to Content]

Manual:IP/Firewall/NAT

Applies to RouterOS: v3, v4 +

Summary

Sub-menu: `/ip firewall nat`

Network Address Translation is an Internet standard that allows hosts on local area networks to use one set of IP addresses for internal communications and another set of IP addresses for external communications. A LAN that uses NAT is referred as natted network. For NAT to function, there should be a NAT gateway in each natted network. The NAT gateway (NAT router) performs IP address rewriting on the way a packet travel from/to LAN.

There are two types of NAT:

- **source NAT or srcnat.** This type of NAT is performed on packets that are originated from a natted network. A NAT router replaces the private source address of an IP packet with a new public IP address as it travels through the router. A reverse operation is applied to the reply packets travelling in the other direction.
- **destination NAT or dstnat.** This type of NAT is performed on packets that are destined to the natted network. It is most comonly used to make hosts on a private network to be acceesible from the Internet. A NAT router performing dstnat replaces the destination IP address of an IP packet as it travel through the router towards a private network.

Hosts behind a NAT-enabled router do not have true end-to-end connectivity. Therefore some Internet protocols might not work in scenarios with NAT. Services that require the initiation of TCP connection from outside the private network or stateless protocols such as UDP, can be disrupted. Moreover, some protocols are inherently incompatible with NAT, a bold example is AH protocol from the IPsec suite.

To overcome these limitations RouterOS includes a number of so-called NAT helpers, that enable NAT traversal for various protocols.

Properties

Property	Description
`action` (*action name*; Default: **accept**)	Action to take if packet is matched by the rule:

- `accept` - accept the packet. Packet is not passed to next NAT rule.
- `add-dst-to-address-list` - add destination address to Address list specified by `address-list` parameter
- `add-src-to-address-list` - add source address to Address list specified by `address-list` parameter
- `dst-nat` - replaces destination address and/or port of an IP packet to values specified by `to-addresses` and `to-ports` parameters
- `jump` - jump to the user defined chain specified by the value of `jump-target` parameter
- `log` - add a message to the system log containing following data: in-interface, out-interface, src-mac, protocol, src-ip:port->dst-ip:port and length of the packet. After packet is matched it is passed to next rule in the list, similar as `passthrough`
- `masquerade` - replace source address of an IP packet to IP determined by routing facility.
- `netmap` - creates a static 1:1 mapping of one set of IP addresses to another one. Often used to distribute public IP addresses to hosts on private networks
- `passthrough` - ignore this rule and go to next one (useful for statistics).
- `redirect` - replaces destination port of an IP packet to one specified by `to-ports` parameter and destination address to one of the router's local addresses
- `return` - passes control back to the chain from where the jump took place
- `same` - gives a particular client the same source/destination IP address from supplied range for each connection. This is most frequently used for services that expect the same client address for multiple connections from the same client
- `src-nat` - replaces source address of an IP packet to values specified by `to-addresses` and `to-ports` parameters

address-list (*string*; Default:)

Name of the address list to be used. Applicable if action is `add-dst-to-address-list` or `add-src-to-address-list`

address-list-timeout (*time*; Default: **00:00:00**)

Time interval after which the address will be removed from the address list specified by `address-list` parameter. Used in conjunction with `add-dst-to-address-list` or `add-src-to-address-list` actions
Value of `00:00:00` will leave the address in the address list forever

chain (*name*; Default:)

Specifies to which chain rule will be added. If the input does not match the name of an already defined chain, a new chain will be created.

comment (*string*; Default:)

Descriptive comment for the rule.

connection-bytes (*integer-integer*; Default:)

Matches packets only if a given amount of bytes has been transfered through the particular connection. 0 - means infinity, for example `connection-bytes=2000000-0` means that the rule matches if more than 2MB has been transfered through the relevant connection

connection-limit (*integer,netmaks*; Default:)

Restrict connection limit per address or address block/td>

connection-mark (*no-mark* | *string*; Default:)

Matches packets marked via mangle facility with particular connection mark. If **no-mark** is set, rule will match any unmarked connection.

connection-rate (*Integer 0..4294967295*; Default:)

Connection Rate is a firewall matcher that allow to capture traffic based on present speed of the connection. `Read more>>`

connection-type (*ftp* | *h323* | *irc* | *pptp* | *quake3* | *sip* | *tftp*; Default:)

Matches packets from related connections based on information from their connection tracking helpers. A relevant connection helper must be enabled under /ip firewall service-port

content (*string*; Default:)

Match packets that contain specified text

dscp (*integer: 0..63*; Default:)

Matches DSCP IP header field.

dst-address (*IP/netmask* | *IP range*; Default:)
Matches packets which destination is equal to specified IP or falls into specified IP range.

dst-address-list (*name*; Default:)
Matches destination address of a packet against user-defined address list

dst-address-type (*unicast* | *local* | *broadcast* | *multicast*; Default:)
Matches destination address type:

- `unicast` - IP address used for point to point transmission
- `local` - if dst-address is assigned to one of router's interfaces
- `broadcast` - packet is sent to all devices in subnet
- `multicast` - packet is forwarded to defined group of devices

dst-limit (*integer,time,integer,dst-address* | *dst-port* | *src-address, time*; Default:)
Matches packets within given pps limit. As opposed to the `limit` matcher, every destination IP address / destination port has it's own limit. Parameters are written in following format: `count,time,burst,mode,expire`.

- **count** - maximum average packet rate measured in packets per `time` interval
- **time** - specifies the time interval in which the packet rate is measured
- **burst** - number of packets which are not counted by packet rate
- **mode** - the classifier for packet rate limiting
- **expire** - specifies interval after which recored ip address /port will be deleted

dst-port (*integer[-integer]: 0..65535*; Default:)
List of destination port numbers or port number ranges

fragment (*yes|no*; Default:)
Matches fragmented packets. First (starting) fragment does not count. If connection tracking is enabled there will be no fragments as system automatically assembles every packet

hotspot (*auth* | *from-client* | *http* | *local-dst* | *to-client*; Default:)

icmp-options (*integer:integer*; Default:)
Matches ICMP type:code fileds

in-bridge-port (*name*; Default:)
Actual interface the packet has entered the router, if incoming interface is bridge

in-interface (*name*; Default:)
Interface the packet has entered the router

ingress-priority (*integer: 0..63*; Default:)
Matches ingress priority of the packet. Priority may be derived from VLAN, WMM or MPLS EXP bit. `Read more>>`

ipv4-options (*any* | *loose-source-routing* | *no-record-route* | *no-router-alert* | *no-source-routing* | *no-timestamp* | *none* | *record-route* | *router-alert* | *strict-source-routing* | *timestamp*; Default:)
Matches IPv4 header options.

- `any` - match packet with at least one of the ipv4 options
- `loose-source-routing` - match packets with loose source routing option. This option is used to route the internet datagram based on information supplied by the source
- `no-record-route` - match packets with no record route option. This option is used to route the internet datagram based on information supplied by the source
- `no-router-alert` - match packets with no router alter option
- `no-source-routing` - match packets with no source routing option
- `no-timestamp` - match packets with no timestamp option
- `record-route` - match packets with record route option
- `router-alert` - match packets with router alter option
- `strict-source-routing` - match packets with strict source routing option
- `timestamp` - match packets with timestamp

jump-target (*name*; Default:)
Name of the target chain to jump to. Applicable only if `action=jump`

layer7-protocol (*name*; Default:)
Layer7 filter name defined in layer7 protocol menu.

limit (*integer,time,integer*; Default:)
Matches packets if given pps limit is exceeded. Parameters are written in following format: `count,time,burst`.
- **count** - maximum average packet rate measured in packets per `time` interval
- **time** - specifies the time interval in which the packet rate is measured
- **burst** - number of packets which are not counted by packet rate

log-prefix (*string*; Default:)
Adds specified text at the beginning of every log message. Applicable if `action=log`

nth (*integer,integer*; Default:)
Matches every nth packet. `Read more >>`

out-bridge-port (*name*; Default:)
Actual interface the packet is leaving the router, if outgoing interface is bridge

out-interface (; Default:)
Interface the packet is leaving the router

packet-mark (*no-mark* | *string*; Default:)
Matches packets marked via mangle facility with particular packet mark. If **no-mark** is set, rule will match any unmarked packet.

packet-size (*integer[-integer]:0..65535*; Default:)
Matches packets of specified size or size range in bytes.

per-connection-classifier (*ValuesToHash:Denominator/Remainder*; Default:)
PCC matcher allows to divide traffic into equal streams with ability to keep packets with specific set of options in one particular stream. `Read more >>`

port (*integer[-integer]: 0..65535*; Default:)
Matches if any (source or destination) port matches the specified list of ports or port ranges. Applicable only if `protocol` is TCP or UDP

protocol (*name or protocol ID*; Default: **tcp**)
Matches particular IP protocol specified by protocol name or number

psd (*integer,time,integer,integer*; Default:)
Attempts to detect TCP and UDP scans. Parameters are in following format `WeightThreshold, DelayThreshold, LopPortWeight, HighPortWeight`
- **WeightThreshold** - total weight of the latest TCP/UDP packets with different destination ports coming from the same host to be treated as port scan sequence
- **DelayThreshold** - delay for the packets with different destination ports coming from the same host to be treated as possible port scan subsequence
- **LowPortWeight** - weight of the packets with privileged (<=1024) destination port
- **HighPortWeight** - weight of the packet with non-priviliged destination port

random (*integer: 1..99*; Default:)
Matches packets randomly with given probability.

routing-mark (*string*; Default:)
Matches packets marked by mangle facility with particular routing mark

same-not-by-dst (*yes* | *no*; Default:)
Specifies whether to take into account or not destination IP address when selecting a new source IP address. Applicable if `action=same`

src-address (*Ip/Netmaks, Ip range*; Default:)
Matches packets which source is equal to specified IP or falls into specified IP range.

src-address-list (*name*; Default:)
Matches source address of a packet against user-defined address list

src-address-type (*unicast* | *local* | *broadcast* | *multicast*; Default:)
Matches source address type:
- `unicast` - IP address used for point to point transmission
- `local` - if address is assigned to one of router's interfaces
- `broadcast` - packet is sent to all devices in subnet
- `multicast` - packet is forwarded to defined group of devices

src-port (*integer[-integer]: 0..65535*; Default:)
List of source ports and ranges of source ports. Applicable only if protocol is TCP or UDP.

src-mac-address (*MAC address*; Default:)
Matches source MAC address of the packet

tcp-flags (*ack* | *cwr* | *ece* | *fin* | *psh* | *rst* | *syn* | *urg*; Default:)	Matches specified TCP flags • `ack` - acknowledging data • `cwr` - congestion window reduced • `ece` - ECN-echo flag (explicit congestion notification) • `fin` - close connection • `psh` - push function • `rst` - drop connection • `syn` - new connection • `urg` - urgent data
tcp-mss (*integer: 0..65535*; Default:)	Matches TCP MSS value of an IP packet
time (*time-time,sat* | *fri* | *thu* | *wed* | *tue* | *mon* | *sun*; Default:)	Allows to create filter based on the packets' arrival time and date or, for locally generated packets, departure time and date
to-addresses (*IP address[-IP address]*; Default: **0.0.0.0**)	Replace original address with specified one. Applicable if action is dst-nat, netmap, same, src-nat
to-ports (*integer[-integer]: 0..255*; Default:)	Replace original port with specified one. Applicable if action is dst-nat, redirect, netmap, same, src-nat
ttl (*integer: 0..255*; Default:)	Matches packets TTL value

`/ip firewall nat print stats` will show additional read-only properties

Property	Description
bytes (*integer*)	Total amount of bytes matched by the rule
packets (*integer*)	Total amount of packets matched by the rule

By default **print** is equivalent to **print static** and shows only static rules.

```
[admin@dzeltenais_burkaans] /ip firewall mangle> print stats
Flags: X - disabled, I - invalid, D - dynamic
 #   CHAIN              ACTION                  BYTES           PACKETS
 0   prerouting         mark-routing            17478158        127631
 1   prerouting         mark-routing            782505          4506
```

To print also dynamic rules use **print all**.

```
[admin@dzeltenais_burkaans] /ip firewall mangle> print all stats
Flags: X - disabled, I - invalid, D - dynamic
 #   CHAIN              ACTION                  BYTES           PACKETS
 0   prerouting         mark-routing            17478158        127631
 1   prerouting         mark-routing            782505          4506
 2 D forward            change-mss              0               0
 3 D forward            change-mss              0               0
 4 D forward            change-mss              0               0
 5 D forward            change-mss              129372          2031
```

Or to print only dynamic rules use **print dynamic**

```
[admin@dzeltenais_burkaans] /ip firewall mangle> print stats dynamic
Flags: X - disabled, I - invalid, D - dynamic
 #   CHAIN              ACTION                  BYTES           PACKETS
 0 D forward            change-mss              0               0
 1 D forward            change-mss              0               0
```

```
 2 D forward            change-mss              0               0
 3 D forward            change-mss              132444          2079
```

Property	Description
reset-counters (*id*)	Reset statistics counters for specified firewall rules.
reset-counters-all ()	Reset statistics counters for all firewall rules.

Basic examples

If you want to "hide" the private LAN 192.168.0.0/24 "behind" one address 10.5.8.109 given to you by the ISP, you should use the source network address translation (masquerading) feature of the MikroTik router. The masquerading will change the source IP address and port of the packets originated from the network 192.168.0.0/24 to the address 10.5.8.109 of the router when the packet is routed through it.

To use masquerading, a source NAT rule with action 'masquerade' should be added to the firewall configuration:

```
/ip firewall nat add chain=srcnat action=masquerade out-interface=Public
```

All outgoing connections from the network 192.168.0.0/24 will have source address 10.5.8.109 of the router and source port above 1024. No access from the Internet will be possible to the Local addresses. If you want to allow connections to the server on the local network, you should use destination Network Address Translation (NAT).

If you want to link Public IP 10.5.8.200 address to Local one 192.168.0.109, you should use destination address translation feature of the MikroTik router. Also if you want allow Local server to talk with outside with given Public IP you should use source address translation, too.

Add Public IP to Public interface:

```
/ip address add address=10.5.8.200/32 interface=Public
```

Add rule allowing access to the internal server from external networks:

```
/ip firewall nat add chain=dstnat dst-address=10.5.8.200 action=dst-nat \
      to-addresses=192.168.0.109
```

Add rule allowing the internal server to talk to the outer networks having its source address translated to 10.5.8.200:

```
/ip firewall nat add chain=srcnat src-address=192.168.0.109 action=src-nat \
      to-addresses=10.5.8.200
```

If you want to link Public IP subnet 11.11.11.0/24 to local one 2.2.2.0/24, you should use destination address translation and source address translation features with action=netmap.

```
/ip firewall nat add chain=dstnat dst-address=11.11.11.0/24 \
      action=netmap to-addresses=2.2.2.0/24

/ip firewall nat add chain=srcnat src-address=2.2.2.0/24 \
      action=netmap to-addresses=11.11.11.0/24
```

Same can be written using different address notation, that still have to match with the described network

```
/ip firewall nat add chain=dstnat dst-address=11.11.11.0-11.11.11.255 \
      action=netmap to-addresses=2.2.2.0-2.2.2.255

/ip firewall nat add chain=srcnat src-address=2.2.2.0-2.2.2.255 \
```

```
      action=netmap to-addresses=11.11.11.0-11.11.11.255
```

If you would like to direct requests for a certain port to an internal machine (sometimes called opening a port, port mapping), you can do it like this:

```
/ip firewall nat add chain=dstnat dst-port=1234 action=dst-nat protocol=tcp to-address=192.168.1.1 to-port=1234
```

This rule translates to: *when an incoming connection requests TCP port 1234, use the DST-NAT action and redirect it to local address 192.168.1.1 and the port 1234*

[Top | Back to Content]

Manual:First time startup

Applies to RouterOS: 2.9, v3, v4

Overview

After you have installed the RouterOS software, or turned on the Router for the first time, there are various ways how to connect to it:

- Accessing Command Line Interface (CLI) via Telnet, ssh, serial cable or even keyboard and monitor if router has VGA card.
- Accessing Web based GUI (WebFig)
- Using WinBox configuration utility

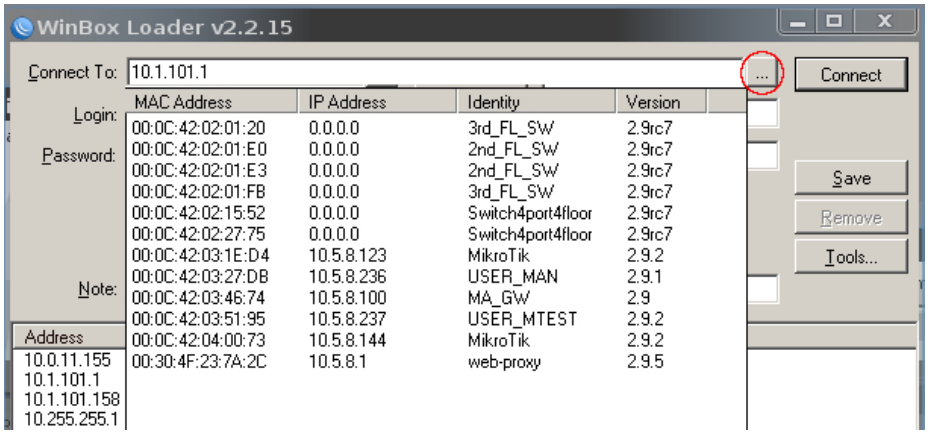
Every router is factory pre-configured with IP address 192.168.88.1/24 on ether1 port. Default username is **admin** with empty password.

Additional configuration may be set depending on RouterBoard model. For example, RB750 ether1 is configured as WAN port and any communication with the router through that port is not possible. List of RouterBOARD models and their default configurations can be found in this article.

Winbox

Winbox is configuration utility that can connect to the router via MAC or IP protocol. Latest winbox version can be downloaded from our demo router [1].

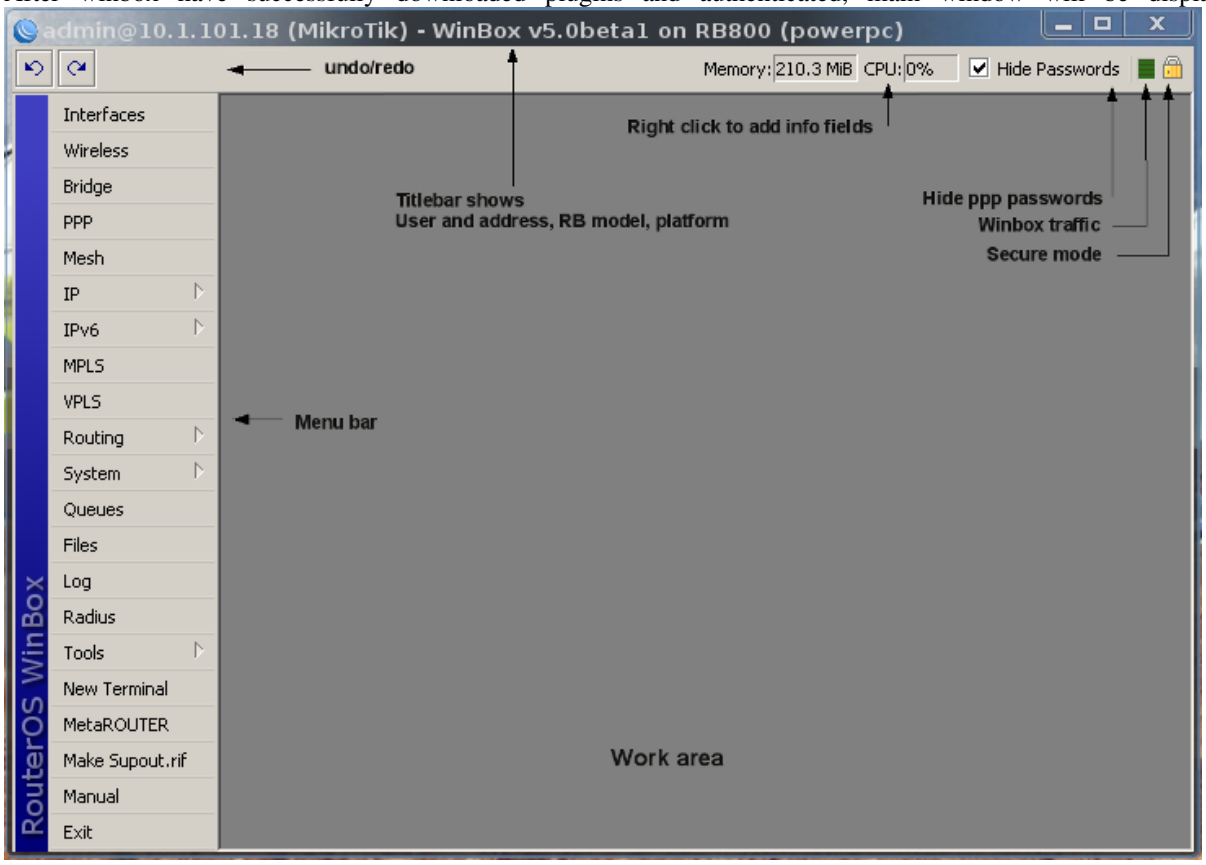
Run Winbox utility, then click the **[...]** button and see if Winbox finds your Router and it's MAC address. Winbox neighbor discovery will discover all routers on the broadcast network. If you see routers on the list, connect to it by clicking on MAC address and pressing **Connect** button.



Winbox will try download plugins from the router, if it is connecting for the first time to the router with current version. Note that it may take about one minute to download all plugins if winbox is connected with MAC protocol.

This method works with any device that runs RouterOS. Your PC needs to have MTU 1500

After winbox have successfully downloaded plugins and authenticated, main window will be displayed:

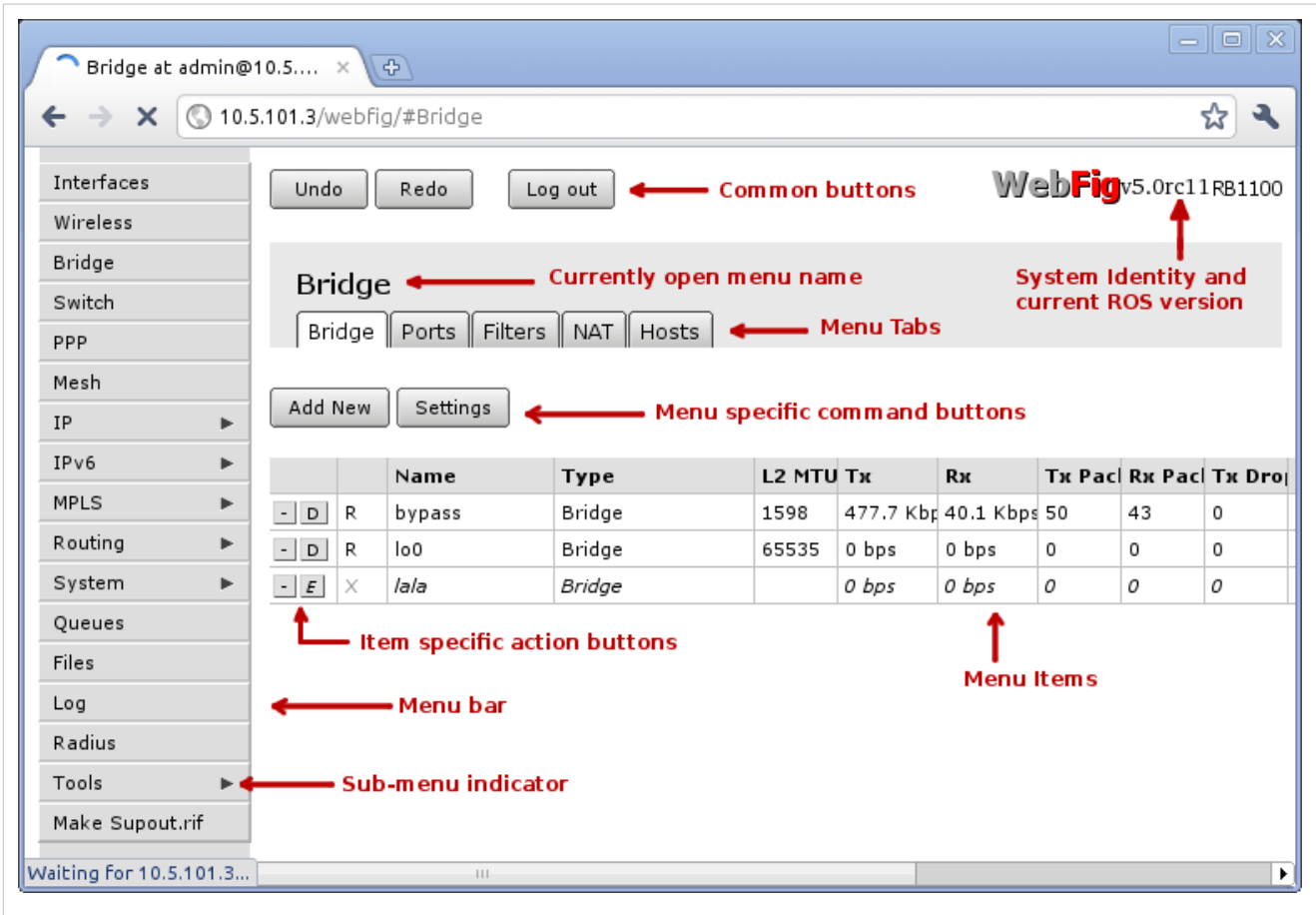


If winbox cannot find any routers, make sure that your Windows computer is directly connected to the router with an Ethernet cable, or at least they both are connected to the same switch. As MAC connection works on Layer2, it is possible to connect to the router even without IP address configuration. Due to the use of broadcasting MAC connection is not stable enough to use continuously, therefore it is not wise to use it on a real production / live network!. MAC connection should be used only for initial configuration.

Follow winbox manual for more information.

WebFig

If you have router with default configuration, then IP address of the router can be used to connect to the Web interface. WebFig has almost the same configuration functionality as Winbox.



Please see following articles to learn more about web interface configuration:

- Initial Configuration with WebFig
- General WebFig Manual

CLI

Command Line Interface (CLI) allows configuration of the router's settings using text commands. Since there is a lot of available commands, they are split into groups organized in a way of hierarchical menu levels. Follow console manual for CLI syntax and commands.

There are several ways how to access CLI:

- winbox terminal
- telnet
- ssh
- serial cable etc.

Serial Cable

If your device has a Serial port, you can use a console cable (or Null modem cable)

Plug one end of the serial cable into the console port (also known as a serial port or DB9 RS232C asynchronous serial port) of the RouterBOARD and the other end in your PC (which hopefully runs Windows or Linux). You can also use a USB-Serial adapter. Run a terminal program (HyperTerminal, or Putty on Windows) with the following parameters for All RouterBOARD models except 230:

```
115200bit/s, 8 data bits, 1 stop bit, no parity, flow control=none by default.
```

RouterBOARD 230 parameters are:

```
9600bit/s, 8 data bits, 1 stop bit, no parity, hardware (RTS/CTS) flow control by default.
```

If parameters are set correctly you should be able to see login prompt. Now you can access router by entering username and password:

```
MikroTik 4.15
MikroTik Login:

  MMM       MMM        KKK                           TTTTTTTTTTT      KKK
  MMMM     MMMM        KKK                           TTTTTTTTTTT      KKK
  MMM MMMM MMM  III  KKK  KKK  RRRRRR     OOOOOO      TTT     III  KKK  KKK
  MMM  MM  MMM  III  KKKKK     RRR  RRR  OOO  OOO     TTT     III  KKKKK
  MMM      MMM  III  KKK KKK   RRRRRR    OOO  OOO     TTT     III  KKK KKK
  MMM      MMM  III  KKK  KKK  RRR  RRR   OOOOOO      TTT     III  KKK  KKK

  MikroTik RouterOS 4.15 (c) 1999-2010       http://www.mikrotik.com/


[admin@MikroTik] >
```

Detailed description of CLI login is in login process section.

Monitor and Keyboard

If your device has a graphics card (ie. regular PC) simply attach a monitor to the video card connector of the computer *(note: RouterBOARD products don't have this, so use Method 1 or 2)* and see what happens on the screen. You should see a login promt like this:

```
MikroTik v3.16
Login:
```

Enter **admin** as the login name, and hit **enter** twice (because there is no password yet), you will see this screen:

```
  MMM       MMM        KKK                           TTTTTTTTTTT      KKK
  MMMM     MMMM        KKK                           TTTTTTTTTTT      KKK
  MMM MMMM MMM  III  KKK  KKK  RRRRRR     OOOOOO      TTT     III  KKK  KKK
  MMM  MM  MMM  III  KKKKK     RRR  RRR  OOO  OOO     TTT     III  KKKKK
  MMM      MMM  III  KKK KKK   RRRRRR    OOO  OOO     TTT     III  KKK KKK
  MMM      MMM  III  KKK  KKK  RRR  RRR   OOOOOO      TTT     III  KKK  KKK

  MikroTik RouterOS 3.16 (c) 2008       http://www.mikrotik.com/
```

```
Terminal ansi detected, using single line input mode
[admin@router] >
```

Now you can start configuring the router, by issuing the **setup** command.

This method works with any device that has a video card and keyboard connector

[Top | Back to Content **]**

References

[1] http://demo2.mt.lv/winbox/winbox.exe

Manual:Flashfig

Applies to RouterOS: v4

Description

Flashfig is an application for mass router configuration. It can be used by MikroTik distributors, ISPs or any other companies who need to apply RouterOS configuration to many routers in shortest possible time.

Flashfig applies MikroTik RouterOS **configuration** to any RouterBOARD within **3 seconds**. You can "flashfig" batch of routers, the only thing you need - **connect** RouterBOARD to network and **power** it.

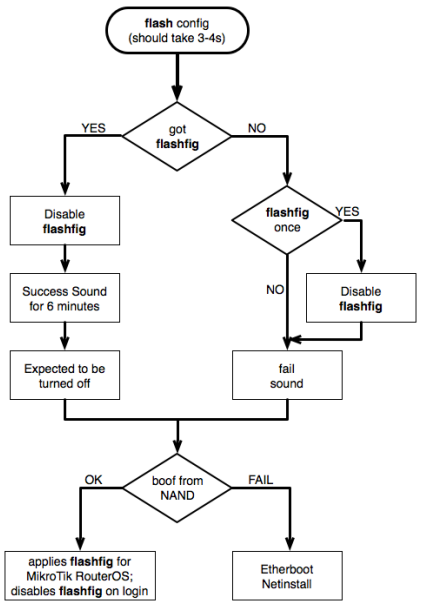
Flashfig runs on a Windows computer, Flashfig is available within Netinstall [1].

Flashfig is supported by all RouterBOARDs [1]. It works between computer with Flashfig and RouterBOARD in the same broadcast domain (direct Ethernet network connection is required).

Flashfig support is enabled on every new RouterBOARD by default from factory (RouterBOARDs manufactured after March 2010). For older models, Flashfig can be enabled via RouterBOOT or from MikroTik RouterOS console.

After Flashfig is used once on a brand new RouterBOARD, it is disabled to avoid unwanted reconfiguation at later time. To use Flashfig a second time on the same router, you need to enable it in Bootloader settings.

If RouterOS *reset-configuration* command is used later, Flashfig configuration is not loaded, but RouterOS default configuration.



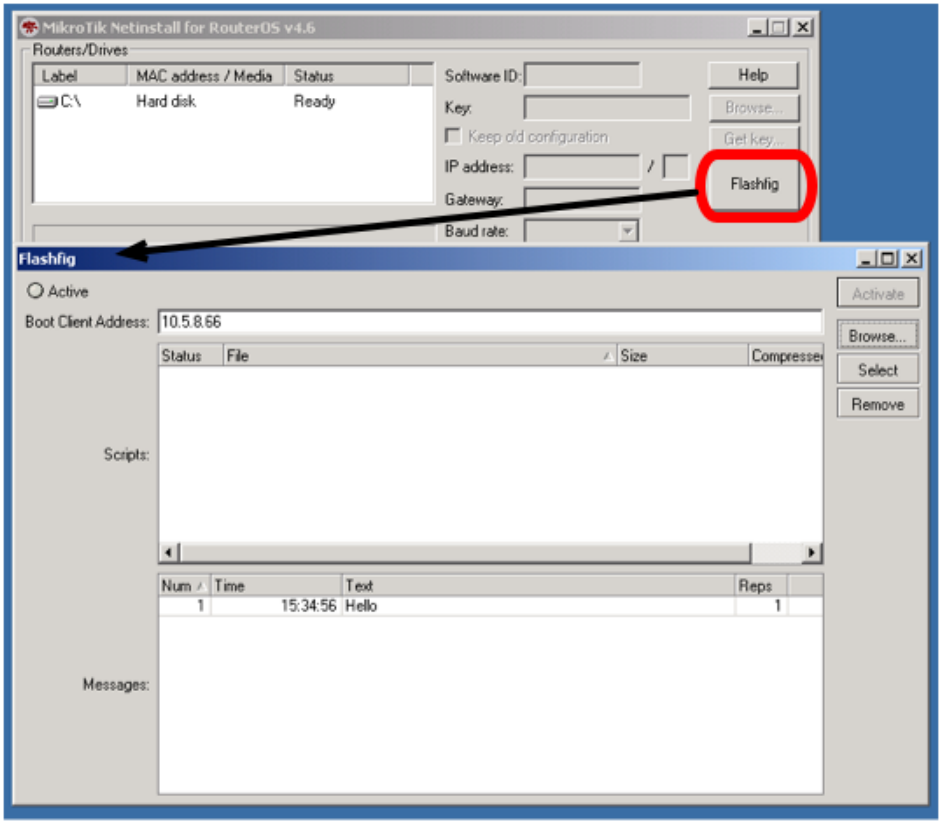
Flashfig diagram shows the procedure of Flashfig,

Flashfig Example

This is a step by step example of how to use Flashfig to set typical MikroTik RouterOS configuration to RouterBOARD.

Introduction

Flashfig is available from Netinstall,



Requirements

The Windows computer must be equipped with the following ports and contain the following files:

- Ethernet port;
- The .rsc file(s) with MikroTik RouterOS configuration (the same as export/import file);
- The latest NetInstall/Flashfig program available from the downloads [1] page;

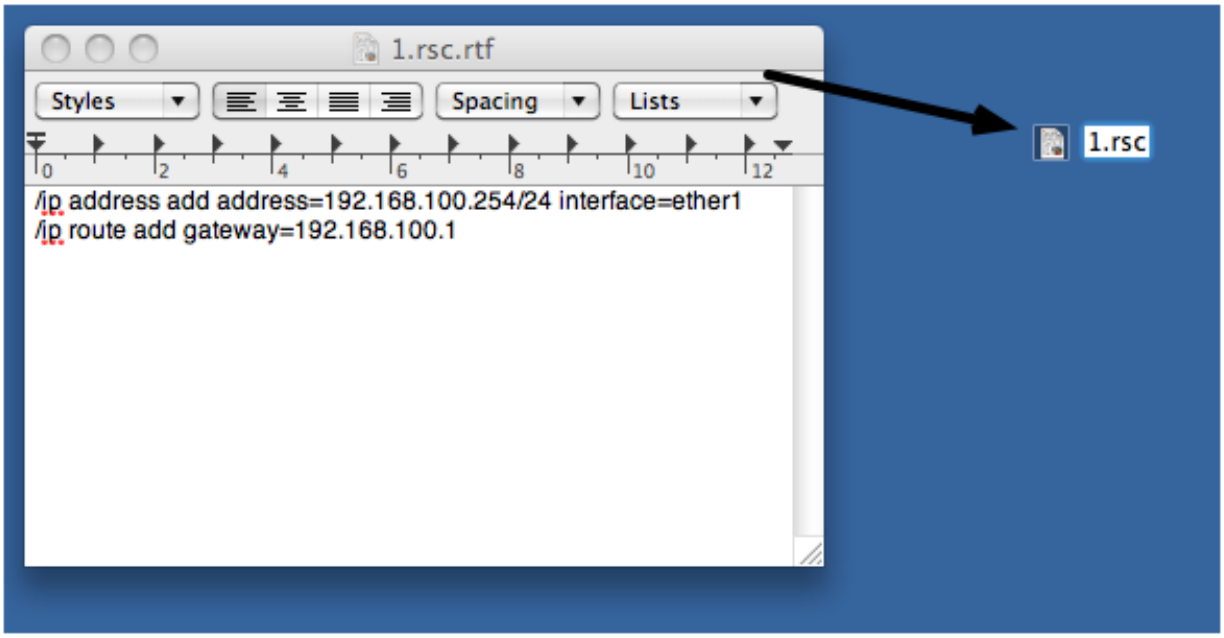
The RouterBOARD:

- Flashfig is supported by first boot of RouterBOARD;

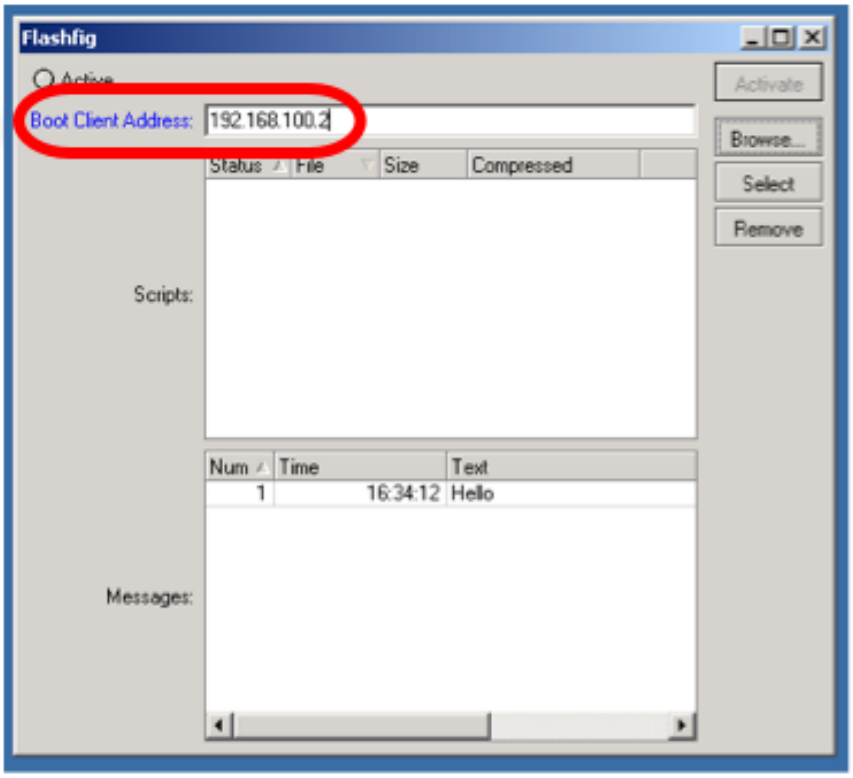
Pre-Configuration

Windows Computer

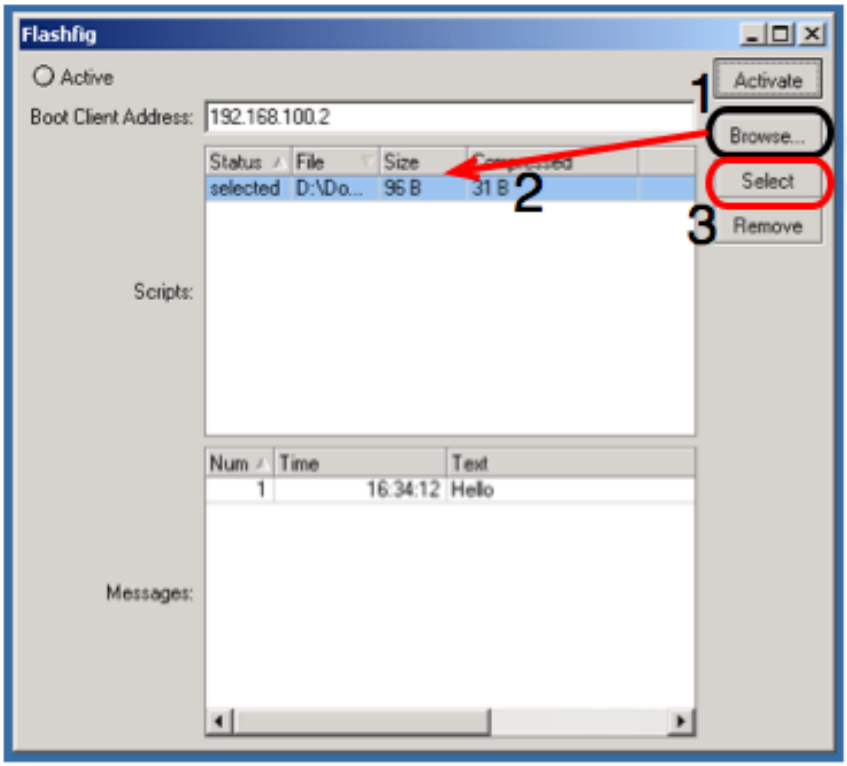
- Run Flashfig;
- Prepare **.rsc** file, **.rsc** file is regular/import file, it accepts valid MikroTik RouterOS CLI commands. You can create .rsc file by any text-editor program (Notepad, Texteditor, TextEdit, Microsoft Word, OpenOffice Writer);



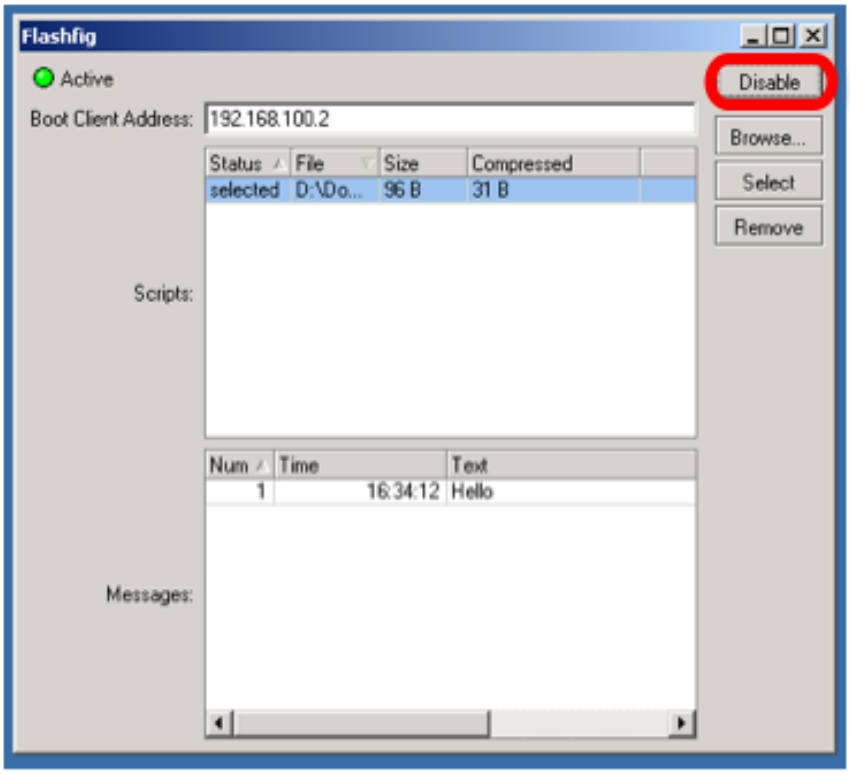
- Assign **Boot Client Address**, which should be address from the same subnet as configured on laptop Ethernet interface,



- **Browse** for **.rsc** MikroTik RouterOS configuration file to apply to RouterBOARD, highlight the file and **Select** to approve it,



- Activate Flashfig server, now it is ready to Flashfig. Note, any RouterBOARD will be flashfiged within the network, which is powered on with boot-device configured to **flash-boot** or **flash-boot-once-then-nand**,



RouterBOARD

- Flashfig mode is enabled on every RouterBOARD from factory by default, which means **no configuration** is required on RouterBOARD.
- If Flashfig is not enabled on your router, access the RouterBOARD with Winbox/Console and set the configuration,

```
/system routerboard settings set boot-device=flash-boot
```

or use more preferable option,

```
/system routerboard settigs set boot-device=flash-boot-once-then-nand
```

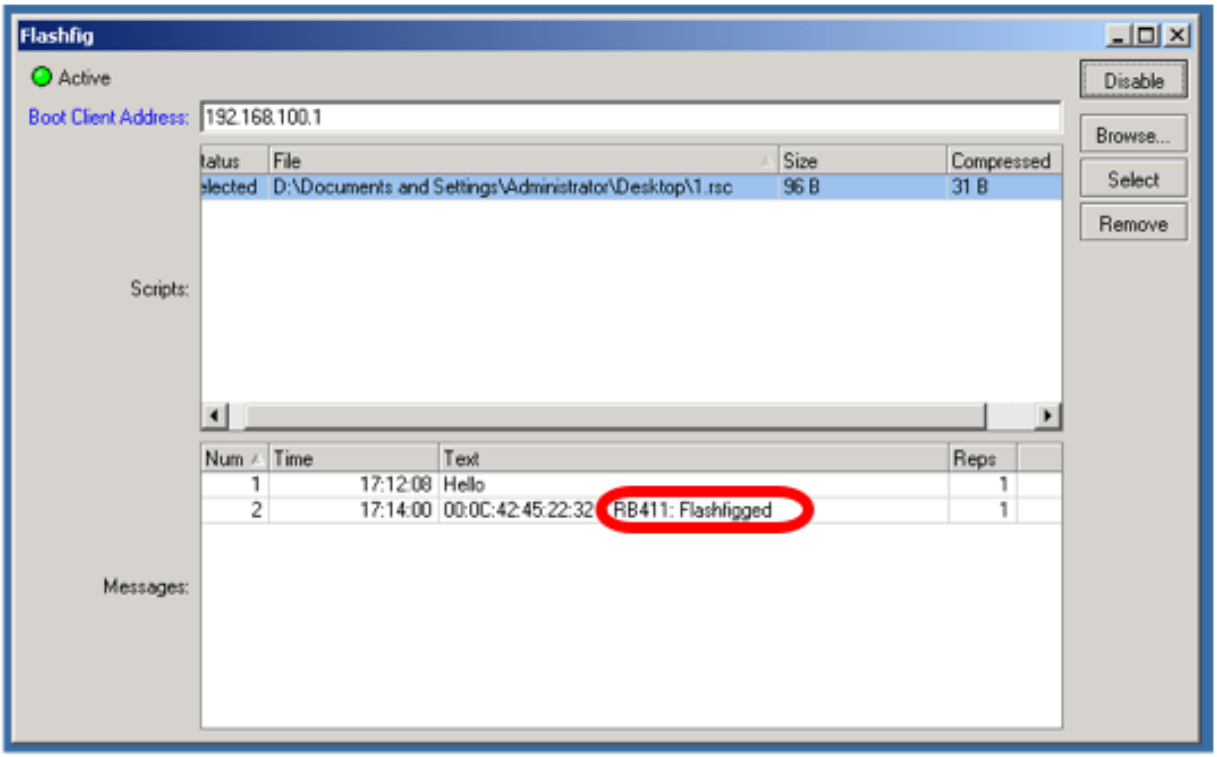
Your router is now ready for Flashfig.

Connect

Connect RouterBOARD and Flashfig computer to the same Local Area Network.

Run Flashfig

- Plug power for RouterBOARD
- Check the status on Flashfig program,



Log shows "RouterBOARD Flashfigged" and RouterBOARD should make sound/LED signal, now it is safe to unplug the router.

- Flashig **configuration** was applied to the RouterBOARD and it is **ready** to be used in production.

References

[1] http://www.mikrotik.com/download/netinstall-4.5b.zip

Manual:FTP server

Applies to RouterOS: 2.9, v3, v4

MikroTik RouterOS implements a File Transfer Protocol (FTP) server feature. It is intended to be used for software packages uploading, configuration script exporting and importing procedures, as well as for storing HotSpot servlet pages.

Specifications

- Packages required: *system*
- License required: *Level1*
- Submenu level: */file*
- Standards and Technologies: *FTP (RFC 959)*
- Hardware usage: *Not significant*

Description

MikroTik RouterOS has an industry standard FTP server facility. It uses ports 20 and 21 for communication with other hosts on the network. Uploaded files as well as exported configuration or backup files can be accessed under **/file** menu. There you can delete unnecessary files from the router.

Authorization for FTP service uses router's system user account names and passwords. The ftp local user policy controls the access rights to the FTP server.

Property Description

- **contents** (text) - file contents (for text files only; size limit - 4kB)
- **creation-time** (read-only: time) - item creation date and time
- **name** (read-only: name) - item name
- **package-architecture** (read-only: [text]) - RouterOS software package target machine architecture (for package files only)
- **package-build-time** (read-only: [date]) - RouterOS software package build time (for package files only)
- **package-name** (read-only: [text]) - RouterOS software package name (for package files only)
- **package-version** (read-only: [text]) - RouterOS software package version number (for package files only)
- **size** (read-only: integer) - package size in bytes
- **type** (read-only: text) - item type. Few file types are recognized by extension: backup, directory, package, script, ssh key, but other files are just marked by their extension (.html file, for example)

Command Description

- **print** - shows a list of files stored
  - *detail* - shows contents of files less that 4kB long
  - *edit [item] contents* - offers to edit file's contents with editor
  - *set [item] contents=[content]* - sets the file's contents to 'content'

Manual:System/GPS

Applies to RouterOS: v3, v4, v5 +

Summary

Sub-menu: `/system gps`
Standards: `GPS, NMEA 0183, Simple Text Output Protocol` [1]

Global Positioning System (GPS) is used for determining precise location of a GPS receiver.

Configuration Properties

Property	Description
channel (*integer [0..4294967295]*; Default: **0**)	Port channel used by device
enabled (*yes* | *no*; Default: **no**)	Whether GPS is enabled
port (*string*; Default:)	Name of the USB/Serial port where GPS receiver is connected
set-system-time (*yes* | *no*; Default: **no**)	Whether to set router's date and time to one received from GPS.

Monitoring Status

Command: `/system gps monitor`

This command is used for monitoring the data received from a GPS receiver

Parameters:

Property	Description
date-and-time (*date*)	Date and time received from GPS
latitude (*none* | *string*)	Latitude in DM (Degrees Minute decimal) format
longitude (*none* | *string*)	Longitude in DM (Degrees Minute decimal) format
speed (*none* | *string*)	Current moving speed of the GPS unit
bearing (*none* | *string*)	The compass direction toward which a GPS is moving
valid (*yes* | *no*)	
satellites (*integer*)	Number of satellites seen by the device.

Note: The time is not stratum 1 as RouterBOARD devices do not have PPS [2] *implemented*

Basic examples

Adjust port settings specific for your device

```
[admin@MikroTik] /port> set 0 baud-rate=4800 parity=odd
[admin@MikroTik] /port> print detail
Flags: I - inactive
 0   name="usb1" used-by="GPS" channels=1 baud-rate=4800 data-bits=8 parity=odd stop-bits=1 flow-control=none
```

Enable GPS

```
[admin@MikroTik] /system gps> set enable=yes port=usb1
[admin@MikroTik] /system gps> print
          enabled: yes
             port: usb1
          channel: 0
  set-system-time: no
```

Monitor status

```
[admin@MikroTik] /system gps> monitor
  date-and-time: sep/05/2011 07:28:54
       latitude: N 56 57' 32.568''
      longitude: E 24 9' 11.568''
       altitude: -23.600000m
          speed: 0.000000 km/h
        bearing: none
          valid: yes
     satellites: 3
```

[Top | Back to Content]

References

[1] http://www8.garmin.com/support/text_out.html
[2] http://en.wikipedia.org/wiki/Pulse_per_second

Manual:Tools/Graphing

Applies to RouterOS: v3, v4, v5 +

Summary

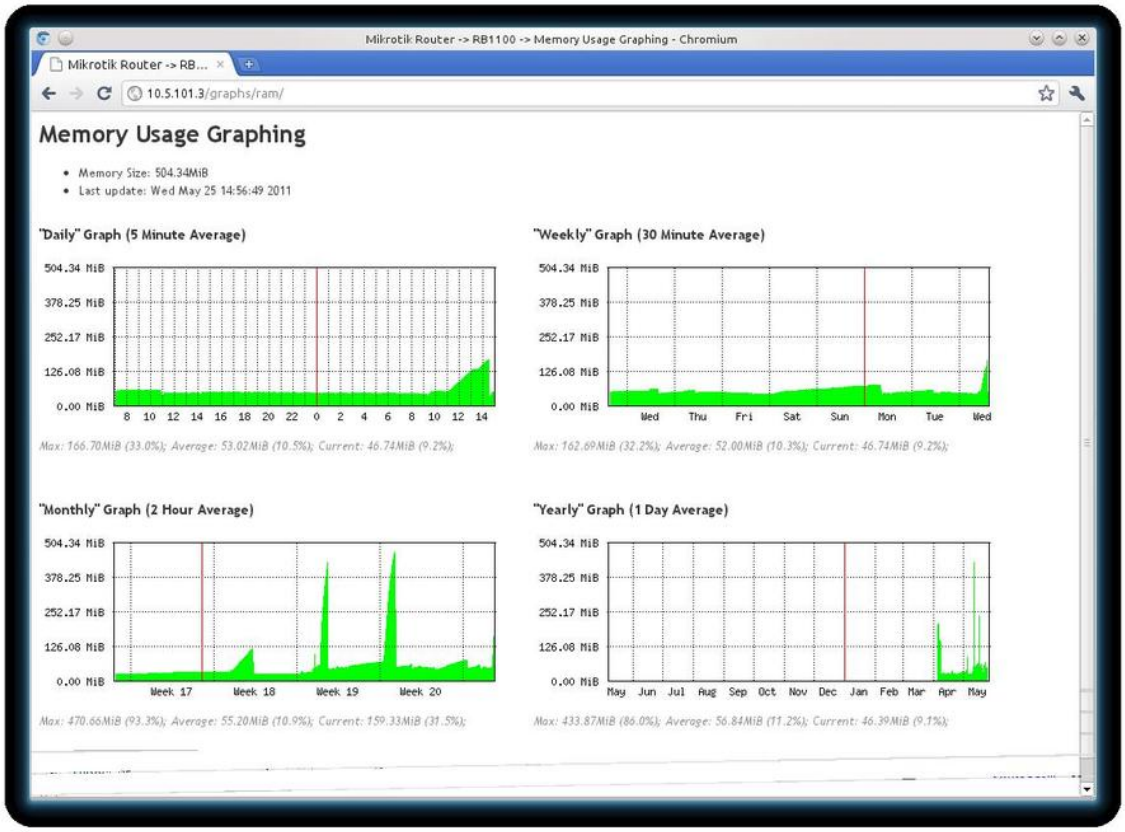
Graphing is a tool to monitor various RouterOS parameters over time and put collected data in nice graphs.

The Graphing tool can display graphics for:

- Routerboard health (voltage and temperature)
- Resource usage (CPU, Memory and Disk usage)
- Traffic which is passed through interfaces
- Traffic which is passed through simple queues

Graphing consists of two parts - first part collects information and other part displays data in a Web page. To access the graphics, type **http://[Router_IP_address]/graphs/** and choose a graphic to display in your Web browser.

Example of memory graphs:



General

Sub-menu `/tool graphing`

Common graphing configuration can be set in this submenu.

Properties

Property	Description
store-every (*24hours* | *5min* | *hour*; Default: **5min**)	How often to write collected data to system drive.
page-refresh (*integer* | *never*; Default: **300**)	How often graph page is refreshed

Interface graphing

Sub-menu `/tool graphing interface`

Sub-menu allows to configure on which interfaces graphing will collect bandwidth usage data.

Properties

Property	Description
allow-address (*IP/IPv6 prefix*; Default: **0.0.0.0/0**)	IP address range from which is allowed to access graphing information
comment (*string*; Default:)	Description of current entry
disabled (*yes* | *no*; Default: **no**)	Defines whether item is used
interface (*all* | *interface name*; Default: **all**)	Defines which interface will be monitored. **all** means that all interfaces on router will be monitored.
store-on-disk (*yes* | *no*; Default: **yes**)	Defines whether to store collected information on system drive.

Queue graphing

Sub-menu `/tool graphing queue`

Sub-menu allows to configure about which simple queues graphing will collect bandwidth usage data.

Properties

Property	Description
allow-address (*IP/IPv6 prefix*; Default: **0.0.0.0/0**)	IP address range from which is allowed to access graphing information
allow-target (*yes* | *no*; Default: **yes**)	Whether to allow access to graphs from queue's target-address
comment (*string*; Default:)	Description of current entry
disabled (*yes* | *no*; Default: **no**)	Defines whether item is used
simple-queue (*all* | *queue name*; Default: **all**)	Defines which queues will be monitored. **all** means that all queues on router will be monitored.
store-on-disk (*yes* | *no*; Default: **yes**)	Defines whether to store collected information on system drive.

Note: If simple queue has target-address set to 0.0.0.0/0 everyone will be able to access queue graphs even if allow address is set to specific address. This happens because by default queue graphs are accessible also from target address.

Resource graphing

Sub-menu `/tool graphing resource`

Sub-menu allows to enable graphing of system resources. Graphing collects data of:

- CPU usage
- Memory usage
- Disk usage

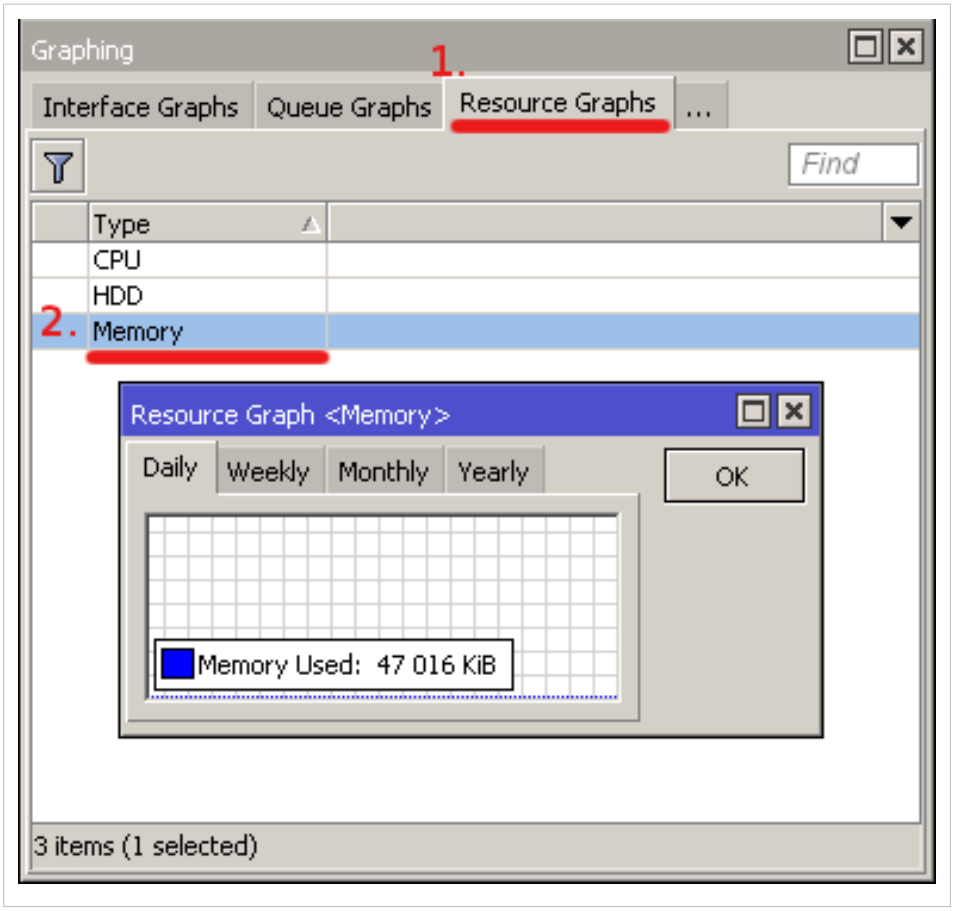
Properties

Property	Description
allow-address (*IP/IPv6 prefix*; Default: **0.0.0.0/0**)	IP address range from which is allowed to access graphing information
comment (*string*; Default:)	Description of current entry
disabled (*yes* | *no*; Default: **no**)	Defines whether item is used
store-on-disk (*yes* | *no*; Default: **yes**)	Defines whether to store collected information on system drive.

Graphing graphics in WinBox

Winbox allows to view the same collected information as in web page. Open **Tools->Graphing** window. Double click on entry of which you want to see graphs.

Image below shows winbox graphs of memory usage:



[Top | Back to Content]

Manual:Grounding

Introduction

The installation infrastructure (towers and masts), as well as antennas and the router itself must be properly grounded, and lightning arrestors must be installed on all external antenna cables (near the antennas or on the antennas themselves) to prevent equipment damage and human injury. Note that lightning arrestors will not have any effect if not grounded.

Use 1 AWG (7mm in diameter) wire with corrosion-resistant connectors for grounding. Be sure to check that the grounding infrastructure you use is indeed functional (as opposed to decorative-only grounding present on some sites). For smaller devices you can use thinner wire.

1. Only shielded and outdoor usage Ethernet cables should be used, magnetic shield should be grounded via shielded RJ-45 connector or via additional wire that is soldered to RJ45 or ground wire.
2. Grounding wire should be connected to RouterBOARD (to the mounting point where board is fastened to the outdoor box), this wire is connected to bottom of the tower and connection to the tower is according to the standards. Antenna grounding wire is connected near RouterBOARD Outdoor case, this wire could be connected to the same RouterBOARD grounding wire.
3. Ethernet port ligthing protectors are not recommended, as most of them are not intended to use for PoE (they are shortening PoE supply). If protectors are used, they could be placed at the outdoor case, where RouterBOARD and grounding pads are connected.

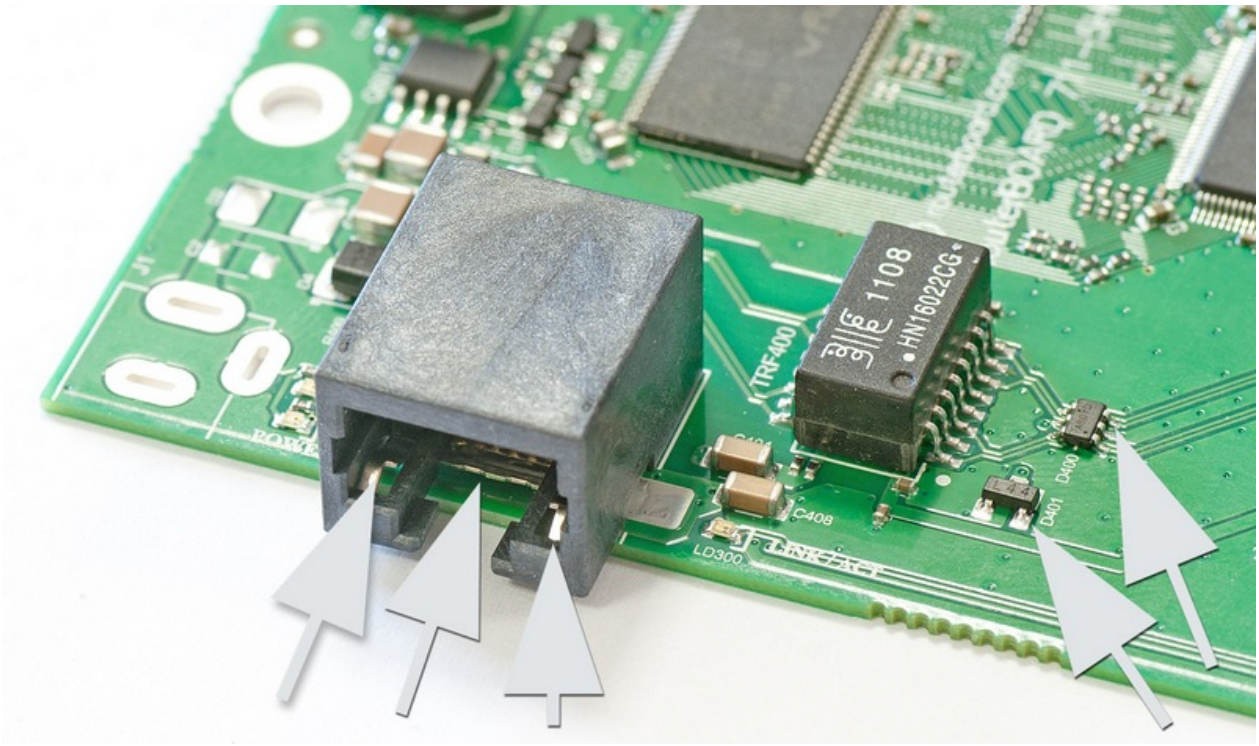
Shielded cable

Example grounding wire attachment screw on an outdoor case:

ESD Protection on RouterBOARD devices

1. Three arrows mark the grounding inside the ethernet port, the shielded cable connects it's shield to these two grounding pins via the metallic ethernet connector.
2. The middle arrow points to the metal plate inside the port, which connects the grounding pins to the board. The board needs to be grounded at the mounting hole (put grounding wire on the screw when you mount the board inside a case). Any surges will go from the grounding pins, to the grounding plate, to the board, and then to the grounding installation.
3. The two separate arrows show the ESD protection chips on the board - in case there was no shielded cable, to protect the CPU and other parts of the board.

The protection is not too effective if you only use shielded cable, and don't ground the board itself. You need to do both things to be successful. See below for possible methods, option 1 is recommended.



Grounding RouterBOARD installations

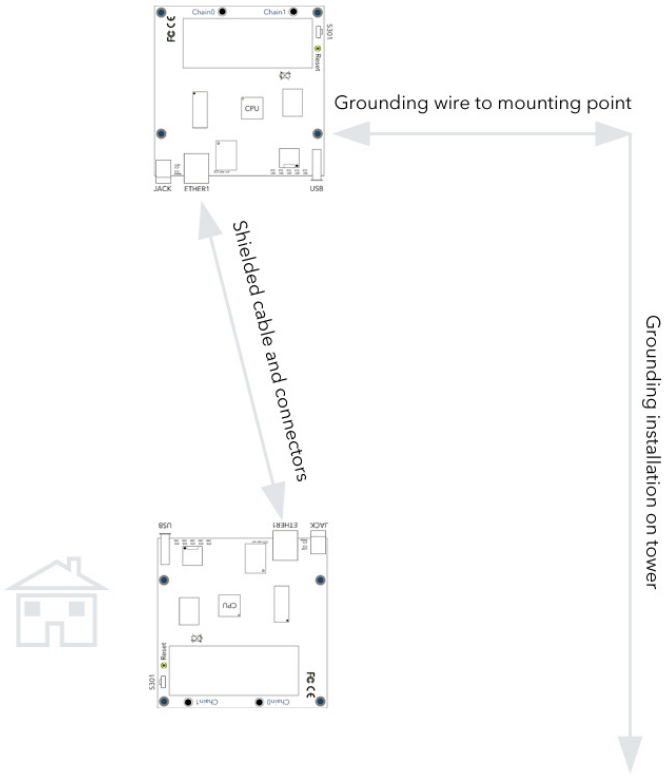
There are two methods, one of them more effective than the other.

1. **Using a Shielded cable + Board is grounded**: If you connect grounding to the mounting point of the RB711 (or the mounting loop inside SXT door), you don't necessary need to ground the device at other end of the shielded cable. Just using a shielded cable is enough. Special PoE is also not needed. This is the best option to protect against all ESD damage.
2. **Using only shielded cable**: If you can't ground the RB711/SXT/other device itself, you can ground the device on the other end of your shielded cable (switch, router, etc). If you need to use PoE, you will need to use our older PoE injector with shielded connectors, because it allows shielded cable to be used. This method is not recommended, better ground the board itself also (option 1).

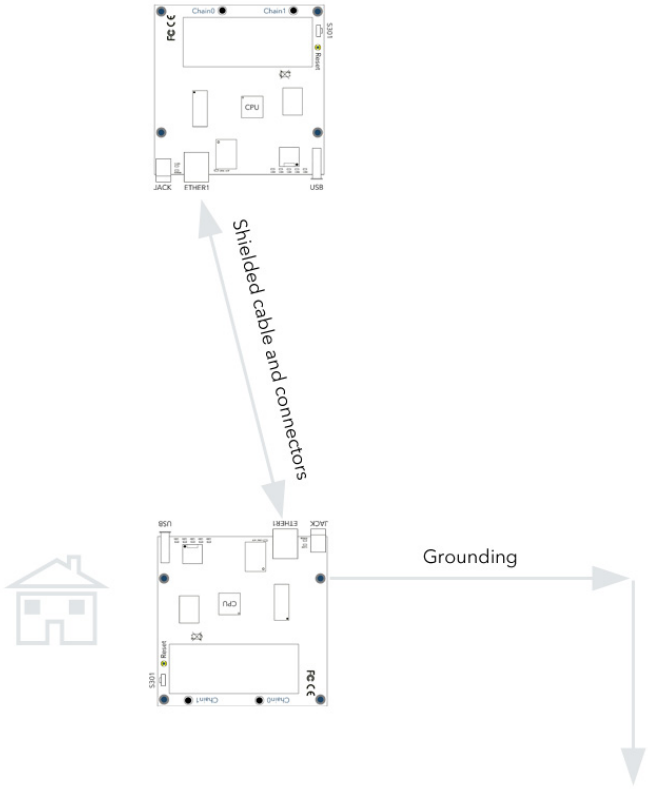
PoE with shielded connectors

Illustrations of the above methods

Method #1 (shielded cable + grounding of the device):



Method #2 (only shielded cable):



Note! Even if you don't ground the outdoor wireless device, and only use a shielded cable, you should still ground the device it's connected to (indoors). Ie. the switch, routerboard or PC.

Manual:Hotspot Introduction

Summary

HotSpot is a way to authorize users to access some network resources, but does not provide traffic encryption. To log in, users may use almost any web browser (either HTTP or HTTPS protocol), so they are not required to install additional software. The gateway is accounting the uptime and amount of traffic each client have used, and also can send this information to a RADIUS server. The HotSpot system may limit each particular user's bitrate, total amount of traffic, uptime and some other parameters mentioned further in this document.

The HotSpot system is targeted to provide authentication within a local network (for the local network users to access the Internet), but may as well be used to authorize access from outer networks to access local resources (like an authentication gateway for the outside world to access your network). It is possible to allow users to access some web pages without authentication using Walled Garden feature.

Getting an Address

First of all, a client have to get an IP address. It may be set on the client statically, or leased from a DHCP server. The DHCP server may provide ways of binding lent IP addresses to clients MAC addresses, if required. The HotSpot system does not care how client get an address before he/she gets to the HotSpot login page.

Moreover, HotSpot server may automatically and transparently change any IP address (yes, meaning really any IP address) of a client to a valid unused address from the selected IP pool. If a user is able to get his/her Internet connection working at their place, he/she will be able to get his/her connection working in the HotSpot network. This feature gives a possibility to provide a network access (for example, Internet access) to mobile clients that are not willing (or are disallowed, not qualified enough or otherwise unable) to change their networking settings. The users will not notice the translation (i.e., there will not be any changes in the users' config), but the router itself will see completely different (from what is actually set on each client) source IP addresses on packets sent from the clients (even the firewall mangle table will 'see' the translated addresses). This technique is called **one-to-one NAT**, but is also known as "Universal Client" as that is how it was called in the RouterOS version 2.8.

One-to-one NAT accepts any incoming address from a connected network interface and performs a network address translation so that data may be routed through standard IP networks. Clients may use any preconfigured addresses. If the one-to-one NAT feature is set to translate a client's address to a public IP address, then the client may even run a server or any other service that requires a public IP address. This NAT is changing source address of each packet just after it is received by the router (it is like source NAT that is performed early in the packet path, so that even firewall mangle table, which normally 'sees' received packets unaltered, can only 'see' the translated address).

Note: arp mode must be enabled on the interface where one-to-one NAT is used

Before the authentication

When enabling HotSpot on an interface, the system automatically sets up everything needed to show login page for all clients that are not logged in. This is done by adding dynamic destination NAT rules, which you can observe on a working HotSpot system. These rules are needed to redirect all HTTP and HTTPS requests from unauthorized users to the HotSpot authentication proxy. Other rules that are also inserted, will be described later in a special section of this manual.

In most common setup, opening any HTTP page will bring up the HotSpot servlet login page (which can be customized extensively, as described later on). As normal user behavior is to open web pages by their DNS names, a valid DNS configuration should be set up on the HotSpot gateway itself (it is possible to reconfigure the gateway so that it will not require local DNS configuration, but such a configuration is impractical and thus not recommended).

Walled Garden

You may wish not to require authorization for some services (for example to let clients access the web server of your company without registration), or even to require authorization only to a number of services (for example, for users to be allowed to access an internal file server or another restricted area). This can be done by setting up Walled Garden system.

When a not logged-in user requests a service allowed in the Walled Garden configuration, the HotSpot gateway does not intercept it, or in case of HTTP, simply redirects the request to the original destination. Other requests are redirected to the HotSpot servlet (login page infrastructure). When a user is logged in, there is no effect of this table on him/her.

Walled Garden for HTTP requests is using the embedded proxy server . This means that all the configured parameters of that proy server will also be effective for the WalledGarden clients (as well as for all clients that have transparent proxy enabled)

Authentication

There are currently 6 different authentication methods. You can use one or more of them simultaneously:

- **HTTP PAP** - simplest method, which shows the HotSpot login page and expect to get the authentication info (i.e. username and password) in plain text. Another use of this method is the possibility of hard-coded authentication information in the servlet's login page simply creating the appropriate link.

Note: passwords are not encrypted when transferred over the network

- **HTTP CHAP** - standard method, which includes CHAP challenge in the login page. The CHAP MD5 hash challenge is used together with the user's password for computing the string which will be sent to the HotSpot gateway. The hash result (as a password) together with username is sent over network to HotSpot service (so, password is never sent in plain text over IP network). On the client side, MD5 algorithm is implemented in JavaScript applet, so if a browser does not support JavaScript (like, for example, Internet Explorer 2.0 or some PDA browsers) or it has JavaScipt disabled, it will not be able to authenticate users. It is possible to allow unencrypted passwords to be accepted by turning on HTTP PAP authentication method, but it is not recommended due to security considerations.
- **HTTPS** - the same as HTTP PAP, but uses SSL protocol to encrypt transmissions. HotSpot user just sends his/her password without additional hashing (note that there is no need to worry about plain-text password exposure over the network, as the transmission itself is encrypted). In either case, **HTTP POST** method (if not possible, then - **HTTP GET** method) is used to send data to the HotSpot gateway.
- **HTTP cookie** - after each successful login, a cookie is sent to the web browser and the same cookie is added to active HTTP cookie list. Next time the same user will try to log in, web browser will send the saved HTTP cookie. This cookie will be compared with the one stored on the HotSpot gateway and only if source MAC address and randomly generated ID matches the ones stored on the gateway, user will be automatically logged in using the login information (username and password pair) was used when the cookie was first generated. Otherwise, the user will be prompted to log in, and in the case authentication is successful, old cookie will be removed from the local HotSpot active cookie list and the new one with different random ID and expiration time will be added to the list and sent to the web browser. It is also possible to erase cookie on user manual logoff (not in the default server pages, but you can modify them to perform this). This method may only be used together

with HTTP PAP, HTTP CHAP or HTTPS methods as there would be nothing to generate cookies in the first place otherwise.

- **MAC address** - try to authenticate clients as soon as they appear in the hosts list (i.e., as soon as they have sent any packet to the HotSpot server), using client's MAC address as username.
- **Trial** - users may be allowed to use the service free of charge for some period of time for evaluation, and be required to authenticate only after this period is over. HotSpot can be configured to allow some amount of time per MAC address to be freely used with some limitations imposed by the provided user profile. In case the MAC address still has some trial time unused, the login page will contain the link for trial login. The time is automatically reset after the configured amount of time (so that, for example, any MAC address may use 30 minutes a day without ever registering). The username of such a user (as seen in the active user table and in the login link) is "T-XX:XX:XX:XX:XX:XX" (where XX:XX:XX:XX:XX:XX is his/her MAC address). The authentication procedure will not ask RADIUS server permission to authorise such a user.

HotSpot can authenticate users consulting the local user database or a RADIUS server (local database is consulted first, then - a RADIUS server). In case of HTTP cookie authentication via RADIUS server, the router will send the same information to the server as it was used when the cookie was first generated. If authentication is done locally, profile corresponding to that user is used, otherwise (in case RADIUS reply did not contain the group for that user) the default profile is used to set default values for parameters, which are not set in RADIUS access-accept message. For more information on how the interaction with a RADIUS server works, see the respective manual section.

The HTTP PAP method also makes it possible to authenticate by requesting the page:

```
/login?username=username&password=password
```

In case you want to log in using telnet connection, the exact HTTP request would look like that:

```
GET /login?username=username&password=password HTTP/1.0
```

Note that the request is case-sensitive.

Authorization

After authentication user gets access to the Internet and receives some limitations (which are user profile specific). HotSpot may also perform a one-to-one NAT for the client, so that a particular user would always receive the same IP address regardless of what PC is used.

The system will automatically detect and redirect requests to a proxy server that client is using (if any; it may be set in his/her settings to use an unknown proxy server) to the proxy server embedded in the router.

Authorization may be delegated to a RADIUS server, which delivers similar configuration options as the local database. For any user requiring authorization, a RADIUS server gets queried first, and if no reply received, the local database is examined. RADIUS server may send a Change of Authorization request according to standards to alter the previously accepted parameters.

Advertisement

The same proxy used for unauthorized clients to provide Walled-Garden facility, may also be used for authorized users to show them advertisement popups. Transparent proxy for authorized users allows to monitor http requests of the clients and to take some action if required. It enables the possibility to open status page even if client is logged in by mac address, as well as to show advertisements time after time

When the time has come to show an advertisement, the server redirects client's web browser to the status page. Only requests, which provide html content, are redirected (images and other content will not be affected). The status page displays the advertisement and next advertise-interval is used to schedule next advertisement. If status page is unable to display an advertisement for configured timeout starting from moment, when it is scheduled to be shown, client access is blocked within walled-garden (just as unauthorized clients are). Client is unblocked when the scheduled page is finally shown. Note that if popup windows are blocked in the browser, the link on the status page may be used to open the advertisement manually.

While client is blocked, FTP and other services are not allowed. Thus requiring client to open an advertisement for any Internet activity not especially allowed by the Walled-Garden.

Accounting

The HotSpot system implement accounting internally, you are not required to do anything special for it to work. The accounting information for each user may be sent to a RADIUS server.

Configuration menus

- /ip hotspot - HotSpot servers on particular interfaces (one server per interface). HotSpot server must be added in this menu in order for HotSpot system to work on an interface /ip hotspot profile - HotSpot server profiles. Settings, which affect login procedure for HotSpot clients are configured here. More than one HotSpot servers may use the same profile
- /ip hotspot host - dynamic list of active network hosts on all HotSpot interfaces. Here you can also find IP address bindings of the one-to-one NAT
- /ip hotspot ip-binding - rules for binding IP addresses to hosts on hotspot interfaces
- /ip hotspot service-port - address translation helpers for the one-to-one NAT
- /ip hotspot walled-garden - Walled Garden rules at HTTP level (DNS names, HTTP request substrings)
- /ip hotspot walled-garden ip - Walled Garden rules at IP level (IP addresses, IP protocols)
- /ip hotspot user - local HotSpot system users
- /ip hotspot user profile - local HotSpot system users profiles (user groups)
- /ip hotspot active - dynamic list of all authenticated HotSpot users
- /ip hotspot cookie - dynamic list of all valid HTTP cookies

[Top | Back to Content]

Manual:System/Health

Summary

Hardware that supports monitoring will display different information about hardware status, like temperature, voltage.

Warning: For feature availablity on RouterBOARD products check routerboard.com [1]

Voltage

Routers that support voltage monitoring will display supplied voltage value. In CLI/Winbox it will display volts. In scripts/API/SNMP this will be dV or value showed in CLI/Winbox multiplied by 10

Note: Routers that have PEXT and PoE power input are calibrated using PEXT, as result value showed over PoE can be lower than input voltage due to additional ethernet protection chains.

Temperature

Routers that support temperature monitoring will display temperature reading. In CLI/Winbox it will display degrees Celsius. In scripts/API/SNMP this will be value showed in CLI/Winbox multiplied by 10

Fan control

Using this menu users will be able to control fan behaviour on the router.

Warning: for *auto* mode to work you have to use fans that support monitoring (it will have 3 wires) If you have fan with only 2 wires (V+,GND) then you have to set fan-mode to manual. If control pulse cannot be detected, then router will switch between main and auxiliary fan and stop only when it detects fan with control

References

[1] http://routerboard.com

Manual:IP/Hotspot

HotSpot

The MikroTik HotSpot Gateway provides authentication for clients before access to public networks .

HotSpot Gateway features:

- different authentication methods of clients using local client database on the router, or remote RADIUS server;
- users accounting in local database on the router, or on remote RADIUS server;
- walled-garden system, access to some web pages without authorization;
- login page modification, where you can put information about the company;
- automatic and transparent change any IP address of a client to a valid address;

Sub Categories

List of reference sub-pages **Case studies** **List of examples**

<splist showparent=yes />

HotSpot Setup

The simplest way to setup HotSpot server on a router is by **`/ip hotspot setup`** command. Router will ask to enter parameters required to successfully set up HotSpot. When finished, default configuration will be added for HotSpot server.

```
[admin@MikroTik] /ip hotspot> setup
Select interface to run HotSpot on

hotspot interface: ether3
Set HotSpot address for interface

local address of network: 10.5.50.1/24
masquerade network: yes
Set pool for HotSpot addresses

address pool of network: 10.5.50.2-10.5.50.254
Select hotspot SSL certificate

select certificate: none
Select SMTP server

ip address of smtp server: 0.0.0.0
Setup DNS configuration

dns servers: 10.1.101.1
DNS name of local hotspot server

dns name: myhotspot
Create local hotspot user
```

```
name of local hotspot user: admin
password for the user:
[admin@MikroTik] /ip hotspot>
```

What was created:

```
[admin@MikroTik] /ip hotspot> print
Flags: X - disabled, I - invalid, S - HTTPS
 #   NAME         INTERFACE       ADDRESS-POOL       PROFILE       IDLE-TIMEOUT
 0   hotspot1     ether3          hs-pool-3          hsprof1       5m
[admin@MikroTik] /ip hotspot>
[admin@MikroTik] /ip pool> print
 # NAME                                      RANGES
 0 hs-pool-3                                 10.5.50.2-10.5.50.254
[admin@MikroTik] /ip pool> /ip dhcp-server
[admin@MikroTik] /ip dhcp-server> print
Flags: X - disabled, I - invalid
 #   NAME      INTERFACE    RELAY           ADDRESS-POOL    LEASE-TIME ADD-ARP
 0   dhcp1     ether3                       hs-pool-3       1h
[admin@MikroTik] /ip dhcp-server> /ip firewall nat
[admin@MikroTik] /ip firewall nat> print
Flags: X - disabled, I - invalid, D - dynamic
 0 X ;;; place hotspot rules here
     chain=unused-hs-chain action=passthrough

 1   ;;; masquerade hotspot network
     chain=srcnat action=masquerade src-address=10.5.50.0/24
[admin@MikroTik] /ip firewall nat>
```

Parameters asked during setup process

Parameter	Description
hotspot interface (*string*; Default: **allow**)	Interface name on which to run HotSpot. To run HotSpot on a bridge interface, make sure public interfaces are not included to the bridge ports.
local address of network (*IP*; Default: **10.5.50.1/24**)	HotSpot gateway address
masquerade network (*yes* | *no*; Default: **yes**)	Whether to masquerade HotSpot network, when **yes** rule is added to */ip firewall nat* with *action=masquerade*
address pool of network (*string*; Default: **yes**)	Address pool for HotSpot network, which is used to change user IP address to a valid address. Useful if providing network access to mobile clients that are not willing to change their networking settings.
select certificate (*none* | *import-other-certificate*; Default:)	Choose SSL certificate, when HTTPS authorization method is required.
ip address of smtp server (*IP*; Default: **0.0.0.0**)	IP address of the SMTP server, where to redirect HotSpot's network SMTP requests (25 TCP port)
dns servers (*IP*; Default: **0.0.0.0**)	DNS server addresses used for HotSpot clients, configuration taken from */ip dns* menu of the HotSpot gateway
dns name (*string*; Default: **""**)	domain name of the HotSpot server, full quality domain name is required, for example www.example.com

name of local hotspot user (*string*; Default: **"admin"**)	username of one automatically created HotSpot user, added to */ip hotspot user*
password for the user' (*string*; Default:)	Password for automatically created HotSpot user

ip hotspot

Menu is designed to manage HotSpot servers of the router. It is possible to run HotSpot on Ethernet, wireless, VLAN and bridge interfaces. One HotSpot server is allowed per interface. When HotSpot is configured on bridge interface, set HotSpot interface as *bridge* interface not as *bridge port*, do not add public interfaces to bridge ports. You can add HotSpot servers manually to */ip hotspot* menu, but it is advised to run */ip hotspot setup*, that adds all necessary settings.

- **name** (text) : HotSpot server's name or identifier
- **address-pool** (name **/** none; default: *none*) : address space used to change HotSpot client *any* IP address to a valid address. Useful for providing public network access to mobile clients that are not willing to change their networking settings
- **idle-timeout** (time **/** none; default: *5m*) : period of inactivity for unauthorized clients. When there is no traffic from this client (literally client computer should be switched off), once the timeout is reached, user is dropped from the HotSpot host list, its used address becomes available
- **interface** (name of interface) : interface to run HotSpot on
- **addresses-per-mac** (integer **/** unlimited; default: 2) : number of IP addresses allowed to be bind with the MAC address, when multiple HotSpot clients connected with one MAC-address
- **profile** (name; default: *default*) - HotSpot server default HotSpot profile, which is located in */ip hotspot profile*

ip hotspot active

HotSpot active menu shows all clients authenticated in HotSpot, menu is informational it is not possible to change anything here.

- **server** (read-only; name) : HotSpot server name client is logged in
- **user** (read-only; name) : name of the HotSpot user
- **domain** (read-only; text) : domain of the user (if split from username), parameter is used only with RADIUS authentication
- **address** (read-only; IP address) : IP address of the HotSpot user
- **mac-address** (read-only; MAC-address) : MAC-address of the HotSpot user
- **login-by** (read-only; multiple choice: cookie **/** http-chap **/** http-pap **/** https **/** mac **/** mac **/** trial) : authentication method used by HotSpot client
- **uptime** (read-only; time) : current session time of the user, it is showing how long user has been logged in
- **idle-time** (read-only; time) : the amount of time user has been idle
- **session-time-left** (read-only; time) : the exact value of session-time, that is applied for user. Value shows how long user is allowed to be online to be logged of automatically by **uptime** reached
- **idle-timeout** (read-only; time) : the exact value of the user's idle-timeout
- **keepalive-timeout** (read-only; time) : the exact value of the keepalive-timeout, that is applied for user. Value shows how long host can stay out of reach to be removed from the HotSpot
- **limit-bytes-in** (read-only; integer) : value shows how many bytes received from the client, option is active when the appropriate parameter is configured for HotSpot user
- **limit-bytes-out** (read-only; integer) : value shows how many bytes send to the client, option is active when the appropriate parameter is configured for HotSpot user

- **limit-bytes-total** (read-only; integer) : value shows how many bytes total were send/received from client, option is active when the appropriate parameter is configured for HotSpot user

ip hotspot host

Host table lists all computers connected to the HotSpot server. Host table is informational and it is not possible to change any value there

- **mac-address** (read-only; MAC-address) : HotSpot user MAC-address
- **address** (read-only; IP address) : HotSpot client original IP address
- **to-address** (read-only; IP address) : New client address assigned by HotSpot, it might be the same as original **address**
- **server** (read-only; name) : HotSpot server name client is connected to
- **bridge-port** (read-only; name) : /interface bridge port client connected to, value is unknown when HotSpot is not configured on the bridge
- **uptime** (read-only; time) : value shows how long user is online (connected to the HotSpot)
- **idle-time** (read-only; time) : time user has been idle
- **idle-timeout** (read-only; time) : value of the client idle-timeout (unauthorized client)
- **keeaplive-timeout** (read-only; time) : keepalive-timeout value of the unauthorized client
- **bytes-in** (read-only; integer) : amount of bytes received from unauthorized client
- **packet-in** (read-only; integer) : amount of packets received from unauthorized client
- **bytes-out** (read-only; integer) : amount of bytes send to unauthorized client
- **packet-out** (read-only; integer) : amount of packets send to unauthorized client

IP Bindings

Sub-menu: `/ip hotspot ip-binding`

IP-Binding HotSpot menu allows to setup static One-to-One NAT translations, allows to bypass specific HotSpot clients without any authentication, and also allows to block specific hosts and subnets from HotSpot network

Property	Description
address (*IP Range*; Default: **""**)	The original IP address of the client
mac-address (*MAC*; Default: **""**)	MAC address of the client
server (*string* | *all*; Default: **"all"**)	Name of the HotSpot server. • **all** - will be applied to all hotspot servers
to-address (*IP*; Default: **""**)	New IP address of the client, translation occurs on the router (client does not know anything about the translation)
type (*blocked* | *bypassed* | *regular*; Default: **""**)	Type of the IP-binding action • **regular** - performs One-to-One NAT according to the rule, translates **address** to **to-address** • **bypassed** - performs the translation, but excludes client from login to the HotSpot • **blocked** - translation is not performed and packets from host are dropped

Cookies

Sub-menu: `/ip hotspot cookie`

Menu contains all cookies sent to the HotSpot clients, which are authorized by cookie method, all the entries are read-only.

Property	Description
domain (*string*)	Domain name (if split from username)
expires-in (*time*)	How long the cookie is valid
mac-address (*MAC*)	Client's MAC-address
user (*string*)	HotSpot username

[Top | Back to Content]

Manual:HTB

Applies to RouterOS: 2.9, v3, v4

Theory

Structure

HTB (Hierarchical Token Bucket) is a classful queuing method that is useful for handling different kind of traffic. We have to follow three basic steps to create HTB:

- **Match and mark traffic –** classify traffic for further use. Consists of one or more matching parameters to select packets for the specific class.
- **Create rules (policy) to mark traffic –** put specific traffic class into specific queue and to define the actions that are taken for each class.
- **Attach policy for specific interface(-s) –** append policy for all interfaces (global-in, global-out or global-total), for specific interface or for specific parent queue.

HTB allows to create a hierarchical queue structure and determine relations between queues, like "parent-child" or "child-child".

As soon as queue has at least one child it becomes a **inner** queue, all queues without children - **leaf** queues. **Leaf** queues make actual traffic consumption, **Inner** queues are responsible only for traffic distribution. All **leaf** queues are treated on equal basis.

In RouterOS it is necessary to specify **parent** option to assign queue as a child to other queue

Dual Limitation

Each queue in HTB has two rate limits:

- **CIR** (Committed Information Rate) – (**limit-at** in RouterOS) worst case scenario, flow will get this amount of traffic no matter what (assuming we can actually send so much data)
- **MIR** (Maximal Information Rate) – (**max-limit** in RouterOS) best case scenario, rate that flow can get up to, if there queue's parent has spare bandwidth

In other words, at first **limit-at** (**CIR**) of the all queues will be satisfied, only then child queues will try to borrow the necessary data rate from their parents in order to reach their **max-limit** (**MIR**).

Note: **CIR** will be assigned to the corresponding queue no matter what. (even if max-limit of the parent is exceeded)

That is why, to ensure optimal (as designed) usage of dual limitation feature, we suggest to stick to these rules:

- Sum of committed rates of all children must be less or equal to amount of traffic that is available to parent.

  $CIR(parent)^* \geq CIR(child1) + ... + CIR(childN)$

  in case if parent is main parent CIR(parent)=MIR(parent)

- Maximal rate of any child must be less or equal to maximal rate of the parent

  $MIR\ (parent) \geq MIR(child1)\ \&\ MIR\ (parent) \geq MIR(child2)\ \&\ ...\ \&\ MIR\ (parent) \geq MIR(childN)$

Queue colors in Winbox:

- 0% - 50% available traffic used - green
- 51% - 75% available traffic used - yellow
- 76% - 100% available traffic used - red

Priority

We already know that **limit-at** (**CIR**) to all queues will be given out no matter what.

Priority is responsible for distribution of remaining parent queues traffic to child queues so that they are able to reach **max-limit**

Queue with higher priority will reach its **max-limit** before the queue with lower priority. 8 is the lowest priority, 1 is the highest.

Make a note that priority only works:

- for **leaf** queues - priority in **inner** queue have no meaning.
- if **max-limit** is specified (not 0)

Examples

In this section we will analyze HTB in action. To do that we will take one HTB structure and will try to cover all the possible situations and features, by changing the amount of incoming traffic that HTB have to recycle. and changing some options.

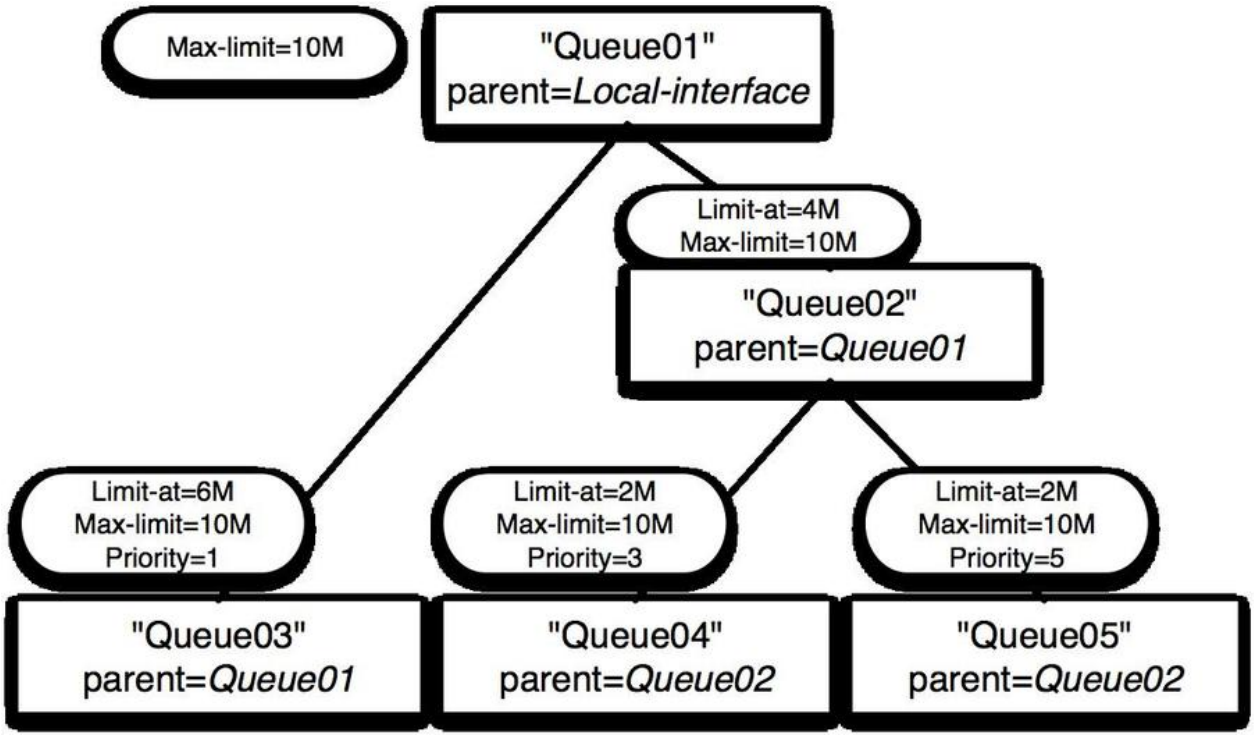
Structure

Our HTB structure will consist of 5 queues:

- **Queue01** inner queue with two children - **Queue02** and **Queue03**
- **Queue02** inner queue with two children - **Queue04** and **Queue05**
- **Queue03** leaf queue
- **Queue04** leaf queue
- **Queue05** leaf queue

Queue03, **Queue04** and **Queue05** are clients who require 10Mbps all the time Outgoing interface is able to handle 10Mbps of traffic.

Example 1 : Usual case

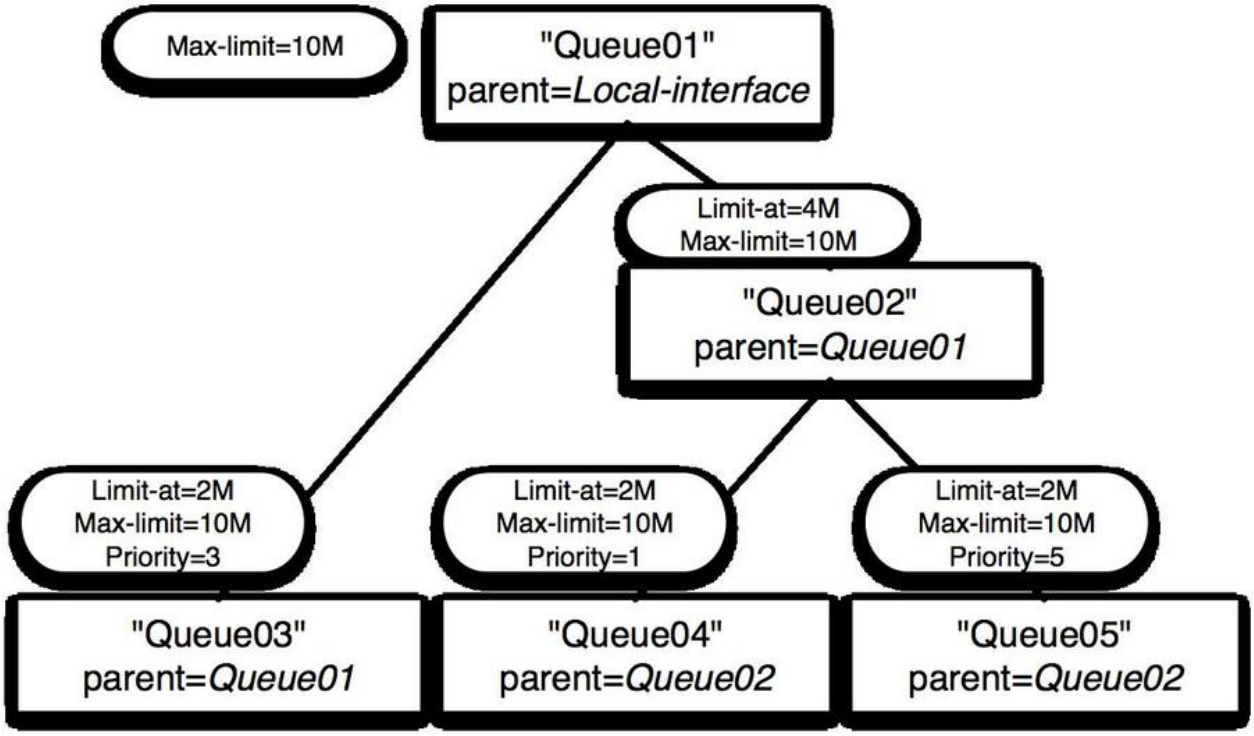


- **Queue01** limit-at=0Mbps max-limit=10Mbps
- **Queue02** limit-at=4Mbps max-limit=10Mbps
- **Queue03** limit-at=6Mbps max-limit=10Mbps priority=1
- **Queue04** limit-at=2Mbps max-limit=10Mbps priority=3
- **Queue05** limit-at=2Mbps max-limit=10Mbps priority=5

Result of Example 1

- **Queue03** will receive 6Mbps
- **Queue04** will receive 2Mbps
- **Queue05** will receive 2Mbps
- **Clarification:** HTB was build in a way, that, by satisfying all **limit-at**s, main queue no longer have throughput to distribute

Example 2 : Usual case with max-limit

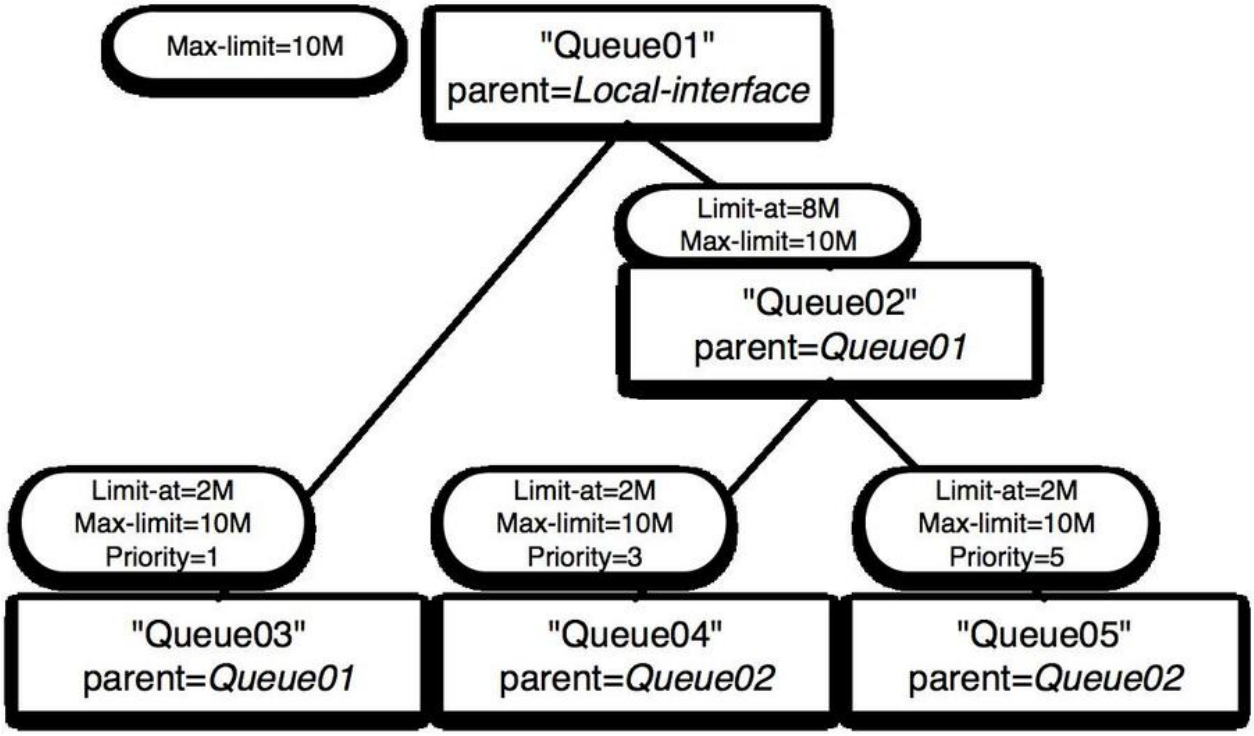


- **Queue01** limit-at=0Mbps max-limit=10Mbps
- **Queue02** limit-at=4Mbps max-limit=10Mbps
- **Queue03** limit-at=2Mbps max-limit=10Mbps priority=3
- **Queue04** limit-at=2Mbps max-limit=10Mbps priority=1
- **Queue05** limit-at=2Mbps max-limit=10Mbps priority=5

Result of Example 2

- **Queue03** will receive 2Mbps
- **Queue04** will receive 6Mbps
- **Queue05** will receive 2Mbps
- **Clarification:** After satisfying all **limit-at**s HTB will give throughput to queue with highest priority.

Example 3 : Inner queue limit-at

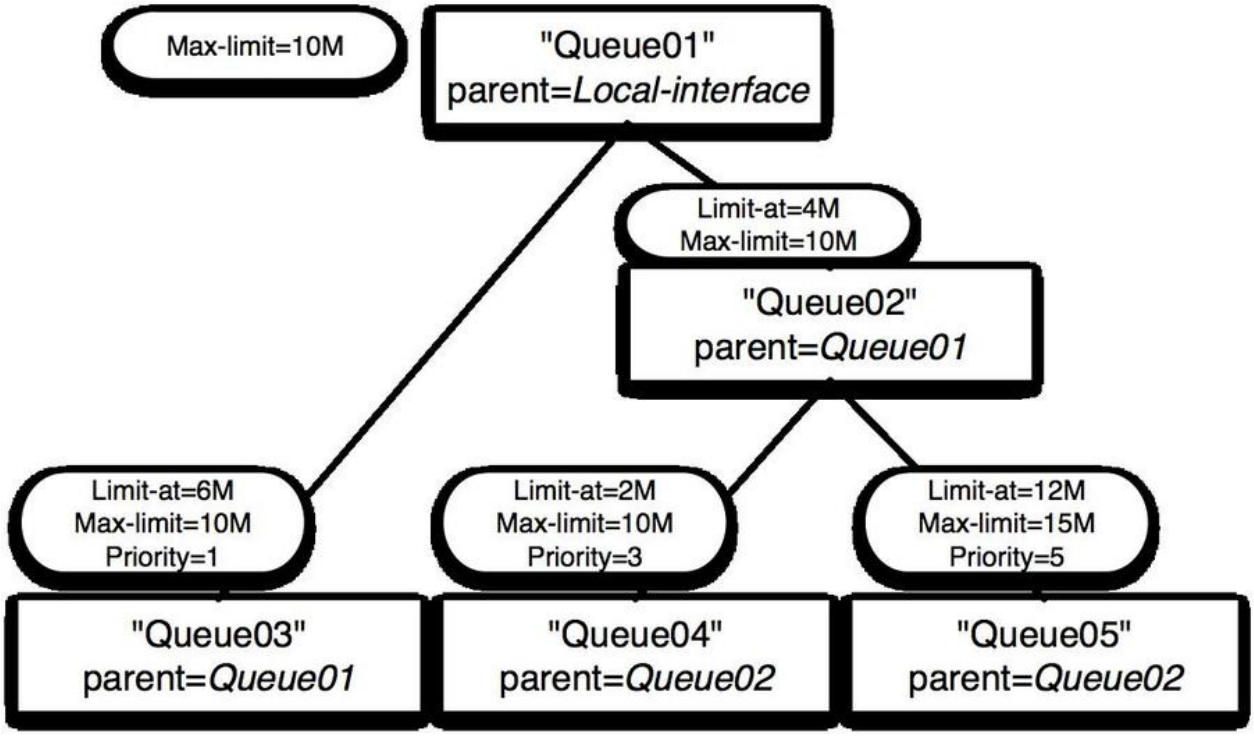


- **Queue01** limit-at=0Mbps max-limit=10Mbps
- **Queue02** limit-at=8Mbps max-limit=10Mbps
- **Queue03** limit-at=2Mbps max-limit=10Mbps priority=1
- **Queue04** limit-at=2Mbps max-limit=10Mbps priority=3
- **Queue05** limit-at=2Mbps max-limit=10Mbps priority=5

Result of Example 3

- **Queue03** will receive 2Mbps
- **Queue04** will receive 6Mbps
- **Queue05** will receive 2Mbps
- **Clarification:** After satisfying all **limit-at**s HTB will give throughput to queue with highest priority. But in this case **inner** queue **Queue02** had **limit-at** specified, by doing so, it reserved 8Mbps of throughput for queues **Queue04** and **Queue05**. From these two **Queue04** have highest priority, that is why it gets additional throughput.

Example 4 : Leaf queue limit-at



- **Queue01** limit-at=0Mbps max-limit=10Mbps
- **Queue02** limit-at=4Mbps max-limit=10Mbps
- **Queue03** limit-at=6Mbps max-limit=10Mbps priority=1
- **Queue04** limit-at=2Mbps max-limit=10Mbps priority=3
- **Queue05** limit-at=12Mbps max-limit=15Mbps priority=5

Result of Example 4

- **Queue03** will receive ~3Mbps
- **Queue04** will receive ~1Mbps
- **Queue05** will receive ~6Mbps
- **Clarification:** Only by satisfying all **limit-at**s HTB was forced to allocate 20Mbps - 6Mbps to **Queue03**, 2Mbps to **Queue04**, 12Mbps to **Queue05**, but our output interface is able to handle 10Mbps. As output interface queue is usually FIFO throughput allocation will keep ratio 6:2:12 or 3:1:6

HTB configuration example

Assume that we want to limit maximum download speed for subnet 10.1.1.0/24 to 2Mbps and distribute this amount of traffic between the server and workstations using HTB (limit upload to 2Mbps). Since HTB works in one direction and is implemented on outbound interface, HTB for download will be on ether2 and HTB for upload will be on ether1.

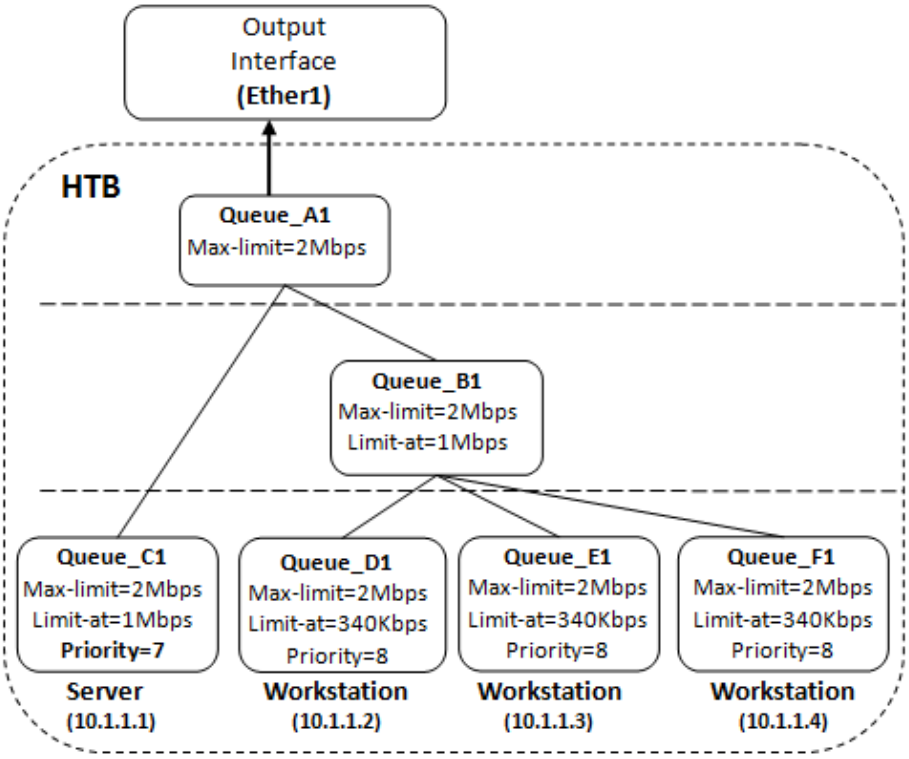


Figure 8.8. HTB for upload limitation

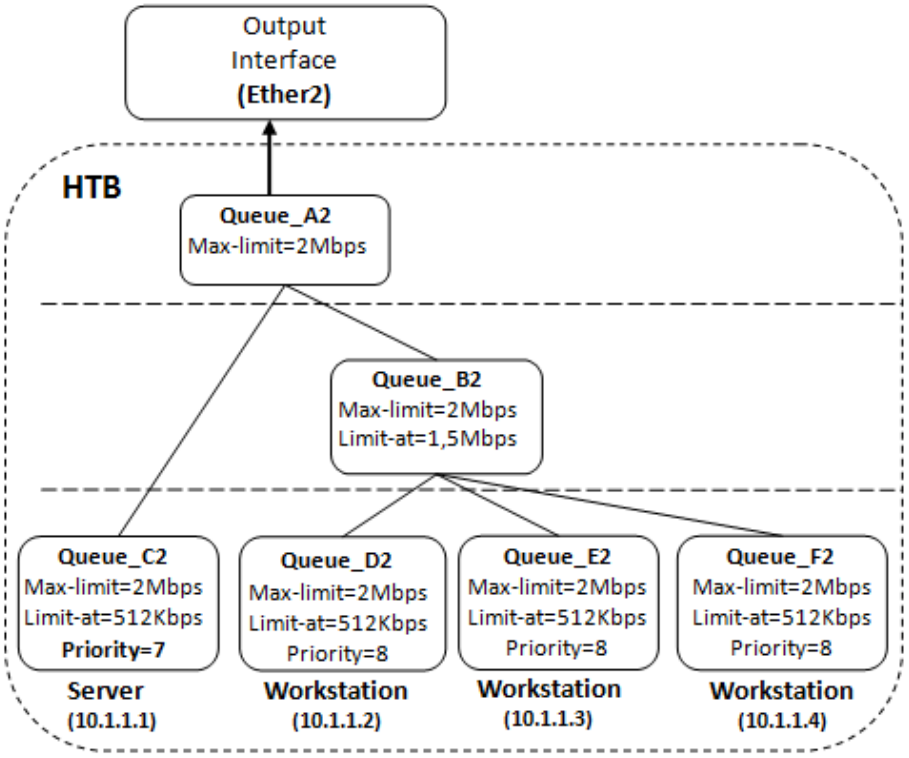


Figure 8.9. HTB for download limitation

The first, we need to classify traffic.

Mark traffic form/to server. The first rule we will mark the outgoing connection from server and with the second one, all packets, which belong to this connection (download and upload packets for this connection):

```
/ip firewall mangle> add chain=prerouting src-address=10.1.1.1/32 action=mark-connection \
new-connection-mark=server_con

/ip firewall mangle> add chain=forward connection-mark=server_con action=mark-packet  \
new-packet-mark=server
```

Do the same for workstation too. Match all workstation connections, mark it with the same mark (new-connection-mark=workstation_con) and after that mark all packets which belong to these workstation.

```
/ip firewall mangle> add chain=prerouting src-address=10.1.1.2
action=mark-connection new-connection-mark=workstation_con
/ip firewall mangle> add chain=prerouting src-address=10.1.1.3
action=mark-connection new-connection-mark=workstation_con
/ip firewall mangle> add chain=prerouting src-address=10.1.1.4
action=mark-connection new-connection-mark=workstation_con

/ip firewall mangle> add chain='''forward''' connection-mark=workstation_con  action=mark-packet \
new-packet-mark=workstations
```

At the end create **/queue tree** for upload and download based on figure 8.8 and figure 8.9.

Queue tree for upload limitation is implemented on ***ether1*** interface.

```
 ;;; Queue_A1 creation
/queue tree> add name=Queue_A1 parent='''ether1''' max-limit=2048k
```

```
 ;;; Queue_B1 creation
/queue tree> add name=Queue_B1 parent=Queue_A1 max-limit=2048k limit-at=1024k

 ;;; Queue_C1 creation
/queue tree> add name=Queue_C1 parent=Queue_A1 max-limit=2048k limit-at=1024k priority=7 \
packet-mark=server

 ;;; Queue_D1, Queue_E1 and Queue_F1 creation
 /queue tree> add name=Queue_D1 parent=Queue_B1 max-limit=2048k limit-at=340k priority=8 \
 packet-mark=workstations
 /queue tree> add name=Queue_E1 parent=Queue_B1 max-limit=2048k limit-at=340k priority=8 \
 packet-mark=workstations
 /queue tree> add name=Queue_F1 parent=Queue_B1 max-limit=2048k limit-at=340k priority=8 \
 packet-mark=workstations
```

```
Priority value by default is 8 so it is not specified here.
```

Queue tree for download limitation is implemented on ***ether2*** interface.

```
 ;;; Queue_A2 creation
/queue tree> add name=Queue_A2 parent='''ether1''' max-limit=2048k

 ;;; Queue_B2 creation
/queue tree> add name=Queue_B2 parent=Queue_A2 max-limit=2048k limit-at=1536k

 ;;; Queue_C creation
 /queue tree> add name=Queue_C2 parent=Queue_A2 max-limit=2048k limit-at=512k priority=7 \
 packet-mark=server

 ;;; Queue_D2, Queue_E2 and Queue_F2 creation
 /queue tree> add name=Queue_D2 parent=Queue_B2 max-limit=2048k limit-at=512k priority=8 \
 packet-mark=workstations
 /queue tree> add name=Queue_E2 parent=Queue_B2 max-limit=2048k limit-at=512k priority=8 \
 packet-mark=workstations
 /queue tree> add name=Queue_F2 parent=Queue_B2 max-limit=2048k limit-at=512k priority=8 \
 packet-mark=workstations
```

[Top | Back to Content **]**

Manual:Interface/HWMPplus

Applies to RouterOS: 3, v4

- *Prerequisites for this article: you understand what WDS is and why to use it*
- *Software versions*: **3.28+** *(earlier versions are incompatible)*

Overview

HWMP+ is a MikroTik specific layer-2 routing protocol for wireless mesh networks. It is based on Hybrid Wireless Mesh Protocol (HWMP) from IEEE 802.11s draft standard. It can be used instead of (Rapid) Spanning Tree protocols in mesh setups to ensure loop-free optimal routing.

The HWMP+ protocol however is not compatible with HWMP from IEEE 802.11s draft standard.

Note that the distribution system you use for your network need not to be Wireless Distribution System (WDS). HWMP+ mesh routing supports not only WDS interfaces, but also Ethernet interfaces inside the mesh. So you can use simple Ethernet based distribution system, or you can combine both WDS and Ethernet links!

Configuration

/interface mesh

Configure mesh interface.

admin-mac (MAC address, default: 00:00:00:00:00:00) -- administratively assigned MAC address, used when **auto-mac** setting is disabled

arp (disabled | enabled | proxy-arp | reply-only; default: enabled) - address resolution protocol setting

auto-mac (boolean, default: no) -- if disabled, then value from **admin-mac** will be used as the MAC address of the mesh interface; else address of some port will be used if ports are present

hwmp-default-hoplimit (integer: 1..255) -- maximum hop count for generated routing protocol packets; after a HWMP+ packet is forwarded "hoplimit" times, it is dropped

hwmp-prep-lifetime (time, default: 5m) -- lifetime for routes created from received PREP or PREQ messages

hwmp-preq-destination-only (boolean, default: yes) -- whether only destination can respond to HWMP+ PREQ message

hwmp-preq-reply-and-forward (boolean, default: yes) -- whether intermediate nodes should forward HWMP+ PREQ message after responding to it. Useful only when **hwmp-preq-destination-only** is disabled

hwmp-preq-retries (integer, default: 2) -- how much times to retry route discovery to a specific MAC address before the address is considered unreachable

hwmp-preq-waiting-time (time, default: 4s) -- how long to wait for a response to the first PREQ message. Note that for subsequent PREQs the waiting time is increased exponentially

hwmp-rann-interval (time, default: 10s) -- how often to send out HWMP+ RANN messages

hwmp-rann-lifetime (time, default: 1s) -- lifetime for routes created from received RANN messages

hwmp-rann-propagation-delay (number, default: 0.5) -- how long to wait before propagating a RANN message. Value in seconds

mesh-portal (boolean, default: no) -- whether this interface is a portal in the mesh network

mtu (number, default: 1500) -- maximum transmit units

name (string) -- interface name

reoptimize-paths (boolean, default: no) -- whether to send out periodic PREQ messages asking for known MAC addresses. Turing on this setting is useful if network topology is changing often. Note that if no reply is received to a reoptimization PREQ, the existing path is kept anyway (until it timeouts itself)

/interface mesh port

Configure mesh interface ports.

hello-interval (time, default: 10s) -- maximum interval between sending out HWMP+ Hello messages. Used only for Ethernet type ports

interface (interface name) -- interface name, which is to be included in a mesh

mesh (interface name) -- mesh interface this port belongs to

path-cost (integer: 0..65535; default: 10) -- path cost to the interface, used by routing protocol to determine the 'best' path

port-type (WDS | auto | ethernet | wireless) -- port type to use

- **auto** - port type is determined automatically based on the underlying interface's type
- **WDS** - a Wireless Distribution System interface, kind of point-to-point wireless link. Remote MAC address is known from wireless connection data
- **ethernet** - Remote MAC addresses are learned either from HWMP+ Hello messages or from source MAC addresses in received or forwarded traffic
- **wireless** - Remote MAC addresses are known from wireless connection data

active-port-type (read-only, wireless | WDS | ethernet-mesh | ethernet-bridge | ethernet-mixed) -- port type and state actually used

/interface mesh fdb

Read-only status of the mesh interface Forwarding Database (FDB).

mac-address (MAC address) -- MAC address corresponding for this FDB entry

seq-number (integer) -- sequence number used in routing protocol to avoid loops

type (local | outsider | direct | mesh | neighbor | larval | unknown) -- type of this FDB entry

- **local** -- MAC address belongs to the local router itself
- **outsider** -- MAC address belongs to a device external to the mesh network
- **direct** -- MAC address belongs to a wireless client on an interface that is in the mesh network
- **mesh** -- MAC address belongs to a device reachable over the mesh network; it can be either internal or external to the mesh network
- **neighbor** -- MAC address belongs to a mesh router that is direct neighbor to this router
- **larval** -- MAC address belongs to an unknown device that is reachable over the mesh network
- **unknown** -- MAC address belongs to an unknown device

mesh (interface name) -- the mesh interface this FDB entry belongs to

on-interface (interface name) -- mesh port used for traffic forwarding, kind of a next-hop value

lifetime (time) -- time remaining to live if this entry is not used for traffic forwarding

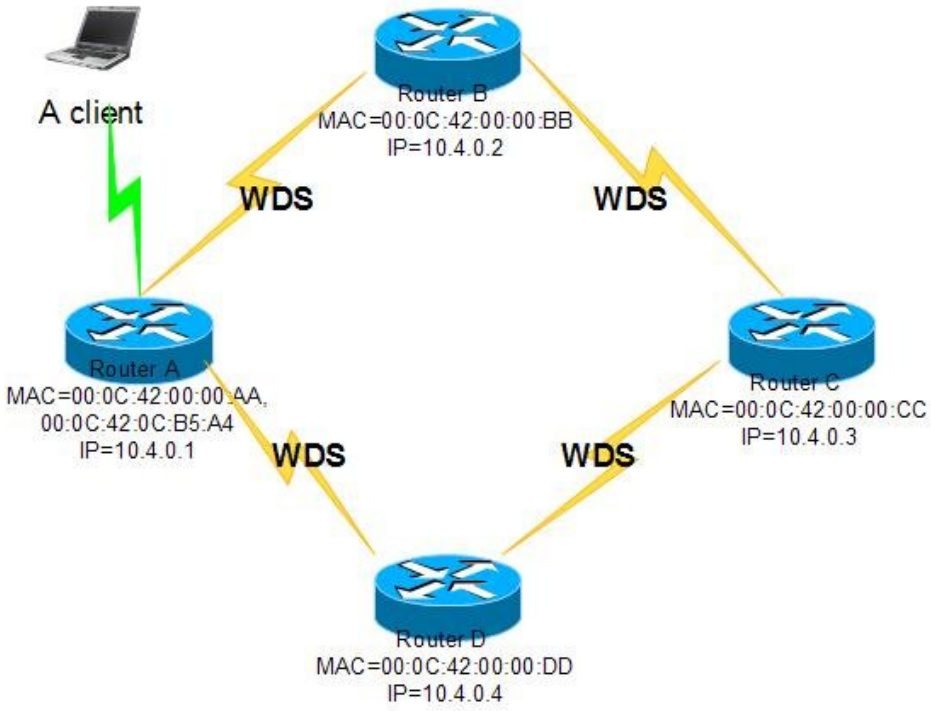
age (time) -- age of this FDB entry

metric (integer) -- metric value used by routing protocol to determine the 'best' path

Additional wireless configuration

Use **wds-default-cost** and **wds-cost-range** wireless interface parameters for controlling the metric that is used in the routing protocol. The WDS cost will be used as **path-cost** for ports dynamically added to the mesh interface.

Example



This example uses static WDS links that are dynamically added as mesh ports when they become active. Two different frequencies are used: one for AP interconnections, and one for client connections to APs, so the AP must have at least two wireless interfaces. Of course, the same frequency for all connections also could be used, but that might not work as good because of potential interference issues.

Repeat this configuration on all APs:

```
/interface mesh add disabled=no

/interface mesh port add interface=wlan1 mesh=mesh1

/interface mesh port add interface=wlan2 mesh=mesh1

# interface used for AP interconnections
/interface wireless set wlan1 disabled=no ssid=mesh frequency=2437 band=2.4ghz-b/g mode=ap-bridge \
  wds-mode=static-mesh wds-default-bridge=mesh1

# interface used for client connections
/interface wireless set wlan2 disabled=no ssid=mesh-clients frequency=5180 band=5ghz mode=ap-bridge

# a static WDS interface for each AP you want to connect to
```

```
/interface wireless wds add disabled=no master-interface=wlan1 name=<descriptive name of remote end> \
  wds-address=<MAC address of remote end>
```

Here WDS interface is added manually, because static WDS mode is used. If you are using **wds-mode**=dynamic-mesh, all WDS interfaces will be created automatically. The **frequency** and **band** parameters are specified here only to produce valid example configuration; mesh protocol operations is by no means limited to, or optimized for, these particular values.

Note: You may want to increase **disconnect-timeout** wireless interface option to make the protocol more stable.

In real world setups you also should take care of securing the wireless connections, using **/interface wireless security-profile**. For simplicity that configuration it's not shown here.

Results on router A (there is one client is connected to wlan2):

```
[admin@A] > /interface mesh pr
Flags: X - disabled, R - running
 0  R name="mesh1" mtu=1500 arp=enabled mac-address=00:0C:42:0C:B5:A4 auto-mac=yes
      admin-mac=00:00:00:00:00:00 mesh-portal=no hwmp-default-hoplimit=32
      hwmp-preq-waiting-time=4s hwmp-preq-retries=2 hwmp-preq-destination-only=yes
      hwmp-preq-reply-and-forward=yes hwmp-prep-lifetime=5m hwmp-rann-interval=10s
      hwmp-rann-propagation-delay=1s hwmp-rann-lifetime=22s

[admin@A] > interface mesh port p detail
Flags: X - disabled, I - inactive, D - dynamic
 0    interface=wlan1 mesh=mesh1 path-cost=10 hello-interval=10s port-type=auto port-type-used=wireless
 1    interface=wlan2 mesh=mesh1 path-cost=10 hello-interval=10s port-type=auto port-type-used=wireless
 2  D interface=router_B mesh=mesh1 path-cost=105 hello-interval=10s port-type=auto port-type-used=WDS
 3  D interface=router_D mesh=mesh1 path-cost=76 hello-interval=10s port-type=auto port-type-used=WDS
```

The FDB (Forwarding Database) at the moment contains information only about local MAC addresses, non-mesh nodes reachable through local interface, and direct mesh neighbors:

```
[admin@A] /interface mesh> fdb print
Flags: A - active, R - root
   MESH         TYPE     MAC-ADDRESS       ON-INTERFACE       LIFETIME     AGE
A  mesh1        local    00:0C:42:00:00:AA                                  3m17s
A  mesh1        neighbor 00:0C:42:00:00:BB router_B                         1m2s
A  mesh1        neighbor 00:0C:42:00:00:DD router_D                         3m16s
A  mesh1        direct   00:0C:42:0C:7A:2B wlan2                            2m56s
A  mesh1        local    00:0C:42:0C:B5:A4                                  2m56s

[admin@A] /interface mesh> fdb print detail
Flags: A - active, R - root
 A  mac-address=00:0C:42:00:00:AA type=local age=3m21s mesh=mesh1 metric=0
     seqnum=4294967196
 A  mac-address=00:0C:42:00:00:BB type=neighbor on-interface=router_B age=1m6s
    mesh=mesh1 metric=132 seqnum=4294967196
 A  mac-address=00:0C:42:00:00:DD type=neighbor on-interface=router_D age=3m20s
     mesh=mesh1 metric=79 seqnum=4294967196
```

```
 A  mac-address=00:0C:42:0C:7A:2B type=direct on-interface=wlan2 age=3m mesh=mesh1
     metric=10 seqnum=0
 A  mac-address=00:0C:42:0C:B5:A4 type=local age=3m mesh=mesh1 metric=0 seqnum=0
```

Test that ping works:

```
[admin@A] > /ping 00:0C:42:00:00:CC
00:0C:42:00:00:CC 64 byte ping time=108 ms
00:0C:42:00:00:CC 64 byte ping time=51 ms
00:0C:42:00:00:CC 64 byte ping time=39 ms
00:0C:42:00:00:CC 64 byte ping time=43 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 39/60.2/108 ms
```

Router A had to discover path to Router C first, hence the slightly larger time for the first ping. Now the FDB also contains an entry for 00:0C:42:00:00:CC, with type "mesh".

Also test that ARP resolving works and so does IP level ping:

```
[admin@A] > /ping 10.4.0.3
10.4.0.3 64 byte ping: ttl=64 time=163 ms
10.4.0.3 64 byte ping: ttl=64 time=46 ms
10.4.0.3 64 byte ping: ttl=64 time=48 ms
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 46/85.6/163 ms
```

Mesh traceroute

There is also mesh traceroute command, that can help you to determine which paths are used for routing.

For example, for this network:

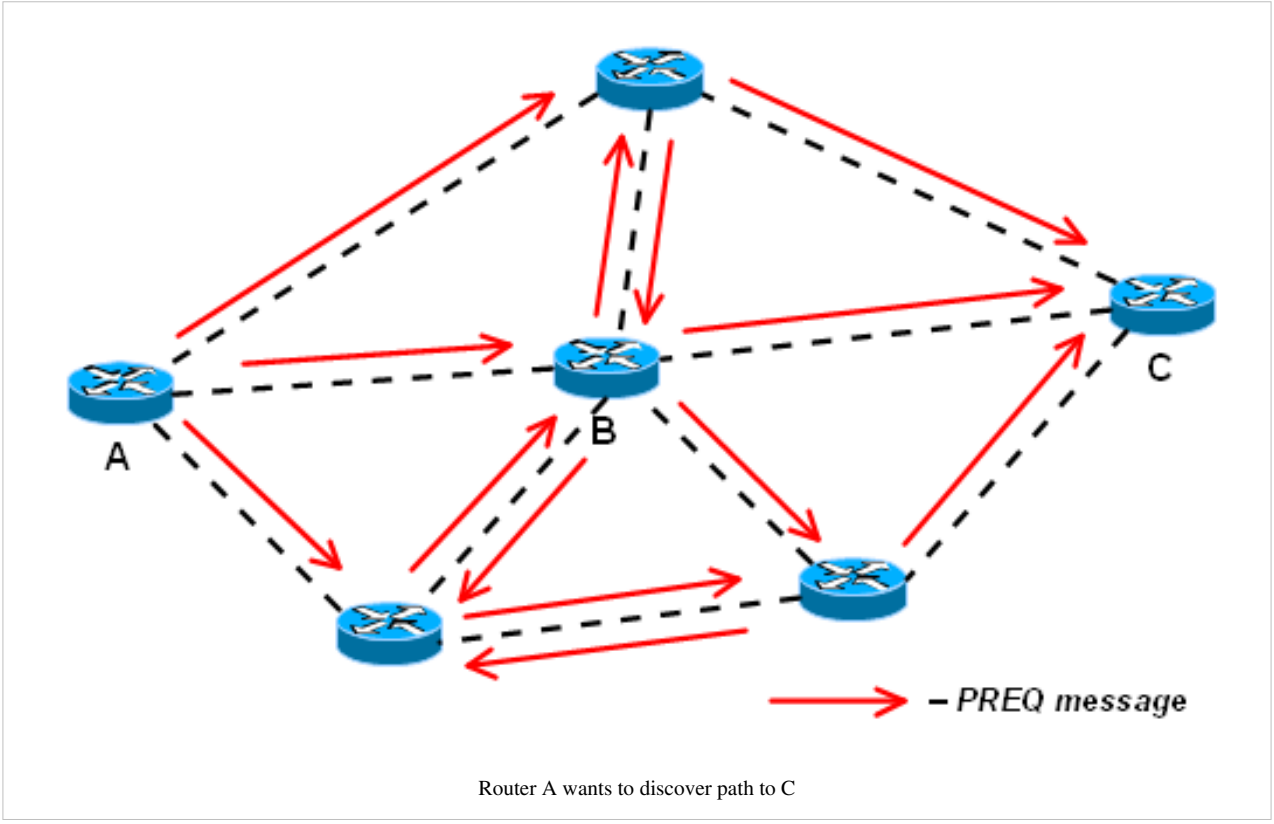
```
[admin@1] /interface mesh> fdb print
Flags: A - active, R - root
   MESH        TYPE     MAC-ADDRESS       ON-INTERFACE       LIFETIME     AGE
A  mesh1       local    00:0C:42:00:00:01                             7m1s
A  mesh1       mesh     00:0C:42:00:00:02 wds4               17s          4s
A  mesh1       mesh     00:0C:42:00:00:12 wds4               4m58s        1s
A  mesh1       mesh     00:0C:42:00:00:13 wds4               19s          2s
A  mesh1       neighbor 00:0C:42:00:00:16 wds4                           7m1s
A  mesh1       mesh     00:0C:42:00:00:24 wds4               18s          3s
```

Traceroute to 00:0C:42:00:00:12 shows:

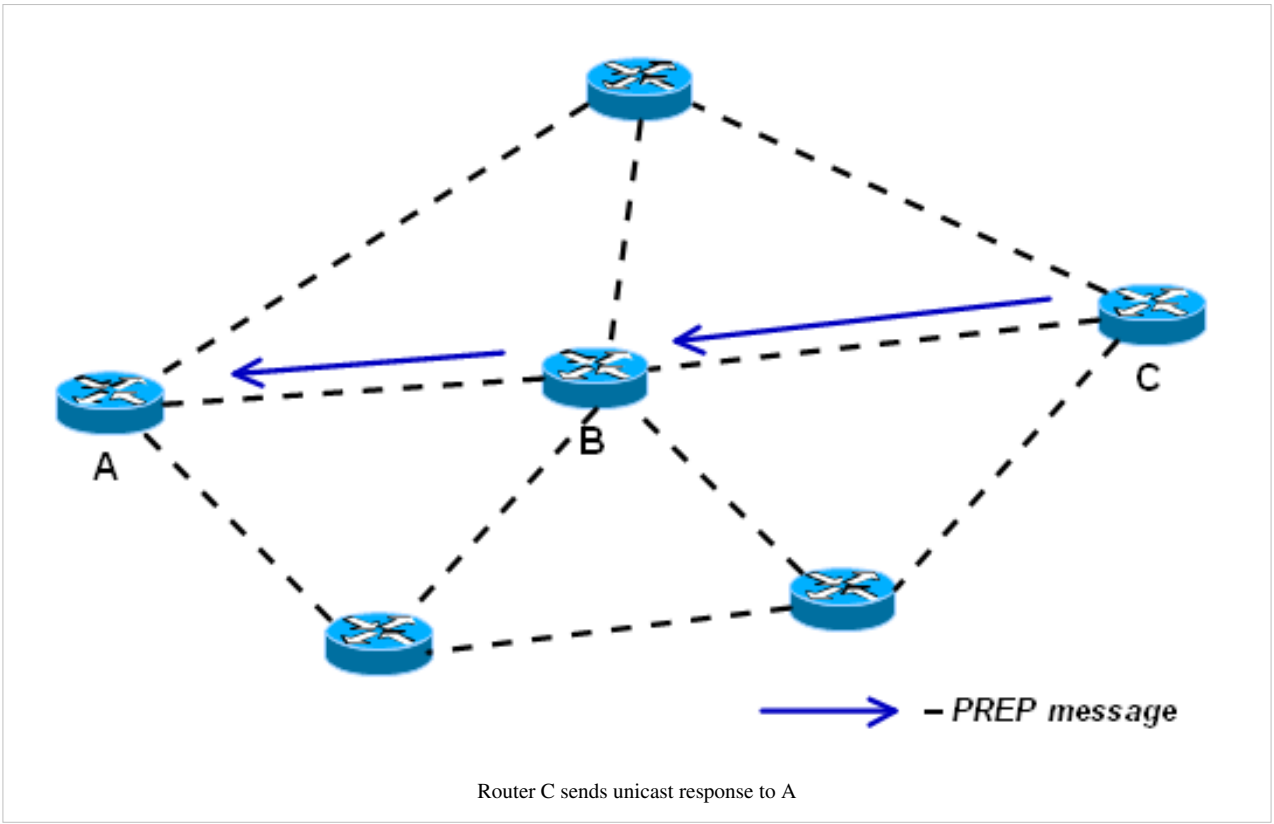
```
[admin@1] /interface mesh> traceroute mesh1 00:0C:42:00:00:12
ADDRESS           TIME         STATUS
00:0C:42:00:00:16 1ms          ttl-exceeded
00:0C:42:00:00:02 2ms          ttl-exceeded
00:0C:42:00:00:24 4ms          ttl-exceeded
00:0C:42:00:00:13 6ms          ttl-exceeded
00:0C:42:00:00:12 6ms          success
```

Protocol description

Reactive mode



Router A wants to discover path to C

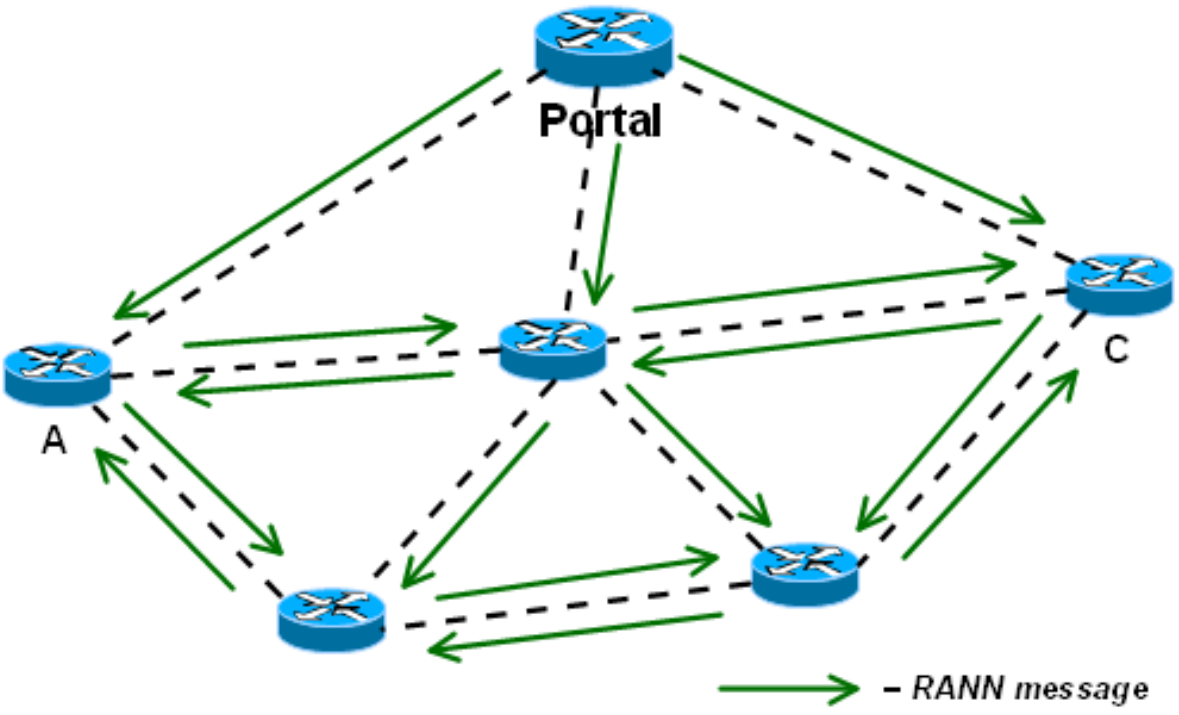


Router C sends unicast response to A

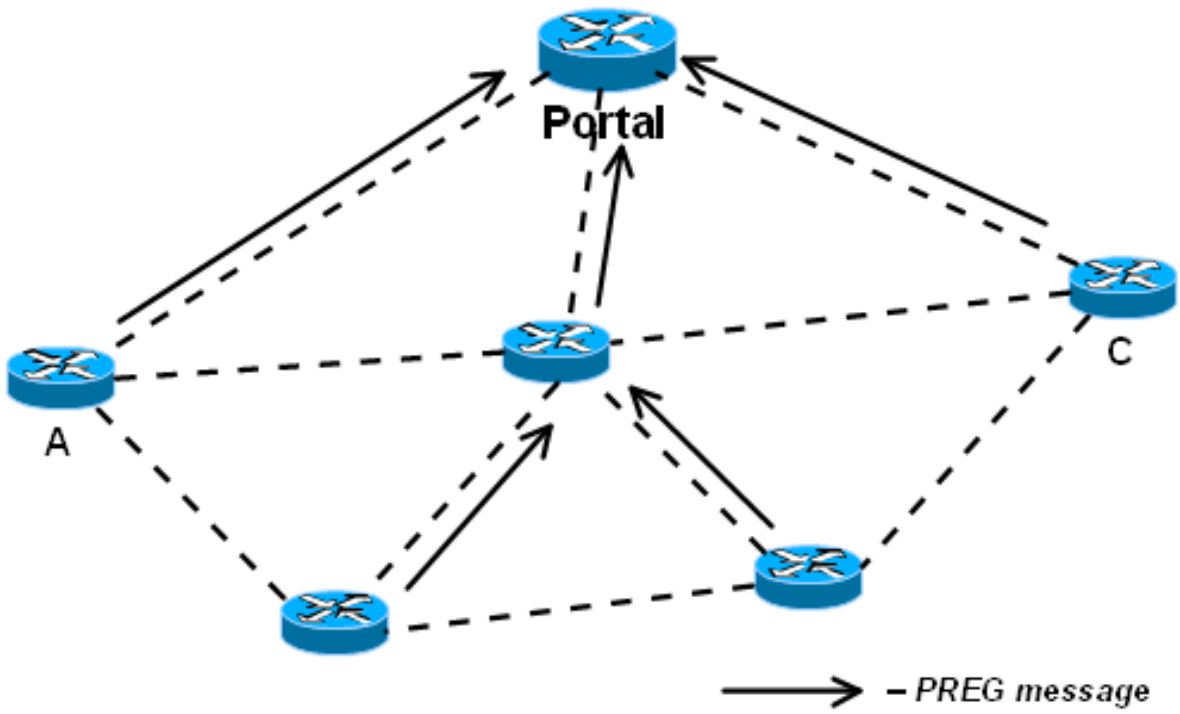
In reactive mode HWMP+ is very much like AODV (Ad-hoc On-demand Distance Vector). All path are discovered on demand, by flooding Path Request (PREQ) message in the network. The destination node or some router that has a path to the destionation will reply with a Path Response (PREP). Note that if the destination address belongs to a client, the AP this client is connected to will serve as proxy for him (i.e. reply to PREQs on his behalf).

This mode is best suited for mobile networks, and/or when most of the communication happens between intra-mesh nodes.

Proactive mode



The root announces itself by flooding RANN



Internal nodes respond with PREGs

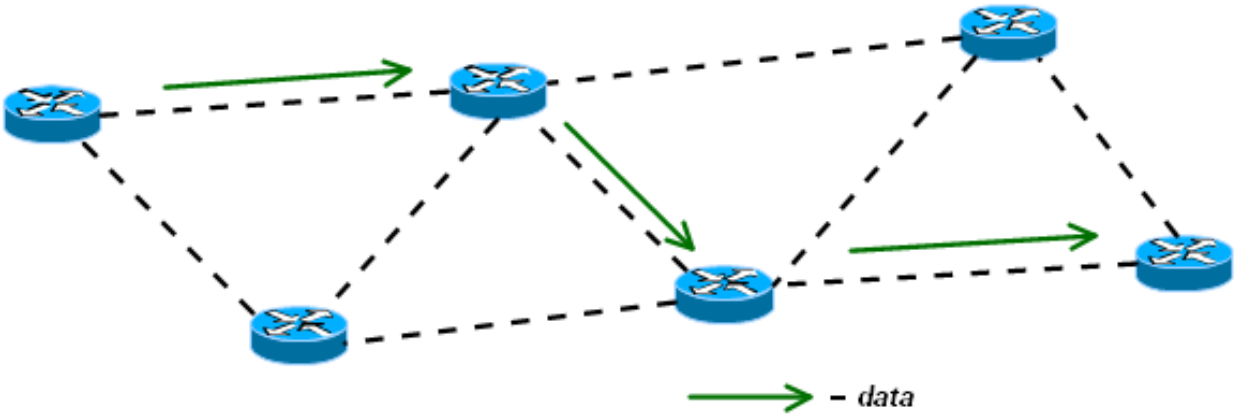
In proactive mode there are some routers configured as portals. In general being a portal means that router has interfaces to some other network,, i.e. it is entry/exit point to the mesh network.

The portals will announce their presence by flooding Root Announcement (RANN) message in the network. Internal nodes will reply with a Path Registration (PREG) message. The result of this process will be routing trees with roots in the portal.

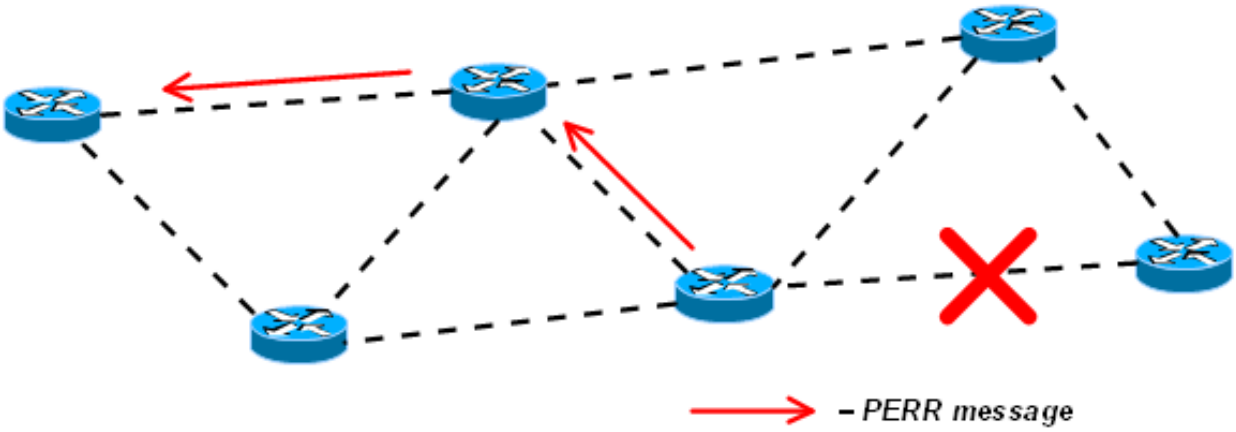
Routes to portals will serve as a kind of default routes. If an internal router does not know path to a particular destination, it will forward all data to its closest portal. The portal will then discover path on behalf of the router, if needed. The data afterwards will flow through the portal. This may lead to suboptimal routing, unless the data is addressed to the portal itself or some external network the portals has interfaces to.

Proactive mode is best suited when most of traffic goes between internal mesh nodes and a few portal nodes.

Topology change detection



Data flow path



After link disappears, error is propagated upstream

HWMP+ uses Path Error (PERR) message to notify that a link has disappeared. The message is propagated to all upstream nodes up to the data source. The source on PERR reception restarts path discovery process.

FAQ

Q. How is this better than RSTP?

A. It gives you optimal routing. RSTP is only for loop prevention.

Q. How the route selection is done?

A. The route with best metric is always selected after the discovery process. There is also a configuration option to periodically reoptimize already known routes.

Route metric is calculated as sum of individual link metrics.

Link metric is calculated in the same way as for (R)STP protocols:

- For Ethernet links the metric is configured statically (like for OSPF, for example).
- For WDS links the metric is updated dynamically depending on actual link bandwidth, which in turn is influenced by wireless signal strength, and the selected data transfer rate.

Currently the protocol does not take in account the amount of bandwidth being used on a link, but that might be also used in future.

Q. How is this better than OSPF/RIP/layer-3 routing in general?

A. WDS networks usually are bridged, not routed. The ability to self-configure is important for mesh networks; and routing generally requires much more configuration than bridging. Of you course, you always can run any L3 routing protocol over a bridged network, but for mesh networks that usually makes little sense.

Note: Since optimized layer-2 multicast forwarding is not included mesh protocol, it is better to avoid forwarding any multicast traffic (including OSPF) over meshed networks. If you need OSPF, then you have to configure OSPF NBMA neighbors that uses unicast instead.

Q. What about performance/CPU requirements?

A. The protocol itself, when properly configured, will take much less resources than OSPF (for example) would. Data forwarding performance on an individual router should be close to that of bridging.

Q. How does it work together with existing mesh setups that are using RSTP?

A. The internal structure of a RSTP network is transparent to the mesh protocol (because mesh hello packets are forwarded inside RSTP network). The mesh will see the path between two entry points in the RSTP network as a single segment. On the other hand, a mesh network is not transparent to the RSTP, since RSTP hello packets are not be forwarded inside the mesh network. *(This is the behaviour since 3.26)*

Warning: Routing loops are possible, if a mesh network is attached to a RSTP network in two or more points!

Note that if you have a WDS link between two access points, then both ends must have the same configuration (either as ports in a mesh on both ends, or as ports in a bridge interface on both ends).

You can also put a bridge interface as a mesh port (to be able to use bridge firewall, for example).

Q. Can I have multiple entry/exit points to the network?

A. If the entry/exit points are configured as portals (i.e. proactive mode is used), each router inside the mesh network will select its closest portal and forward all data to it. The portal will then discover path on behalf of the router, if needed.

Q. How to control or filter mesh traffic?

A. At the moment the only way is to use bridge firewall. Create a bridge interface, put the WDS interfaces and/or Ethernets in that bridge, and put that bridge in a mesh interface. Then configure bridge firewall rules.

To match MAC protocol used for mesh traffic encapsulation, use MAC protocol number 0x9AAA, and to mathc mesh routing tafffic, use MAC protocol number 0x9AAB. Example:

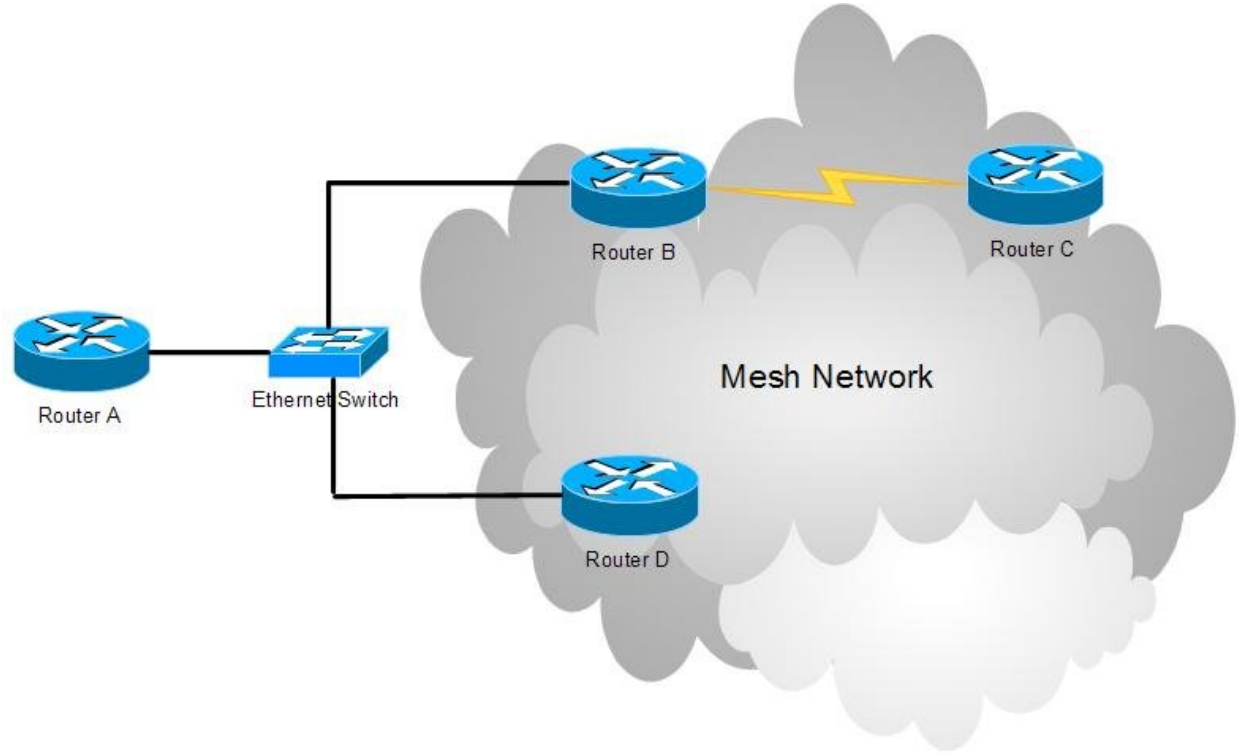
```
interface bridge settings set use-ip-firewall=yes
interface bridge filter add chain=input action=log mac-protocol=0x9aaa
interface bridge filter add chain=input action=log mac-protocol=0x9aab
```

Note that it is perfectly possible to create mixed mesh/bridge setups that will not work (e.g. *Problematic example 1* with bridge instead of switch). The recommended fail-safe way that will always work is to create a separate bridge interface per each physical interfaces; then add all these bridge interfaces as mesh ports.

Advanced topics

We all know that it's easy to make problematic layer-2 bridging or routing setups and hard to debug them. (Compared to layer-3 routing setups.) So there are a few bad configuration examples which could create problems for you. Avoid them!

Problematic example 1: Ethernet switch inside a mesh



Router A is outside the mesh, all the rest of routers: inside. For routers B, C, D all interfaces are added as mesh ports.

Router A will not be able to communicate reliably with router C. The problem manifests itself when D is the designated router for Ethernet; if B takes this role, everything is OK. The main cause of the problem is MAC address learning on Ethernet switch.

Consider what happens when router A wants to send something to C. We suppose router A either knowns or floods data to all interfaces. Either way, data arrives at switch. The switch, not knowing anything about destination's MAC address, forwards to data both to B and D.

What happens now:

1. B receives the packet on a mesh interface. Since the MAC address is not local for B, and B knows that he is not the designated router for the Ethernet network, he simply ignores the packet.
2. D receives the packet on a mesh interface. Since the MAC address is not local for B, and D is the designated router for the Ethernet network, he initiates path discovery process to C.

After path discovery is completed, D has information that C is reachable over B. Now D encapsulates the packet and forwards back to Ethernet network. The encapsulated packet forwarded by switch, received and forwarded by B, and received by C. So far everything is good.

Now C is likely to respond to the packet. Since B already knows where A is, he will decapsulate and forward the reply packet. But now switch will learn that the MAC address of C is reachable through B! That means, next time when something arrives from A addressed to C, the switch will forward data *only* to B. (And B, of course, will silently ignore the packet).

In contrast, if B took up the role of designated router, everything would be OK, because traffic would not have to go through the Ethernet switch twice.

Troubleshooting: either avoid such setup or disable MAC address learning on the switch. Note that on many switches it's not possible.

Also note that there will be no problem, if either:

- router A supports and is configured to use HWMP+;

- or Ethernet switch is replaced with and router that supports HWMP+ and has Ethernet interfaces added as mesh ports.

Problematic example 2: wireless modes

Consider this setup (invalid):

Routers A and B are inside the mesh, routers C: outside. For routers A and B all interfaces are added as mesh ports.

It is not possible to bridge wlan1 and wlan2 on router B now. The reason for this is pretty obvious if you understand how WDS works. For WDS communications four address frames are used. This is because for wireless multihop forwarding you need to know both the addresses of intermediate hops, as well as the original sender and final receiver. In contrast, non-WDS 802.11 communication includes only three MAC addresses in a frame. That's why it's not possible to do multihop forwarding in station mode.

Troubleshooting: depends on what you want to achieve:

1. If you want to router C as a repeater either for wireless or Ethernet traffic, configure WDS link between router B and router C, and run mesh routing protocol on all nodes.
2. In other cases configure wlan2 on router B in AP mode, and wlan on router C in station mode.

See also:

- A presentation about mesh networks and MikroTik *(in Portuguese)* [1]

References

[1] http://mum.mikrotik.com/presentations/BR08/Brasil_Mesh_Maia.pdf

Manual:Creating IPv6 loopback address

In some cases it is necessary to have a kind of loopback interface. It can be used to hold addresses that belong to the "router itself" and not to any particular outgoing interface. Such addresses are useful, for example, as source addresses for TCP connections between two routers that have more that one physical interfaces between them.

In MT RouterOS the recommended way to add a loopback interface for IPv4 is to create a new empty bridge interface:

```
/interface bridge add name=lobridge
# loopback address
/ip address add address=10.0.0.1/24 interface=lobridge
```

However, for IPv6 this won't work.

Empty bridge interface has zero MAC byte default. MT RouterOS does not generate IPv6 link-local addresses on interfaces with zero MAC address (because of high address collision probability).

Since IPv6 link-local address is needed for IPv6 to function properly on an interface, this means that by default the empty bridge interface cannot be used as IPv6 loopback interface.

Recommended solution

Add an empty bridge, and specify bridge MAC address manually:

```
/interface bridge add name=lobridge auto-mac=no admin-mac=01:00:00:00:01:00
# loopback address
/ipv6 address add address=2003::1/64 advertise=no interface=lobridge
```

Alternative solution is to use a fake EoIP tunnel interface instead of bridge. A random MAC address will be generated in this case.

Results

Test that you are able to ping the loopback address:

```
/ping 2003::1
2003::1 64 byte ping: ttl=64 time=5 ms
2003::1 64 byte ping: ttl=64 time=5 ms
```

Manual:Interface

Applies to RouterOS: v3, v4 +

Sub Categories

List of reference sub-pages | **Case studies** | **List of examples**

<splist showparent=yes />

Summary

Sub-menu: `/interface`

MikroTik RouterOS supports a variety of Network Interface Cards as well as virtual interfaces (like Bonding, Bridge, VLAN etc.). Each of them has its own submenu, but common properties of all interfaces can be configured and read in general interface menu.

Properties

Property	Description
l2mtu (*integer*; Default:)	Layer2 Maximum transmission unit. Note that this property can not be configured on all interfaces. Read more>>
mtu (*integer*; Default:)	Layer3 Maximum transmission unit
name (*string*; Default:)	Name of an interface

Read-only properties

Property	Description
bytes (*integer/integer*)	Total received and transmitted bytes by interface since startup. Read more>>
drops (*integer/integer*)	packets not sent/received because interface queue is full (no free descriptors), dma engine overrun/underrun. Read more>>
dynamic (*yes|no*)	Whether interface is dynamically created
errors (*integer/integer*)	Packets received with some kind of error or not transimitted because of some error. Read more>>
packets (*integer/integer*)	Total count of packets on interface since startup. Read more>>
running (*yes|no*)	Whether interface is running. Note that some interface does not have running check and they are always reported as "running"
slave (*yes|no*)	Whether interface is configured as a slave of another interface (for example Bonding)
dynamic (*yes|no*)	Whether interface is dynamically created
type (*string*)	Type of an interface (ethernet, wireless, etc.)

Traffic monitor

The traffic passing through any interface can be monitored using following command:
`/interface monitor-traffic [id | name]`

For example monitor ether2 and aggregate traffic. `Aggregate` is used to monitor total ammount of traffic handled by the router:

```
[maris@maris_main] > /interface monitor-traffic ether2,aggregate
    rx-packets-per-second: 9        14
      rx-drops-per-second: 0        0
     rx-errors-per-second: 0        0
       rx-bits-per-second: 6.6kbps  10.2kbps
    tx-packets-per-second: 9        12
      tx-drops-per-second: 0        0
     tx-errors-per-second: 0        0
       tx-bits-per-second: 13.6kbps 15.8kbps
```

Stats

RouterOS v3.22 introduces a new command:

```
 /interface print stats
```

This command prints total packets, bytes, drops and errors.

All interfaces that support this feature will be displayed. Some interfaces are not supporting Error and Drop counters at the moment (RB4XX except RB450G ether 2-5), these devices will not display these counters.

Traffic monitor now also displays errors per second, in addition to the usual stats:

```
 /interface monitor-traffic
```

`/interface ethernet print stats` will display all kinds of other statistics if the interface is supporting them (currently only RB450G ether2-ether5 and also RB750 ether2-ether5).

[Top | Back to Content **]**

Manual:Interface/IPIP

Applies to RouterOS: 2.9, v3, v4+

Summary

Sub-menu: `/interface ipip`
Standards: `IPIP RFC 2003`

The IPIP tunneling implementation on the MikroTik RouterOS is RFC 2003 compliant. IPIP tunnel is a simple protocol that encapsulates IP packets in IP to make a tunnel between two routers. The IPIP tunnel interface appears as an interface under the interface list. Many routers, including Cisco and Linux based, support this protocol. This protocol makes multiple network schemes possible.
IP tunneling protocol adds the following possibilities to a network setups:

- to tunnel Intranets over the Internet
- to use it instead of source routing

Properties

Property	Description
local-address (*IP*; Default:)	IP address on a router that will be used by IPIP tunnel
mtu (*integer*; Default: **1500**)	Layer3 Maximum transmission unit
name (*string*; Default:)	Interface name
remote-address (*IP*; Default:)	IP address of remote end of IPIP tunnel

Note: There is no authentication or 'state' for this interface. The bandwidth usage of the interface may be monitored with the monitor feature from the interface menu.

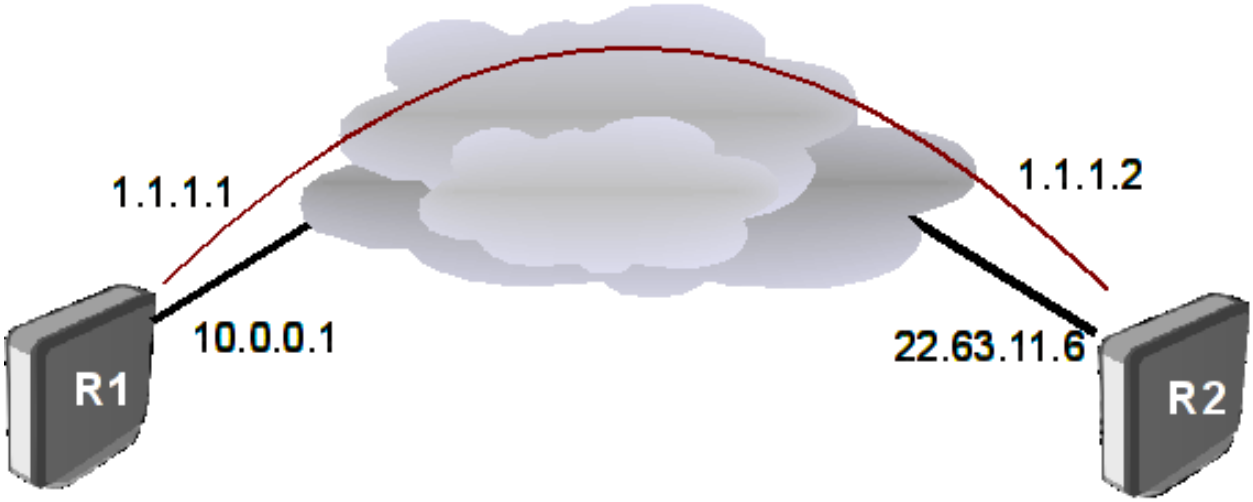
IPv6

Sub-menu: `/interface ipipv6`

IP/IPv6 over IPv6 tunnel functionality is added in v5RC6 and is configurable from another menu: `/interface ipipv6` IPv6 version uses the same properties as IPv4 version.

Setup examples

Suppose we want to add an IPIP tunnel between routers R1 and R2:



At first, we need to configure IPIP interfaces and then add IP addresses to them.
The configuration for router **R1** is as follows:

```
[admin@MikroTik] interface ipip> add
local-address: 10.0.0.1
remote-address: 22.63.11.6
[admin@MikroTik] interface ipip> print
Flags: X - disabled, R - running
  #    NAME                                MTU   LOCAL-ADDRESS   REMOTE-ADDRESS
  0 X  ipip1                               1480  10.0.0.1        22.63.11.6

[admin@MikroTik] interface ipip> en 0
[admin@MikroTik] interface ipip> /ip address add address=1.1.1.1/24 interface=ipip1
```

The configuration of the **R2** is shown below:

```
[admin@MikroTik] interface ipip> add local-address=22.63.11.6 remote-address=10.
0.0.1
[admin@MikroTik] interface ipip> print
Flags: X - disabled, R - running
  #    NAME                                MTU   LOCAL-ADDRESS   REMOTE-ADDRESS
  0 X  ipip1                               1480  22.63.11.6      10.0.0.1

[admin@MikroTik] interface ipip> enable 0
[admin@MikroTik] interface ipip> /ip address add address=1.1.1.2/24 interface=ipip1
```

Now both routers can ping each other:

```
[admin@MikroTik] interface ipip> /ping 1.1.1.2
1.1.1.2 64 byte ping: ttl=64 time=24 ms
1.1.1.2 64 byte ping: ttl=64 time=19 ms
1.1.1.2 64 byte ping: ttl=64 time=20 ms
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 19/21.0/24 ms
[admin@MikroTik] interface ipip>
```

[Top | Back to Content]

Manual:IP

List of reference sub-pages **Case studies** **List of examples**

<splist showparent=yes />

Manual:IPv6

List of reference sub-pages **Case studies** **List of examples**

<splist showparent=yes />

Manual:IPv6 Overview

Applies to RouterOS: v3beta10+, v4, v5+

IPv6 overview

Package requirement: `ipv6`

Internet Protocol version 6 (IPv6) is the new version of the Internet Protocol (IP). It was initially expected to replace IPv4 in short enough time, but for now it seems that these two version will coexist in Internet in foreseeable future. Nevertheless, IPv6 becomes more important, as the date of unallocated IPv4 address pool's exhaustion approaches.

The two main benefits of IPv6 over IPv4 are:

- much larger address space;
- support of stateless and statefull address autoconfiguration;
- built-in security;
- new header format (faster forwarding).

Supported programms

MikroTik IPv6 support at the moment:

- DHCPv6 prefix delegation for DHCP server.
- DHCPv6-PD client.
- IPv6 Prefix Delegation over PPP interfaces.
- static addressing and routing;
- router advertisement daemon (for address autoconfiguration);
- dynamic routing: BGP+, OSPFv3, and RIPng protocols;
- firewall (filter, mangle, address lists, connection table);
- queue tree, simple queue, pcq;
- DNS name servers;
- 6in4 (SIT) tunnels;
- EoIPv6, ip/ipv6 over ipv6 (IPIPv6) tunnel interface (starting from v5RC6)
- IPSEC;
- VRRPv3;
- all PPP (Point-to-point protocols);
- SSH, telnet, FTP, WWW access, Winbox, API;
- ping;
- traceroute;
- web proxy;
- sniffer and fetch tools;
- IP services and User allowed IPv6 address support;
- torch, bandwidth test and other tools;

Features not yet supported:

- automatic tunnel creation;
- policy routing;
- multicast routing;
- MPLS;

Addressing

IPv6 uses 16 bytes addresses compared to 4 byte addresses in IPv4. IPv6 address syntax and types are described in RFC 4291.

Read more>>

Stateless Autoconfiguration

Read more >>

Routing

For static routing, the basic principles of IPv6 are exactly the same as for IPv4. `Read more >>`

Note: Link local addresses are required for dynamic routing protocols to function!

Warning: All dynamic routing protocols also require a valid Router ID to function. If the Router ID is not configured manually, one of router's IPv4 addresses are used as the Router ID. If no IPv4 addresses are present, the router ID selection process will fail. This means that dynamic routing will not work on a router that has no IPv4 addresses, unless you configure the Router ID manually!

BGP

Because of it's design BGP naturally supports multiple address families, and migration to IPv6 is straightforward here.

Example: configure iBGP between routers A and B, AS 65000, that will exchange IPv4 and IPv6 routes.

Router A:

```
[admin@A] > routing bgp peer add remote-address=10.0.0.134 remote-as=65000 address-families=ip,ipv6
```

Router B:

```
[admin@B] > routing bgp peer add remote-address=10.0.0.133 remote-as=65000 address-families=ip,ipv6
```

Redistribute a route from router A to router B:

```
[admin@A] > ipv6 route add dst-address=2001::/16 gateway=fe80::1%ether1
[admin@A] > routing bgp network add network=2001::/16
[admin@A] > routing bgp advertisements print
PEER     PREFIX               NEXTHOP          AS-PATH  ORIGIN     LOCAL-PREF
peer1    2001::/16            fe80::1200:ff...          igp        100
```

```
[admin@B] > ipv6 route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, o - ospf, b - bgp, U - unreachable
 #      DST-ADDRESS              GATEWAY                   DISTANCE
 0 ADb  2001::/16                fe80::1200:ff:fe00:10...  200
```

IPv6 addresses can also be used in peer configuration in **remote-address** and **update-source** fields - to make a BGP connection over IPv6.

OSPF

Unlike to BGP, adding IPv6 support to OSPF required a lot of changes and resulted in a new, incompatible, version of OSPF: protocol version 3. (For IPv4, OSPF version 2 is used). The new version is described in RFC 2740.

OSPFv3 uses the same fundamental mechanisms as OSPFv2 — LSAs, flooding, the SPF algorithm, etc. However, it adds not only support to a new address family, but also some improvements to the protocol itself. The new version avoids some potential problems and inefficiencies present in the operation of OSPFv2.

OSPFv3 configuration syntax largely remains the same as for OSPFv2. One mayor difference is that there is no configuration for networks anymore, and interface configuration becomes mandatory, since OSPFv3 runs on link, not IP subnet, basis.

Example:

Configure OSPF on router A:

```
[admin@A] > routing ospf-v3 interface add interface=ether1 area=backbone
```

Configure OSPF on router B:

```
[admin@B] > routing ospf-v3 interface add interface=ether1 area=backbone
```

Redistribute a route from router A to router B:

```
[admin@A] > ipv6 route add dst-address=2001::/16 gateway=fe80::1%ether1
[admin@A] > routing ospf-v3 instance set default redistribute-static=as-type-1
[admin@A] > routing ospf-v3 route print
 # DESTINATION                                STATE          COST
 0 2001::/16                                  imported-ext-1 20
```

```
[admin@B] > ipv6 route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, o - ospf, b - bgp, U - unreachable
 #      DST-ADDRESS              GATEWAY                  DISTANCE
 0 ADo  2001::/16                fe80::1200:ff:fe00:10... 110
```

RIP

Similarly to OSPF, a new version of RIP was required to add IPv6 support. The new version is called RIPng (RIP new generation) and described in RFC 2080. Just like OSPFv3, RIPng runs on link, not IP subnet, basis - this means that you need to configure interfaces, not IP networks, on which to run RIPng.

Example:

Configure RIP on router A:

```
[admin@A] > routing ripng interface add interface=ether1
```

Configure RIP on router B:

```
[admin@B] > routing ripng interface add interface=ether1
```

Redistribute a route from router A to router B:

```
[admin@A] > ipv6 route add dst-address=2001::/16 gateway=fe80::1%ether1
[admin@A] > routing ripng set redistribute-static=yes
[admin@A] > routing ripng route print
Flags: C - connect, S - static, R - rip, O - ospf, B - bgp
 #   DST-ADDRESS
 0 S 2001::/16
```

```
[admin@B] > ipv6 route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, o - ospf, b - bgp, U - unreachable
 #      DST-ADDRESS              GATEWAY                  DISTANCE
 0 ADr  2001::/16                fe80::1200:ff:fe00:10... 120
```

6to4 (6in4) tunnels

This describes solution using global 6to4 relay address. For a solution using a tunnel broker see Setting up an IPv6 tunnel via a tunnel broker.

First, you will need a global routable IPv4 address. We assume the address 1.2.3.4 for the sake of this example.

Then you need to make user that the global 6to4 relay anycast address 192.88.99.1 is reachable and that it really provides relay services (since it's anycast address, your connection should be routed to the host having this addresses that is the closest to your location).

Then add *6to4* interface without specifying remote address and using your global IPv4 address as *local-address*:

```
interface 6to4 add mtu=1280 local-address=1.2.3.4 disabled=no
```

Now you need to add a IPv6 address to the tunnel interface. The address should be in form *"2002 + <IPv4 address in hex> + <custom id>"* . A bash script can be used to generate such IPv6 address for you:

```
 atis@atis-desktop:~$ ipv4="1.2.3.4"; id="1"; printf "2002:%02x%02x:%02x%02x::$id\n"
 `echo $ipv4 | tr "." " "`
2002:0102:0304::1
```

Add the generated address to the 6to4 interface:

```
ipv6 address add address=2002:0102:0304::1/128 interface=sit1
```

Add route to global IPv6 Internet through the tunnel interface using the anycast IPv4 address:

```
ipv6 route add dst-address=2000::/3 gateway=::192.88.99.1,sit1
```

Syntax for RouterOS v4.x, or RouterOS 3.x with **routing-test**:

```
ipv6 route add dst-address=2000::/3 gateway=::192.88.99.1%sit1
```

Now try to ping some IPv6 host (e.g. ipv6.google.com, 2001:4860:a003::68) to check your IPv6 connectivity.

See also 6in4 [1] and 6to4 [2] in Wikipedia.

Using dual stack

All IP services that listen to IPv6 also accept IPv4 connections. We take the web proxy for an example.

To force the web proxy to listen to IPv6 connections:

```
/ip proxy set src-address=::
```

To demonstrate that the dual stack is working, we connect to the web proxy at *10.0.0.131/fc00:1::1* using telnet, issue "GET /" request, and observe generated error message.

Connecting via IPv4:

```
$ telnet 10.0.0.131 8080
Trying 10.0.0.131...
Connected to 10.0.0.131.
Escape character is '^]'.
GET /

HTTP/1.0 404 Not Found
Content-Length: 518
...
```

```
Generated Mon, 18 Dec 2006 12:40:03 GMT by 10.0.0.131 (Mikrotik HttpProxy)
```

Connecting via IPv6:

```
$ telnet -6 fc00:1::1 8080
Trying fc00:1::1...
Connected to fc00:1::1.
GET /

HTTP/1.0 404 Not Found
Content-Length: 525
...
Generated Mon, 18 Dec 2006 12:38:51 GMT by ::ffff:10.0.0.131 (Mikrotik HttpProxy)
```

[Top | Back to Content]

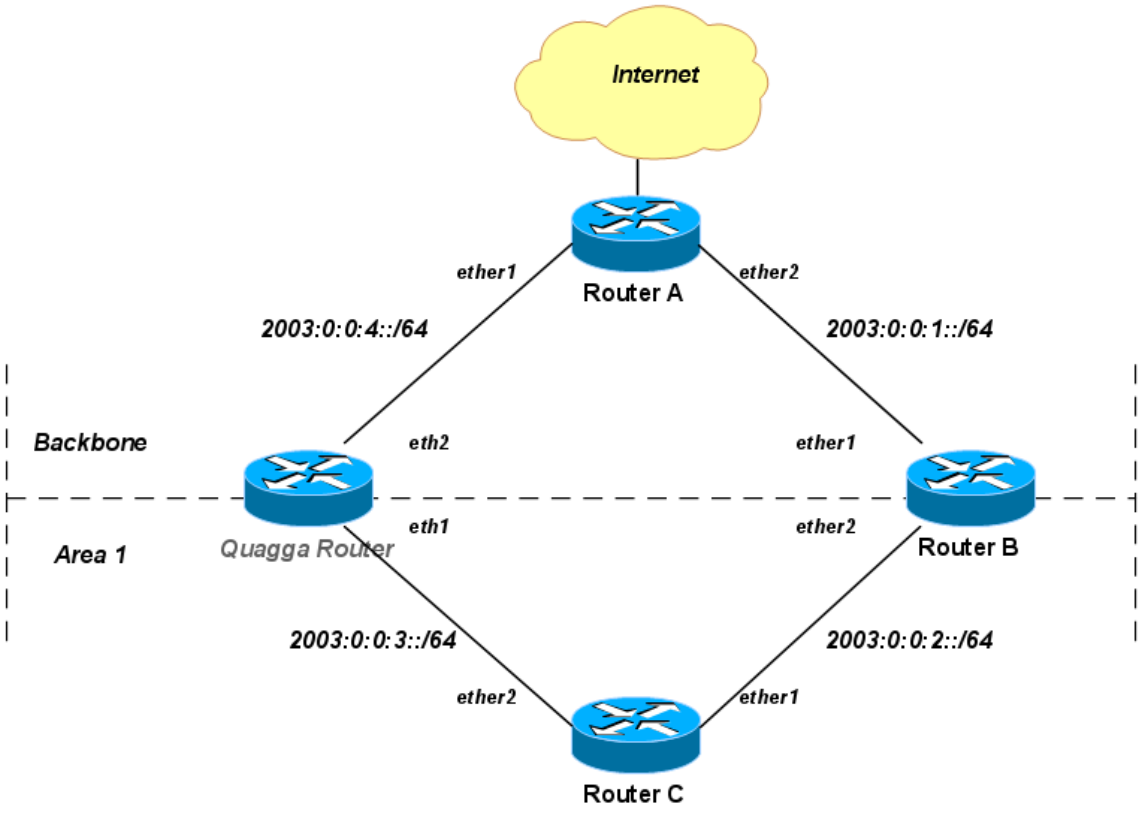
References

[1] http://en.wikipedia.org/wiki/6in4
[2] http://en.wikipedia.org/wiki/6to4

Manual:OSPFv3 with Quagga

In this example we demonstrate interoperability of MikroTik 3.x with Quagga in multi-area OSPF setup with load balancing.

RouterOS version 3.16 and Quagga 0.99.11 are used respectively.



Router A

```
/ipv6 address
 add address=2003::1:0:0:0:1/64 advertise=no interface=ether2
 add address=2003::4:0:0:0:1/64 advertise=no interface=ether1
 add address=2003::1/64 advertise=no interface=ToInternet

/routing ospf-v3
 set router-id=0.0.0.1 distribute-default=always-as-type-1

/routing ospf-v3 interface
 add interface=ether1 area=backbone
 add interface=ether2 area=backbone
```

Router B

```
/ipv6 address
 add address=2003::1:0:0:0:2/64 advertise=no interface=ether1
 add address=2003::2:0:0:0:2/64 advertise=no interface=ether2

/routing ospf-v3
 set router-id=0.0.0.2
/routing ospf-v3 area
 add area-id=0.0.0.1 name=area1
/routing ospf-v3 interface
 add interface=ether1 area=backbone
 add interface=ether2 area=area1
```

Quagga Router

```
debian:~# ip -6 addr add 2003:0:0:3::4/64 dev eth1
debian:~# ip -6 addr add 2003:0:0:4::4/64 dev eth2
debian:~#
debian:~# cat /etc/quagga/ospf6d.conf
...
interface eth1
 ipv6 ospf6 cost 10

interface eth2
 ipv6 ospf6 cost 10

router ospf6
 router-id 0.0.0.4
 interface eth1 area 0.0.0.1
 interface eth2 area 0.0.0.0

debian:~# telnet ::1 2606
Hello, this is Quagga (version 0.99.11).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
```

```
...

quagga# show ipv6 ospf6 route
*N E1 ::/0                          fe80::1200:ff:fe00:100      eth2 00:33:50
*N IA 2003:0:0:1::/64               fe80::1200:ff:fe00:100      eth2 00:32:55
*N IE 2003:0:0:2::/64               fe80::1200:ff:fe00:100      eth2 00:02:44
*N IA 2003:0:0:2::/64               fe80::1200:ff:fe00:301      eth1 00:02:37
*N IE 2003:0:0:3::/64               fe80::1200:ff:fe00:100      eth2 00:02:39
 N IA 2003:0:0:3::/64               ::                          eth1 00:02:46
*N IA 2003:0:0:4::/64               ::                          eth2 00:33:50
```

Router C

```
/ipv6 address
 add address=2003::2:0:0:0:3/64 advertise=no interface=ether1
 add address=2003::3:0:0:0:3/64 advertise=no interface=ether2

/routing ospf-v3
 set router-id=0.0.0.3
/routing ospf-v3 area
 add area-id=0.0.0.1 name=area1
/routing ospf-v3 interface
 add interface=ether1 area=area1
 add interface=ether2 area=area1

[admin@C] /routing ospf-v3> route print
 # DESTINATION                                 STATE      COST
 0 ::/0                                        ext-1      21
 1 2003::1:0:0:0:0/64                          inter-area 20
 2 2003::2:0:0:0:0/64                          intra-area 10
 3 2003::3:0:0:0:0/64                          intra-area 10
 4 2003::4:0:0:0:0/64                          inter-area 20

[admin@C] /routing ospf-v3> route print detail
 0 destination=::/0 state=ext-1 gateway=fe80::1200:ff:fe00:201,fe80::1200:ff:fe00:ff00
   interface=ether1,ether2 cost=21 area=external

 1 destination=2003::1:0:0:0:0/64 state=inter-area gateway=fe80::1200:ff:fe00:201
   interface=ether1 cost=20 area=area1

 2 destination=2003::2:0:0:0:0/64 state=intra-area gateway=:: interface=ether1 cost=10
   area=area1

 3 destination=2003::3:0:0:0:0/64 state=intra-area gateway=:: interface=ether2 cost=10
   area=area1

 4 destination=2003::4:0:0:0:0/64 state=inter-area gateway=fe80::1200:ff:fe00:ff00
   interface=ether2 cost=20 area=area1
```

Ping an "Internet" address from Router C (traffic will go through ECMP route):

```
[admin@C] > /ping 2003::1
2003::1 64 byte ping: ttl=63 time=20 ms
2003::1 64 byte ping: ttl=63 time=12 ms
2003::1 64 byte ping: ttl=63 time=9 ms
2003::1 64 byte ping: ttl=63 time=12 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 9/13.2/20 ms

[admin@C] > /tool traceroute 2003::1
     ADDRESS                                  STATUS
   1 2003::2:0:0:0:2 19ms 7ms 15ms
   2         2003::1 13ms 13ms 12ms
```

Manual:Routing/IGMP-Proxy

Applies to RouterOS: v4.5

- *Packages required:* **multicast**
- *Incompatible with:* **routing-test (v3)**

Summary

Internet Group Management Protocol (IGMP) proxy can be used to implement multicast routing. It is forwarding IGMP frames and commonly is used when there is no need for more advanced protocol like PIM.

IGMP proxy features:

- The simplest way how to do multicast routing;
- Can be used in topologies where PIM-SM is not suitable for some reason;
- Takes slightly less resources than PIM-SM;
- Ease of configuration.

On the other hand, IGMP proxy is not well suited for complicated multicast routing setups. Compared to PIM based solutions, IGMP proxy does not support more than one upstream interface and routing loops are not detected or avoided.

MikroTik RouterOS IGMP proxy supports IGMP version 2 (RFC 2236).

Example

To forward all multicast data coming from *ether1* interface to all other interfaces, where subscribers are connected:

```
[admin@MikroTik] /routing igmp-proxy> interface add interface=ether1 upstream=yes
[admin@MikroTik] /routing igmp-proxy> interface add interface=all
[admin@MikroTik] /routing igmp-proxy> interface print
Flags: X - disabled, I - inactive, D - dynamic, U - upstream
 #    INTERFACE                                                          THRESHOLD
 0  U ether1                                                             1
 1    all                                                                1
```

```
 2 D  ether2                                                                1
 3 D  ether3                                                                1
```

You may also need to configure **alternative-subnets** on upstream interface - in case if the multicast sender address is in an IP subnet that is not directly reachable from from the local router.

```
[admin@MikroTik] /routing igmp-proxy> interface set [find upstream=yes] \
 alternative-subnets=1.2.3.0/24,2.3.4.0/24
```

/routing igmp-proxy

General configuration.

- **query-interval** (time, 00:00:01 - 01:00:00) : how often to send out IGMP Query messages over upstream interface
- **query-response-interval** (time, 00:00:01 - 01:00:00) : how long to wait for responses to an IGMP Query message
- **quick-leave** (yes|no) : specifies action on IGMP Leave message. If quick-leave is on, then an IGMP Leave message is sent upstream as soon as a leave is received from the first client on downstream interface

/routing igmp-proxy interface

Used to configure what interfaces will participate as IGMP proxy interfaces on router. If an interface is not configured as IGMP proxy interface, then all IGMP traffic received on it will be ignored.

- **alternative-subnets** (list of IP prefixes) : by default, only packets from directly attached subnets are accepted. This parameter can be used to specify a list of alternative valid packet source subnets, both for data or IGMP packets. Has effect only on upstream interface. Should be used when the source of multicast data often is in a different IP network.
- **interface** (interface name) : RouterOS interface
- **threshold** (integer) : minimal TTL; packets received with a lower TTL value are ignored
- **upstream** (yes|no) : interface is called "upstream" if it's in the direction of the root of the multicast tree. An IGMP forwarding router must have exactly one upstream interface configured. The upstream interface is used to sent out IGMP membership requests.

/routing igmp-proxy mfc

Multicast forwarding cache (MFC) status.

- **group** (IP address) : IGMP group address
- **source** (IP address) : multicast data originator address
- **incoming-interface** (interface name) : packet stream is coming in router through this interface
- **outgoing-interface** (interface name) : packet stream is going out of router through this interface

Static multicast forwarding cache (MFC) entries

Since RouterOS 4.5 MFC is enabled to add static multicast forwarding rules. If a static rule is added, all dynamic rules for that group will be ignored.

Configuration

These rules will take effect only if IGMP-proxy interfaces are configured (upstream and downstram interfaces should be set) or these rules wont be active.

- **downstream-interfaces** (list of interfaces) : received stream will be sent out to listed interfaces only.
- **group** (multicast group address) : multicast stream group address this rule applies *should be set*
- **source** (IP address) : IP address we are receiving stream from *should be set*
- **upstream-interface** (interface) : interface that is receiving stream data *should be set*

Example

Example #1 will forward stream unconditionally if it comes in from ether1 with set source and will be sent out to ether2, clients that will try to get stream on interface ether3 will not receive that stream.

```
/routing igmp-proxy interface add comment="" disabled=no interface=ether1 threshold=1 upstream=yes
/routing igmp-proxy interface add comment="" disabled=no interface=ether2 threshold=1
/routing igmp-proxy interface add comment="" disabled=no interface=ether3 threshold=1
/routing igmp-proxy mfc add source=192.168.0.1 upstream-interface=ether1 downstream-interface=ether2 \
```

group=224.10.10.11 disabled=no

Example #2 224.10.10.10 group will not be sent at all

```
/routing igmp-proxy mfc add source=192.168.0.1 upstream-interface=ether1 group=224.10.10.11 disabled=no
```

References

RFC 4605 IGMP/MLD - Based Multicast Forwarding [1]

References

[1] http://www.ietf.org/rfc/rfc4605.txt

Manual:Tools/IP-Scan

Summary

Sub-menu: `/tool ip-scan`

Standards:

IP Scan tool allows user to scan network based on some network prefix or by setting interface to listen to. Either way tool collects certain data from the network.

Properties

Property	Description
address (*string*; Default:)	IP address of network device.
mac-address (*string*; Default:)	MAC address of network device.
time (*integer in ms*; Default:)	response time of seen network device when found
DNS (*string*; Default:)	DNS name of network device
SNMP (*string*; Default:)	SNMP name of the device
NET-BIOS (*string*; Default:)	NET-BIOS name of device if advertised by device

How to use

When using IP scan tool user must choose what it wants to scan for:

- certain IPv4 prefix

  tool will attempt to scan all the addreses or address set in address.

- interface of the router

  tool will attempt to listen to packets that are "passing by" and attempt to compile results when something is found

There is a possibility to set both but then results may be inconclusive

[Top | Back to Content **]**

Manual:Initial Configuration

Summary

Congratulations, you have got hold of MikroTik router for your home network. This guide will help you to do initial configuration of the router to make your home network a safe place to be.

The guide is mostly intended in case if default configuration did not get you to the internet right away, however some parts of the guide is still useful.

Connecting wires

Router's initial configuration should be suitable for most of the cases. Description of the configuration is on the back of the box and also described in the online manual.

The best way to connect wires as described on the box:

- Connect ethernet wire from your internet service provider (ISP) to port *ether1*, rest of the ports on the router are for local area network (LAN). At this moment, your router is protected by default firewall configuration so you should not worry about that;
- Connect LAN wires to the rest of the ports.

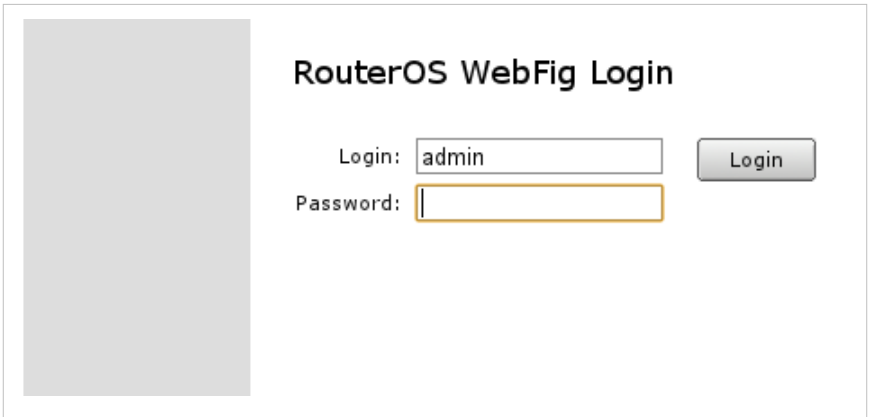
Configuring router

Initial configuration has DHCP client on WAN interface (ether1), rest of the ports are considered your local network with DHCP server configured for automatic address configuration on client devices. To connect to the router you have to set your computer to accept DHCP settings and plug in the ethernet cable in one of the LAN ports (please check routerboard.com for port numbering of the product you own, or check front panel of the router).

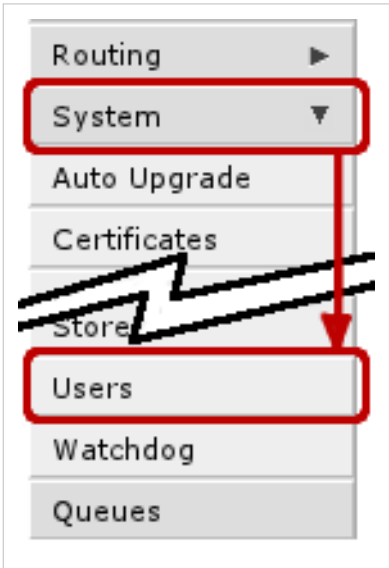
Logging into the router

To access the router enter address *192.168.88.1* in your browser. Main RouterOS page will be shown as in the screen shot below. Click on WebFig from the list.

You will be prompted for login and password to access configuration interface. Default login name is *admin* and blank password (leave empty field as it is already).



Router user accounts

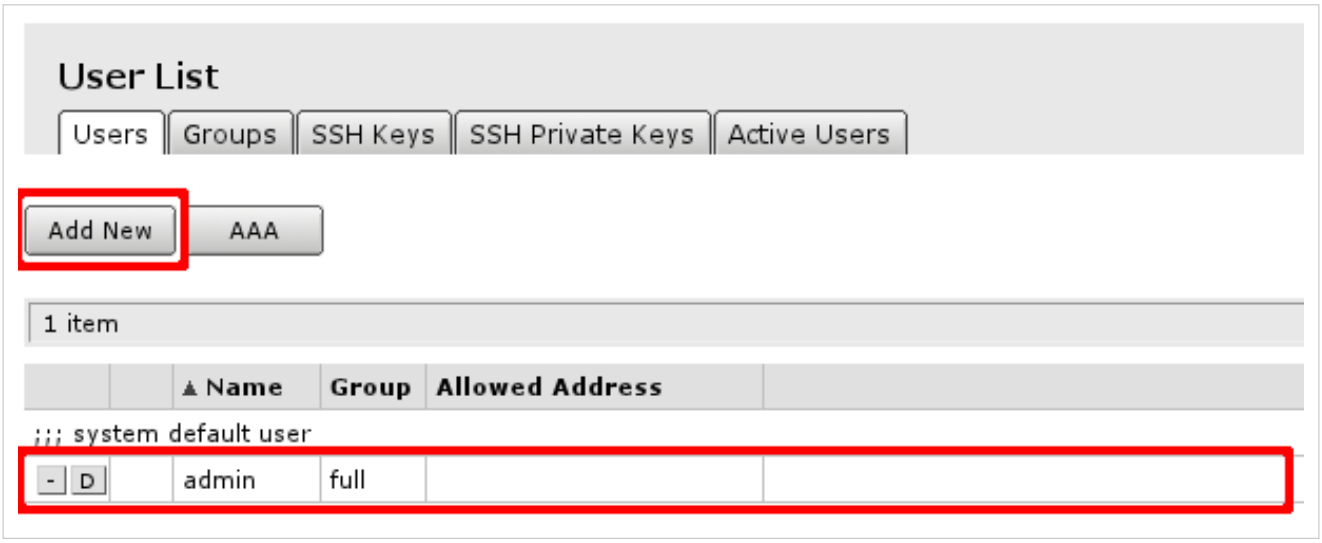


It is good idea to start with password setup or add new user so that router is not accessible by anyone on your network. User configuration is done form **System -> Users** menu.

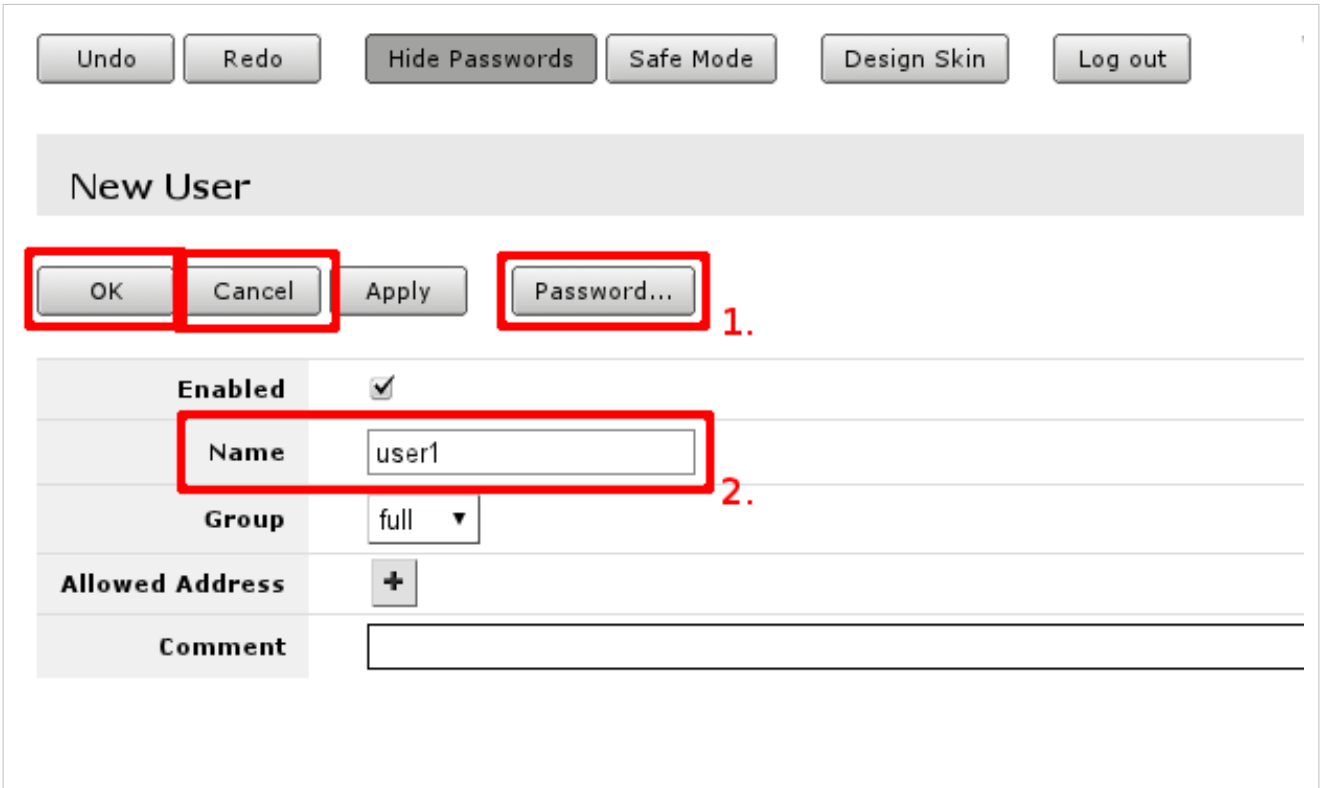
To access this menu, click on **System** on the left panel and from the dropdown menu choose **Users** (as shown in screenshot on the left)

You will see this screen, where you can manage users of the router. In this screen you can edit or add new users:

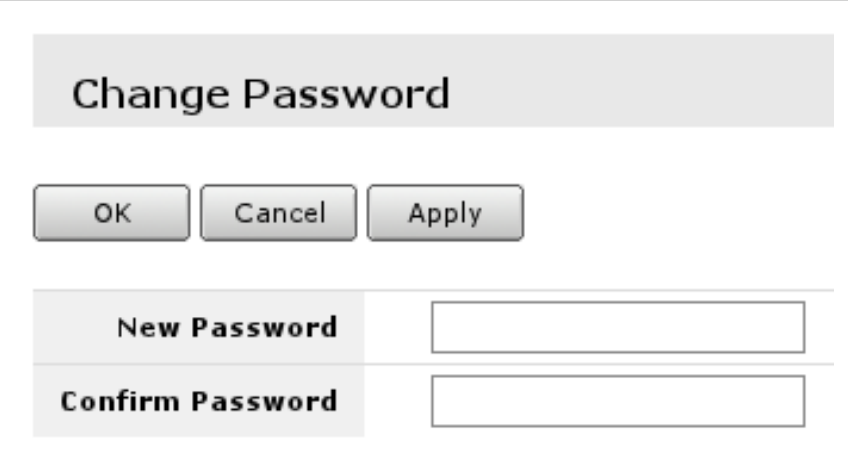
- When you click on account name (in this case *admin*), edit screen for the user will be displayed.
- If you click on **Add new** button, new user creation screen will be displayed.



Both screens are similar as illustrated in screenshot below. After editing user's data click *OK* (to accept changes) or *Cancel*. It will bring you back to initial screen of user management.



In user *edit/Add new* screen you can alter existing user or create new. Field marked with *2.* is the user name, field *1.* will open password screen, where old password for the user can be changed or added new one (see screenshot below).



Configure access to internet

If initial configuration did not work (your ISP is not providing DHCP server for automatic configuration) then you will have to have details from your ISP for static configuration of the router. These settings should include

- IP address you can use
- Network mask for the IP address
- Default gateway address

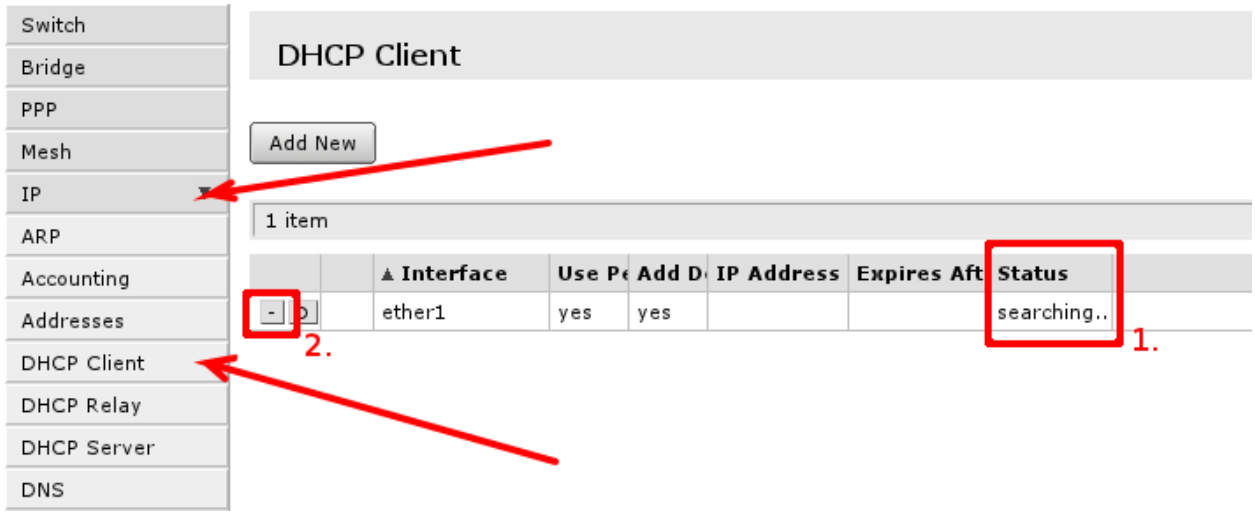
Less important settings regarding router configuration:

- DNS address for name resolution
- NTP server address for time automatic configuration

- Your previous MAC address of the interface facing ISP

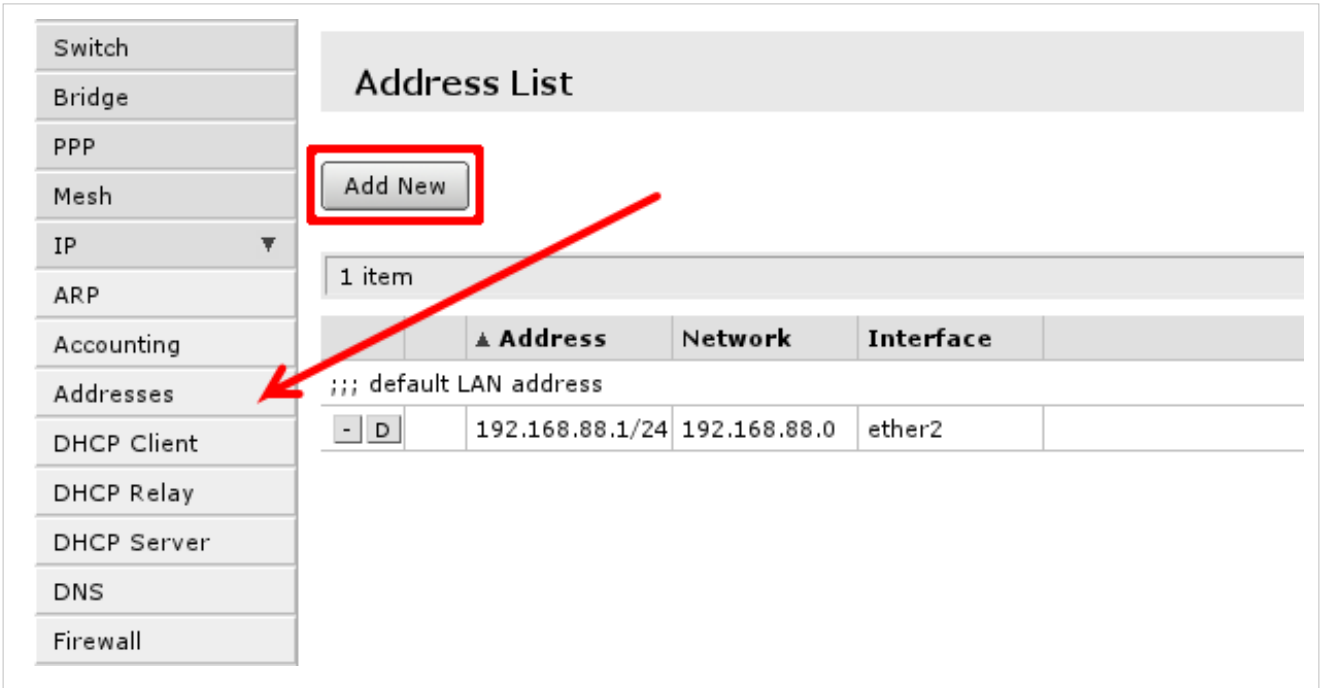
DHCP Client

Default configuration is set up using DHCP-Client on interface facing your ISP or wide area network (WAN). It has to be disabled if your ISP is not providing this service in the network. Open 'IP -> DHCP Client' and inspect field *1.* to see status of DHCP Client, if it is in state as displayed in screenshot, means your ISP is not providing you with automatic configuration and you can use button in selection *2.* to remove DHCP-Client configured on the interface.

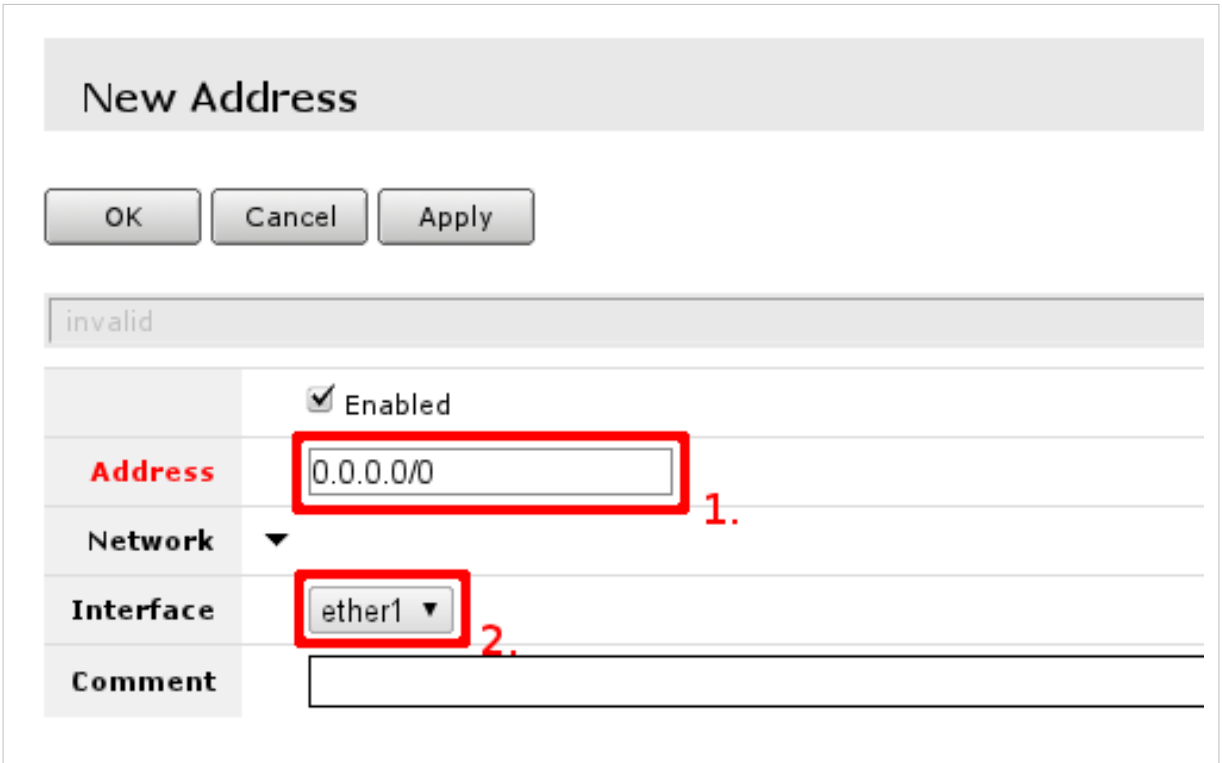


Static IP Address

To manage IP addresses of the router open 'IP -> Address'



You will have one address here - address of your local area network (LAN) *192.168.88.1* one you are connected to router. Select *Add new* to add new static IP address to your router's configuration.



You have to fill only fields that are marked. Field *1.* should contain *IP address* provided by your ISP and *network mask'. Examples:*

```
172.16.88.67/24
```

both of these notations mean the same, if your ISP gave you address in one notation, or in the other, use one provided and router will do the rest of calculation.

Other field of interest is *interface* this address is going to be assigned. This should be interface your ISP is connected to, if you followed this guide - interface contains name - *ether1*

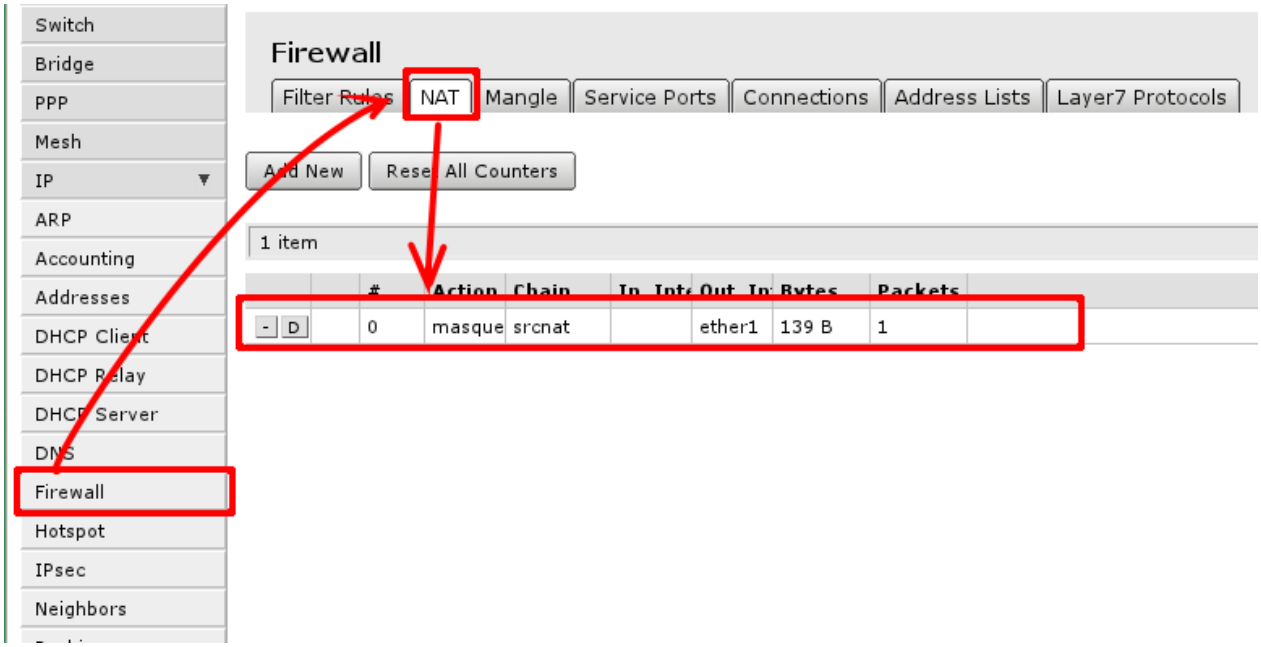
Note: While you type in the address, webfig will calculate if address you have typed is acceptable, if it is not label of the field will turn red, otherwise it will be blue

Note: It is good practice to add comments on the items to give some additional information for the future, but that is not required

Configuring network address translation (NAT)

Since you are using local and global networks, you have to set up network masquerade, so that your LAN is hidden behind IP address provided by your ISP. That should be so, since your ISP does not know what LAN addresses you are going to use and your LAN will not be routed from global network.

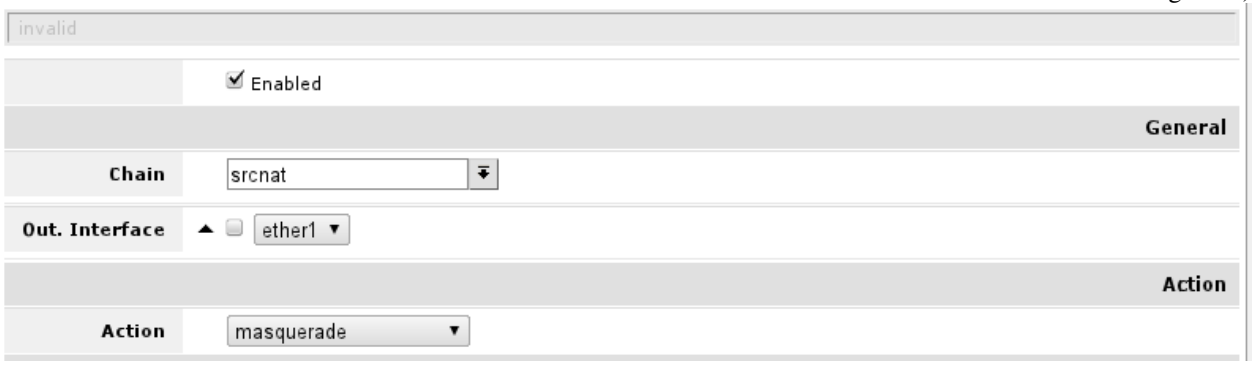
To check if you have the source NAT open 'IP -> Firewall -> tab NAT' and check if item highlighted (or similar) is in your configuration.



Essential fields for masquerade to work:

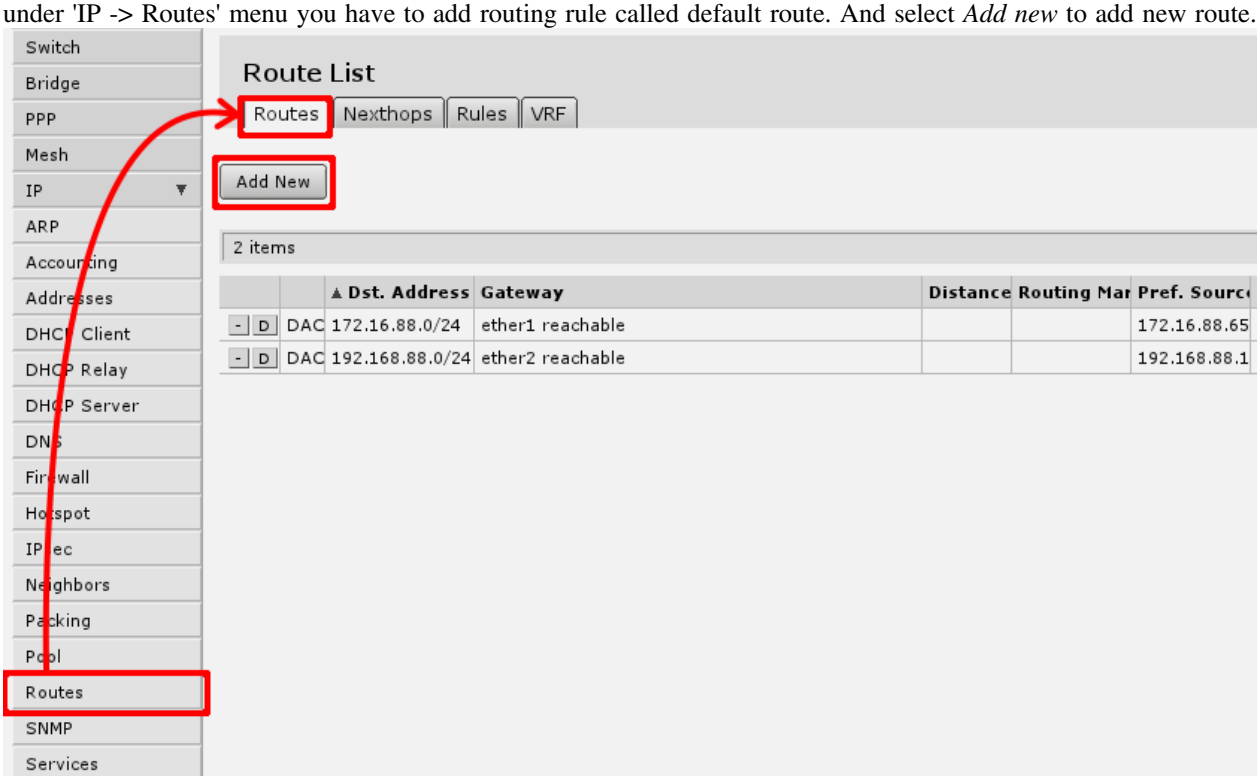
- enabled is checked;
- chain - should be *srcnat*;
- out-interface is set to interface connected to your ISP network, Following this guide *ether1*;
- action should be set to *masquerade*.

In screenshot correct rule is visible, note that irrelevant fields that should not have any value set here are hidden (and can be ignored)

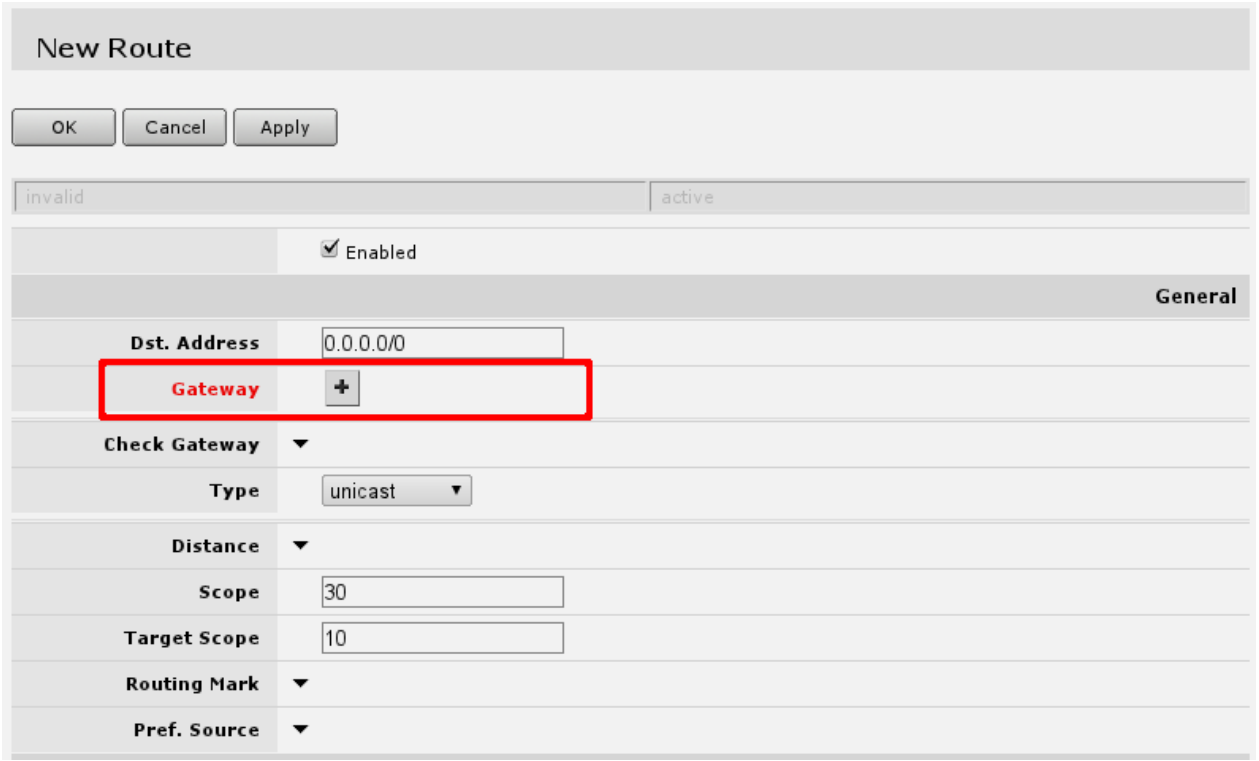


Default gateway

under 'IP -> Routes' menu you have to add routing rule called default route. And select *Add new* to add new route.

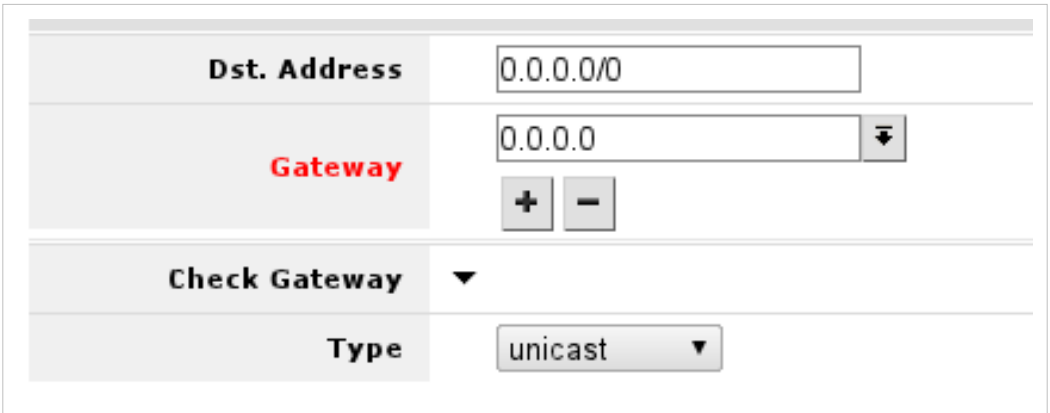


In screen presented you will see the following screen:



here you will have to press button with **+** near red *Gateway* label and enter in the field default gateway, or simply gateway given by your ISP.

This should look like this, when you have pressed the **+** button and enter gateway into the field displayed.



After this, you can press OK button to finish creation of the default route.

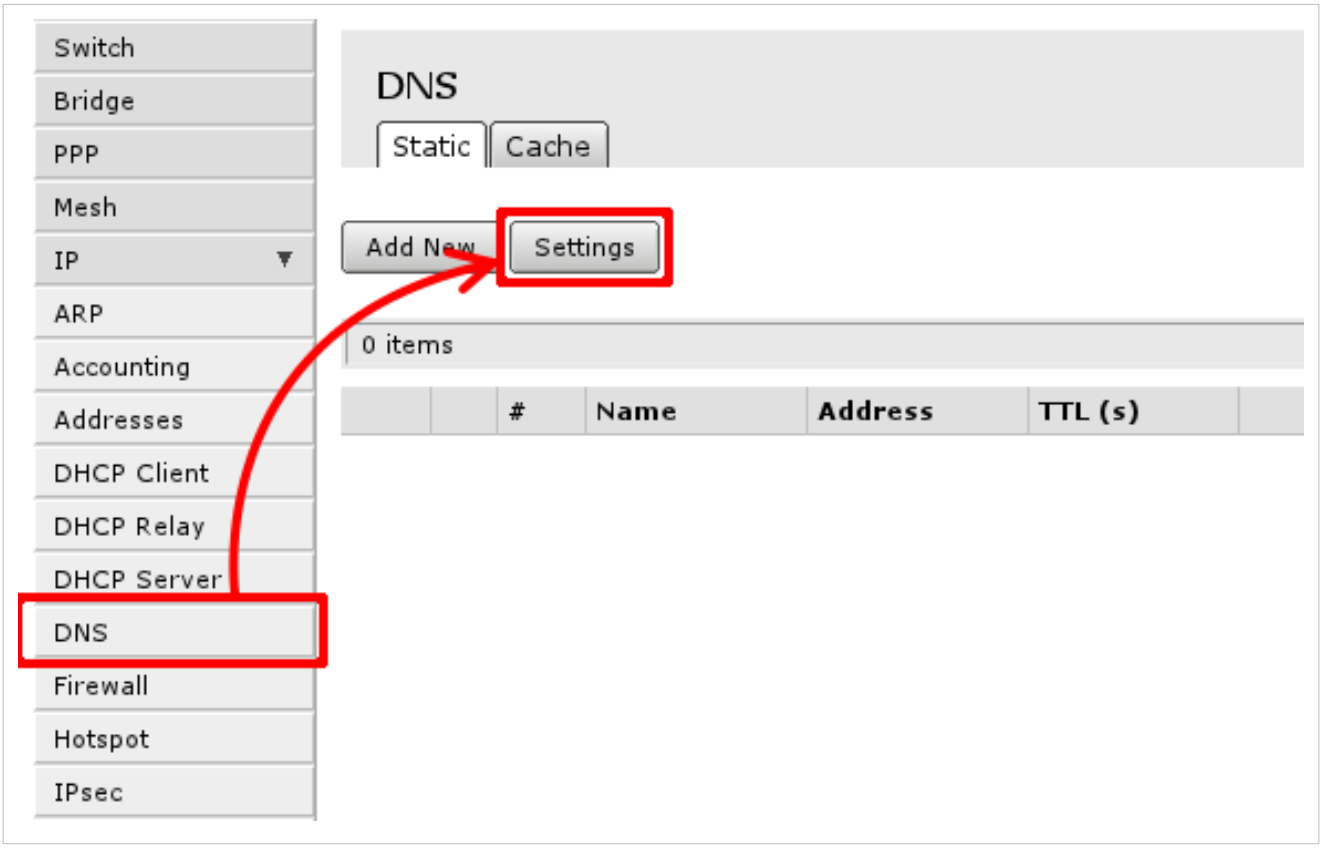
At this moment, you should be able to reach any globally available host on the Internet using IP address.

To check weather addition of default gateway was successful use *Tools -> Ping*

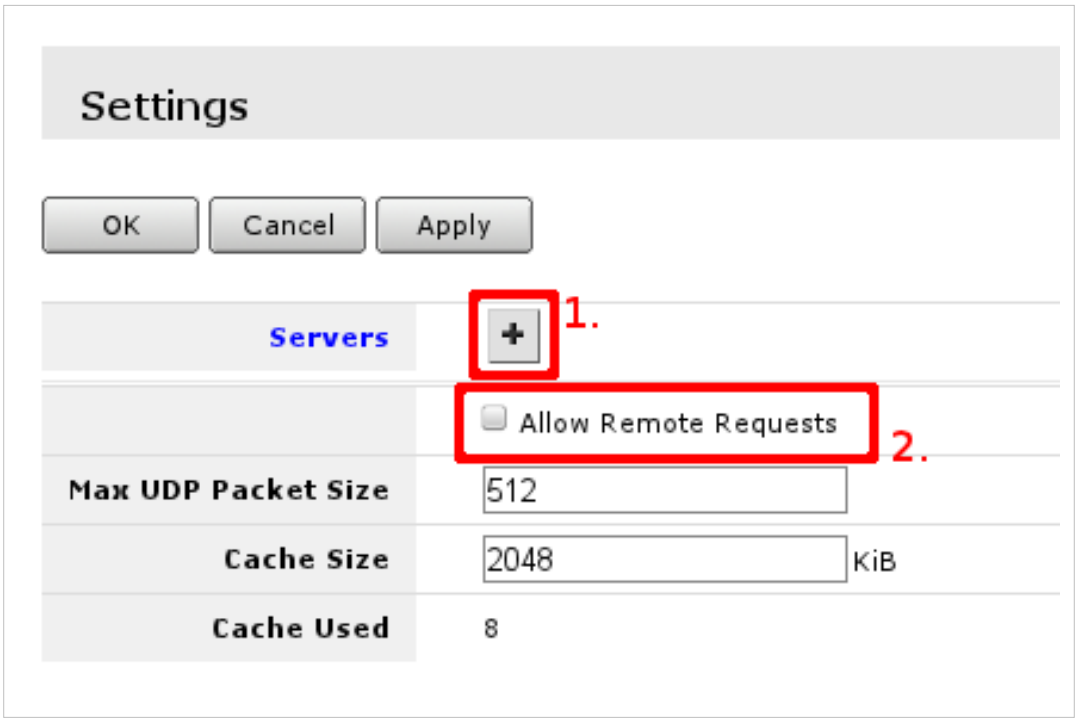
Domain name resolution

To be able to open web pages or access Internet hosts by domain name DNS should be configured, either on your router or your computer. In scope of this guide, i will present only option of router configuration, so that DNS addresses are given out by DHCP-Server that you are already using.

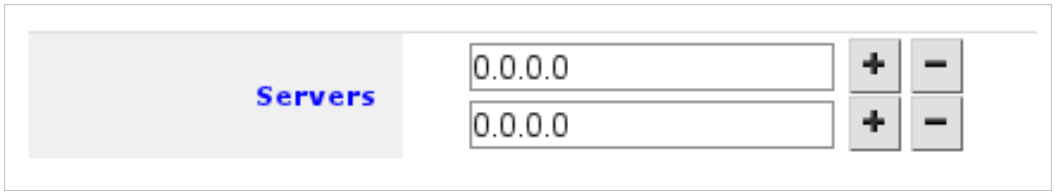
This can be done in 'IP -> DNS ->Settings', first Open 'IP ->DNS':



Then select *Settings* to set up DNS cacher on the router. You have to add field to enter DNS IP address, section *1.* in image below. and check *Allow Remote Requests* marked with *2.*



The result of pressing **+** twice will result in 2 fields for DNS IP addresses:

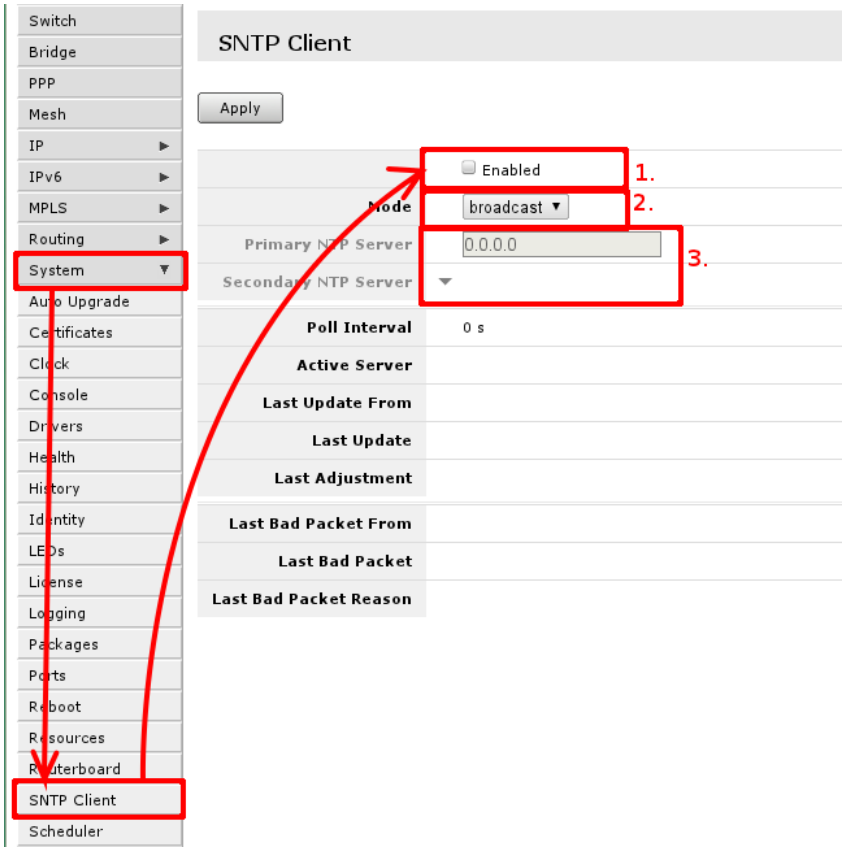


Note: Filling acceptable value in the field will turn field label blue, other way it will be marked red.

SNTP Client

RouterBOARD routers do not keep time between restarts or power failuers. To have correct time on the router set up SNTP client if you require that.

To do that, go to 'System -> SNTP' where you have to enable it, first mark, change mode from broadcast to unicast, so you can use global or ISP provided NTP servers, that will allow to enter NTP server IP addresses in third area.



Setting up Wireless

For ease of use bridged wireless setup will be used, so that your wired hosts will be in same ethernet broadcast domain as wireless clients.

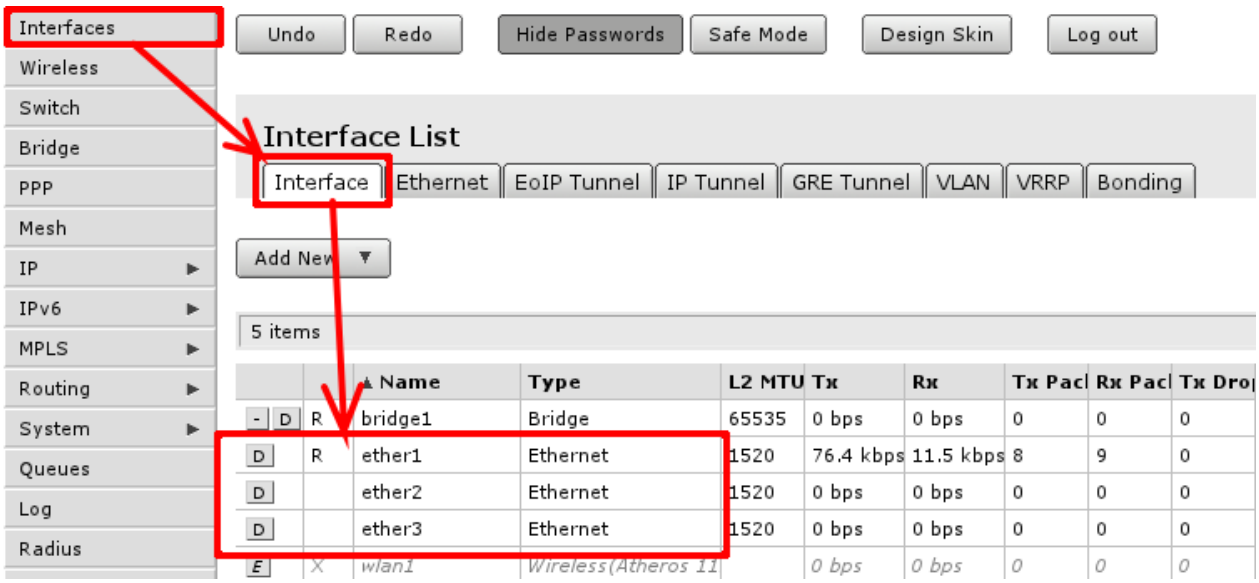
To make this happen several things has to be checked:

- Ethernet interfaces designated for LAN are swtiched or bridged, or they are separate ports;
- If bridge interface exists;
- Wireless interface *mode* is set to *ap-bridge* (in case, router you have has level 4 or higher license level), if not, then *mode* has to be set to *bridge* and only one client (station) will be able to connect to the router using wireless network;
- There is appropriate security profile created and selected in interface settings.

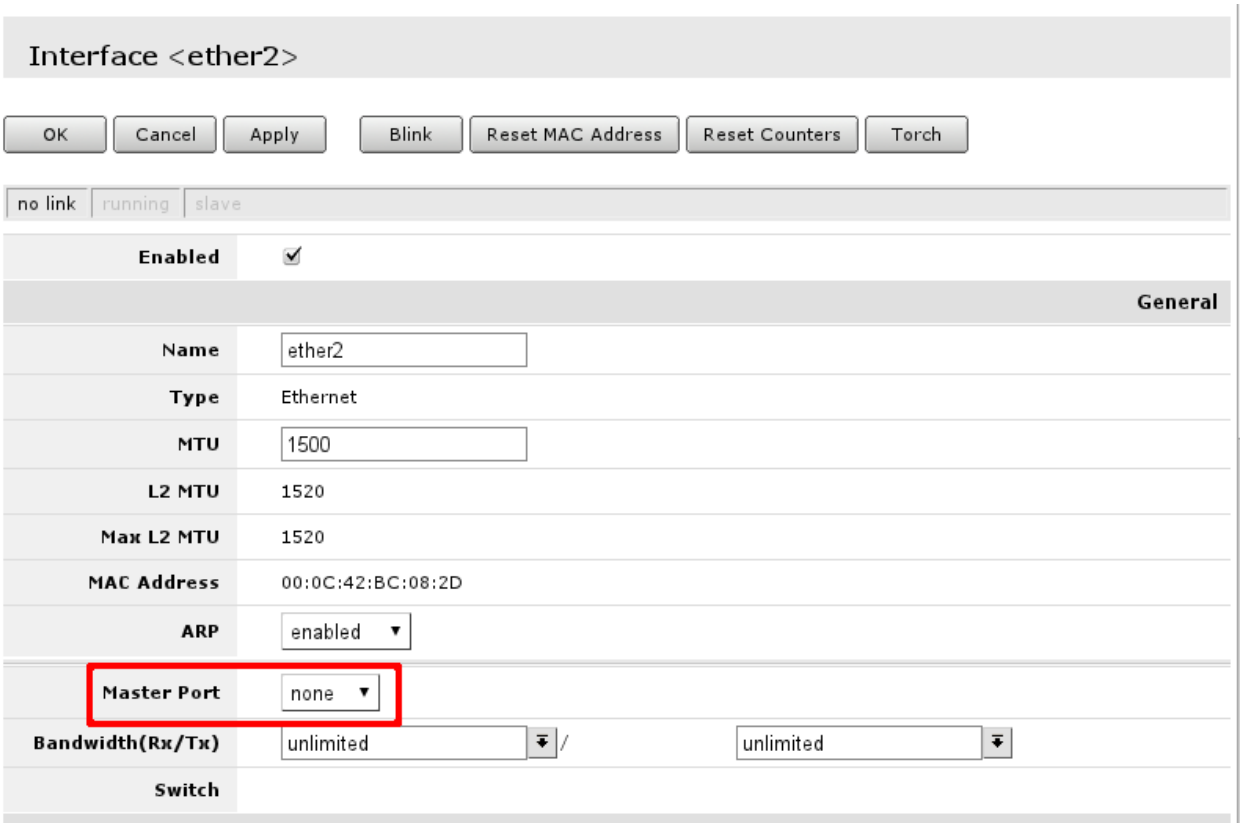
Check Ethernet interface state

Warning: Changing settings may affect connectivity to your router and you can be disconnected from the router. Use *Safe Mode* so in case of disconnection made changes are reverted back to what they where before you entered safe mode

To check if ethernet port is switched, in other words, if ethernet port is set as slave to another port go to 'Interface' menu and open Ethernet interface details. They can be distinguished by Type column displaying *Ethernet*.



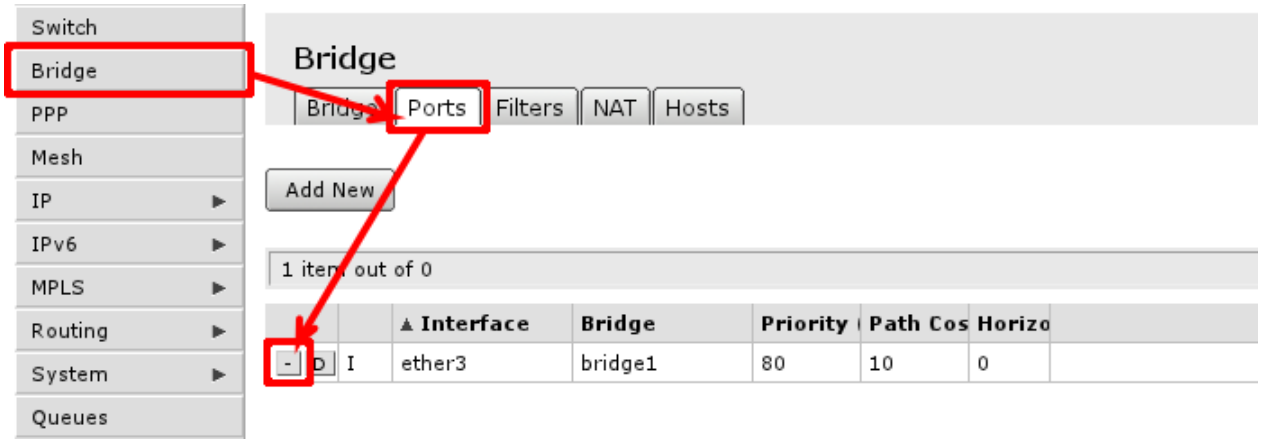
When interface details are opened, look up *Master Port* setting.



Available settings for the attribute are none, or one of Ethernet interface names. If name is set, that mean, that interface is set as slave port. Usually RouterBOARD routers will come with *ether1* as intended WAN port and rest of ports will be set as slave ports of *ether2* for LAN use.

Check if all intended LAN Ethernet ports are set as slave ports of the rest of one of the LAN ports. For example, if *ether2. ether3, ether4* and *ether5* are intended as LAN ports, set on ether3 to ether5 attribute *Master Port* to *ether2*.

In case this operation fails - means that Ethernet interface is used as port in bridge, you have to remove them from bridge to enable hardware packet switching between Ethernet ports. To do this, go to *Bridge -> Ports* and remove slave ports (in example, *ether3* to *ether5*) from the tab.



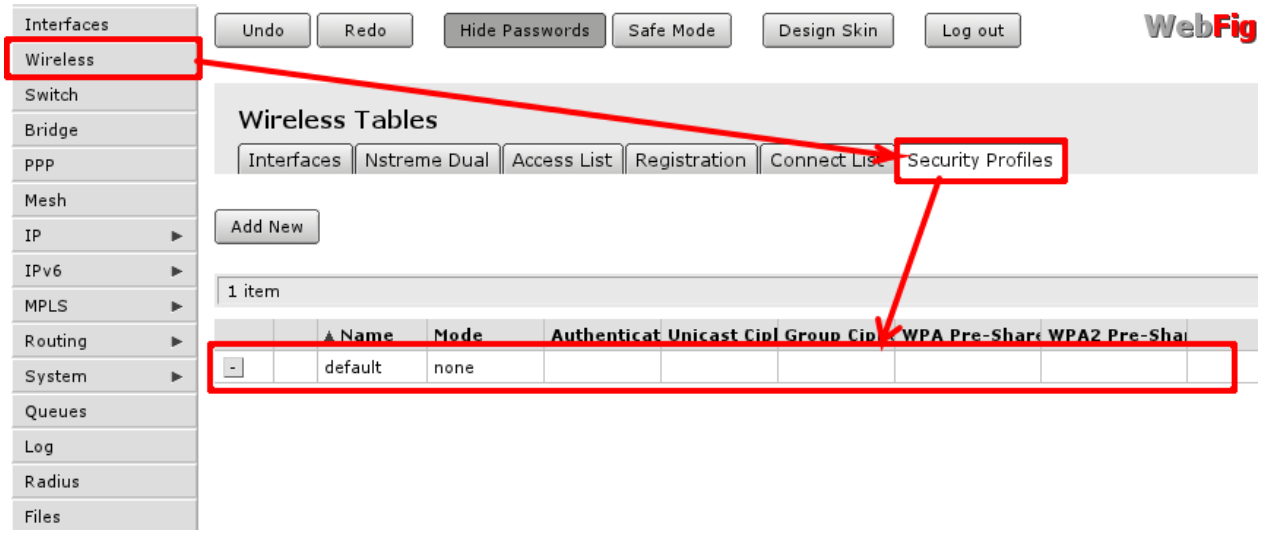
Note: If master port is present as bridge port, that is fine, intended configuration requires it there, same applies to wireless interface (*wlan*)

Security profile

It is important to protect your wireless network, so no malicious acts can be performed by 3rd parties using your wireless access-point.

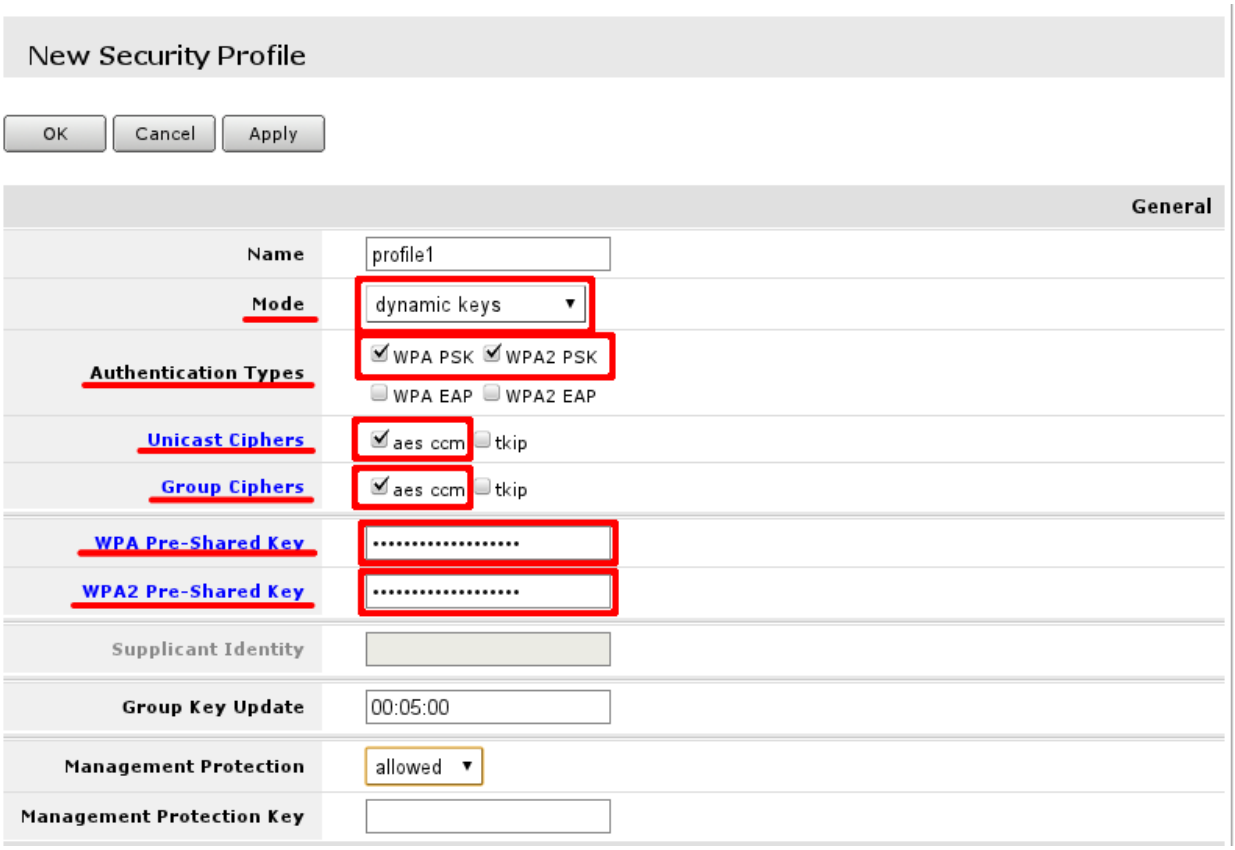
To edit or create new security profile head to 'Wireless -> tab 'Security Prodiles' and choose one of two options:

- Using *Add new* create new profile;
- Using highlighted path in screenshot edit default profile that is already assigned to wireless interface.



In This example i will create new security profile, editing it is quite similar. Options that has to be set are highlighted with read and recommended options are outlined by red boxes and pre-set to recommended values. WPA and WPA2 is used since there are still legacy equipment around (Laptops with Windows XP, that do not support WPA2 etc.)

WPA Pre- shared key and WPA2 Pre- shared key should be entered with sufficient length. If key length is too short field label will indicate that by turning red, when sufficient length is reached it will turn blue.



Note: WPA and WPA2 pre-shared keys should be different

Note: When configuring this, you can deselect *Hide passwords* in page header to see the actual values of the fields, so they can be successfully entered into device configuration that are going to connect to wireless access-point

Wireless settings

Adjusting wireless settings. That can be done here:

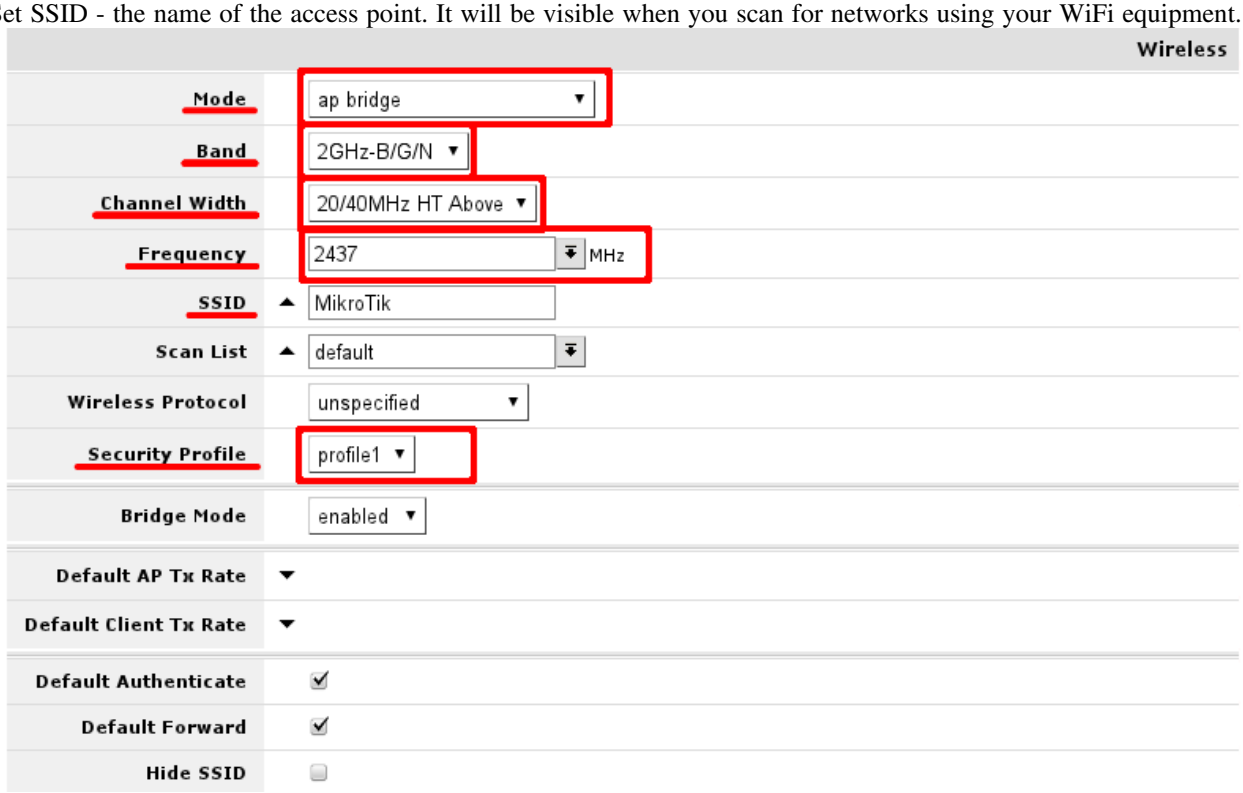
In *General* section adjust settings to settings as shown in screenshot. Consider these safe, however it is possible, that these has to be adjusted slightly.

Interface mode has to be set to *ap-bridge*, if that is not possible (license resctrictions) set to bridge, so one client will be able to connect to device.

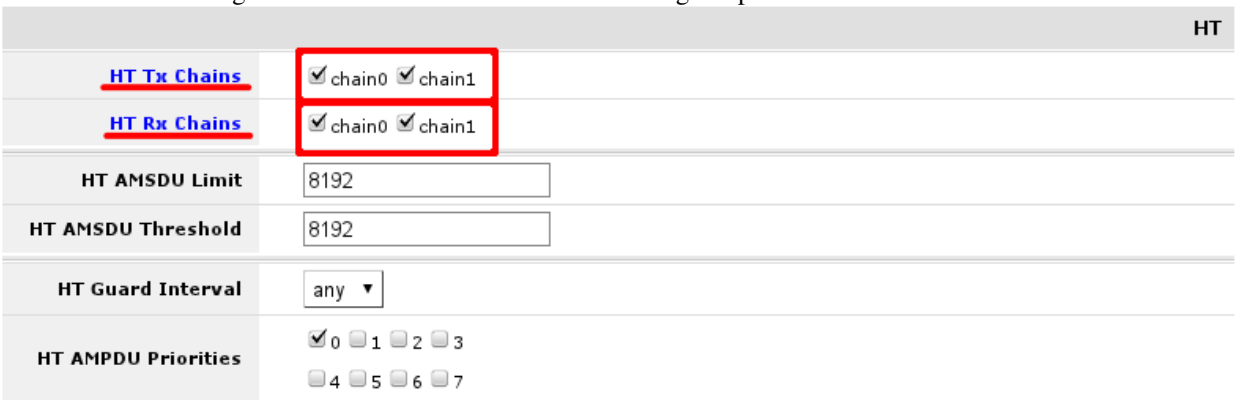
WiFI devices usually are designed with 2.4GHz modes in mind, setting band to 2GHz-b/g/n will enable clients with 802.11b, 802.11g and 802.11n to connect to the access point

Adjust channel width to enable faster data rates for 802.11n clients. In example channel 6 is used, as result, *20/40MHz HT Above* or *20/40 MHz HT Below* can be used. Choose either of them.

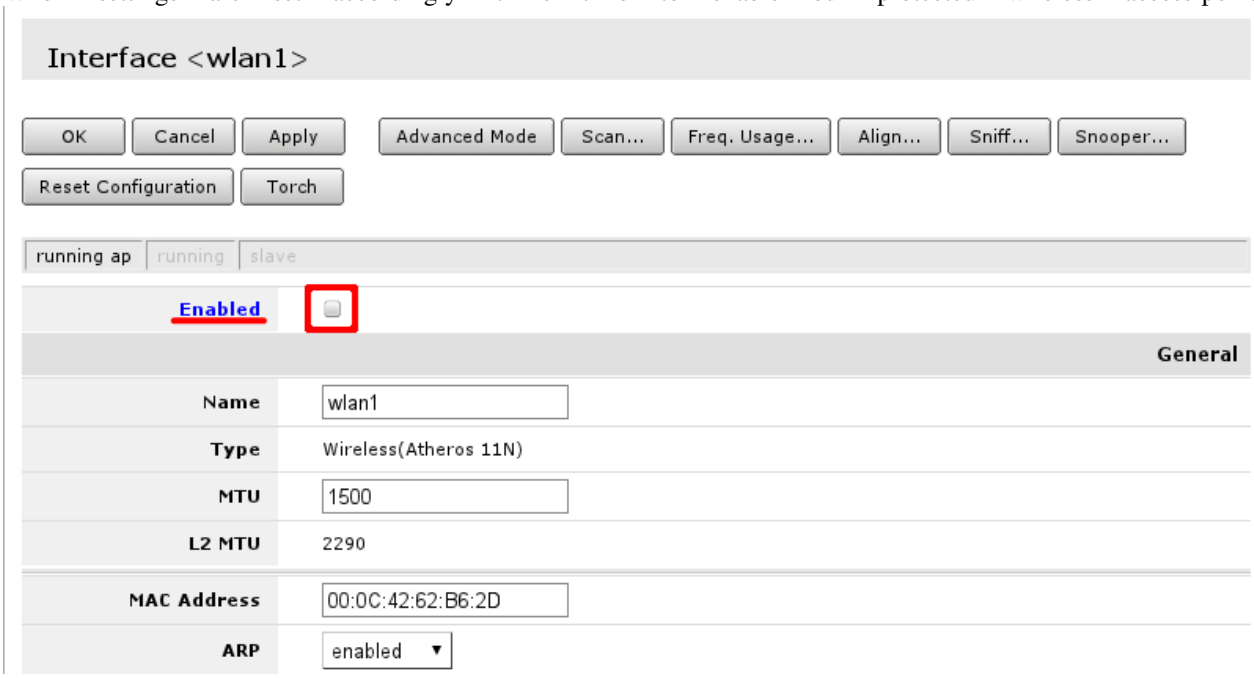
Set SSID - the name of the access point. It will be visible when you scan for networks using your WiFi equipment.



In section *HT* set change HT transmit and receive chains. It is good practice to enable all chains that are available



When settings are set accordingly it is time to enable our protected wireless access-point

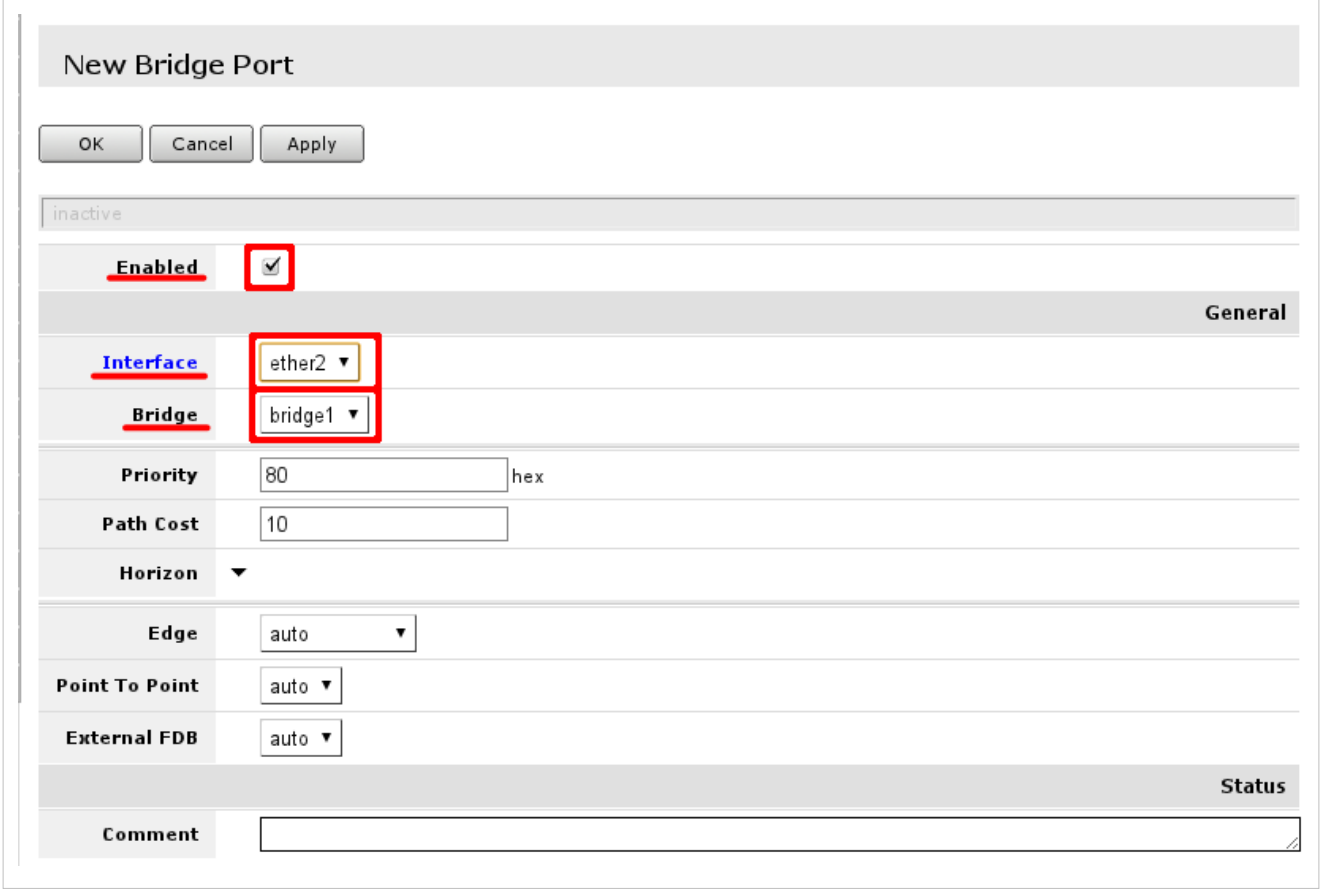


Bridge LAN with Wireless

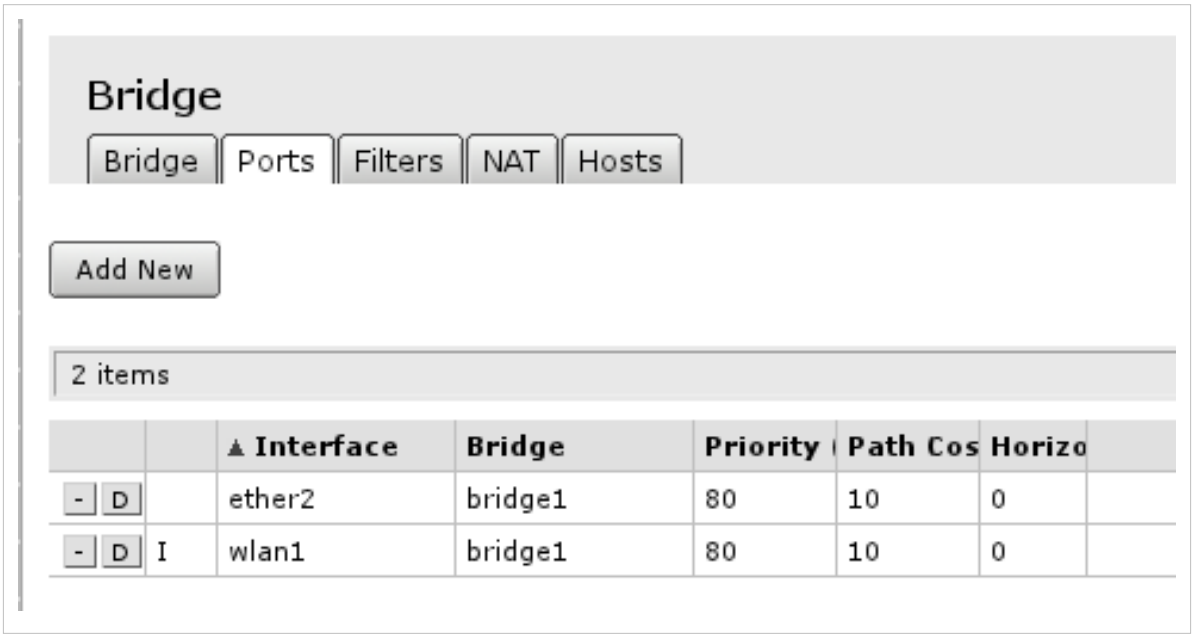
Open *Bridge* menu and check if there are any bridge interface available first mark. If there is not, select *Add New* marked with second mark and in the screen that opens just accept the default settings and create interface. When bridge interface is availbe continue to *Ports* tab where master LAN interface and WiFI interface have to be added.

First marked area is where interfaces that are added as ports to bridge interface are visible. If there are no ports added, choose *Add New* to add new ports to created bridge interfaces.

When new bridge port is added, select that it is enabled (part of active configuration), select correct bridge interface, following this guide - there should be only 1 interface. And select correct port - LAN interface master port and WiFi port



Finished look of bridge configured with all ports required



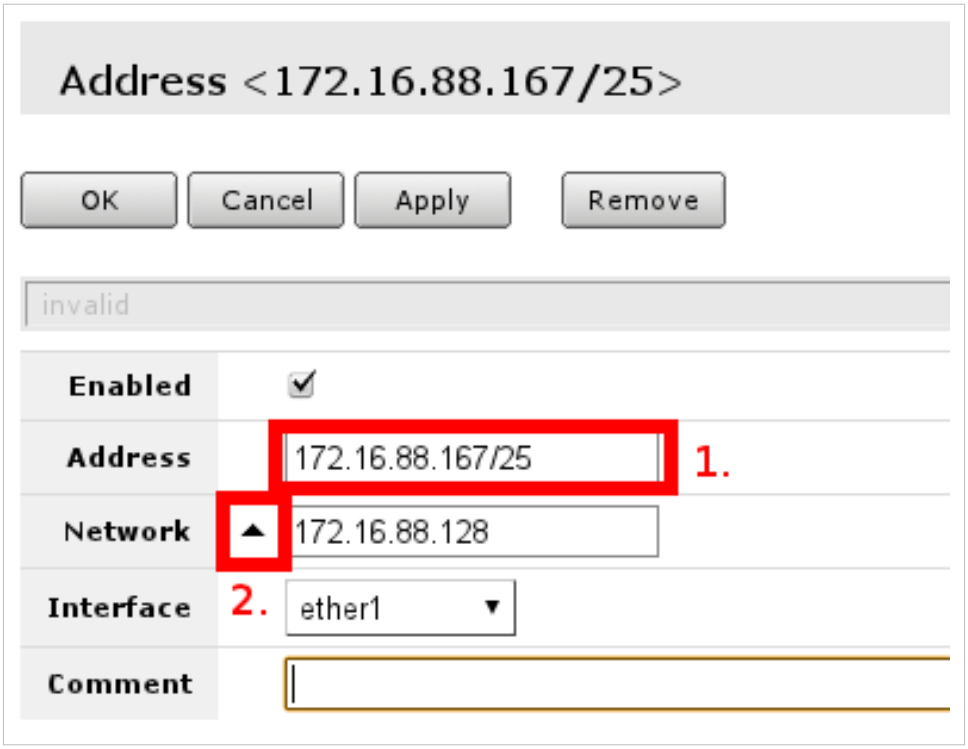
Troubleshooting & Advanced configuration

This section is here to make some deviations from configuration described in the guide itself. It can require more understanding of networking, wireless networks in general.

General

Check IP address

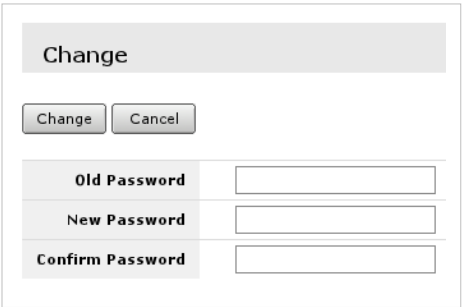
Adding IP address with wrong network mask will result in wrong network setting. To correct that problem it is required to change *address* field, first section, with correct address and network mask and *network* field with correct network, or unset it, so it is going to be recalculated again



Change password for current user

To change password of the current user, safe place to go is *System -> Password*

Where all the fields has to be filled. There is other place where this can be done in case you have full privileges on the router.



Change password for existing user

If you have full privileges on the router, it is possible to change password for any user without knowledge of current one. That can be done under *System -> Users* menu.

Steps are:

- Select user;
- type in password and re-type it to know it is one you intend to set

No access to the Internet or ISP network

If you have followed this guide to the letter but even then you can only communicate with your local hosts only and every attempt to connect to Internet fails, there are certain things to check:

- If masquerade is configured properly;
- If setting MAC address of previous device on WAN interface changes anything
- ISP has some captive portal in place.

Respectively, there are several ways how to solve the issue, one - check configuration if you are not missing any part of configuration, second - set MAC address. Change of mac address is available only from CLI - *New Terminal* from the left side menu. If new window is not opening check your browser if it is allowing to open popup windows for this place. There you will have to write following command by replacing MAC address to correct one:

```
/interface ethernet set ether1 mac-address=XX:XX:XX:XX:XX:XX
```

Or contact your ISP for details and inform that you have changed device.

Checking link

There are certain things that are required for Ethernet link to work:

- Link activity lights are on when Ethernet wire is plugged into the port
- Correct IP address is set on the interface
- Correct route is set on the router

What to look for using ping tool:

- If all packets are replied;
- If all packets have approximately same round trip time (RTT) on non-congested Ethernet link

It is located here: *Tool -> Ping* menu. Fill in *Ping To* field and press start to initiate sending of ICMP packets.

Wireless

Wireless unnamed features in the guide that are good to know about. Configuration adjustments.

Channel frequencies and width

It is possible to choose different frequency, here are frequencies that can be used and channel width settings to use 40MHz HT channel (for 802.11n). For example, using *channel 1 or 2412MHz frequency* setting *20/40MHz HT below* will not yield any results, since there are no 20MHz channels available below set frequency.

Channel #	Frequency	Below	Above
1	2412 MHz	no	yes
2	2417 MHz	no	yes
3	2422 MHz	no	yes
4	2427 MHz	no	yes
5	2432 MHz	yes	yes
6	2437 MHz	yes	yes
7	2442 MHz	yes	yes
8	2447 MHz	yes	yes
9	2452 MHz	yes	yes
10	2457 MHz	yes	yes
11	2462 MHz	yes	no

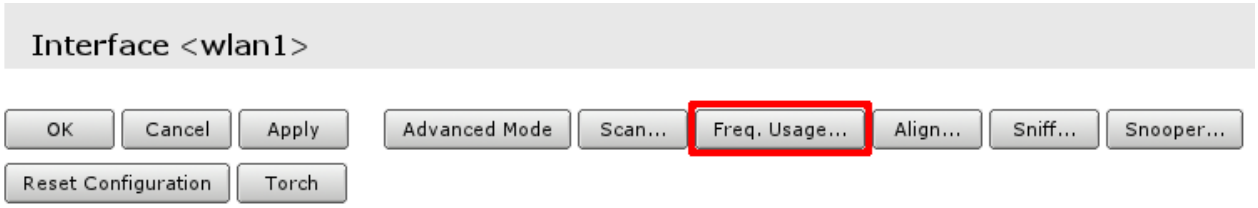
12	2467 MHz	yes	no
13	2472 MHz	yes	no

Warning: You should check how many and what frequencies you have in your regulatory domain before. If there are 10 or 11 channels adjust settings accordingly. With only 10 channels, channel #10 will have no sense of setting *20/40MHz HT above* since no full 20MHz channel is available

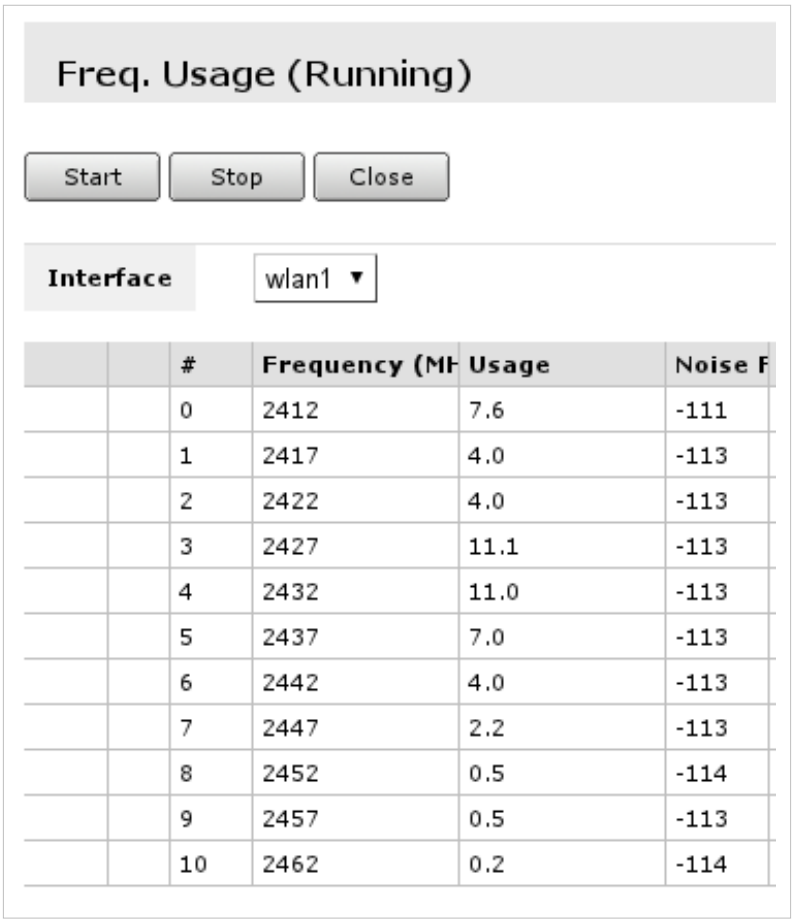
Wireless frequency usage

If wireless is not performing very well even when data rates are reported as being good, there might be that your neighbours are using same wireless channel as you are. To make sure follow these steps:

- Open frequency usage monitoring tool *Freq. Usage...* that is located in wireless interface details;



- Wait for some time as scan results are displayed. Do that for minute or two. Smaller numbers in *Usage* column means that channel is less crowded.



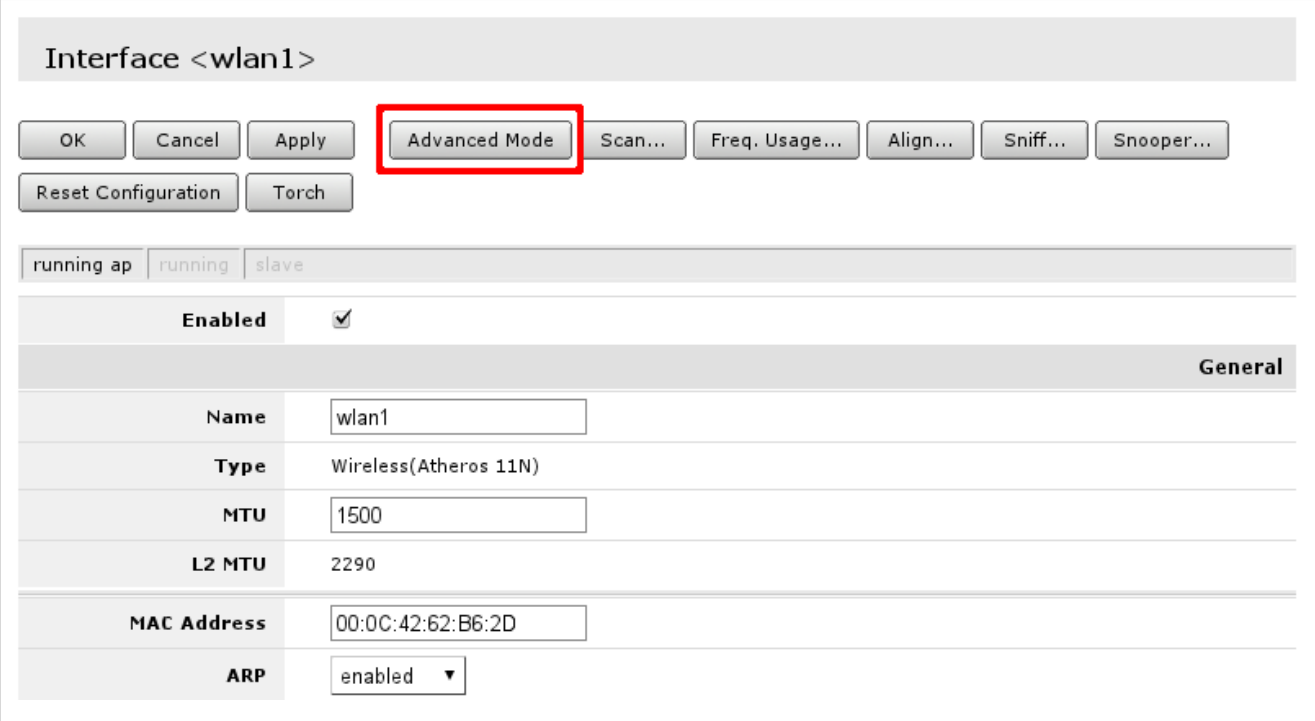
		#	Frequency (MH	Usage	Noise F
		0	2412	7.6	-111
		1	2417	4.0	-113
		2	2422	4.0	-113
		3	2427	11.1	-113
		4	2432	11.0	-113
		5	2437	7.0	-113
		6	2442	4.0	-113
		7	2447	2.2	-113
		8	2452	0.5	-114
		9	2457	0.5	-113
		10	2462	0.2	-114

Note: Monitoring is performed on default channels for *Country* selected in configuration. For example, if selected country would be Latvia, there would have been 13 frequencies listed as at that country have 13 channels allowed.

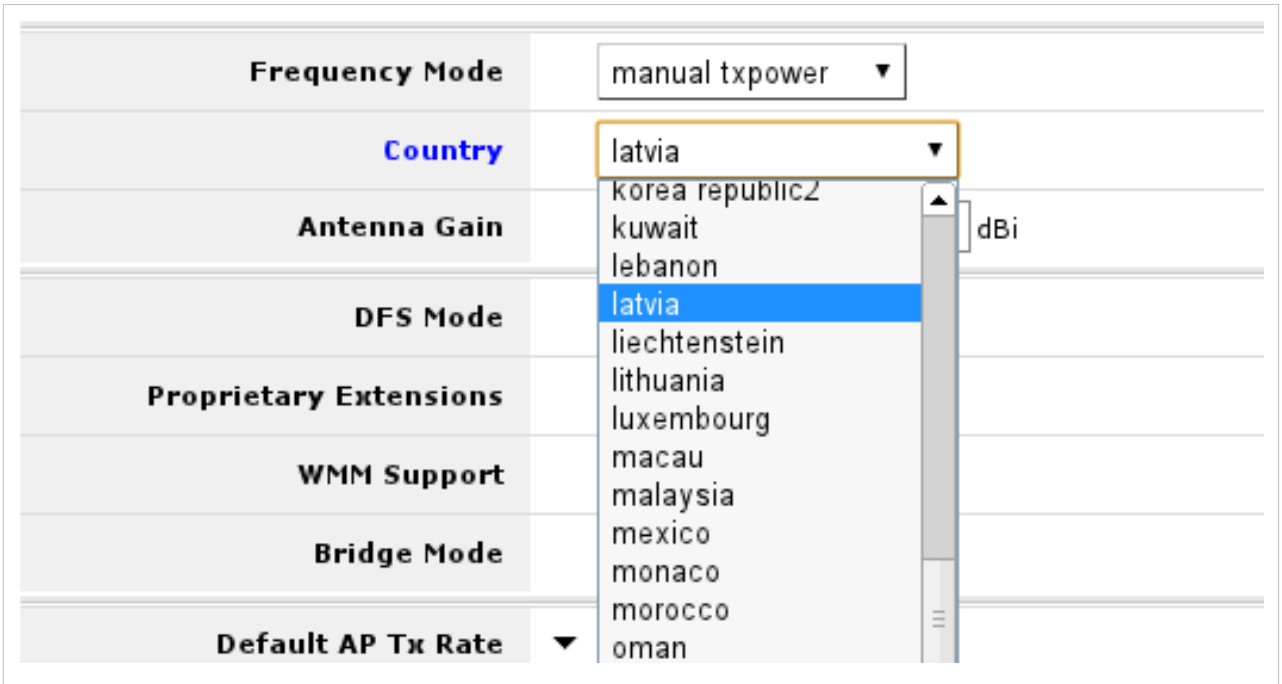
Change Country settings

By default *country* attribute in wireless settings is set to *no_country_set*. It is good practice to change this (if available) to change country you are in. To do that do the following:

- Go to wireless menu and select *Advanced mode*;



- Look up *Country* attribute and from drop-down menu select country



Note: Advanced mode is toggle button that changes from Simple to Advanced mode and back.

Port forwarding

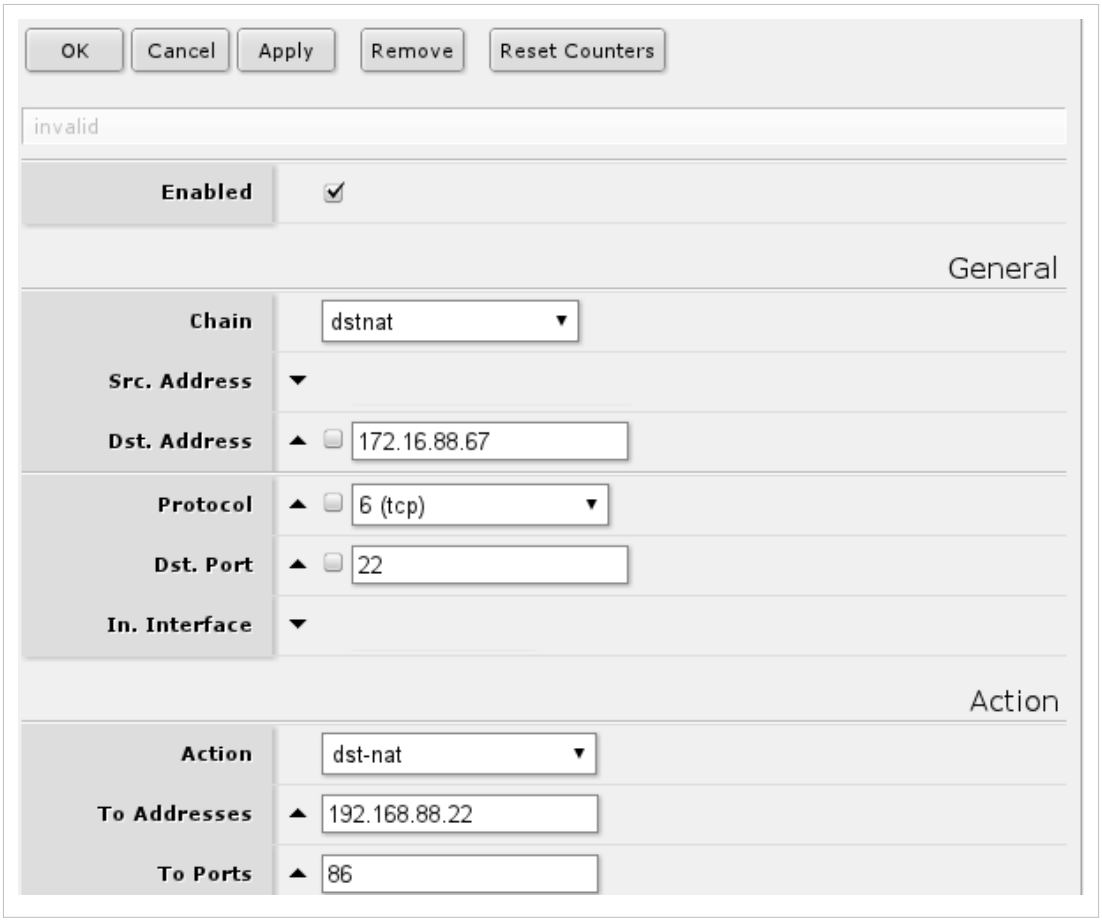
To make services on local servers/hosts available to general public it is possible to forward ports from outside to inside your NATed network, that is done from */ip firewall nat* menu. For example, to make possible for remote helpdesk to connect to your desktop and guide you, make your local file cache available for you when not at location etc.

Static configuration

A lot of users prefer to configure these rules statically, to have more control over what service is reachable from outside and what is not. This also has to be used when service you are using does not support dynamic configuration.

Following rule will forward all connections to port 22 on the router external ip address to port 86 on your local host with set IP address:

if you require other services to be accessible you can change protocol as required, but usually services are running TCP and dst-port. If change of port is not required, eg. remote service is 22 and local is also 22, then to-ports can be left unset.



Comparable command line command:

```
/ip firewall nat add chain=dstnat dst-address=172.16.88.67 protocol=tcp dst-port=22 \
action=dst-nat to-address=192.168.88.22 to-ports=86
```

Note: Screenshot contain only minimal set of settings are left visible

Dynamic configuration

uPnP is used to enable dynamic port forwarding configuration where service you are running can request router using uPnP to forward some ports for it.

Warning: Services you are not aware of can request port forwarding. That can compromise security of your local network, your host running the service and your data

Configuring uPnP service on the router:

- Set up what interfaces should be considered external and what internal;

```
/ip upnp interface add interface=ether1 type=external
/ip upnp interface add interface=ether2 type=internal
```

- Enable service itself

```
/ip upnp set allow-disable-external-interface=no show-dummy-rule=no enabled=yes
```

Limiting access to web pages

Using *IP -> Web Proxy* it is possible to limit access to unwanted web pages. This requires some understanding of use of WebFig interface.

Set up Web Proxy for page filtering

From *IP -> Web Proxy* menu *Access* tab open *Web Proxy Settings* and make sure that these attributes are set follows:

```
Enabled -> checked
Port -> 8080
Max. Cache Size -> none
Cache on disk -> unchecked
Parent proxy -> unset
```

When required alterations are done *apply*settings to return to *Access* tab.

Set up Access rules

This list will contain all the rules that are required to limit access to sites on the Internet.

To add sample rule to deny access to any host that contain example.com do the following when adding new entry:

```
Dst. Host -> .*example\.com.*
Action -> Deny
```

With this rule any host that has example.com will be unaccessible.

Limitation strategies

There are two main approaches to this problem

- deny only pages you know you want to deny *(A)*
- allow only certain pages and deny everything else *(B)*

For approach *A* each site that has to be denied is added with *Action* set to *Deny*

For approach *B* each site that has to be allowed should be added with *Action* set to *Allow* and in the end is rule, that matches everything with *Action* set to *Deny*.

[Top | Back to Content]

Manual:Internet access from VRF

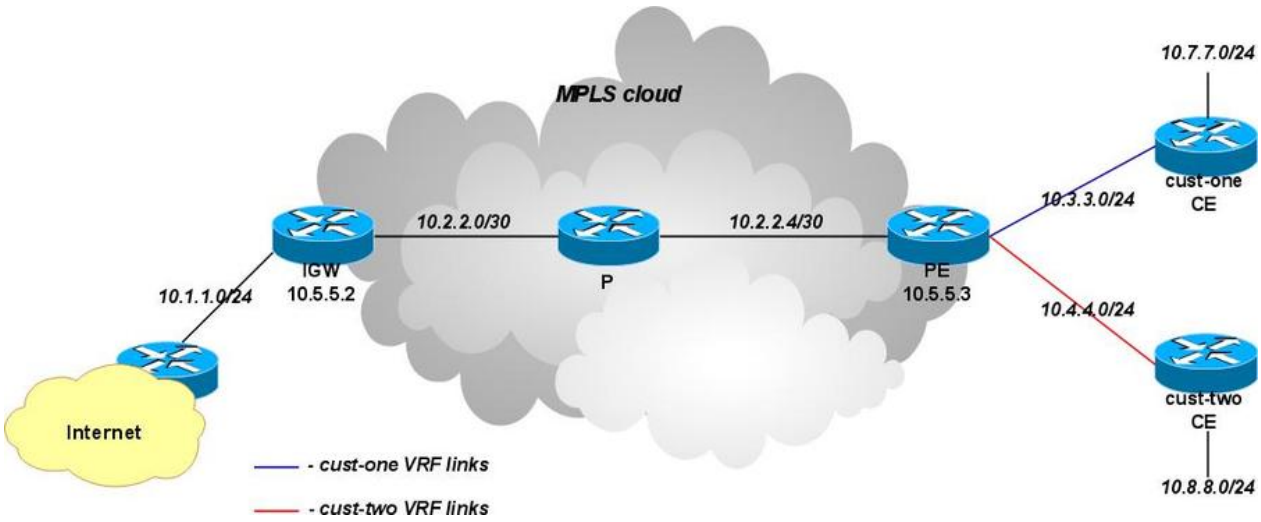
Description

Packages required: ***routing-test***, ***mpls-test***, RouterOS version 3.23+

There are multiple ways how Internet access could be provided to VRF clients. They are outlined in RFC 4364 section 11, for example.

Here we show the way how to configure access using global routing table.

Example



Default routes

Add default routes to VRF routing tables on PE:

```
/ip route add routing-mark=cust-one gateway=10.0.0.1@main
/ip route add routing-mark=cust-two gateway=10.0.0.1@main
```

Note that we must explicitly specify that the gateway should be resolved in the **@main** routing table, otherwise the routes will not become active.

Routes to client's networks

Routes to client's networks should be added in the main routing table, while their nexthops should be reachable via client's VRF interfaces, and as such belong to the VRF tables.

On the other hand, there is no way how to explicitly specify that gateway must be resolved in any other table, except the main table. So instead we specify the interface (which is in a VRF) and nexthop gateway address (which must be directly reachable on that interface). For point-to-point interfaces even the gateway address is not required.

Add these routes to PE's route table and redistribute them via OSPF:

```
/ip route add dst-address=10.7.7.0/24 gateway=10.3.3.4%ether2
/ip route add dst-address=10.8.8.0/24 gateway=10.4.4.5%ether3
```

And this is how should look in print:

```
[admin@PE2] > /ip route print detail where !routing-mark
 ...
 5 A S  dst-address=10.7.7.0/24 gateway=10.3.3.4 on cust-one reachable ether2
    distance=1 scope=30 target-scope=10

 6 A S  dst-address=10.8.8.0/24 gateway=10.4.4.5 on cust-two reachable ether3
     distance=1 scope=30 target-scope=10
```

Manual:Internet access from VRF with NAT

Packages required: ***routing-test, mpls-test***, RouterOS version 3.28+

MPLS Per-VRF NAT for internet access to L3VPNs

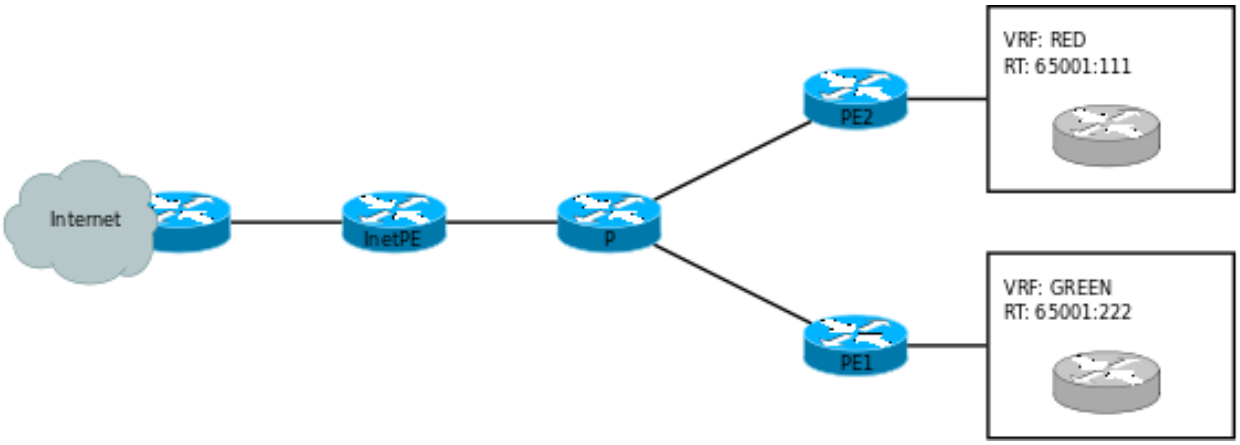
Abstract

This article will describe the basic configuration of how to provide internet access to L3VPN customers in an MPLS infrastructure. It has been tested with RouterOS version 3.28 with mpls-test and routing-test. This article assumes basic knowledge of MPLS operation as well as knowledge of NAT and routing.

Requirements

The concepts in this article requires at least one routable public ip address per VRF that needs to have internet access. It also requires you to have a dedicated PE-router to be placed between your internet-connected router and the MPLS network in order to do the actuall NAT translation before the data is transmitted to the internet-facing router. This article does not require you to have your own AS, although it may be convenient, just as long as you have the routable public IP addresses to spare for your customers.

Example topology



In this example topology we have two customers, RED and GREEN, who both reside in a separate VRF. Their LAN addressing is of no concern to this setup, and could possibly overlap. They receive internet access on the InetPE router. This design is not an actual MPLS network, but just a simple illustration of the basic concept.

InetPE configuration

We assume that the example network here has a public network of 1.1.1.0/24. The link between the InetPE and the actual internet gateway is 1.1.1.0/30, and 1.1.1.16/28 is assigned for VRFs terminating here. A default route to the internet gateway exists on the InetPE in some form, pointing to 1.1.1.1, and 1.1.1.1 should have a route to 1.1.1.16/28 via 1.1.1.2 (the InetPE).

VRF configuration

The VRFs are configured like this:

```
/ip route vrf add routing-mark=RED route-distinguisher=65001:111 import-route-targets=65001:111 \
  export-route-targets=65001:111 disabled=no

/ip route vrf add routing-mark=GREEN route-distinguisher=65001:222 import-route-targets=65001:222 \
  export-route-targets=65001:222 disabled=no

/routing bgp instance vrf add routing-mark=RED redistribute-static=yes instance=default

/routing bgp instance vrf add routing-mark=GREEN redistribute-static=yes instance=default
```

Default Route

To add a default route, the following commands should be used:

```
/ip route add routing-mark=RED dst-address=0.0.0.0/0 gateway=1.1.1.1@main
/ip route add routing-mark=GREEN dst-address=0.0.0.0/0 gateway=1.1.1.1@main
```

Notice the @main part. This indicates that the address 1.1.1.1 should be resolved on the main routing table instead of inside the VRF routing table.

NAT

In this step, we will source NAT the traffic from the RED VRF to the address 1.1.1.16 and the GREEN VRF to 1.1.1.17. This requires both a NAT entry and a MANGLE entry, since the return traffic does not automatically go back into the correct VRF.

- NAT:

```
/ip firewall nat add action=src-nat chain=srcnat out-interface=ether1 routing-mark=RED \
  to-addresses=1.1.1.16 disabled=no

/ip firewall nat add action=src-nat chain=srcnat out-interface=ether1 routing-mark=GREEN \
  to-addresses=1.1.1.17 disabled=no
```

- MANGLE:

```
/ip firewall mangle add chain=prerouting action=mark-routing disabled=no dst-addres=1.1.1.16 \
  new-routing-mark=RED passthrough=yes

/ip firewall mangle add chain=prerouting action=mark-routing disabled=no dst-address=1.1.1.17 \
  new-routing-mark=GREEN passthrough=yes
```

Further design considerations

There are several ways you could enhance this design. You could consider breaking out internet access to a separate RT, allowing you a little more flexibility with the internet routes. I have yet to find out how to do this without a default route that points to a next-hop router, eg. if you want to terminate the VRF's on a box with full BGP feed. If you discover, please update this Wiki article.

Conclusion

This configuration is enough to get simple src-nat working. You may want to fine tune these rules to fit into your setup. Dst-nat is not covered by this guide, but should be simple to set up as long as you remember to set up the corresponding mangle rules. It has not been thoroughly tested, so I cannot say what kind of performance you might expect from this.

Manual:IP/Hotspot/Profile

Applies to RouterOS: v3, v4, v5+

Summary

Sub-menu: `/ip hotspot profile`

This submenu contains list of Hotspot server profiles. There may be various different HotSpot systems, defined as HotSpot Server Profiles, on the same gateway machine. One or more interfaces can be grouped into one server profile. There are very few settings for the servers on particular interfaces - most of the configuration is set in the server profiles. For example, it is possible to make completely different set of servlet pages for each server profile, and define different RADIUS servers for authentication.

Properties

Property	Description
dns-name (*string*; Default: **""**)	DNS name of the HotSpot server (it appears as the location of the login page). This name will automatically be added as a static DNS entry in the DNS cache.
hotspot-address (*IP*; Default: **0.0.0.0**)	IP address of HotSpot service.
html-directory (*string*; Default: **hotspot**)	Directory name in which HotSpot HTML pages are stored (by default *hotspot* directory). It is possible to specify different directory with modified HTML pages. To change HotSpot login page, connect to the router with FTP and download hotspot directory contents. `Read more >>`
http-cookie-lifetime (*time*; Default: **3d**)	HTTP cookie validity time, the option is related to *cookie* HotSpot login method
http-proxy (*IP:Port*; Default: **0.0.0.0:0**)	Address and port of the proxy server for HotSpot service, when default value is used all request are resolved by the local /ip proxy

login-by (*cookie|http-chap|http-pap|https|mac|trial*; Default: **http-chap, cookie**)

Used HotSpot authentication method

- **cookie** - may only be used with other HTTP authentication method. HTTP cookie is generated, when user authenticates in HotSpot for the first time. User is not asked for the login/password and authenticated automatically, until *cookie-lifetime* is active
- **http-chap** - login/password is required for the user to authenticate in HotSpot. CHAP challenge-response method with MD5 hashing algorithm is used for protecting passwords.
- **http-pap** - login/password is required for user to authenticate in HotSpot. Username and password are sent over network in plain text.
- **https** - login/password is required for user to authenticate in HotSpot. Client login/password exchange between client and server is encrypted with SSL tunnel
- **mac** - client is authenticated without asking login form. Client MAC-address is added to /ip hotspot user database, client is authenticated as soon as connected to the HotSpot
- **trial** - client is allowed to use internet without HotSpot login for the specified amount of time

mac-auth-password (*string*; Default:)

Used together with MAC authentication, field used to specify password for the users to be authenticated by their MAC addresses. The following option is required, when specific RADIUS server rejects authentication for the clients with blank password

name (*string*; Default:)

Descriptive name of the profile

nas-port-type (*string*; Default: **wireless-802.11**)

NAS-Port-Type value to be sent to RADIUS server, NAS-Port-Type values are described in the RADIUS RFC 2865. This optional value attribute indicates the type of the physical port of the HotSpot server.

radius-accounting (*yes | no*; Default: **yes**)

Send RADIUS server accounting information for each user, when yes is used

radius-default-domain (*string*; Default:)

Default domain to use for RADIUS requests. Allows to use separate RADIUS server per */ip hotspot profile*

radius-interim-update (*time* | *received*; Default: **received**)

How often to send accounting updates . When *received* is set, interim-time is used from RADIUS server. **0s** is the same as *received*.

radius-location-name (*string*; Default:)

RADIUS-Location-Id to be sent to RADIUS server. Used to identify location of the HotSpot server during the communication with RADIUS server. Value is optional and used together with RADIUS server.

radius-mac-format (*"XX XX XX XX XX XX"|XX:XX:XX:XX:XX:XX|XXXXXX-XXXXXX|XXXXXXXXXXXX|XX-XX-XX-XX-XX-XX|XXXX:XXXX:XXXX|XXXXXX:XXXXXX*; Default: **XX:XX:XX:XX:XX:XX**)

rate-limit (*string*; Default: **""**)

Rate limitation in form of **rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time]]]] [priority] [rx-rate-min[/tx-rate-min]]** from the point of view of the router (so "rx" is client upload, and "tx" is client download). All rates should be numbers with optional 'k' (1,000s) or 'M' (1,000,000s). If tx-rate is not specified, rx-rate is as tx-rate too. Same goes for tx-burst-rate and tx-burst-threshold and tx-burst-time. If both rx-burst-threshold and tx-burst-threshold are not specified (but burst-rate is specified), rx-rate and tx-rate is used as burst thresholds. If both rx-burst-time and tx-burst-time are not specified, 1s is used as default. rx-rate-min and tx-rate min are the values of limit-at properties

smtp-server (*IP*; Default: **0.0.0.0**)

SMTP server address to be used to redirect HotSpot users SMTP requests.

split-user-domain (*yes* | *no*; Default: **no**)

Split username from domain name when the username is given in "user@domain" or in "domain\user" format from RADIUS server

ssl-certificate (*string* | *none*; Default: **none**)

Name of the SSL certificate on the router to to use only for HTTPS authentication.

trial-uptime (*time/time*; Default: **30m/1d**)

Used only with *trial* authentication method. First time value specifies, how long trial user identified by MAC address can use access to public networks without HotSpot authentication. Second time value specifies amount of time, that has to pass until user is allowed to use trial again.

trial-user-profile (*string*; Default: **default**)

Specifies hotspot user profile for trial users.

use-radius (*yes* | *no*; Default: **no**) Use RADIUS to authenticate HotSpot users.

[Top | Back to Content]

Manual:IP/IPsec

Applies to RouterOS: v5.0 +

Summary

Sub-menu: `/ip ipsec`
Package required: `security`
Standards: `RFC 4301`

Internet Protocol Security (IPsec) is a set of protocols defined by the Internet Engineering Task Force (IETF) to secure packet exchange over unprotected IP/IPv6 networks such as Internet.

IpSec protocol suite can be divided in following groups:

- **Authentication Header (AH)** RFC 4302
- **Encapsulating Security Payload (ESP)** RFC 4303
- **Internet Key Exchange (IKE)** protocols. Dynamically generates and distributes cryptographic keys for AH and ESP.

Authentication Header (AH)

AH is a protocol that provides authentication of either all or part of the contents of a datagram through the addition of a header that is calculated based on the values in the datagram. What parts of the datagram are used for the calculation, and the placement of the header, depends whether tunnel or transport mode is used.

The presence of the AH header allows to verify the integrity of the message, but doesn't encrypt it. Thus, AH provides authentication but not privacy (Another protocol ESP is used to provide encryption).

RouterOS supports the following authentication algorithms for AH:

- SHA1
- MD5

Transport mode

In transport mode AH header is inserted after IP header. IP data and header is used to calculate authentication value. IP fields that might change during transit, like TTL and hop count, are set to zero values before authentication.

Tunnel mode

In tunnel mode original IP packet is encapsulated within a new IP packet. All of the original IP packet is authenticated.

Encapsulating Security Payload

Encapsulating Security Payload (ESP) uses shared key encryption to provide data privacy. ESP also supports its own authentication scheme like that used in AH, or can be used in conjunction with AH.

ESP packages its fields in a very different way than AH. Instead of having just a header, it divides its fields into three components:

- **ESP Header** - Comes before the encrypted data and its placement depends on whether ESP is used in transport mode or tunnel mode.
- **ESP Trailer** - This section is placed after the encrypted data. It contains padding that is used to align the encrypted data.
- **ESP Authentication Data** - This field contains an Integrity Check Value (ICV), computed in a manner similar to how the AH protocol works, for when ESP's optional authentication feature is used.

Transport mode

In transport mode ESP header is inserted after original IP header. ESP trailer and authentication value is added to the end of the packet. In this mode only IP payload is encrypted and authenticated, IP header is not secured.

Tunnel mode

In tunnel mode original IP packet is encapsulated within a new IP packet thus securing IP payload and IP header.

Encryption algorithms

RouterOS ESP supports various encryption and authentication algorithms.

Authentication:

- SHA1
- MD5

Encryption:

- **DES** - 56-bit DES-CBC encryption algorithm;
- **3DES** - 168-bit DES encryption algorithm;
- **AES** - 128, 192 and 256-bit key AES-CBC encryption algorithm;
- **Blowfish** - added since v4.5
- **Twofish** - added since v4.5
- **Camellia** - 128, 192 and 256-bit key Camellia encryption algorithm added since v4.5

Hardware encryption

Hardware encryption allows to do faster encryption process by using built-in encryption engine inside CPU. AES is the only algorithm that will be accelerated in hardware.

List of RouterBoards with enabled hardware support:

- RB1000
- RB1100AHx2

For comparison RB1000 with enabled HW support can forward up to 550Mbps encrypted traffic. When HW support is disabled it can forward only 150Mbps encrypted traffic in AES-128 mode.

Some configuration advices on how to get maximum ipsec throughput on multicore RB1100AHx2:

- Avoid using ether12 and ethet13. Since these prots are pci-x they will be slowest ones.
- Fastest forwarding is from switch chip ports (ether1-ether10) to ether11 (directly connected to CPU) and vice versa.
- Set hardware queue on all interfaces

```
/queue interface set [find] set queue=only-hardware-queue
```

- Disable RPS:

```
/system resource irq rps disable [find]
```

- Assign one CPU core to ether11 and other CPU core to everything else. Forwarding over ether11 requires more CPU that is why we are giving one core only for that interface (in IRQ setting ether11 is listed as ether12 tx,rx and error).

```
/system resource irq
set [find] cpu=1
set [find users="eth12 tx"] cpu=0
set [find users="eth12 rx"] cpu=0
set [find users="eth12 error"] cpu=0
```

- disable connection tracking

With all above recommendations it is possible to forward 820Mbps (1470byte packets two streams).

With enabled connection tracking 700Mbps (1470 byte packets two streams).

Internet Key Exchange Protocol

The Internet Key Exchange (IKE) is a protocol that provides authenticated keying material for Internet Security Association and Key Management Protocol (ISAKMP) framework. There are other key exchange schemes that work with ISAKMP, but IKE is the most widely used one. Together they provide means for authentication of hosts and automatic management of security associations (SA).

Most of the time IKE daemon is doing nothing. There are two possible situations when it is activated:

There is some traffic caught by a policy rule which needs to become encrypted or authenticated, but the policy doesn't have any SAs. The policy notifies IKE daemon about that, and IKE daemon initiates connection to remote host. IKE daemon responds to remote connection. In both cases, peers establish connection and execute 2 phases:

- **Phase 1** - The peers agree upon algorithms they will use in the following IKE messages and authenticate. The keying material used to derive keys for all SAs and to protect following ISAKMP exchanges between hosts is generated also. This phase should match following settings:
  - authentication method

- DH group
- encryption algorithm
- exchange mode
- hash alorithm
- NAT-T
- DPD and lifetime (optional)

- **Phase 2** - The peers establish one or more SAs that will be used by IPsec to encrypt data. All SAs established by IKE daemon will have lifetime values (either limiting time, after which SA will become invalid, or amount of data that can be encrypted by this SA, or both). This phase should match following settings:
  - Ipsec protocol
  - mode (tunnel or transport)
  - authentication method
  - PFS (DH) group
  - lifetime

Note: There are two lifetime values - soft and hard. When SA reaches it's soft lifetime treshold, the IKE daemon receives a notice and starts another phase 2 exchange to replace this SA with fresh one. If SA reaches hard lifetime, it is discarded.

IKE can optionally provide a Perfect Forward Secrecy (PFS), which is a property of key exchanges, that, in turn, means for IKE that compromising the long term phase 1 key will not allow to easily gain access to all IPsec data that is protected by SAs established through this phase 1. It means an additional keying material is generated for each phase 2.

Generation of keying material is computationally very expensive. Exempli gratia, the use of modp8192 group can take several seconds even on very fast computer. It usually takes place once per phase 1 exchange, which happens only once between any host pair and then is kept for long time. PFS adds this expensive operation also to each phase 2 exchange.

Diffie-Hellman Groups

Diffie-Hellman (DH) key exchange protocol allows two parties without any initial shared secret to create one securely. The following Modular Exponential (MODP) and Elliptic Curve (EC2N) Diffie-Hellman (also known as "Oakley") Groups are supported:

Diffie-Hellman Group	Name	Reference
Group 1	768 bit MODP group	RFC 2409
Group 2	1024 bits MODP group	RFC 2409
Group 3	EC2N group on GP(2^155)	RFC 2409
Group 4	EC2N group on GP(2^185)	RFC 2409
Group 5	1536 bits MODP group	RFC 3526

IKE Traffic

To avoid problems with IKE packets hit some SPD rule and require to encrypt it with not yet established SA (that this packet perhaps is trying to establish), locally originated packets with UDP source port 500 are not processed with SPD. The same way packets with UDP destination port 500 that are to be delivered locally are not processed in incoming policy check.

Setup Procedure

To get IPsec to work with automatic keying using IKE-ISAKMP you will have to configure policy, peer and proposal (optional) entries.

Warning: Ipsec is very sensitive to time changes. If both ends of the IpSec tunnel are not synchronizing time equally(for example, different NTP servers not updating time with the same timestamp), tunnels will break and will have to be established again.

Peer configuration

Sub-menu: `/ip ipsec peer`

Peer configuration settings are used to establish connections between IKE daemons (**phase 1** configuration). This connection then will be used to negotiate keys and algorithms for SAs.

Property	Description
address (*IP/IPv6 Prefix*; Default: **0.0.0.0/0**)	If remote peer's address matches this prefix, then the peer configuration is used in authentication and establishment of **Phase 1**. If several peer's addresses match several configuration entries, the most specific one (i.e. the one with largest netmask) will be used.
auth-method (*pre-shared-key* | *rsa-signature*; Default: **pre-shared-key**)	Authentication method: • `pre-shared-key` - authenticate by a password (secret) string shared between the peers • `rsa-signature` - authenticate using a pair of RSA certificates • `rsa-key` - authenticate using a RSA key imported in Ipsec key menu.
certificate (*string*; Default:)	Name of a certificate listed in certificate table (signing packets; the certificate must have private key). Applicable if RSA signature authentication method (auth-method=rsa-signature) is used.
comment (*string*; Default:)	Short description of the peer.
dh-group (*ec2n155* | *ec2n185* | *modp1024* | *modp1536* | *modp2048* | *modp3072* | *modp4096* | *modp6144* | *modp768*; Default: **modp1024**)	Diffie-Hellman group (cipher strength)
disabled (*yes* | *no*; Default: **no**)	Whether peer is used to match remote peer's prefix.
dpd-interval (*time* | *disable-dpd*; Default: **2m**)	Dead peer detection interval. If set to `disable-dpd`, dead peer detection will not be used.
dpd-maximum-failures (*integer: 1..100*; Default: **5**)	Maximum count of failures until peer is considered to be dead. Applicable if DPD is enabled.
enc-algorithm (*3des* | *aes-128* | *aes-192* | *aes-256* | *blowfish* | *camellia-128* | *camellia-192* | *camellia-256* | *des*; Default: **3des**)	Encryption algorithm.
exchange-mode (*aggressive* | *base* | *main* | *main-l2tp*; Default: **main**)	Different ISAKMP phase 1 exchange modes according to RFC 2408. Do not use other modes then main unless you know what you are doing. **main-l2tp** mode relaxes rfc2409 section 5.4, to allow pre-shared-key authentication in main mode.

generate-policy (*yes* | *no*; Default: **no**)	Allow this peer to establish SA for non-existing policies. Such policies are created dynamically for the lifetime of SA. Automatic policies allows, for example, to create IPsec secured L2TP tunnels, or any other setup where remote peer's IP address is not known at the configuration time.
hash-algorithm (*md5* | *sha1*; Default: **md5**)	Hashing algorithm. SHA (Secure Hash Algorithm) is stronger, but slower.
key (*string*; Default:)	Name of the key from key menu. Applicable if auth-method=rsa-key.
lifebytes (*Integer: 0..4294967295*; Default: **0**)	Phase 1 lifetime: specifies how much bytes can be transferred before SA is discarded. If set to **0**, SA will not be discarded due to byte count excess.
lifetime (*time*; Default: **1d**)	Phase 1 lifetime: specifies how long the SA will be valid.
my-id-user-fqdn (*string*; Default:)	By default IP address is used as ID. This parameter replaces ID with specified value. Can be used, for example, in cases if DNS name as ID is required.
nat-traversal (*yes* | *no*; Default: **no**)	Use Linux NAT-T mechanism to solve IPsec incompatibility with NAT routers inbetween IPsec peers. This can only be used with ESP protocol (AH is not supported by design, as it signs the complete packet, including IP header, which is changed by NAT, rendering AH signature invalid). The method encapsulates IPsec ESP traffic into UDP streams in order to overcome some minor issues that made ESP incompatible with NAT.
port (*integer:0..65535*; Default: **500**)	Communication port used for ipsec traffic.
proposal-check (*claim* | *exact* | *obey* | *strict*; Default: **obey**)	Phase 2 lifetime check logic: • `claim` - take shortest of proposed and configured lifetimes and notify initiator about it • `exact` - require lifetimes to be the same • `obey` - accept whatever is sent by an initiator • `strict` - if proposed lifetime is longer than the default then reject proposal otherwise accept proposed lifetime
remote-certificate (*string*; Default:)	Name of a certificate (listed in certificate table) for authenticating the remote side (validating packets; no private key required). Applicable if RSA signature authentication method is used
secret (*string*; Default:)	Secret string (in case pre-shared key authentication is used). If it starts with '0x', it is parsed as a hexadecimal value
send-initial-contact (*yes* | *no*; Default: **yes**)	Specifies whether to send initial IKE information or wait for remote side.

Note: IPSec phases information is erased, when /ip ipsec peer configuration is modified on the fly, however packets are being encrypted/decrypted because of installed-sa (for example remote-peers information is erased, when peer configuration is modified.

Keys

Sub-menu: `/ip ipsec key`

This submenu list all imported public/private keys, that can be used for peer authentication. Submenu also has several commands to work with keys.

For example print below shows two imported 1024-bit keys, one public and one private.

```
[admin@PoETik] /ip ipsec key> print
Flags: P - private-key, R - rsa
 #    NAME                                                                KEY-SIZE
 0 PR priv                                                                1024-bit
 1  R pub                                                                 1024-bit
```

Commands

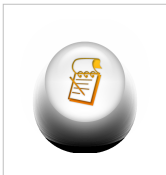
Property	Description
export-pub-key (*file-name; key*)	Export public key to file from one of existing private keys.
generate-key (*key-size; name*)	Generate private key. Takes two parameters, name of newly generated key and key size 1024,2048 and 4096.
import (*file-name; name*)	Import key from file.

Policy

Sub-menu: `/ip ipsec policy`

Policy table is used to determine whether security settings should be applied to a packet.

Property	Description
action (*discard* | *encrypt* | *none*; Default: **encrypt**)	Specifies what to do with packet matched by the policy. • `none` - pass the packet unchanged • `discard` - drop the packet • `encrypt` - apply transformations specified in this policy and it's SA
comment (*string*; Default:)	Short description of the policy
disabled (*yes* | *no*; Default: **no**)	Whether policy is used to match packets.
dst-address (*IP/IPv6 prefix*; Default: **0.0.0.0/32**)	Destination address to be matched in packets.
dst-port (*integer:0..65535* | *any*; Default: **any**)	Destination port to be matched in packets. If set to **any** all ports will be matched
ipsec-protocols (*ah* | *esp*; Default: **esp**)	Specifies what combination of Authentication Header and Encapsulating Security Payload protocols you want to apply to matched traffic
level (*require* | *unique* | *use*; Default: **require**)	Specifies what to do if some of the SAs for this policy cannot be found: • `use` - skip this transform, do not drop packet and do not acquire SA from IKE daemon • `require` - drop packet and acquire SA • `unique` - drop packet and acquire a unique SA that is only used with this particular policy
manual-sa (*string* | *none*; Default: **none**)	Name of the manual SA template
priority (*integer:-2147483646..2147483647*; Default: **0**)	Policy ordering classificator (signed integer). Larger number means higher priority.
proposal (*string*; Default: **default**)	Name of the proposal template that will be sent by IKE daemon to establish SAs for this policy.
protocol (*all* | *egp* | *ggp*| *icmp* | *igmp* | ...; Default: **all**)	IP packet protocol to match.
sa-dst-address (*ip/ipv6 address*; Default: **::**)	SA destination IP/IPv6 address (remote peer).
sa-src-address (*ip/ipv6 address*; Default: **::**)	SA source IP/IPv6 address (local peer).
src-address (*ip/ipv6 prefix*; Default: **0.0.0.0/32**)	Source IP prefix
src-port (*any* | *integer:0..65535*; Default: **any**)	Source Port of the packet
tunnel (*yes* | *no*; Default: **no**)	Specifies whether to use tunnel mode



Note: All packets are IPIP encapsulated in tunnel mode, and their new IP header's src-address and dst-address are set to sa-src-address and sa-dst-address values of this policy. If you do not use tunnel mode (id est you use transport mode), then only packets whose source and destination addresses are the same as sa-src-address and sa-dst-address can be processed by this policy. Transport mode can only work with packets that originate at and are destined for IPsec peers (hosts that established security associations). To encrypt traffic between networks (or a network and a host) you have to use tunnel mode.

Policy Stats

Command **`/ip ipsec policy print stats`** will show current status of the policy. Additional read-only parameters will be printed.

Property	Description
`in-accepted` (*integer*)	How many incoming packets were passed by the policy without an attempt to decrypt.
`in-dropped` (*integer*)	How many incoming packets were dropped by the policy without an attempt to decrypt
`in-transformed` (*integer*)	How many incoming packets were decrypted (ESP) and/or verified (AH) by the policy
`out-accepted` (*integer*)	How many outgoing packets were passed by the policy without an attempt to encrypt.
`out-dropped` (*integer*)	How many outgoing packets were dropped by the policy without an attempt to encrypt.
`out-transformed` (*integer*)	How many outgoing packets were encrypted (ESP) and/or verified (AH) by the policy.
`ph2-state` (*expired* | *no-phase2* | *established*)	Indication of the progress of key establishing.

Dumping Policies

It is possible to dump policies installed into the kernel for debugging purposes with command:

```
/ip ipsec policy  dump-kernel-policies
```

After executing this command check the logs to see the result, there should be three policies in the kernel: forward, in and out.

```
[admin@test-host] >/log print
07:28:34 ipsec,debug,packet policy ipsec fwd: 10.5.101.9[0] - 10.5.101.13[0]
07:28:34 ipsec,debug,packet policy ipsec in: 10.5.101.9[0] - 10.5.101.13[0]
07:28:34 ipsec,debug,packet policy ipsec out: 10.5.101.13[0] - 10.5.101.9[0]
```

Proposal settings

Sub-menu: `/ip ipsec proposal`

Proposal information that will be sent by IKE daemon to establish SAs for this policy (**Phase 2**). Configured proposals are set in policy configuration.

Property	Description
auth-algorithms (*md5|sha1|null*; Default: **sha1**)	Allowed algorithms for authorization. sha1 is stronger, but slower algorithm.
comment (*string*; Default:)	Short description of an item.
disabled (*yes* | *no*; Default: **no**)	Whether item is disabled.
enc-algorithms (*null|des|3des|aes-128|aes-192|aes-256|blowfish|camellia-128|camellia-192|camellia-256|twofish*; Default: **3des**)	Allowed algorithms and key lengths to use for SAs.
lifetime (*time*; Default: **30m**)	How long to use SA before throwing it out.
name (*string*; Default:)	Name of the proposal template, that will be identified in other parts of ipsec configuration.
pfs-group (*ec2n155* | *ec2n185* | *modp1024* | *modp1536* | *modp2048* | *modp3072* | *modp4096* | *modp6144* | *modp768* | *none*; Default: **modp1024**)	Diffie-Helman group used for Perfect Forward Secrecy.

Manual SA

Sub-menu: `/ip ipsec manual-sa`

Menu is used to configure SAs manually. Created SA template then can be used in policy configuration.

Property	Description
ah-algorithm (*in/out* *in,out = md5|null|sha1*; Default: **null**)	Authentication Header encryption algorithm.
ah-key (*string/string*; Default:)	Incoming-authentication-key/outgoing-authentication-key
ah-spi (*0x100..FFFFFFFF/0x100..FFFFFFFF*; Default: **0x100**)	Incoming-SA-SPI/outgoing-SA-SPI
disabled (*yes* | *no*; Default: **no**)	Defines whether item is ignored or used
esp-auth-algorithm (*in/out* *in,out = md5|null|sha1*; Default: **null**)	Encapsulating Security Payload authentication encryption algorithm
esp-auth-key (*string/string*; Default:)	Incoming-authentication-key/outgoing -authentication-key
esp-enc-algorithm (*in/out* *in,out = 3des* | *aes-128* | *aes-192* | *aes-256* | *des* | ...; Default: **null**)	Incoming-encryption-algorithm
esp-enc-key (*string/string*; Default:)	Incoming-encryption-key/outgoing-encryption-key
esp-spi (*0x100..FFFFFFFF/0x100..FFFFFFFF*; Default: **0x100**)	Incoming-SA-SPI/outgoing-SA-SPI
lifetime (*time*; Default: **0s**)	Lifetime of this SA
name (*string*; Default:)	Name of the item for reference from policies

Installed SA

Sub-menu: `/ip ipsec installed-sa`

This facility provides information about installed security associations including the keys.

Property	Description
AH (*yes* | *no*)	
ESP (*yes* | *no*)	
add-lifetime (*time/time*)	Added lifetime for the SA in format soft/hard • soft - time period after which ike will try to establish new SA • hard - time period after which SA is deleted
addtime (*time*)	Date and time when this SA was added.
auth-algorithm (*sha1* | *md5*)	Shows currently used authentication algorithm
auth-key (*string*)	Shows used authentication key
current-bytes (*integer*)	Shows number of bytes seen by this SA.
dst-address (*IP*)	
enc-algorithm (*des* | *3des* | *aes ...*)	Shows currently used encryption algorithm
pfs (*yes* | *no*)	
replay (*integer*)	
spi (*string*)	
src-address (*IP*)	
state (*string*)	Shows the current state of the SA ("mature", "dying" etc)

Flushing SAs

Sometimes after incorrect/incomplete negotiations took place, it is required to flush manually the installed SA table so that SA could be renegotiated. This option is provided by the `/ip ipsec installed-sa flush` command.

This command accepts only one property:

Property	Description
sa-type (*ah* | *all* | *esp*; Default: **all**)	Specifies SA types to flush: • `ah` - delete AH protocol SAs only • `esp` - delete ESP protocol SAs only • `all` - delete both ESP and AH protocols SAs

Remote Peers

Sub-menu: `/ip ipsec remote-peers`

This submenu provides you with various statistics about remote peers that currently have established phase 1 connections with this router. Note that if peer doesn't show up here, it doesn't mean that no IPsec traffic is being exchanged with it.

Read only properties:

Property	Description
local-address (*ip/ipv6 address*)	Local ISAKMP SA address on the router used by the peer
remote-address (*ip/ipv6 address*)	Remote peer's ip/ipv6 address
side (*initiator | responder*)	Shows which side initiated the Phase1 negotiation.
state (*string*)	State of phase 1 negotiation with the peer. For example when phase1 and phase 2 are negotiated it will show state "established".
established (*time*)	How long peers are in established state.

Closing all IPsec connections

Menu has a command to quickly close all established ipsec connections. This command will clear all installed SAs (Phase2) and remove all entries from remote-peers menu (Phase1).

Usage:

```
/ip ipsec remote-peers kill-connections
```

Statistics

Sub-menu: `/ip ipsec statistics`

This menu shows various ipsec statistics

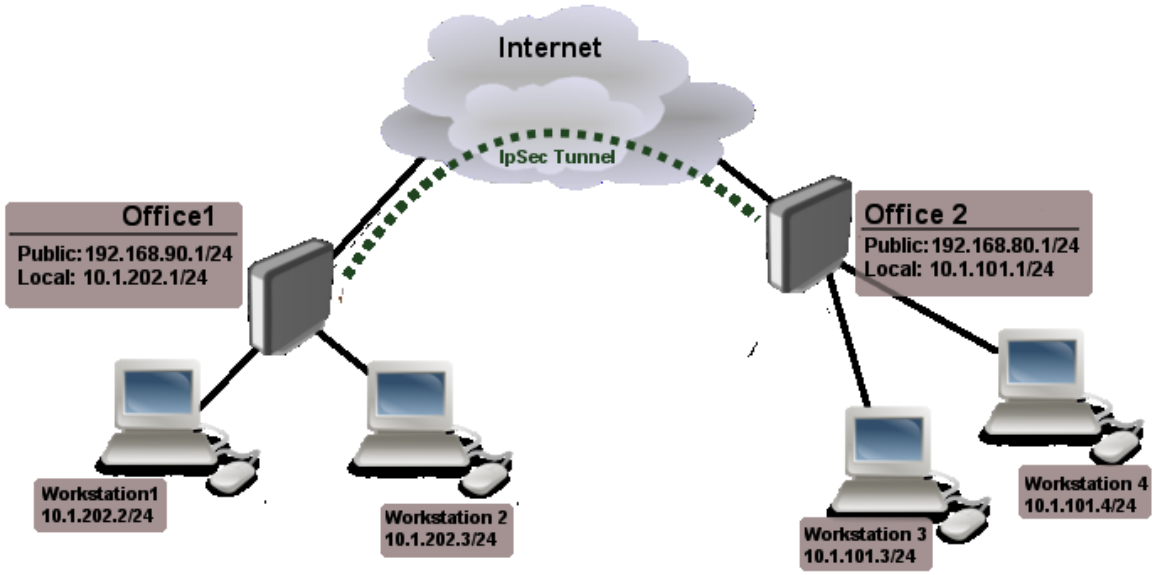
Property	Description
in-errors (*integer*)	All inbound errors that are not matched by other counters.
in-buffer-errors (*integer*)	No free buffer.
in-header-errors (*integer*)	Header error
in-no-states (*integer*)	No state is found i.e. Either inbound SPI, address, or IPsec protocol at SA is wrong
in-state-protocol-errors (*integer*)	Transformation protocol specific error, for example SA key is wrong or hardware accelerator is unable to handle amount of packets.
in-state-mode-errors (*integer*)	Transformation mode specific error
in-state-sequence-errors (*integer*)	Sequence number is out of window
in-state-expired (*integer*)	State is expired
in-state-mismatches (*integer*)	State has mismatched option, for example UDP encapsulation type is mismatched.
in-state-invalid (*integer*)	State is invalid
in-template-mismatches (*integer*)	No matching template for states, e.g. Inbound SAs are correct but SP rule is wrong
in-no-policies (*integer*)	No policy is found for states, e.g. Inbound SAs are correct but no SP is found

in-policy-blocked (*integer*)	Policy discards
in-policy-errors (*integer*)	Policy errors
out-errors (*integer*)	All outbound errors that are not matched by other counters
out-bundle-errors (*integer*)	Bundle generation error
out-bundle-check-errors (*integer*)	Bundle check error
out-no-states (*integer*)	No state is found
out-state-protocol-errors (*integer*)	Transformation protocol specific error
out-state-mode-errors (*integer*)	Transformation mode specific error
out-state-sequence-errors (*integer*)	Sequence errors, for example Sequence number overflow
out-state-expired (*integer*)	State is expired
out-policy-blocked (*integer*)	Policy discards
out-policy-dead (*integer*)	Policy is dead
out-policy-errors (*integer*)	Policy error

Application Examples

Site to Site IpSec Tunnel

Consider setup as illustrated below



Two remote office routers are connected to internet and office workstations behind routers are NATed. Each office has its own local subnet, 10.1.202.0/24 for Office1 and 10.1.101.0/24 for Office2. Both remote offices needs secure tunnel to local networks behind routers.

IP Connectivity

On both routers ether1 is used as wan port and ether2 is used to connect workstations. Also NAT rules are set tu masquerade local networks.

Office1 router:

```
/ip address
add address=192.168.90.1/24 interface=ether1
add address=10.1.202.1/24 interface=ether2

/ip route
add gateway=192.168.90.254

/ip firewall nat
add chain=srcnat out-interface=ether1 action=masquerade
```

Office2 router:

```
/ip address
add address=192.168.80.1/24 interface=ether1
add address=10.1.101.1/24 interface=ether2

/ip route
add gateway=192.168.80.254

/ip firewall nat
add chain=srcnat out-interface=ether1 action=masquerade
```

IpSec Peer's config

Next step is to add peer's configuration. We need to specify peers address and port and pre-shared-key. Other parameters are left to default values.

Office1 router:

```
/ip ipsec peer
add address=192.168.80.1/32 port=500 auth-method=pre-shared-key secret="test"
```

Office2 router:

```
/ip ipsec peer
add address=192.168.90.1/32 port=500 auth-method=pre-shared-key secret="test"
```

Policy and proposal

It is important that proposed authentication and encryption algorithms match on both routers. In this example we can use predefined "default" proposal

```
[admin@MikroTik] /ip ipsec proposal> print
Flags: X - disabled
 0   name="default" auth-algorithms=sha1 enc-algorithms=3des lifetime=30m
     pfs-group=modp1024
```

As we already have proposal as a next step we need correct IpSec policy. We want to encrypt traffic coming form 10.1.202.0/24 to 10.1.101.0/24 and vice versa.

Office1 router:

```
/ip ipsec policy
add src-address=10.1.202.0/24 src-port=any dst-address=10.1.101.0/24 dst-port=any \
sa-src-address=192.168.90.1 sa-dst-address=192.168.80.1 \
tunnel=yes action=encrypt proposal=default
```

Office2 router:

```
/ip ipsec policy
add src-address=10.1.101.0/24 src-port=any dst-address=10.1.202.0/24 dst-port=any \
sa-src-address=192.168.80.1 sa-dst-address=192.168.90.1 \
tunnel=yes action=encrypt proposal=default
```

Note that we configured tunnel mode instead of transport, as this is site to site encryption.

NAT Bypass

At this point if you will try to establish IpSec tunnel it will not work, packets will be rejected. This is because both routers have NAT rules that is changing source address after packet is encrypted. Remote router reiceves encrypted packet but is unable to decrypt it because source address do not match address specified in policy configuration. For more information see packet flow ipsec example.

To fix this we need to set up NAT bypass rule.

Office1 router:

```
/ip firewall nat
add chain=srcnat action=accept  place-before=0 \
 src-address=10.1.202.0/24 dst-address=10.1.101.0/24
```

Office2 router:

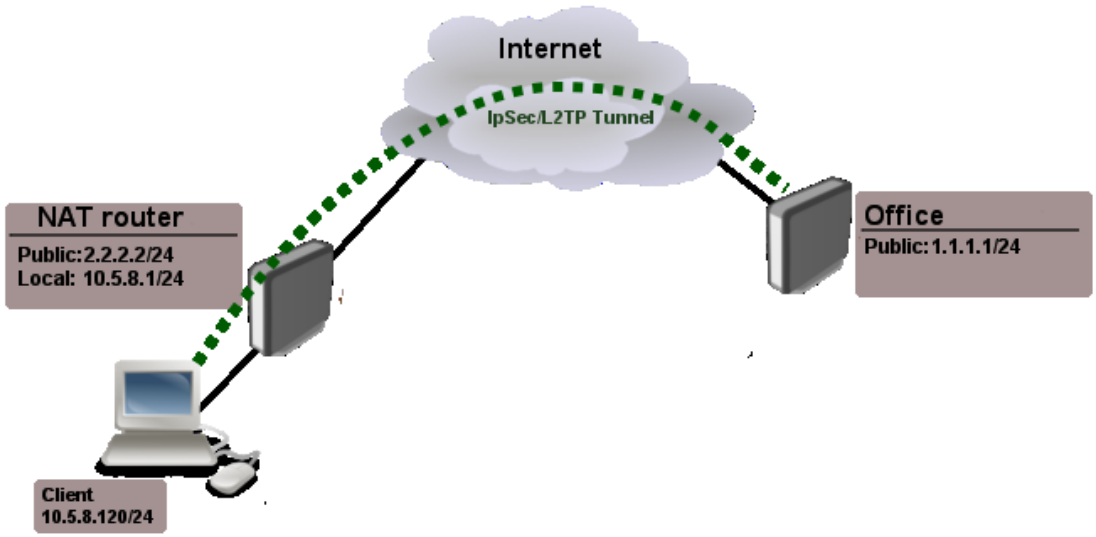
```
/ip firewall nat
add chain=srcnat action=accept  place-before=0 \
 src-address=10.1.101.0/24 dst-address=10.1.202.0/24
```

It is very important that bypass rule is placed at the top of all other NAT rules.

Note: If you previously tried to establish tunnel before NAT bypass rule was added, you have to clear connection table from existing connection or restart the routers

Ipsec/L2TP behind NAT

Consider setup as illustrated below



Client needs secure connection to the office with public address 1.1.1.1, but server does not know what will be the source address from which client connects. It is so called road-warrior setup. Our client will also be located behind the router with enabled NAT.

For the setup RouterOS router will be used as the client device behind NAT (it can be any device: Windows PC, Smartphone, Linux PC, etc.)

IP Connectivity

On the server:

```
/ip address
add address=1.1.1.1/24 interface=ether1

/ip route
add gateway=1.1.1.2
```

On the clients router:

```
/ip address
add address=2.2.2.2/24 interface=ether1
add address=10.5.8.0/24 interface=ether2

/ip route
add gateway=2.2.2.1

/ip firewall net
add chain=srcnat action=masquerade out-interface=ether1
```

On the client:

```
/ip address
add address=10.5.8.120/24 interface=ether1
```

L2TP Config

On the server:

```
/interface l2tp-server
set enabled=yes profil=default

/ip pool
add name=l2tp-pool ranges=192.168.1.2-192.168.1.20

/ppp profile
set default local-address=192.168.1.1 remote-address=l2tp-pool

/ppp secret
add name=l2tp-test password=test123456
```

On the client:

```
/interface l2tp-client
add connect-to=1.1.1.1 disabled=no name=l2tp-out1 password=password user=l2tp-test
```

IpSec Config

On server side:

```
/ip ipsec proposal
set [ find default=yes ] enc-algorithms=3des,aes-128,aes-192,aes-256
/ip ipsec peer
add generate-policy=yes hash-algorithm=sha1 nat-traversal=yes secret=test123456 \
      send-initial-contact=no
```

RouterOS as client:

```
/ip ipsec proposal
set [ find default=yes ] enc-algorithms=aes-128
/ip ipsec peer
add address=1.1.1.1/32 hash-algorithm=sha1 nat-traversal=yes secret=test123456

/ip ipsec policy
add dst-address=1.1.1.1/32 protocol=udp sa-dst-address=1.1.1.1 \
      sa-src-address=10.5.8.120 src-address=10.5.8.120/32
```

Notice that `nat-traversal` is enabled. This option is required because Ipsec connection will be established through the NAT router otherwise Ipsec will not be able to establish phase2.

Note: Only one L2TP/ipsec connection can be established through the NAT. Which means that only one client can connect to the sever located behind the same router.

Connecting with Shrew Client and allowing only Encrypted traffic

- See example here

[Top | Back to Content]

Manual:IPv6/Firewall/Address-list

Manual:IPv6/Firewall/Mangle

Manual:IPv6/ND

Applies to RouterOS: v3, v4 +

Summary

Sub-menu: `/ipv6 nd`
Standards: `RFC 2462, RFC 2461`
Package : `IPv6`

RouterOS has Ipv6 Neighbor Detection and stateless address autoconfiguration support using Router Advertisement Daemon (RADVD).

Node description

Node is a device that implements IPv6. In IPv6 networks nodes are divided into two types:

- **Routers** - a node that forwards IPv6 packets not explicitly addressed to itself.
- **Hosts** - any node that is not a router.

Routers and hosts are strictly separated, meaning that router cannot be host and host cannot be router at the same time.

Stateless address autoconfiguration

There are several types of autoconfiguration:

- *stateless* - address configuration is done by received Router Advertisement messages. These messages include stateless address prefixes and require that host is not using stateful address configuration protocol.
- *stateful* - address configuration is done by using stateful address configuration protocol (DHCPv6). Stateful protocol is used if RA messages do not include address prefixes.
- *both* - RA messages include stateless address prefixes and require that hosts use a stateful address configuration protocol.

A highly useful feature of IPv6 is the ability to automatically configure itself without the use of a stateful configuration protocol like DHCP (See example).

Note: Address autoconfiguration can only be performed on multicast-capable interfaces.

It is called stateless address autoconfiguration, since there is no need to manage state in the router side. It is a very simple, robust and effective autoconfiguration mechanism.

RouterOS uses RADVD to periodically advertise information about the link to all nodes on the same link. The information is carried by ICMPv6 "router advertisement" packet, and includes following fields:

- IPv6 subnet prefix
- Default router link local address
- Other parameters that may be optional: link MTU, default hoplimit, and router lifetime.

Then host catches the advertisement, and configures the global IPv6 address and the default router. Global IPv6 address is generated from advertised **subnet prefix** and EUI-64 interface identifier.

Optionally, the host can ask for an advertisement from the router by sending an ICMPv6 "router solicitation" packet. On linux **rtsol** utility transmits the router solicitation packet. If you are running a mobile node, you may want to transmit router solicitations periodically.

Note: Due to restrictions of IPv6, address auto-configuration can not be performed on routers. Routers require manual address configuration.

Address states

When auto-configuration address is assigned it can be in one of the following states:

- ***tentative*** - in this state host verifies that the address is unique. Verification occurs through duplicate address detection.
- ***preferred*** - at this state address is verified as unique and node can send and receive unicast traffic to and from a preferred address. The period of time of preferred state is included in the RA message.
- ***deprecated*** - address is still valid, but is not used for new connections.
- ***invalid*** - node can no longer send or receive unicast traffic. An address enters the invalid state after the valid lifetime expires.

Image belove ilustrates relation between states and lifetimes.

Neighbor discovery

Sub-menu: `/ipv6 nd`

In this submenu IPv6 Neighbor Discovery (ND) protocol is configured.

Neighbor Discovery (ND) is a set of messages and processes that determine relationships between neighboring nodes. ND, compared to IPv4, replaces Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP) Router Discovery, and ICMP Redirect and provides additional functionality.

ND is used by hosts to:

- Discover neighboring routers.

- Discover addresses, address prefixes, and other configuration parameters.

ND is used by routers to:

- Advertise their presence, host configuration parameters, and on-link prefixes.
- Inform hosts of a better next-hop address to forward packets for a specific destination.

ND is used by nodes to:

- Both resolve the link-layer address of a neighboring node to which an IPv6 packet is being forwarded and determine when the link-layer address of a neighboring node has changed.
- Determine whether IPv6 packets can be sent to and received from a neighbor.

Properties

Property	Description
advertise-dns (*yes* | *no*; Default: **no**)	Option to redistribute DNS server information using RADVD. You will need a running client side software with Router Advertisement DNS support to take advantage of the advertised DNS information. `Read more >>`
advertise-mac-address (*yes* | *no*; Default: **yes**)	When set, the link-layer address of the outgoing interface is included in the RA.
comment (*string*; Default:)	Descriptive name of an item
disabled (*yes* | *no*; Default: **no**)	Whether item is disabled or not. By default entry is enabled.
hop-limit (*unspecified* | *integer[0..4294967295]*; Default: **unspecified**)	The default value that should be placed in the Hop Count field of the IP header for outgoing (unicast) IP packets.
interface (*all* | *string*; Default:)	Interface on which to run neighbor discovery. • `all` - run ND on all running interfaces.
managed-address-configuration (*yes* | *no*; Default: **no**)	Flag indicates whether hosts should use stateful autoconfiguration (DHCPv6) to obtain addresses.
mtu (*unspecified* | *integer[0..4294967295]*; Default: **unspecified**)	The MTU option is used in router advertisement messages to insure that all nodes on a link use the same MTU value in those cases where the link MTU is not well known. • **unspecified** - do not send MTU option.
other-configuration (*yes* | *no*; Default: **no**)	Flag indicates whether hosts should use stateful autoconfiguration to obtain additional information (excluding addresses).
ra-delay (*time*; Default: **3s**)	The minimum time allowed between sending multicast router advertisements from the interface.
ra-interval (*time[3s..20m50s]-time[4s..30m]*; Default: **3m20s-10m**)	min-max interval allowed between sending unsolicited multicast router advertisements from the interface.
ra-lifetime (*none* | *time*; Default: **30m**)	
reachable-time (*unspecified* | *time[0..1h]*; Default: **unspecified**)	The time that a node assumes a neighbor is reachable after having received a reachability confirmation. Used by the Neighbor Unreachability Detection algorithm (see Section 7.3 of RFC 2461)
retransmit-interval (*unspecified* | *time*; Default: **unspecified**)	The time between retransmitted Neighbor Solicitation messages. Used by address resolution and the Neighbor Unreachability Detection algorithm (see Sections 7.2 and 7.3 of RFC 2461)

Prefix

Sub-menu: `/ipv6 nd prefix`

Prefix information sent in RA messages used by stateless address auto-configuration.

Note: The autoconfiguration process applies only to hosts and not routers.

Properties

Property	Description
6to4-interface (*none* | *string*; Default:)	If this option is specified, this prefix will be combined with the IPv4 address of interface name to produce a valid 6to4 prefix. The first 16 bits of this prefix will be replaced by 2002 and the next 32 bits of this prefix will be replaced by the IPv4 address assigned to interface name at configuration time. The remaining 80 bits of the prefix (including the SLA ID) will be advertised as specified in the configuration file.
autonomous (*yes* | *no*; Default: **yes**)	When set, indicates that this prefix can be used for autonomous address configuration. Otherwise prefix information is silently ignored.
comment (*string*; Default:)	Descriptive name of an item
disabled (*yes* | *no*; Default: **no**)	Whether item is disabled or not. By default entry is enabled.
on-link (*yes* | *no*; Default: **yes**)	When set, indicates that this prefix can be used for on-link determination. When not set the advertisement makes no statement about on-link or off-link properties of the prefix. For instance, the prefix might be used for address configuration with some of the addresses belonging to the prefix being on-link and others being off-link.
preferred-lifetime (*infinity* | *time*; Default: **1w**)	Timeframe (relative to the time the packet is sent) after which generated address becomes "deprecated". Deprecated is used only for already existing connections and is usable until `valid-lifetime` expires. `Read more >>`
prefix (*ipv6 prefix*; Default: **::/64**)	Prefix from which stateless address autoconfiguration generates the valid address.
valid-lifetime (*infinity* | *time*; Default: **4w2d**)	The length of time (relative to the time the packet is sent) an address remains in the valid state. The `valid-lifetime` must be greater than or equal to the `preferred-lifetime`. `Read more >>`
interface (*string*; Default:)	Interface name on which stateless auto-configuration will be running.

Examples

Stateless autoconfiguration example

```
[admin@MikroTik] > ipv6 address print
 Flags: X - disabled, I - invalid, D - dynamic, G - global, L - link-local
  #    ADDRESS                                     INTERFACE         ADVERTISE
  0  G 2001:db8::1/64                                ether1             yes
```

As in example above **advertise** flag is enabled which indicates that dynamic `/ipv6 nd prefix` entry is added.

```
 [admin@MikroTik] > ipv6 nd prefix print
Flags: X - disabled, I - invalid, D - dynamic
 0 D prefix=2001:db8::/64 interface=ether1 on-link=yes autonomous=yes
```

```
       valid-lifetime=4w2d preferred-lifetime=1w
```

On a host that is directly attached to the router we see that an address was added. The address consists of prefix part (first 64 bits) that takes prefix from the prefix advertisement, and host part (last 64 bits) that is automatically generated from local MAC address:

```
atis@atis-desktop:~$ ip -6 addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:db8::21a:4dff:fe56:1f4d/64 scope global dynamic
       valid_lft 2588363sec preferred_lft 601163sec
    inet6 fe80::21a:4dff:fe56:1f4d/64 scope link
       valid_lft forever preferred_lft forever
```

The host has received the *2001:db8::/64* prefix from the router and configured an address with it.

There is also an option to redistribute DNS server information using RADVD:

```
 [admin@MikroTik] > ip dns set server=2001:db8::2
 [admin@MikroTik] > ip dns print
           servers: 2001:db8::2
 ...
 [admin@MikroTik] > ipv6 nd set [f] advertise-dns=yes
```

You will need a running client side software with Router Advertisement DNS support to take advantage of the advertised DNS information.

On Ubuntu/Debian linux distributions you can install **rdnssd** package which is capable of receiving advertised DNS address.

```
mrz@bumba:/$ sudo apt-get install rdnssd
```

```
mrz@bumba:/$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 2001:db8::2

mrz@bumba:/$ ping6 www.mikrotik.com
PING www.mikrotik.com(2a02:610:7501:1000::2) 56 data bytes
64 bytes from 2a02:610:7501:1000::2: icmp_seq=1 ttl=61 time=2.11 ms
64 bytes from 2a02:610:7501:1000::2: icmp_seq=2 ttl=61 time=1.33 ms
^C
--- www.mikrotik.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.334/1.725/2.117/0.393 ms
mrz@bumba:/$
```

See Also

- http://www.tcpipguide.com/free/t_IPv6Addressing.htm

[Top | Back to Content]

Manual:IPv6/Neighbors

Applies to RouterOS: v3, v4 +

Summary

Sub-menu: `/ipv6 neighbor`
Standards: `RFC 2461`
Package : `IPv6`

List of all discovered nodes by IPv6 neighbor discovery protocol (neighbor cache).

```
[admin@test_host] /ipv6 neighbor> print
Flags: R - router
 0   address=ff02::5 interface=main mac-address=33:33:00:00:00:05 status="noarp"

 1   address=ff02::1 interface=main mac-address=33:33:00:00:00:01 status="noarp"

 2 R address=fe80::d7:4cff:fec1:2e32 interface=main mac-address=00:0C:42:28:79:45
     status="stale"
```

Read more about ND >>

Read-only Properties

Property	Description
address (*ipv6 address*)	link-local address of the node.
comment (*string*)	
inteface (*string*)	Interface on which node was detected.
mac-address (*string*)	Mac address of discovered node.
router (*yes* | *no*)	Whether discovered node is router

status (*noarp* | *incomplete* | *stale* | *reachable* | *delay* | *probe*)	Status of the cached entry: • **noarp** - • **incomplete** - address resolution is in progress and the link-layer address of the neighbor has not yet been determined; • **reachable** - the neighbor is known to have been reachable recently (within tens of seconds ago); • **stale** - the neighbor is no longer known to be reachable but until traffic is sent to the neighbor, no attempt should be made to verify its reachability; • **delay** - the neighbor is no longer known to be reachable, and traffic has recently been sent to the neighbor, probes are delayed for a short period in order to give upper layer protocol a chance to provide reachability confirmation; • **probe** - the neighbor is no longer known to be reachable, and unicast Neighbor Solicitation probes are being sent to verify reachability.

Manual:IPv6/Route

Applies to RouterOS: v3, v4 +

Summary

Sub-menu: `/ipv6 route`
Standards: `RFC 4291`

For static routing, the basic principles of IPv6 are exactly the same as for IPv4.

Simple ipv6 routing example:

```
[admin@MikroTik] > ipv6 route add dst-address=2001::/16 gateway=fc00:dead:beef::2
[admin@MikroTik] > ipv6 route print detail
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, o - ospf, b - bgp, U - unreachable
 0 A S  dst-address=2001::/16 gateway=fc00:dead:beef::2 reachable ether1 distance=1
        scope=30 target-scope=10
```

Most notable difference between ipv4 and ipv6 is that link local addresses can be used as route nexthops if interface is specified:

```
[admin@MikroTik] > ipv6 route add dst-address=2002::/16 gateway=fe80::21a:4dff:fe56:1f4d%ether1
[admin@MikroTik] > ipv6 route print detail
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, o - ospf, b - bgp, U - unreachable
 ...
 1 A S  dst-address=2002::/16
        gateway=fe80::21a:4dff:fe56:1f4d%ether1 reachable distance=1
        scope=30 target-scope=10
```

Another small difference is that there are no blackhole or prohibit routes, only unreachable.

IPv4 and IPv6 routing also differs in the area of multipath route. Technically speaking, in Linux kernel there is no support for multiple nexthops for a IPv6 route. However, RouterOS allows to set more than one gateway address for a single route. In this case, a route is installed in the kernel for each of the different interfaces to which route's nexthops belong.

Example:

```
[admin@MikroTik] > ipv6 address p
Flags: X - disabled, I - invalid, D - dynamic, G - global, L - link-local
 #    ADDRESS                                     INTERFACE         ADVERTISE
 0  G fc00:1::1/64                                ether1            no
 1  G fc00:2::1/64                                ether2            no
[admin@MikroTik] > ipv6 route add dst-address=2001::/16 gateway=fc00:1::2,fc00:2::2
[admin@MikroTik] > ipv6 route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, o - ospf, b - bgp, U - unreachable
 #      DST-ADDRESS              GATEWAY                      DISTANCE
 0 A S  2001::/16                fc00:2::2 reachable ether1,  1
                                 fc00:1::2 reachable ether2
```

When printing the Linux kernel route table, we see that two routes were added, not one:

```
# ip -6 route
2001::/16 via fc00:2::2 dev eth1  proto static  metric 1024  mtu 1500 advmss 1440 metric10 4294967295
2001::/16 via fc00:1::2 dev eth0  proto static  metric 1024  mtu 1500 advmss 1440 metric10 4294967295
...
```

Properties

Property	Description
bgp-as-path (*list of AS numbers*; Default:)	Value of BGP AS_PATH attribute. Read more>>
bgp-atomic-aggregate (*yes* | *no*; Default:)	
bgp-communities (*list of two integers separated by :*; Default:)	Value of BGP communities list. This attribute can be used to group or filter routes. Named values have special meanings: • internet - advertise this route to the Internet community (i.e. all routers) • no-advertise - do not advertise this route to any peers • no-export - do not advertise this route to EBGP peers • local-as - same as no-export, except that route is also advertised to EBGP peers inside local confederation
bgp-local-pref (*integer*; Default: **100**)	Value of BGP LOCAL_PREF attribute. Read more>>
bgp-med (*integer*; Default: **0**)	Value of BGP MULTI_EXIT_DISC BGP attribute. Read more>>
bgp-origin (*igp* | *egp* | *incomplete*; Default:)	Value of BGP ORIGIN attribute. Read more>>
bgp-prepend (*integer [0..16]*; Default:)	How many times to prepend router's own AS number to AS_PATH attribute when announcing route via BGP. Affects only routes sent to eBGP peers (for iBGP value 0 is always used). Read more>>

check-gateway (*ping* | *arp*; Default:)
Periodically (every 10 seconds) check gateway by sending either ICMP echo request (ping) or ARP request (arp). If no response from gateway is received for 10 seconds, request times out. After two timeouts gateway is considered unreachable. After receiving reply from gateway it is considered reachable and timeout counter is reset.

comment (*string*; Default:)
Descriptive name of an item

disabled (*yes* | *no*; Default: **yes**)
Whether interface is disabled or not. By default it is disabled.

distance (*integer*; Default:)
Value used in route selection. Routes with smaller distance value are given preference. If value of this property is not set, then the default depends on route protocol:

- connected routes: 0
- static routes: 1
- eBGP: 20
- OSPF: 110
- RIP: 120
- MME: 130
- iBGP: 200

dst-address (*IPv6/Netmask*; Default: **::/0**)
IPv6 prefix of route, specifies destination addresses that this route can be used for. **Netmask (integer [0..128])** part of this property specifies how many of the most significant bits in packet destination address must match this value. If there are several active routes that match destination address of packet, then the most specific one (with largest netmask value) is used.

gateway (*ipv6 address[,ipv6 address[,..]]*; Default:)
Specifies which host or interface packets should be sent to. Link Local addresses can also be used as gateways if interface is specified. `Read more>>`

route-tag (*integer*; Default:)
Value of route tag attribute for RIP or OSPF. For RIP only values 0..65535 are valid.

scope (*integer [0..255]*; Default:)
Used in nexthop resolution. Route can resolve nexthop only through routes that have scope less than or equal to the target-scope of this route. Default value depends on route protocol:

- connected routes: 10 (if interface is running)
- OSPF, RIP, MME routes: 20
- static routes: 30
- BGP routes: 40
- connected routes: 200 (if interface is not running)

target-scope (*integer [0..255]*; Default: **10 (30 for iBGP)**)
Used in nexthop resolution. This is the maximum value of scope for a route through which a nexthop of this route can be resolved. See nexthop lookup.

type (*unicast* | *unreachabe*; Default: **unicast**)
Routes that do not specify nexthop for packets, but instead perform some other action on packets have type different from the usual `unicast`.

Read-only properties

Property	Description
active (*yes* | *no*)	Whether route is currently active and is used for packet forwarding.
bgp (*yes* | *no*)	BGP route
bgp-weight (*integer*)	BGP weight attribute
connect (*yes* | *no*)	Directly connected route
dynamic (*yes* | *no*)	Dynamically added route
gateway-status ()	
ospf (*yes* | *no*)	OSPF route
ospf-metric (*integer*)	
ospf-type (*external-type-1* | *intra-area* | ...)	Type of the OSPF route
received-from (*string*)	Name of the BGP peer from which this route was received.

rip (*yes* | *no*)	RIP route
static (*yes* | *no*)	Statically added route by user.
unreachable (*yes* | *no*)	Discard packet forwarded by this route. Notify sender with ICMP host unreachable (type 3 code 1) message.

See Also

- Ipv4 Routing and route selection
- Simple IPv6 routing example

[Top | Back to Content]

Manual:KVM

Applies to RouterOS: v4.3+ on x86

Overview

Kernel-based Virtual Machine (KVM) is the method to run multiple guest operating systems on one RouterOS host. KVM can be used only on x86 machines that have CPU with virtualization support .

Requirements

KVM requires **Intel VT-x** or **AMD-V** CPU virtualization support. Here [1] you can find a list of supported CPUs, for more detailed information look on vendor's web site.

Each guest requires at least 16 MB of RAM and sufficient storage space on image file. Once image file have been created, its size cannot be increased.

KVM support in RouterOS is enabled if *kvm* package is installed.

Where it can be used?

Virtual Router is useful to allow clients or lower-privilege users access their own 'router' and adjust configure as they like without the need for a second hardware.

For example; a WISP can create a virtual router for the clients ethernet port allowing them to define their own firewall settings, while leaving the WISP's wireless settings untouched.

Another useful method is to run guest OS that supports functionality which is not available in RouterOS, for example, Intrusion detection (SNORT), Asterisk or Squid web proxy.

It can also be used as test environment. it is possible to create virtual network within one x86 machine very similar to real network and test how RouterOS behaves before implementing the setup in your production network.

Creating KVM Guest

Before creating KVM guest we need image file. RouterOS has built in commands to make and modify RouterOS image easily without external tools.

```
/kvm make-routeros-image file-name=ros1.img file-size=128
```

We can proceed with Guest configuration when disk image is created.

```
/kvm add name=ROS memory=128MiB cpu-count=2 disabled=no disk-images=hda:ros1.img \
    initrd="" kernel="" kernel-cmdline="console=ttyS0"
```

As you noticed **initrd** and **kernel** properties are empty, which means that hosts kernel and initrd is used.

For example, to add guest without SMP support we can explicitly set initrd and kernel:

```
/kvm add name=ROS memory=128MiB cpu-count=2 disabled=no disk-images=hda:ros1.img \
    initrd=/boot/initrd.rgz kernel=/boot/vmlinuz kernel-cmdline="console=ttyS0"
```

Note: Leaving **initrd** and **kernel** properties empty is dangerous if Host and Guest will be running different RouterOS versions . Guests other than RouterOS also can break if you leave these values empty.

KVM Guest when created is not automatically started. We must start it manually

```
[admin@proxy] /kvm> start ROS;
```

```
[admin@proxy] /kvm> print
Flags: X - disabled
 0   name="ROS" cpu-count=2 memory=128MiB disk-images=hda:ros1.img kernel="/boot/vmlinuz"
      kernel-cmdline="" initrd="/boot/initrd.rgz" vnc-server=0.0.0.0:0 snapshot=no  state=running
[admin@proxy] /kvm>
```

Adding Interfaces

Lets add to our previously created Virtual Router one interface.

```
[admin@proxy] /kvm interface> add virtual-machine=ROS type=dynamic
[admin@proxy] /kvm interface> print
Flags: X - disabled, A - active
 #   VIRTUAL-MACHINE            INTERFACE                  TYPE         VM-MAC-ADDRESS
 0   ROS                                                   dynamic      02:D9:52:31:11:CC
[admin@proxy] /kvm interface>
```

In this case dynamic type is used which creates dynamic virtual interface on the host:

```
[admin@proxy] /interface virtual-ethernet> print
Flags: D - dynamic, X - disabled, R - running
 #      NAME                              MTU   ARP         MAC-ADDRESS
 0 D R tap1                               1500  enabled     02:3F:9F:AE:10:34
[admin@proxy] /interface virtual-ethernet>
```

Note: Add and remove interfaces only when KVM guest is shut-down, stopped or disabled. Making changes to running guest may lead to host system crash.

If mac addresses are not specified when creating virtual interfaces, addresses are generated automatically. Generate MAC addresses will be in form of 02:XX:XX:XX:XX:XX. For *static* interfaces this address will not change during use of guest, for *dynamic* interface will change every time dynamic interface is created.

More information about virtual interfaces are in virtual-ethernet manual

Connecting to the virtual machine

There are two ways how to connect to KVM Guest:

- virtual console;
- vnc.

Console

To connect using console:

```
[admin@proxy] /kvm> console ROS
```

You will see your newly added virtual interface here:

```
[admin@mr0] > interface print
 Flags: D - dynamic, X - disabled, R - running, S - slave
 #     NAME                                                TYPE               MTU
 0  R  ether1                                              ether              1500
```

To disconnect from the metarouter virtual machine console, hit **CTRL + A** and then **Q** to Quit back to your **Host** console (if you are using minicom, hit **CTRL + A** twice):

```
[admin@MikroTik] >
[Q - quit connection]       [B - send break]
[A - send Ctrl-A prefix]    [R - autoconfigure rate]

 Q

 Welcome back!
```

VNC

Before connecting with VNC client guest needs some configuration changes.

```
[admin@proxy] /kvm> print
Flags: X - disabled
 0   name="ROS" cpu-count=2 memory=128MiB disk-images=hda:ros1.img kernel="/boot/vmlinuz"
      kernel-cmdline="" initrd="/boot/initrd.rgz" vnc-server=0.0.0.0:0 snapshot=no state=running
[admin@proxy] /kvm> shut-down 0
[admin@proxy] /kvm> set 0 vnc-server=10.5.100.99:1
[admin@proxy] /kvm> start 0
```

```
[admin@proxy] /kvm> print
Flags: X - disabled
```

```
 0   name="ROS" cpu-count=2 memory=128MiB disk-images=hda:ros1.img kernel="/boot/vmlinuz"
      kernel-cmdline="" initrd="/boot/initrd.rgz" vnc-server=10.5.100.99:1 snapshot=no
      state=running
[admin@proxy] /kvm>
```

VNC servers address in this case is the address on the host reachable from remote locations. Address is followed by screen number.

Now we can try to connect from remote location:

```
mrz@bumba:/$ vncviewer 10.5.100.99:1
```

Configuring a virtual network

Right now you saw that the virtual interface is visible in the **Host** *Interfaces* menu as *tap1* and also in the **guest** *interfaces* menu as *ether1*. You can add an IP address on both interfaces, and set up networking. Creating a bridge between the virtual interface and a physical interface allows traffic to pass.

As an example lets make three virtual routers connected to each other on the same broadcast domain.

Create images and guests:

```
/kvm
make-routeros-image file-name=R1.img file-size=64
make-routeros-image file-name=R2.img file-size=64
make-routeros-image file-name=R3.img file-size=64

add name=R1 disk-image=hda:R1.img
add name=R2 disk-image=hda:R2.img
add name=R3 disk-image=hda:R3.img
```

Create a bridge interface which will simulate broadcast domain and add virtual interfaces:

```
/interface bridge
add name=kvm_bridge

/kvm interface
add virtual-machine=R1 type=dynamic dynamic-bridge=kvm_bridge
add virtual-machine=R2 type=dynamic dynamic-bridge=kvm_bridge
add virtual-machine=R3 type=dynamic dynamic-bridge=kvm_bridge
```

Now we can start virtual machines and verify if dynamic interfaces are created:

```
[admin@proxy] /kvm> start [find]
```

```
[admin@proxy] > /interface virtual-ethernet print
Flags: D - dynamic, X - disabled, R - running
 #     NAME                              MTU   ARP        MAC-ADDRESS
 0 D R tap2                              1500  enabled    02:20:94:67:D6:D5
 1 D R tap3                              1500  enabled    02:95:EE:EA:43:FF
 2 D R tap4                              1500  enabled    02:05:7E:4B:86:F9
[admin@proxy] > /interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE               BRIDGE                    PRIORITY PATH-COST  HORIZON
```

```
 0  D tap2                    kvm_bridge              0x80     10         none
 1  D tap3                    kvm_bridge              0x80     10         none
 2  D tap4                    kvm_bridge              0x80     10         none
[admin@proxy] >
```

Now we can connect with console to each of guests and set up ip addresses from the same network and verify reachability. R1

```
[admin@proxy] > /kvm console R1

[Ctrl-A is the prefix key]

MikroTik 5.0rc8
MikroTik Login: admin
Password:

 [admin@MikroTik] > /ip address add address=192.168.1.1/24 interface=ether1
```

R2

```
<pre>
[admin@proxy] > /kvm console R2

[Ctrl-A is the prefix key]

MikroTik 5.0rc8
MikroTik Login: admin
Password:

 [admin@MikroTik] > /ip address add address=192.168.1.2/24 interface=ether1
```

R3

```
<pre>
[admin@proxy] > /kvm console R1

[Ctrl-A is the prefix key]

MikroTik 5.0rc8
MikroTik Login: admin
Password:

[admin@MikroTik] > /ip address add address=192.168.1.3/24 interface=ether1
[admin@MikroTik] > /ping 192.168.1.1
HOST                                     SIZE  TTL TIME  STATUS
192.168.1.1                              56    64  11ms
192.168.1.1                              56    64  2ms
    sent=2 received=2 packet-loss=0% min-rtt=2ms avg-rtt=6ms max-rtt=11ms

[admin@MikroTik] > /ping 192.168.1.2
```

```
HOST                                     SIZE  TTL TIME  STATUS
192.168.1.2                              56    64  12ms
    sent=1 received=1 packet-loss=0% min-rtt=12ms avg-rtt=12ms max-rtt=12ms
```

Removing KVM guest

KVM guest has two parts in RouterOS - configuration (kvm, virtual-ethernet, /kvm interface) and image file (/file). If image file is removed, but KVM guest is still running, then file will be removed from file menu but still exist until guest is shut-down or disabled, at that moment file will be removed and storage space returned to available storage on the router.

Additional information

Information useful for running KVM guests

Host shutdown

When host is shutting down each guest receives shut-down notification and are give 10 seconds to shut down. After time-out value is reached, guests are killed.

Host and guest update

When new version of RouterOS is updated to host system and you have RouterOS guest with *initrd* and *kernel* fields empty, it is good practice to update guest first (even it it does not boot up at current host versions. Then update host and see if guests are running. After guest update incompatibilities between host kernel and guest drivers might prevent guest from booting up properly.

Reference

General

Sub-menu: `/kvm`

KVM Guest Properties

To add new KVM guest you will have to issue command *add* under */kvm* menu with attributes as follows:

Property	Desciption
comment (*text, default: ''*)	to add simple *text* description of the KVM guest
cpu-count (*1 .. 32, default: 1*)	available count of processing cores for guest. Allowed values are [1..32]
disabled (*yes | no, default: no*)	to set guest state after creation, values: *yes* or *no*

disk-images (*list of images used in guest*)	list of image assignment to drives for guest OS. If type will be set to ***cdrom*** then guest will automatically boot from that, instead of any other drive configured in this field. It can be single drive specified
	`disk-images=hda:ros.img`
	or it can be comma seperated list:
	`disk-images=hda:system.img,hdb:swap.img`

initrd (*path*)	path to initrd file, can be left empty if running RouterOS as guest
kernel (*path*)	path to kernel image file, if using RouterOS image created on host this field can be left empty

kernel-cmdline (*text*)	parameters that are passed to kernel, it is space separated string.
memory (*integer default:32*)	to set up amount of memory that is available to KVM guest
name (*text*)	name of KVM guest that it will be accessible though the system
snapshot (*yes | no*)	will try to run virtual machine with image file in read-only mode.
vnc-server-address (*IP address*)	address to bind VNC server port that will connect to guest virtual screen. If left empty it will bind to all IP addresses. If address set is not ready at the moment when guest is started then system will automatically attempt to start guest for the next 20 seconds. If IP address to bind VNC does not become available in that time automatic start of guest will fail and guest will not be started. IP address is considered unavailable if either address or interface address is assigned to is invalid or does not exist.
vnc-server-display (*number (0..99) default:0*)	will try to run virtual machine with image file in read-only
copy-from (*number*)	use configuration from already existing KVM guest

Warning: vnc-server attribute has been changed since RouterOS 5.0. in older versions instead of vnc-server-address and vnc-server-display was used combine attribute named vnc-server <IP address>:<display number>

Note: If start of guest failed for the first time, then next 20 seconds KVM will attempt to start guest. After 20 seconds it will fail and guest will stay in stopped state.

States of KVM guest

This field is read-only and is set by RouterOS. These are possible values that can be set:

- *stopped* - KVM guest is not running, either successful shut-down or disabled.
- *stopping* - KVM guest is shutting down
- *starting* - KVM guest is starting
- *running* - KVM guest has started successfully and is executing guest operating system
- *restarting* - KVM guest is reloading its guest operating system
- *failed* - KVM guest has encountered an error and is not operational.
- *image-busy* - image file set in configuration is already in use by other KVM guest entry
- *no-kernel-or-initrd* - initrd or kernel was not found in files set in configuration, mentioned files could not be found or no values in those fields where set
- *no-disk-image* - either disk image was not found or disk image was not set in configuration.
- *kernel-extract-failed* - when in guest configuration field *kernel* is left empty and and KVM cannot extract kernel from image file supplied
- *vnc-cant-bind* - vnc server for guest cannot bind to setting specified in *vnc-server-address* and/or *vnc-server-display*

KVM commands

Sub-menu allows to manage KVM guests on RouterOS host.

Command	Desciption
add	Create new KVM guest entry
comment	Set comment for KVM guest entry
console	to connect to KVM guest console display
continue	resume KVM guest if it was paused
disable	change global state of KVM guest. If enabled KVM guest will be started when RouterOS boots. KVM guest cannot change
edit	edit selected value of KVM guest entry
enable	change KVM guest global state to enable operation of KVM guest. If guest where disabled before - KVM guest is automatically started.
export	Print or save an export script that can be used to restore configuration of current sub-menu, KVM guest configuration, image files will not be saved
find	Find items by value
get	Gets value of item's property
make-routeros-image	creates RouterOS image from current installation installed on the router with no configuration. It is advised to create Image file larger than minimal, so you are able to upload new package files and upgrade/update RouterOS installation. Also, all the additional files created in KVM guest will be stored in file image. This image file is not connected to host RouterOS and user is able to run different RouterOS versions on host and guest. This command will create RAW image file containing RouterOS installation. parameters: • `file-name` - name of ROS image file; • `file-size` - image size in Meba Bytes; • `configuration-script` - file name where configuration script is located;
pause	suspend operation of KVM guest
print	Print values of item properties
reboot	issue ACPI ***shut-down*** command to KVM guest, if guest does not support ACPI, command have no effect. After KVM guest is shut-downed it will be automatically started by host when shut down is complete.
reconfigure-routeros-image	sets up default configuration for RouterOS image. Parameters: • `file-name` - name of ROS image to be reconfigured; • `configuraton-script` - file name where configuration script is located; • `configuration-string` - string containing ROS commands to be configured on ROS image.
remove	Remove item
set	Change item properties
shut-down	issue ACPI shut-down command to KVM guest, if guest does not support ACPI, command have no effect.
start	to start KVM guest

Interface

Sub-menu: `/kvm interface`

Property	Desciption
comment (*text*)	description of interface
disabled (*yes|no, default: no*)	state of interface after creation
host-mac-address (*MAC Address, default: generated*)	MAC address of virtual interface that host will use
model (*virto | e1000 | pcnet, default: virtio*)	mode of virtual interface. Available options are: • *virtio - default* value. Fastest available option, should be chosen if no other problems are encountered • *e1000* - emulates card that uses e1000 driver. *This option where added for compatibility with some guest operating systems that where not able to communicate with host RouterOS if* virtio *interface model where used.* • *pcnet* - emulates card that uses pcnet driver. *This option where added for compatibility with some guest operating systems that where not able to communicate with host RouterOS if* virtio *interface model where used.*
vm-mac-address (*MAC Address, default: generated*)	MAC address of virtual interface that guest will use
copy-from (*number*)	use configuration from existing virtual interface
dynamic-bridge (*interface name, default: none*)	if set, ***dynamic*** interface will be automatically added as port to bridge interface
interface	is set for ***static*** interface, to assign it to already created virtual-ethernet interface
type (*dynamic | static, default: static*)	to set if interface is either *static* or *dynamic*. • *dynamic* interface will add virtual-ethernet automatically when virtual machine starts. • *static* interface have to have created virtual-ethernet interface at the time of creation of the entry.
virtual-machine (*KVM machine name, must be set)*	name of virtual machine this interface will be assigned to

References

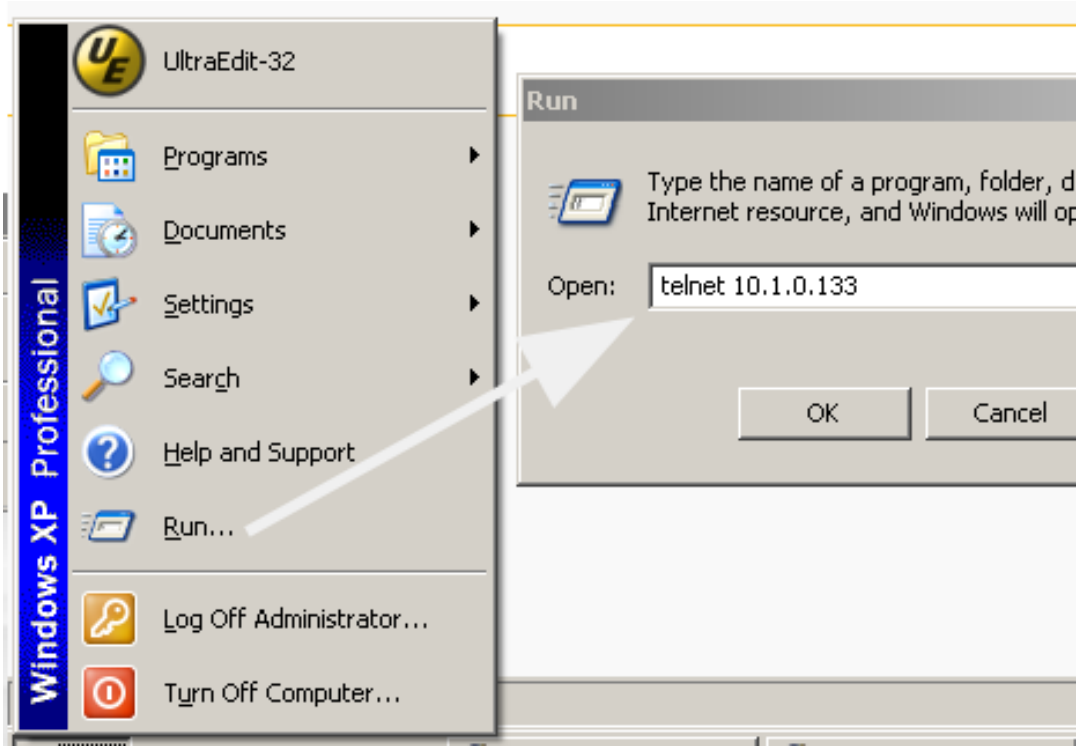
[1] http://en.wikipedia.org/wiki/X86_virtualization

Manual:Entering a RouterOS License key

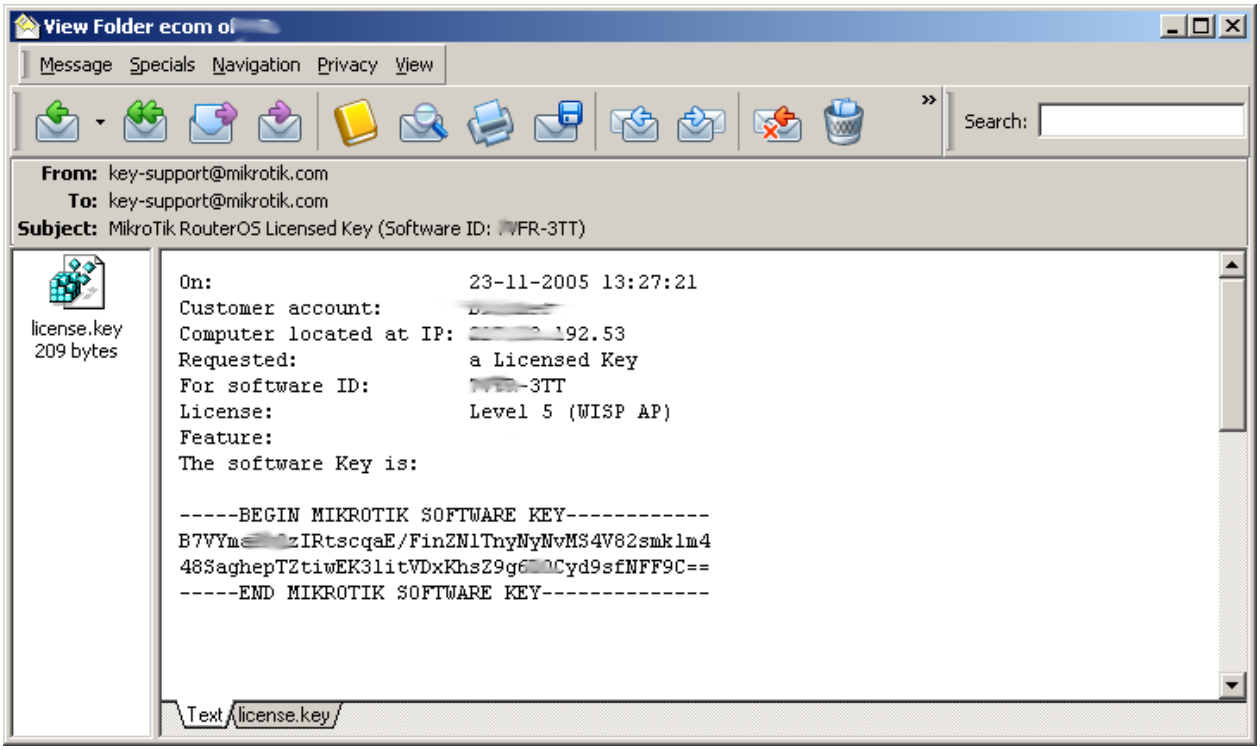
First method

If you have installed the Router OS onto a PC (i.e. it is not a RouterBoard), you will initially have no key, but for 24 hours the router will be fully operable and working. During this period configure the router to have an IP address, for example **10.1.0.133**, then purchase a key on the www.mikrotik.com account server. To enter this key follow this short guide:

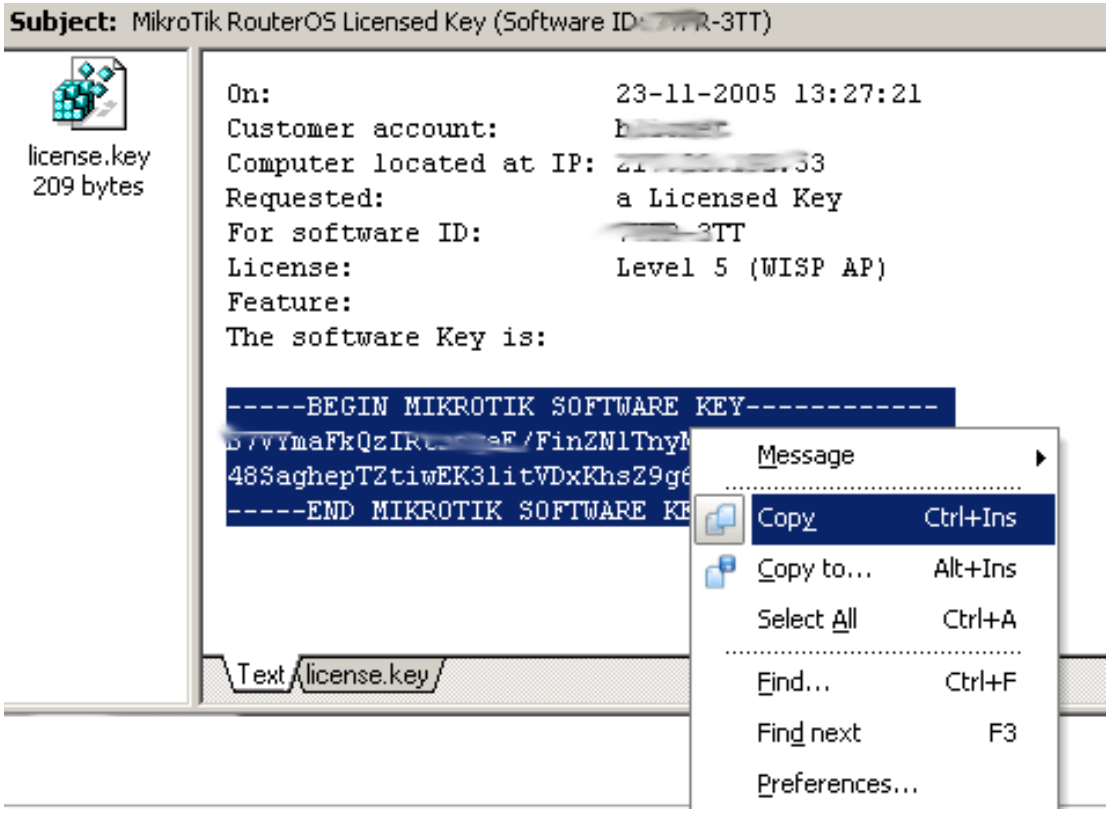
- **Telnet** to the router:



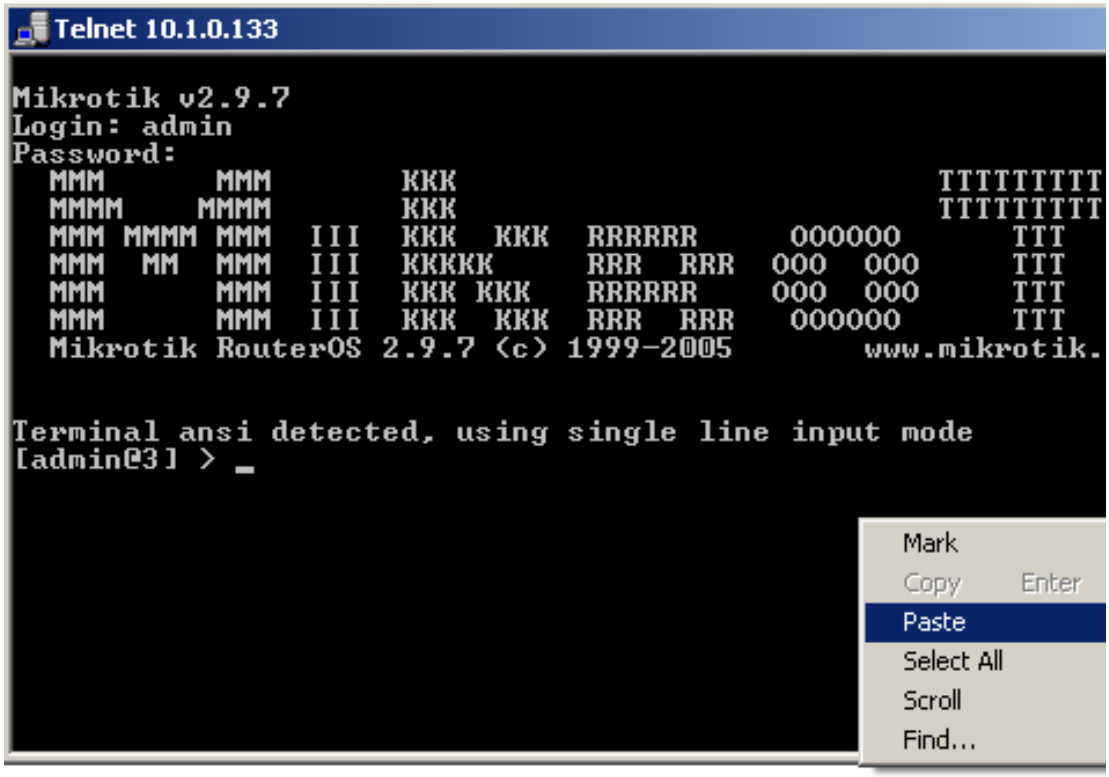
- find the email from mikrotik which contains your key



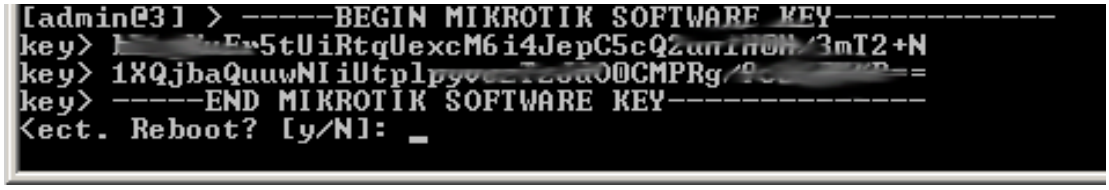
- select this key and click **copy**



- in the telnet window right-click the screen and choose **paste**



- type **y** and hit enter to reboot the router



- For fans of the serial console, you may enter the license information via the serial console on certain equipment. Perform the same operation as in the telnet session above, i.e., at the console prompt, paste the license information as if it were a command; the paste buffer or clipboard should contain the full text including the lines containing "BEGIN" and "END" as mentioned above.

Manual:Interface/L2TP

Applies to RouterOS: v3, v4, v5+

Summary

Standards: `RFC 2661`

L2TP is a secure tunnel protocol for transporting IP traffic using PPP. L2TP encapsulates PPP in virtual lines that run over IP, Frame Relay and other protocols (that are not currently supported by MikroTik RouterOS). L2TP incorporates PPP and MPPE (Microsoft Point to Point Encryption) to make encrypted links. The purpose of this protocol is to allow the Layer 2 and PPP endpoints to reside on different devices interconnected by a packet-switched network. With L2TP, a user has a Layer 2 connection to an access concentrator - LAC (e.g., modem bank, ADSL DSLAM, etc.), and the concentrator then tunnels individual PPP frames to the Network Access Server - NAS. This allows the actual processing of PPP packets to be separated from the termination of the Layer 2 circuit. From the user's perspective, there is no functional difference between having the L2 circuit terminate in a NAS directly or using L2TP.

It may also be useful to use L2TP just as any other tunneling protocol with or without encryption. The L2TP standard says that the most secure way to encrypt data is using L2TP over IPsec (Note that it is default mode for Microsoft L2TP client) as all L2TP control and data packets for a particular tunnel appear as homogeneous UDP/IP data packets to the IPsec system.

Multilink PPP (MP) is supported in order to provide MRRU (the ability to transmit full-sized 1500 and larger packets) and bridging over PPP links (using Bridge Control Protocol (BCP) that allows to send raw Ethernet frames over PPP links). This way it is possible to setup bridging without EoIP. The bridge should either have an administratively set MAC address or an Ethernet-like interface in it, as PPP links do not have MAC addresses.

L2TP includes PPP authentication and accounting for each L2TP connection. Full authentication and accounting of each connection may be done through a RADIUS client or locally.

MPPE 40bit RC4 and MPPE 128bit RC4 encryption are supported.

L2TP traffic uses UDP protocol for both control and data packets. UDP port 1701 is used only for link establishment, further traffic is using any available UDP port (which may or may not be 1701). This means that L2TP can be used with most firewalls and routers (even with NAT) by enabling UDP traffic to be routed through the firewall or router.

L2TP Client

Sub-menu: `/interface l2tp-client`

Property	Description
add-default-route (*yes* | *no*; Default: **no**)	Whether to add L2TP remote address as a default route.
allow (*mschap2* | *mschap1* | *chap* | *pap*; Default: **mschap2, mschap1, chap, pap**)	Allowed authentication methods.
connect-to (*IP*; Default:)	Remote address of L2TP server
dial-on-demand (*yes* | *no*; Default: **no**)	
disabled (*yes* | *no*; Default: **yes**)	Whether interface is disabled or not. By default it is disabled
max-mru (*integer*; Default: **1460**)	Maximum Receive Unit. Max packet size that L2TP interface will be able to receive without packet fragmentation.
max-mtu (*integer*; Default: **1460**)	Maximum Transmission Unit. Max packet size that L2TP interface will be able to send without packet fragmentation.
mrru (*disabled* | *integer*; Default: **disabled**)	Maximum packet size that can be received on the link. If a packet is bigger than tunnel MTU, it will be split into multiple packets, allowing full size IP or Ethernet packets to be sent over the tunnel. Read more >>
name (*string*; Default:)	Descriptive name of the interface.
password (*string*; Default: "")	Password used for authentication.
profile (*name*; Default: **default-encryption**)	Used PPP profile.
user (*string*; Default:)	User name used for authentication.

This example demonstrates how to set up L2TP client with username "l2tp-hm", password "123" and server 10.1.101.100

```
[admin@dzeltenais_burkaans] /interface l2tp-client>add name=l2tp-hm user=l2tp-hm password=123 \
\... connect-to=10.1.101.100 disabled=no
[admin@dzeltenais_burkaans] /interface l2tp-client> print detail
Flags: X - disabled, R - running
 0    name="l2tp-hm" max-mtu=1460 max-mru=1460 mrru=disabled
      connect-to=10.1.101.100 user="l2tp-hm" password="123"
      profile=default-encryption add-default-route=no dial-on-demand=no
      allow=pap,chap,mschap1,mschap2
```

L2TP Server

Sub-menu: `/interface l2tp-server`

This sub-menu shows interfaces for each connected L2TP clients.

An interface is created for each tunnel established to the given server. There are two types of interfaces in L2TP server's configuration

- Static interfaces are added administratively if there is a need to reference the particular interface name (in firewall rules or elsewhere) created for the particular user.
- Dynamic interfaces are added to this list automatically whenever a user is connected and its username does not match any existing static entry (or in case the entry is active already, as there can not be two separate tunnel interfaces referenced by the same name).

Dynamic interfaces appear when a user connects and disappear once the user disconnects, so it is impossible to reference the tunnel created for that use in router configuration (for example, in firewall), so if you need a persistent rules for that user, create a static entry for him/her. Otherwise it is safe to use dynamic configuration.

Note: in both cases PPP users must be configured properly - static entries do not replace PPP configuration.

Sub-menu: `/interface l2tp-server server`

Properties:

Property	Description
authentication (*pap* | *chap* | *mschap1* | *mschap2*; Default: **mschap1,mschap2**)	Authentication methods that server will accept.
default-profile (*name*; Default: **default-encryption**)	
enabled (*yes* | *no*; Default: **no**)	Defines whether L2TP server is enabled or not.
max-mru (*integer*; Default: **1460**)	Maximum Receive Unit. Max packet size that L2TP interface will be able to receive without packet fragmentation.
max-mtu (*integer*; Default: **1460**)	Maximum Transmission Unit. Max packet size that L2TP interface will be able to send without packet fragmentation.
mrru (*disabled* | *integer*; Default: **disabled**)	Maximum packet size that can be received on the link. If a packet is bigger than tunnel MTU, it will be split into multiple packets, allowing full size IP or Ethernet packets to be sent over the tunnel. `Read more >>`

To enable L2TP server:

```
[admin@MikroTik] interface l2tp-server server> set enabled=yes
[admin@MikroTik] interface l2tp-server server> print
          enabled: yes
          max-mtu: 1460
          max-mru: 1460
             mrru: disabled
   authentication: pap,chap,mschap1,mschap2
  default-profile: default-encryption
[admin@MikroTik] interface l2tp-server server>
```

Monitoring

Monitor command can be used to monitor status of the tunnel on both client and server.

```
[admin@dzeltenais_burkaans] /interface l2tp-client> monitor 0
     status: "connected"
     uptime: 7h24m18s
  idle-time: 6h21m4s
   encoding: "MPPE128 stateless"
        mtu: 1460
        mru: 1460
```

Read-only properties

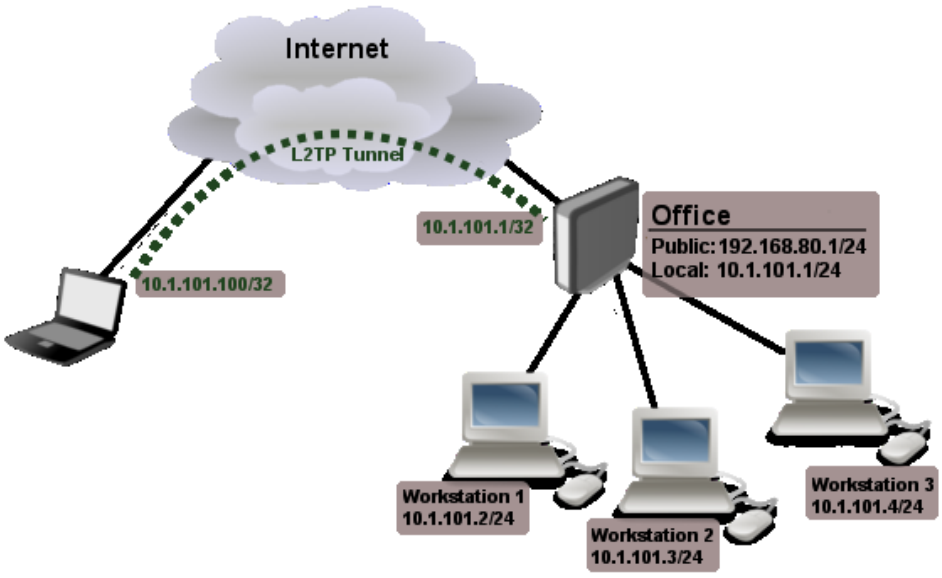
Property	Description
status ()	Current L2TP status. Value other than "connected" indicates that there are some problems estabising tunnel. • **dialing** - attempting to make a connection • **verifying password** - connection has been established to the server, password verification in progress • **connected** - tunnel is successfully established • **terminated** - interface is not enabled or the other side will not establish a connection
uptime (*time*)	Elapsed time since tunnel was established.
idle-time (*time*)	Elapsed time since last activity on the tunnel.
encoding ()	Used encryption method
mtu (*integer*)	Negotiated and used MTU
mru (*integer*)	Negotiated and used MRU

Application Examples

Connecting Remote Client

The following example shows how to connect a computer to a remote office network over L2TP encrypted tunnel giving that computer an IP address from the same network as the remote office has (without need of bridging over EoIP tunnels)

Consider following setup



Office router is connected to internet through **ether1**. Workstations are connected to **ether2**. Laptop is connected to the internet and can reach Office router's public IP (in our example it is 192.168.80.1).

First step is to create a user

```
[admin@RemoteOffice] /ppp secret> add name=Laptop service=l2tp password=123
local-address=10.1.101.1 remote-address=10.1.101.100
[admin@RemoteOffice] /ppp secret> print detail
Flags: X - disabled
  0   name="Laptop" service=l2tp caller-id="" password="123" profile=default
```

```
      local-address=10.1.101.1 remote-address=10.1.101.100 routes==""

[admin@RemoteOffice] /ppp secret>
```

Notice that L2TP local address is the same as routers address on local interface and remote address is form the same range as local network (10.1.101.0/24).

Next step is to enable L2TP server and L2TP client on the laptop.

```
[admin@RemoteOffice] /interface l2tp-server server> set enabled=yes
[admin@RemoteOffice] /interface l2tp-server server> print
            enabled: yes
            max-mtu: 1460
            max-mru: 1460
               mrru: disabled
     authentication: mschap2
    default-profile: default-encryption
[admin@RemoteOffice] /interface l2tp-server server>
```

L2TP client from the laptop should connect to routers public IP which in our example is 192.168.80.1.
Please, consult the respective manual on how to set up a L2TP client with the software You are using.

Note: By default Windows sets up L2TP with IPsec. To disable IpSec registry modifications are required. `Read more >>`

At this point (when L2TP client is successfully connected) if you will try to ping any workstation form the laptop, ping will time out, because Laptop is unable to get ARPs from workstations. Solution is to set up `proxy-arp` on local interface

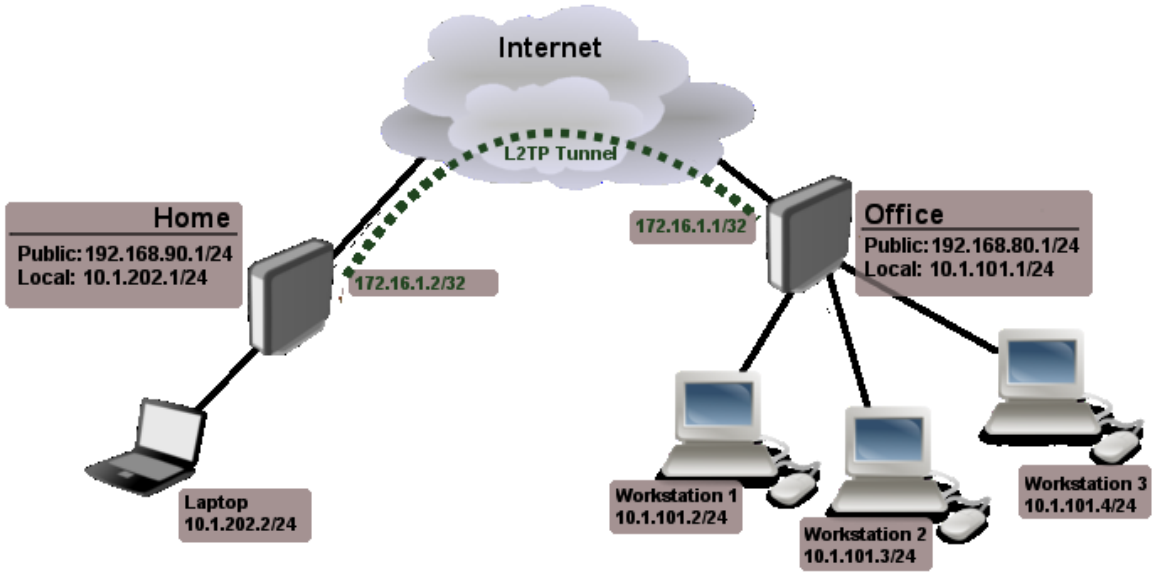
```
[admin@RemoteOffice] interface ethernet> set ether2 arp=proxy-arp
[admin@RemoteOffice] interface ethernet> print
Flags: X - disabled, R - running
  #    NAME                 MTU   MAC-ADDRESS         ARP
  0  R ether1               1500  00:30:4F:0B:7B:C1 enabled
  1  R ether2               1500  00:30:4F:06:62:12 proxy-arp
[admin@RemoteOffice] interface ethernet>
```

After `proxy-arp` is enabled client can successfully reach all workstations in local network behind the router.

Site-to-Site L2TP

The following is an example of connecting two Intranets using L2TP tunnel over the Internet.

Consider following setup



Office and Home routers are connected to internet through **ether1**, workstations and laptops are connected to **ether2**. Both local networks are routed through L2TP client, thus they are not in the same broadcast domain. If both networks should be in the same broadcast domain then you need to use BCP and bridge L2TP tunnel with local interface.

First step is to create a user

```
[admin@RemoteOffice] /ppp secret> add name=Home service=l2tp password=123
local-address=172.16.1.1 remote-address=172.16.1.2 routes="10.1.101.0/24 172.16.1.1 1"
[admin@RemoteOffice] ppp secret> print detail
Flags: X - disabled
  0   name="Home" service=l2tp caller-id="" password="123" profile=default
      local-address=172.16.1.1 remote-address=172.16.1.2 routes=="10.1.101.0/24 172.16.1.1 1"

[admin@RemoteOffice] /ppp secret>
```

Notice that we set up L2TP to add route whenever client connects. If this option is not set, then you will need static routing configuration on the server to route traffic between sites through L2TP tunnel.

Next step is to enable L2TP server on the office router and configure L2TP client on the Home router.

```
[admin@RemoteOffice] /interface l2tp-server server> set enabled=yes
[admin@RemoteOffice] /interface l2tp-server server> print
            enabled: yes
            max-mtu: 1460
            max-mru: 1460
               mrru: disabled
     authentication: mschap2
    default-profile: default-encryption
[admin@RemoteOffice] /interface l2tp-server server>
```

```
[admin@Home] /interface l2tp-client> add user=Home password=123 connect-to=192.168.80.1 disabled=no
[admin@Home] /interface l2tp-client> print
Flags: X - disabled, R - running
 0 R  name="l2tp-out1" max-mtu=1460 max-mru=1460 mrru=disabled connect-to=192.168.80.1 user="Home"
       password="123" profile=default-encryption add-default-route=no dial-on-demand=no
       allow=pap,chap,mschap1,mschap2
[admin@Home] /interface l2tp-client>
```

Now we need to add route to reach local network behind Home router

```
[admin@RemoteOffice] /ip route> add dst-address=10.1.202.0/24 gateway=172.16.1.2
```

After tunnel is established and routes are set, you should be able to ping remote network.

Read More

- BCP (Bridge Control Protocol)
- Disable IpSec used with L2TP on Windows [1]
- MikroTik RouterOS and Windows XP IPSec/L2TP

[Top | Back to Content]

References

[1] http://support.microsoft.com/default.aspx?scid=kb%3Ben-us%3B258261.php

Manual:License

Overview

RouterBOARD devices come preinstalled with a RouterOS license, if you have purchased a RouterBOARD device, nothing must be done regarding the license.

For X86 systems (ie. PC devices), you need to obtain a license key.

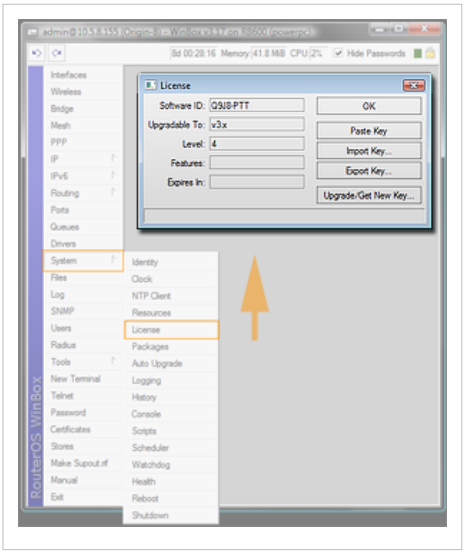
The license key is a block of symbols that needs to be copied from your mikrotik.com account, or from the email you received in, and then it can be pasted into the router. You can paste the key anywhere in the terminal, or by clicking "Paste key" in Winbox License menu. A reboot is required for the key to take effect.

RouterOS licensing scheme is based on SoftwareID number that is bound to storage media (HDD, NAND).

Licensing information can be read from CLI system console:

```
[admin@RB1100] > /system license print
    software-id: "43NU-NLT9"
  upgradable-to: v7.x
         nlevel: 6
       features:
[admin@RB1100] >
```

or from equivalent winbox, webfig menu.



License Levels

You can purchase a Level 3, 4, 5 and 6. Level 1 is the demo license. The difference between license levels is shown in the table.

Level 3 is a wireless station (client) only license. Level 3 can only be obtained in large quantities.

Level 2 was a transitional license from old legacy (pre 2.8) license format. These licenses are not available anymore, if you have this kind of license, it will work, but to upgrade it - you will have to purchase a new license.

Note: current RouterOS version is **5** table modified according to that. The Upgradable-to below applies only to Keys purchased after release of v5

Level number	0 (Demo mode)	1 (Free)	3 (WISP CPE)	4 (WISP)	5 (WISP)	6 (Controller)
Price	no key [1]	registration required [1]	volume only [1]	$45	$95	$250
Upgradable To	-	no upgrades	ROS v6.x	ROS v6.x	ROS v7.x	ROS v7.x
Initial Config Support	-	-	-	15 days	30 days	30 days
Wireless AP	24h trial	-	-	yes	yes	yes
Wireless Client and Bridge	24h trial	-	yes	yes	yes	yes
RIP, OSPF, BGP protocols	24h trial	-	yes(*)	yes	yes	yes
EoIP tunnels	24h trial	1	unlimited	unlimited	unlimited	unlimited
PPPoE tunnels	24h trial	1	200	200	500	unlimited
PPTP tunnels	24h trial	1	200	200	500	unlimited
L2TP tunnels	24h trial	1	200	200	500	unlimited
OVPN tunnels	24h trial	1	200	200	unlimited	unlimited
VLAN interfaces	24h trial	1	unlimited	unlimited	unlimited	unlimited
HotSpot active users	24h trial	1	1	200	500	unlimited
RADIUS client	24h trial	-	yes	yes	yes	yes
Queues	24h trial	1	unlimited	unlimited	unlimited	unlimited
Web proxy	24h trial	-	yes	yes	yes	yes
User manager active sessions	24h trial	1	10	20	50	Unlimited
Number of KVM guests	none	1	Unlimited	Unlimited	Unlimited	Unlimited

(*) - BGP is included in License Level3 only for RouterBOARDs, for other devices you need Level4 or above to have BGP.

All Licenses:

- never expire
- include 15-30 day free support over e-mail
- can use unlimited number of interfaces

- are for one installation each
- Level3 is not available for purchase individually. For ordering more than 100 L3 licenses, contact sales[at]mikrotik.com

Licenses and RouterOS upgrades

RouterOS upgrade capabilities are not limited by time, but by version, and this depends on the RouterOS license level. For example if you are running RouterOS v5, your license could restrict the upgrade only to v6, and not to v7. The following examples describe how this is determined:

- There are two types of keys, Level3/L4 and Level5/L6
- The difference between these is that L3 and L4 only allow RouterOS upgrades until the last update of the next version. L5 and L6 however, give you the ability to use one more major version
- There are also differences between all License levels (L3-L6) that are unrelated to RouterOS upgrades, see License levels

So the math is:

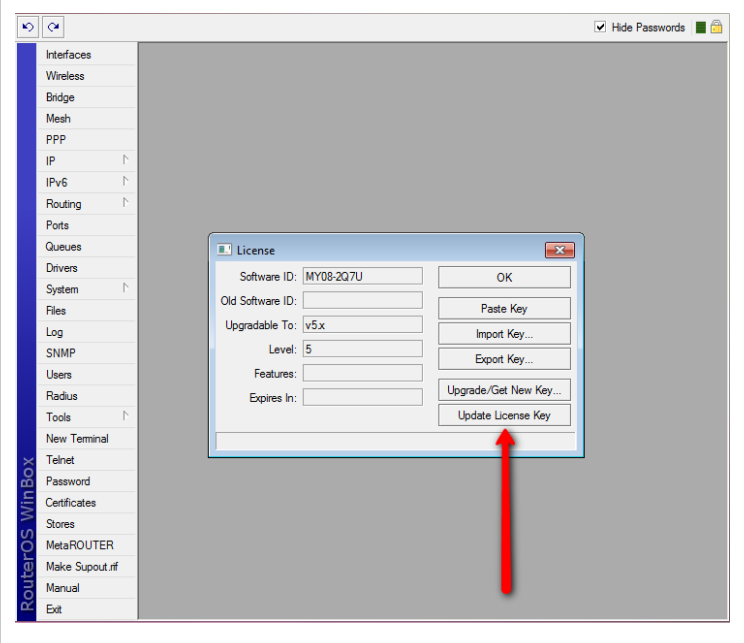
- L3/4 = current version + 1 = can use
- L5/6 = current version + 2 = can use

eg. L5/6 = v3 + 2 = v5.21 you can use

Examples:

- If current version is ROS v3, L3 and L4 will work with v3.1, v3.20, v4,1, v4.20 but NOT v5.0 and beyond
- If current version is ROS v3, L5 and L6 will work with v3.1, v3.20, v4.1, v4.20 and also v5beta1 but NOT v6.0 and beyond
- If current version would be **ROS v4**, L5 and L6 will work with v4.1, v4.20, v5.1, v5.20 and also v6beta to v6.99 but NOT v7

New 8 symbol SoftID



Since RouterOS 3.25 and 4.0beta3 **new SoftID format** is introduced. Your license menu will show both the old and the new SoftID. Even by upgrading to a new version, RouterOS will **still work as before**, but to use some of the new features, LICENSE UPDATE will be necessary. To do this, just click on "Update license key" button in Winbox (currently only in Winbox).

New SoftID's are in the form of XXXX-XXXX (Four symbols, dash, four symbols).

The following actions will be taken:

1. Winbox will contact www.mikrotik.com with your old SoftID
2. www.mikrotik.com will check the database and see details about your key
3. the server will generate a new key as "upgrade" and put it into the same account as old one
4. Winbox will receive the new key and automatically License your router with the new key

5. Reboot will be required
6. New RouterOS features will be unlocked

Important Note!: If you see this button also in v3.24, don't use it, it will not work.

If you ever wish to downgrade RouterOS, you will have to apply the OLD key before doing so. When RouterOS applies the NEW key, the OLD key is saved to a file, in the FILES folder, to make sure you have the old key handy.

Even more important: Don't downgrade v4.0b3 to v3.23 or older. Use only v3.24 for downgrading, or you might lose your new format key.

Change license Level

1. There are no license level upgrades, if you wish to use a different license Level, please purchase the appropriate level. Be very careful when purchasing for the first time, choose the correct option.
2. Why is it not possible to change license level (ie. upgrade license)? Just like you can't easily upgrade your car's engine from 2L to 4L just by paying the difference, you can't switch license levels as easily. This is a policy used by many software companies, choose wisely when making your purchase! Instead we have lowered the prices, and removed the software update time limit.

Using the License

Can I Format or Re-Flash the drive?

Formatting, and Re-Imaging the drive with non-mikrotik tools (like DD and Fdisk) will destroy your license! Be very careful and contact mikrotik support before doing this. It is not recommended, as mikrotik support might deny your request for a replacement license. For this use MikroTik provided tools Netinstall or CD-install that are freely available from our download page.

How many computers can I use the License on?

At the same time, the RouterOS license can be used only in one system. The License is bound to the HDD it is installed on, but you have the ability to move the HDD to another computer system. You cannot move the License to another HDD, neither can you format or overwrite the HDD with the RouterOS license. It will be erased from the drive, and you will have to get a new one. If you accidently removed your license, contact the support team for help.

Can I temporary use the HDD for something else, other than RouterOS?

As stated above, no.

Can I move the license to another HDD ?

If your current HDD drive is destroyed, or can no longer be used, it is possible to transfer the license to another HDD. You will have to request a replacement key (see below) which will cost 10$

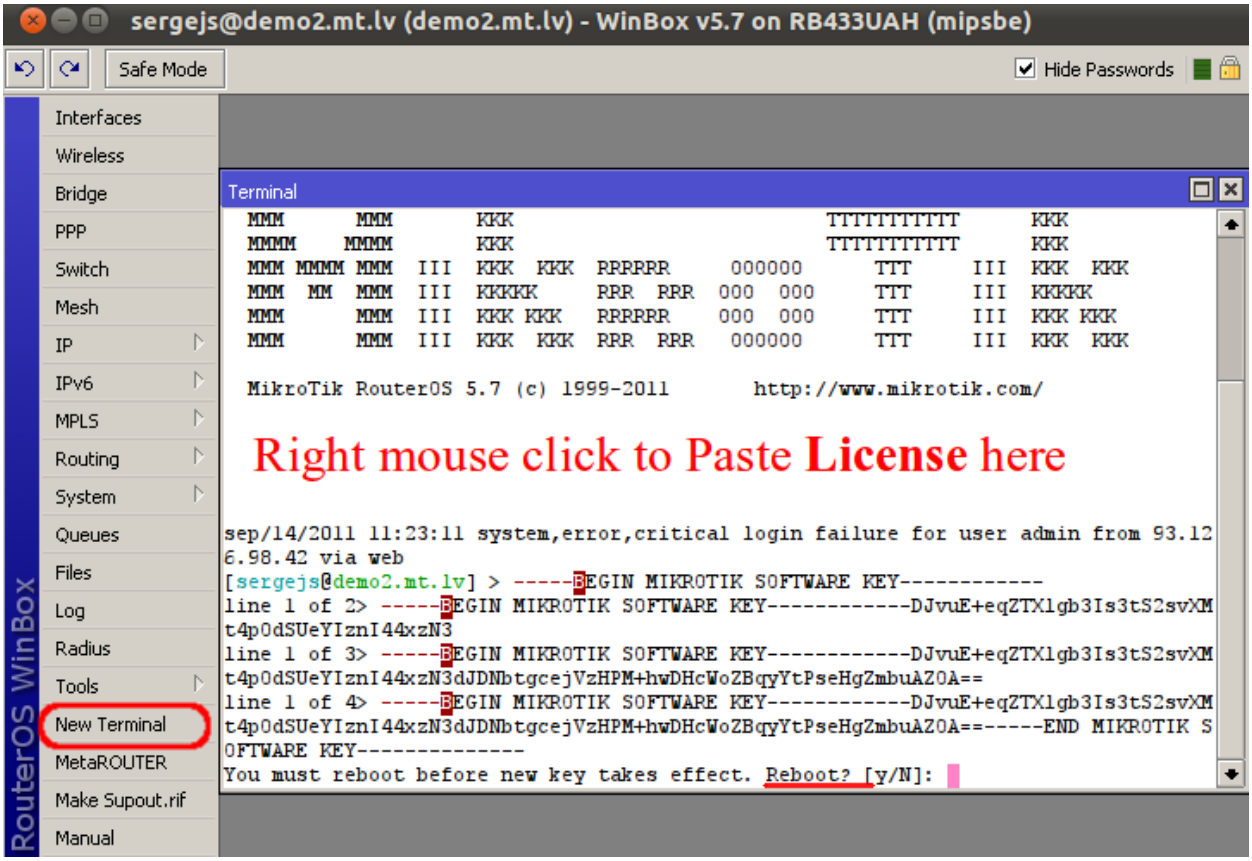
What is a Replacement Key

It is a special key which is issued by the Support Team if you accidently lose the license, and the Mikrotik Support decides that it is not directly your fault. It costs 10$ and has the same features as the key that you lose. Note that before issuing such key, the Mikrotik Support can ask you to prove that the old drive is failed, in some cases this means sending us the dead drive.

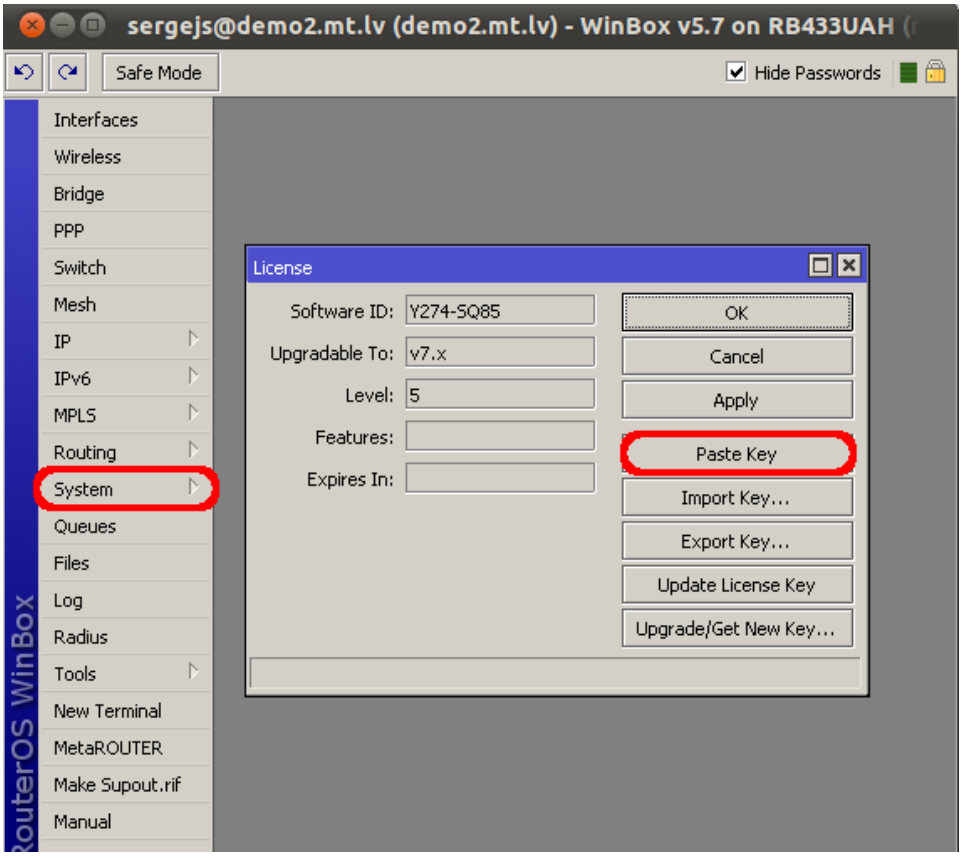
Must I type the whole key into the router?

No, simply copy it and paste into the Telnet window, or License menu in Winbox.

Copy license to Telnet Window (or Winbox New Terminal),



Another option to use Winbox License Window, click on **System** ---> **License**,



Can I install another OS on my drive and then install RouterOS again later?

No, because if you use formatting or partitioning utilities, or tools that do something to the MBR, you will lose the license and you will have to make a new one. This process is not free (see Replacement Key above)

I lost my RouterBOARD, can you give me the license to use on another system?

The RouterBOARD comes with an embedded license. You cannot move this license to a new system in any way, this includes upgrades applied to the RouterBOARD while it was still working.

Licenses Purchased from Resellers

The keys that you purchase from other vendors and resellers, are not in your account. Your mikrotik.com account only contains licenses purchased from MikroTik directly. However, you can use the "Request key" link in your account, to get the key into your account for reference, or for some upgrades (if available).

Obtaining Licenses and working with them

Where can I buy a RouterOS license key?

In the Account Server, which is located on www.mikrotik.com

If I have purchased my key elsewhere

You must contact the company who sold you the license, they will provide support

If I have a license and want to put it on another account?

You can give access to keys with the help of Virtual Folders

References

[1] mailto:sales@mikrotik.com

Manual:Lua

Summary

- Version 4.0beta3 introduces preliminary support for Lua scripting language [1]. Integration with console is still in progress.
- RouterOS v4 RC1 removes Lua support indefinetly

Changes in console

- ':' and '/' namespaces are merged. Lookup rules have been changed so as not to affect existing scripts:
  - Without leading ':' or '/' names are looked up starting from the current path.
  - With leading ':' and '/' names are looked up starting from the root of the hierarchy.
  - With leading '/' current path of all subcommands is set to the path of command.
  - With leading ':' current path of subcommands is kept the same as the current path of command.

Example:

```
/ip address { #changes current path
    print   #/ip address print
    :queue simple print where interface=[get 0 interface]
        #this 'get' is '/ip address get', because
        #leading ':' does not change current path
        #for subcommands
}
```

- Value of type 'nothing' is gone.
- Command 'nothing' returns same value as the empty command '[]'. It is kept for compatibility with earlier versions.
- Value of type 'error' is gone, console error handling is unified with the Lua error hadling.
- 'error' command now immedeately raises an exception.
- New value type 'lua', holds arbitrary Lua values.
- New command 'lua' that returns lua function created from the given source string.
- Command can now also be a variable substitution, an expression or a command substitution. Result is evaluated. Example:

```
global a [parse "/interface print"]; $a
($a)
[parse "/interface print"]
[lua "io.write 'this works\\n'"]
```

- Global variables are shared between Lua and console scripts.
- There is no ':log' command anymore, it is replace by four commands 'log info', 'log error', 'log warning', 'log debug'. This change was necessary because of the root namespace merge. It preserves compatibility with previous scripts, but note that name of log topic cannot be a result of an expression.
- '/system script' items has new property 'language' that can be either 'cmd' or 'lua'. 'cmd' is for console scripts, 'lua' is for Lua scripts.
- New operator '%' that computes remainder.
- Logical operators now treat all values except nil and 'false' as 'true'. Note that empty string, empty array, number 0, IP address 0.0.0.0 and similar values are treated as a 'true', this is consistent with the Lua behaviour. Previously

values of non-boolean type were causing an error.

- Logical 'and' and 'or' operators ('&&' and '||') now use shortcut evaluation. If left hand value is sufficient for computing the operation, it is returned and the right hand value is not computed. Otherwise, operation returns the right hand value. Example:

```
put (9 or (1 / 0)) #prints 9, division is not computed
```

Changes in Lua compared to the standard release

- Lua base version is 5.1.4
- Number type is 64 bit signed integer. Floating point constants are not supported. Exponentiation does not work with negative exponents.
- Following patches are applied:
  - Byte swapping for loading bytecode [2].
  - Patch by Thierry Grellier that adds '&' '|' '^^' '<<' '>>' '~' bitwise operators. Integer division operator is not included. [3]
- Following libraries are included:
  - bitlib [4], adapted for 64 bit integer numbers.
  - md5 1.1.2 [5]
  - lpeg 0.9 [6]
- Following features of standard libraries are not available:
  - 'io.popen', 'io.pclose', 'io.tmpfile'.
  - 'os.execute', 'os.tmpname', 'os.getenv', 'os.setlocale', 'os.exit'.
  - 'debug' library.
  - All functions and constants from 'math', except 'math.abs', 'math.ceil', 'math.floor', 'math.max', 'math.min' and 'math.random'.
  - 'print'
- Following changes are made to standard functions:
  - 'pairs' now calls "__pairs" metamethod of it's argument, and falls back to 'rawpairs' call. Original 'pairs' is now renamed to 'rawpairs'. These changes allow simple way of implementing default iteration behaviour for objects, by providing "__pairs" metamethod.
  - math.random without arguments can return any 64 bit integer number. All bits of the result are random.
- "/LIB" is a virtual path that contains additional libraries. This path is not accessible for file operations.

References

[1] http://www.lua.org/
[2] http://lua-users.org/lists/lua-l/2006-02/msg00507.html
[3] http://lua-users.org/wiki/LuaPowerPatches
[4] http://luaforge.net/projects/bitlib
[5] http://www.keplerproject.org/md5
[6] http://www.inf.puc-rio.br/~roberto/lpeg/lpeg.html

Manual:System/LEDS

Applies to RouterOS: v5 +

Summary

Sub-menu: `/system leds`
Standards:

RouterOS allows to configure each leds activity the way that user wishes. It is possible to configure the leds to display wireless strength, blink the leds on interface traffic activity and many other options.

For example default led configuration on Groove

```
[admin@MikroTik] /system leds> print
Flags: X - disabled
 #   TYPE                      INTERFACE                 LEDS
 0   wireless-signal-strength                            led1
                                                         led2
                                                         led3
                                                         led4
                                                         led5
 1   interface-activity        ether1                    user-led
```

RB Groove uses five leds for wireless strength and one for ethernet activity monitoring.

Configuration Properties

Property	Description
disabled (*yes* | *no*; Default: **no**)	Whether item is disabled
interface (*string*; Default:)	Name of the interface which will be used for led status. Applicable only if **type** is interface specific.
modem-signal-treshold (*integer [-113..-51]*; Default:)	Applicable if type is **modem-signal**
leds (*list of leds*; Default:)	List of led names used for status report. For example wireless signal strength will require more than one led.

type (*flash-access* | *interface-activity* | *interface-receive* | *interface-status* | *interface-transmit* | *modem-signal* | *wireless-signal-strength* | *wireless-status*; Default:)

Type of the status:

- `flash-access` - blink the led on flash access
- `interface-activity` - blink the led on interface (traffic) activity
- `interface-receive` - blink the led on interface received traffic
- `interface-status` - light the led on interface status change
- `interface-transmit` - blink the led on interface transmitted traffic
- `modem-signal` - blink the led on modem signal
- `wireless-signal-strength` - light the leds displaying wireless signal (requires more than one led)
- `wireless-status` - light the led on wireless status change.

Basic examples

[Top | Back to Content]

Manual:System/Log

Applies to RouterOS: v3, v4 +

Summary

RouterOS is capable of logging various system events and status information. Logs can be saved in routers memory (RAM), disk, file, sent by email or even sent to remote syslog server (RFC 3164).

Log messages

Sub-menu level: `/log`

All messages stored in routers local memory can be printed from `/log` menu. Each entry contains time and date when event occurred, topics that this message belongs to and message itself.

```
[admin@ZalaisKapots] /log> print
jan/02/1970 02:00:09 system,info router rebooted
sep/15 09:54:33 system,info,account user admin logged in from 10.1.101.212 via winbox
sep/15 12:33:18 system,info item added by admin
sep/15 12:34:26 system,info mangle rule added by admin
sep/15 12:34:29 system,info mangle rule moved by admin
sep/15 12:35:34 system,info mangle rule changed by admin
sep/15 12:42:14 system,info,account user admin logged in from 10.1.101.212 via telnet
sep/15 12:42:55 system,info,account user admin logged out from 10.1.101.212 via telnet
01:01:58 firewall,info input: in:ether1 out:(none), src-mac 00:21:29:6d:82:07, proto UDP,
                          10.1.101.1:520->10.1.101.255:520, len 452
```

If logs are printed at the same date when log entry was added, then only time will be shown. In example above you can see that second message was added on sep/15 current year (year is not added) and the last message was added today so only the time is displayed.

Note: print command accepts several parameters that allows to detect new log entries, print only necessary messages and so on. For more information about parameters refer to scripting manual

For example following command will print all log messages where one of the topics is info and will detect new log entries until Ctrl+C is pressed

```
[admin@ZalaisKapots] /log > print follow where topics~".info"
12:52:24 script,info hello from script
-- Ctrl-C to quit.
```

If print is in follow mode you can hit 'space' on keyboard to insert separator:

```
[admin@ZalaisKapots] /log > print follow where topics~".info"
12:52:24 script,info hello from script

 = = =   = = =   = = =      = = =   = = =   = = =      = = =   = = =   = = =

-- Ctrl-C to quit.
```

Logging configuration

Sub-menu level: `/system logging`

Property	Description
`action` (*name*; Default: **memory**)	specifies one of the system default actions or user specified action listed in actions menu
`prefix` (*string*; Default:)	prefix added at the beginning of log messages
`topics` (*account, async, backup, bgp, calc, critical, ddns, debug, dhcp, e-mail, error, event, firewall, gsm, hotspot, igmp-proxy, info, ipsec, iscsi, isdn, l2tp, ldp, manager, mme, mpls, ntp, ospf, ovpn, packet, pim, ppp, pppoe, pptp, radius, radvd, raw, read, rip, route, rsvp, script, sertcp, state, store, system, telephony, tftp, timer, ups, warning, watchdog, web-proxy, wireless, write*; Default: **info**)	log all messages that falls into specified topic or list of topics. '**!**' character can be used before topic to exclude messages falling under this topic. For example, we want to log NTP debug info without too much details: `/system logging add topics=ntp,debug,!packet`

Actions

Sub-menu level: `/system logging action`

Property	Description
bsd-syslog (*yes|no*; Default:)	whether to use bsd-syslog as defined in RFC 3164
disk-file-count (*integer [1..65535]*; Default: **2**)	specifies number of files used to store log messages, applicable only if action=disk
disk-file-name (*string*; Default: **log**)	name of the file used to store log messages, applicable only if action=disk
disk-lines-per-file (*integer [1..65535]*; Default: **100**)	specifies maximum size of file in lines, applicable only if action=disk
disk-stop-on-full (*yes|no*; Default: **no**)	whether to stop to save log messages to disk after the specified disk-lines-per-file and disk-file-count number is reached, applicable only if action=disk
email-to (*string*; Default:)	email address where logs are sent, applicable only if action=email
memory-lines (*integer [1..65535]*; Default: **100**)	number of records in local memory buffer, applicable only if action=memory
memory-stop-on-full (*yes|no*; Default: **no**)	whether to stop to save log messages in local buffer after the specified memory-lines number is reached
name (*string*; Default:)	name of an action
remember (*yes|no*; Default:)	whether to keep log messages, which have not yet been displayed in console, applicable if action=echo
remote (*IP/IPv6 Address[:Port]*; Default: **0.0.0.0:514**)	remote logging server's IP/IPv6 address and UDP port, applicable if action=remote
src-address (*IP address*; Default: **0.0.0.0**)	source address used when sending packets to remote server
syslog-facility (*auth, authpriv, cron, daemon, ftp, kern, local0, local1, local2, local3, local4, local5, local6, local7, lpr, mail, news, ntp, syslog, user, uucp*; Default: **daemon**)	
syslog-severity (*alert, auto, critical, debug, emergency, error, info, notice, warning*; Default: **auto**)	Severity level indicator defined in RFC 3164: • Emergency: system is unusable • Alert: action must be taken immediately • Critical: critical conditions • Error: error conditions • Warning: warning conditions • Notice: normal but significant condition • Informational: informational messages • Debug: debug-level messages
target (*disk, echo, email, memory, remote*; Default: **memory**)	storage facility or target of log messages • disk - logs are saved to the hard drive more>> • echo - logs are displayed on the console screen • email - logs are sent by email • memory - logs are stored in local memory buffer • remote - logs are sent to remote host

Note: default actions can not be deleted or renamed.

Topics

Each log entry have topic which describes the origin of log message. There can be more than one topic assigned to log message. For example, OSPF debug logs have four different topics: route, ospf, debug and raw.

```
11:11:43 route,ospf,debug SEND: Hello Packet 10.255.255.1 -> 224.0.0.5 on lo0
11:11:43 route,ospf,debug,raw PACKET:
11:11:43 route,ospf,debug,raw     02 01 00 2C 0A FF FF 03 00 00 00 00 E7 9B 00 00
11:11:43 route,ospf,debug,raw     00 00 00 00 00 00 00 00 FF FF FF FF 00 0A 02 01
11:11:43 route,ospf,debug,raw     00 00 00 28 0A FF FF 01 00 00 00 00
```

List of Facility independent topics

Topic	Description
critical	Log entries marked as critical, these log entries are printed to console each time you log in.
debug	Debug log entries
error	Error messages
info	Informative log entry
packet	Log entry that shows contents from received/sent packet
raw	Log entry that shows raw contents of received/sent packet
warning	Warning message.

Topics used by various RouterOS facilities

Topic	Description
account	Log messages generated by accounting facility.
async	Log messages generated by asynchronous devices
backup	Log messages generated by backup creation facility.
bfd	Log messages generated by Manual:Routing/BFD protocol
bgp	Log messages generated by Manual:Routing/BGP protocol
calc	Routing calculation log messages.
ddns	Log messages generated by Manual:Tools/Dynamic DNS tool
dhcp	DHCP client, server and relay log messages
e-mail	Messages generated by Manual:Tools/email tool.
event	Log message generated at routing event. For example, new route have been installed in routing table.
firewall	Firewall log messages generated when **action=log** is set in firewall rule
gsm	Log messages generated by GSM devices
hotspot	Hotspot related log entries
igmp-proxy	IGMP Proxy related log entries
ipsec	IpSec log entries
iscsi	
isdn	
l2tp	Log entries generated by Manual:Interface/L2TP client and server
ldp	Manual:MPLS/LDP protocol related messages
manager	User manager log messages.
mme	MME routing protocol messages
mpls	MPLS messages
ntp	sNTP client generated log entries

ospf	Manual:Routing/OSPF routing protocol messages
ovpn	OpenVPN tunnel messages
pim	Multicast PIM-SM related messages
ppp	ppp facility messages
pppoe	PPPoE server/client related messages
pptp	PPTP server/client related messages
radius	Log entries generated by RADIUS Client
radvd	IPv6 radv deamon log messages.
read	SMS tool messages
rip	RIP routing protocol messages
route	Routing facility log entries
rsvp	Resource Reservation Protocol generated messages.
script	Log entries generated from scripts
sertcp	Log messages related to facility responsible for "/ports remote-access"
simulator	
state	DHCP Client and routing state messages.
store	Log entries generated by Store facility
system	Generic system messages
telephony	
tftp	TFTP server generated messages

timer	Log messages that are related to timers used in RouterOS. For example bgp keepalive logs
	`12:41:40 route,bgp,debug,timer KeepaliveTimer expired`
	`12:41:40 route,bgp,debug,timer RemoteAddress=2001:470:1f09:131::1`

ups	Messages generated by UPS monitoring tool
watchdog	Watchdog generated log entries
web-proxy	Log messages generated by web proxy
wireless	M:Interface/Wireless log entries.
write	SMS tool messages.

Logging to file

To log everything to file, add new log action:

```
/system logging action add name=file target=disk disk-file-name=log
```

and then make everything log using this new action:

```
/system logging action=file
```

You can log only errors there by issuing command:

```
/system logging topics=error action=file
```

This will log into files **log.0.txt** and **log.1.txt**.

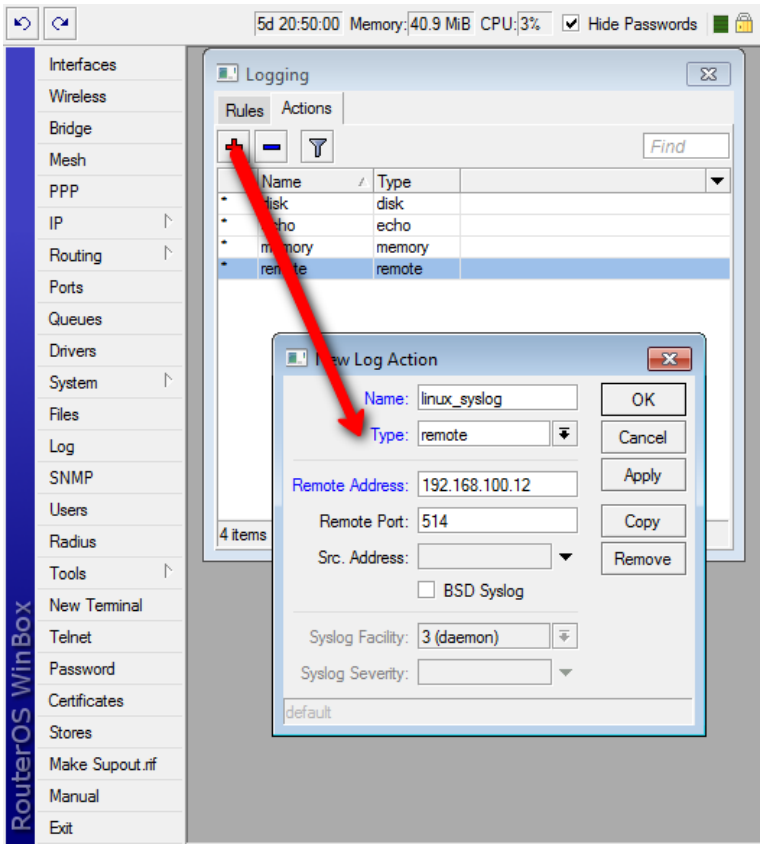
You can specify maximum size of file in lines by specifying *disk-lines-per-file*. **<file>.0.txt** is active file were new logs are going to be appended and once it size will reach maximum it will become **<file>.1.txt**, and new empty **<file>.0.txt** will be created.

You can log into USB flashes or into *MicroSD/CF* (on Routerboards) by specifying it's directory name before file name. For example, if you have accessible usb flash as **usb1** directory under */files*, you should issue following command:

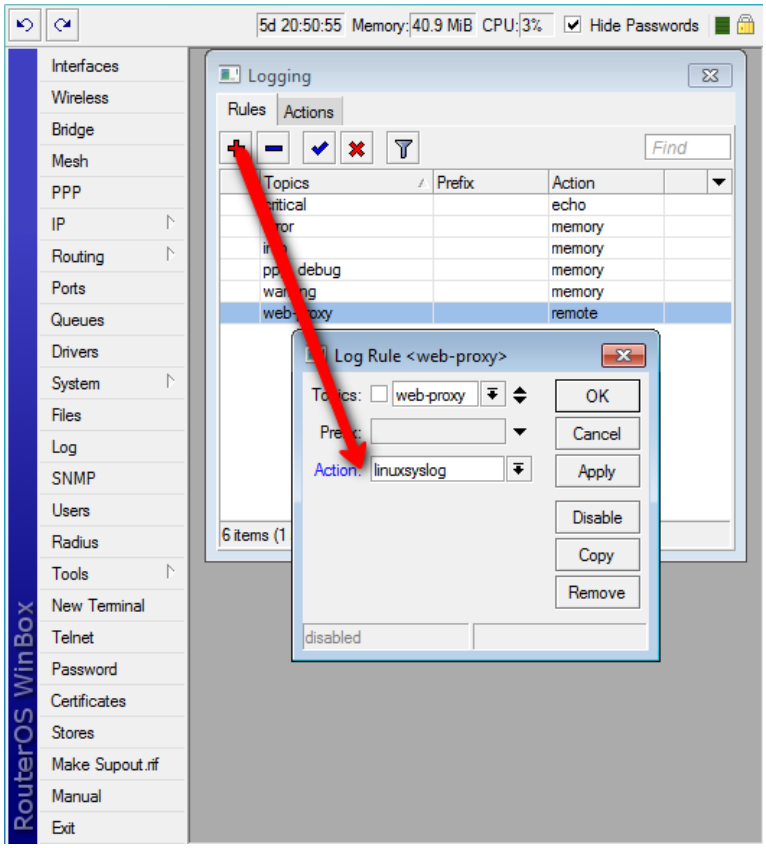
```
/system logging action add name=usb target=disk disk-file-name=usb1/log
```

Example:Webproxy logging

These two screenshots will show you how to configure the RouterOS logging facility to send Webrpoxy logs to a remote syslog server, in this example, located at 192.168.100.12. The syslog server can be any software that supports receiving syslogs, for example Kiwi syslog.

- 

Add a new logging action, with "remote" and the IP of the remote server. Call it whatever you like

- 

Then add a new logging rule with the topic "webproxy" and then newly created action. Note that you must have webproxy running on this router already, for this to work. To test, you can temporary change the action to "memory" and see the "log" window if the webproxy visited websites are logged. If it works, change it back to your new remote action

Note: it's a good idea to add another topic in the same rule: ***!debug****. This would be to ensure you don't get any debug stuff, only the visited sites.*

Manual:System/UPS

Applies to RouterOS: v3, v4 +

Summary

Sub-menu: `/system ups`
Standards: `APC Smart Protocol` [1]

The UPS monitor feature works with APC UPS units that support “smart” signaling over serial RS232 or USB connection. This feature enables the network administrator to monitor the UPS and set the router to ‘gracefully’ handle any power outage with no corruption or damage to the router. The basic purpose of this feature is to ensure that the router will come back online after an extended power failure. To do this, the router will monitor the UPS and set itself to hibernate mode when the utility power is down and the UPS battery is has less than 10% of its battery power left. The router will then continue to monitor the UPS (while in hibernate mode) and then restart itself after when the utility power returns. If the UPS battery is drained and the router loses all power, the router will power back to full operation when the ‘utility’ power returns.

The UPS monitor feature on the MikroTik RouterOS supports

- hibernate and safe reboot on power and battery failure
- UPS battery test and run time calibration test
- monitoring of all "smart" mode status information supported by UPS
- logging of power changes

The serial APC UPS (BackUPS Pro or SmartUPS) requires a special serial cable (unless connected with USB). If no cable came with the UPS, a cable may be ordered from APC or one can be made "in-house". Use the following diagram:

Router Side (DB9f)	Signal	Direction	UPS Side (DB9m)
2	Receive	IN	2
3	Send	OUT	1
5	Ground		4
7	CTS	IN	6

General Properties

Property	Description
alarm-setting (*delayed* | *immediate* | *low-battery* | *none*; Default: **immediate**)	UPS sound alarm setting: • `delayed` - alarm is delayed to the on-battery event • `immediate` - alarm immediately after the on-battery event • `low-battery` - alarm only when the battery is low • `none` - do not alarm
min-runtime (*time*; Default: **5m**)	Minimal run time remaining. After a 'utility' failure, the router will monitor the runtime-left value. When the value reaches the min-runtime value, the router will go to hibernate mode. • 0 - the router will go to hibernate mode when the "battery low" signal is sent indicating that the battery power is below 10%
offline-time (*time*; Default: **5m**)	How long to work on batteries. The router waits that amount of time and then goes into hibernate mode until the UPS reports that the 'utility' power is back • 0 - the router will go into hibernate mode according the min-runtime setting and 10% of battery power event. In this case, the router will wait until the UPS reports that the battery power is below 10%
port (*string*; Default:)	Communication port of the router.

Read-only properties:

Property	Description
load (*percent*)	The UPS's output load as a percentage of full rated load in Watts. The typical accuracy of this measurement is ±3% of the maximum of 105%
manufacture-date (*string*)	UPS's date of manufacture in the format "mm/dd/yy" (month, day, year).
model (*string*)	Less than 32 ASCII character string consisting of the UPS model name (the words on the front of the UPS itself)
nominal-battery-voltage (*integer*)	UPS's nominal battery voltage rating (this is not the UPS's actual battery voltage)
offline-after (*time*)	When will the router go offline
serial (*string*)	A string of at least 8 characters directly representing the UPS's serial number as set at the factory. Newer SmartUPS models have 12-character serial numbers
version (*string*)	UPS version, consists of three fields: SKU number, firmware revision, country code. The county code may be one of the following: • I - 220/230/240 Vac • D - 115/120 Vac • A - 100 Vac • M - 208 Vac • J - 200 Vac

Note: In order to enable UPS monitor, the serial port should be available.

Example

To enable the UPS monitor for port serial1:

```
[admin@MikroTik] system ups> add port=serial1 disabled=no
[admin@MikroTik] system ups> print
Flags: X - disabled, I - invalid
 0    name="ups" port=serial1 offline-time=5m min-runtime=5m
      alarm-setting=immediate model="SMART-UPS 1000" version="60.11.I"
```

```
      serial="QS0030311640" manufacture-date="07/18/00"
      nominal-battery-voltage=24V
[admin@MikroTik] system ups>
```

Runtime Calibration

Command: `/system ups rtc <id>`

The rtc command causes the UPS to start a run time calibration until less than 25% of full battery capacity is reached. This command calibrates the returned run time value.

Note: The test begins only if the battery capacity is 100%.

Monitoring

Command: `/system ups monitor <id>`

Property	Description
battery-charge ()	the UPS's remaining battery capacity as a percent of the fully charged condition
battery-voltage ()	the UPS's present battery voltage. The typical accuracy of this measurement is ±5% of the maximum value (depending on the UPS's nominal battery voltage)
frequency ()	when operating on-line, the UPS's internal operating frequency is synchronized to the line within variations within 3 Hz of the nominal 50 or 60 Hz. The typical accuracy of this measurement is ±1% of the full scale value of 63 Hz
line-voltage ()	the in-line utility power voltage
load ()	the UPS's output load as a percentage of full rated load in Watts. The typical accuracy of this measurement is ±3% of the maximum of 105%
low-battery (*yes* | *no*)	only shown when the UPS reports this status
on-battery (*yes* | *no*)	Whether UPS battery is supplying power
on-line (*yes* | *no*)	whether power is being provided by the external utility (power company)
output-voltage ()	the UPS's output voltage
overloaded-output (*yes* | *no*)	only shown when the UPS reports this status
replace-battery (*yes* | *no*)	only shown when the UPS reports this status
runtime-calibration-running (*yes* | *no*)	only shown when the UPS reports this status
runtime-left (*time*)	the UPS's estimated remaining run time in minutes. You can query the UPS when it is operating in the on-line, bypass, or on-battery modes of operation. The UPS's remaining run time reply is based on available battery capacity and output load
smart-boost-mode (*yes* | *no*)	only shown when the UPS reports this status
smart-ssdd-mode ()	only shown when the UPS reports this status
transfer-cause (*string*)	the reason for the most recent transfer to on-battery operation (only shown when the unit is on-battery)

Example

When running on utility power:

```
[admin@MikroTik] system ups> monitor 0
          on-line: yes
       on-battery: no
      RTC-running: no
     runtime-left: 20m
   battery-charge: 100%
  battery-voltage: 27V
     line-voltage: 226V
   output-voltage: 226V
             load: 45%
      temperature: 39C
        frequency: 50Hz
  replace-battery: no
      smart-boost: no
       smart-trim: no
         overload: no
      low-battery: no

[admin@MikroTik] system ups>
```

When running on battery:

```
[admin@MikroTik] system ups> monitor 0
          on-line: no
       on-battery: yes
   transfer-cause: "Line voltage notch or spike"
      RTC-running: no
     runtime-left: 19m
    offline-after: 4m46s
   battery-charge: 94%
  battery-voltage: 24V
     line-voltage: 0V
   output-voltage: 228V
             load: 42%
      temperature: 39C
        frequency: 50Hz
  replace-battery: no
      smart-boost: no
       smart-trim: no
         overload: no
      low-battery: no

      [admin@MikroTik] system ups>
```

[Top | Back to Content **]**

References

[1] http://www.exploits.org/nut/library/protocols/apcsmart.html

Manual:Layer-3 MPLS VPN example

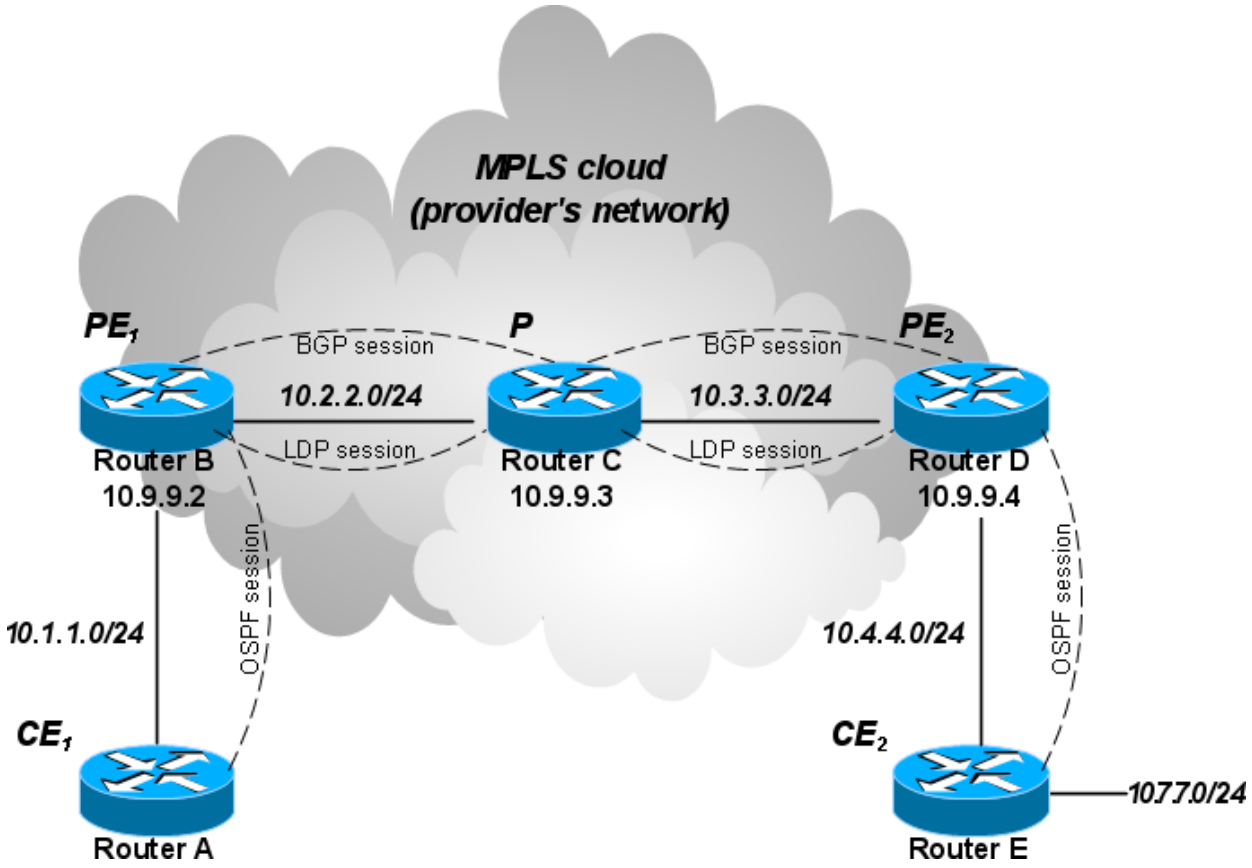
This is a kind of "putting it all together" setup.

Technologies used:

- LDP for MPLS label distribution
- BGP for VPNv4 route distribution
- OSPF as CE - PE routing protocol

Software:

- PE and P routers have RouterOS 3.17 with **routing-test** and **mpls-test** packages.
- CE routers have RouterOS 3.17 with routing-test package. (**routing** package and older versions can be used here as well.)



IP addressing & routing

Provider's network

On Router B:

```
/ip address add address=10.1.1.2/24 interface=ether2
/ip address add address=10.2.2.2/24 interface=ether3

# put PE-CE interface in a VRF
```

```
/ip route vrf add routing-mark=vrf1 interfaces=ether2 \
 route-distinguisher=10.1.1.1:111 import-route-targets=10.1.1.1:111 export-route-targets=10.1.1.1:111

# loopback interface
/interface bridge add name=lobridge
/ip address add address=10.9.9.2/32 interface=lobridge

# add routes to loopback addresses
# (static routing is used for destinations inside providers network)
/ip route add dst-address=10.9.9.3/32 gateway=10.2.2.3
/ip route add dst-address=10.9.9.4/32 gateway=10.2.2.3
```

On Router C:

```
/ip address add address=10.2.2.3/24 interface=ether3
/ip address add address=10.3.3.3/24 interface=ether2

# loopback interface
/interface bridge add name=lobridge
/ip address add address=10.9.9.3/32 interface=lobridge

# add routes to loopback addresses
/ip route add dst-address=10.9.9.2/32 gateway=10.2.2.2
/ip route add dst-address=10.9.9.4/32 gateway=10.3.3.4
```

On Router D:

```
/ip address add address=10.3.3.4/24 interface=ether2
/ip address add address=10.4.4.4/24 interface=ether3

# put PE-CE interface in a VRF
/ip route vrf add routing-mark=vrf1 interfaces=ether3 \
 route-distinguisher=10.1.1.1:111 import-route-targets=10.1.1.1:111 export-route-targets=10.1.1.1:111

# loopback interface
/interface bridge add name=lobridge
/ip address add address=10.9.9.4/32 interface=lobridge

# add routes to loopback addresses
/ip route add dst-address=10.9.9.2/32 gateway=10.3.3.3
/ip route add dst-address=10.9.9.3/32 gateway=10.3.3.3
```

Client's sites

On Router A:

```
/ip address add address=10.1.1.1/24 interface=<ToRouterB>
```

On Router E:

```
/ip address add address=10.4.4.5/24 interface=<ToRouterD>
/ip address add address=10.7.7.5/24 interface=<ToLocalNetwork>
```

LDP

On Router B:

```
/mpls ldp set enabled=yes transport-address=10.9.9.2
/mpls ldp interface add interface=ether3
```

On Router C:

```
/mpls ldp set enabled=yes transport-address=10.9.9.3
/mpls ldp interface add interface=ether2
/mpls ldp interface add interface=ether3
```

On Router D:

```
/mpls ldp set enabled=yes transport-address=10.9.9.4
/mpls ldp interface add interface=ether2
```

Setting transport address for LDP is not required, but very recommended. If the address is not set, the router will pick any address at random, which may be an address belonging to VRF, and as such not connectible from internal P routers.

Results

```
[admin@C] > /mpls ldp neighbor print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                      SEN ADDRESSES
 0  O   10.9.9.2        10.9.9.3        10.1.1.2:0                no  10.1.1.2
                                                                      10.2.2.2
                                                                      10.9.9.2
 1      10.3.3.4                                                  no
 2  O   10.9.9.4        10.9.9.3        10.3.3.4:0                no  10.3.3.4
                                                                      10.4.4.4
                                                                      10.9.9.4
```

BGP

On Router B:

```
/routing bgp instance vrf add instance=default routing-mark=vrf1 redistribute-connected=yes \
 redistribute-ospf=yes
/routing bgp peer add remote-address=10.9.9.3 remote-as=65530 address-families=vpnv4 \
 update-source=lobridge
```

On Router C:

```
/routing bgp peer add remote-address=10.9.9.2 remote-as=65530 route-reflect=yes \
  address-families=vpnv4 update-source=lobridge
/routing bgp peer add remote-address=10.9.9.4 remote-as=65530 route-reflect=yes \
  address-families=vpnv4 update-source=lobridge
# client-to-client-reflection is on by default
#/routing bgp instance set default client-to-client-reflection=yes
```

On Router D:

```
/routing bgp instance vrf add instance=default routing-mark=vrf1 redistribute-connected=yes \
  redistribute-ospf=yes
/routing bgp peer add remote-address=10.9.9.3 remote-as=65530 address-families=vpnv4 \
  update-source=lobridge
```

Note that route reflection here is used for the sake of an example. A simpler configuration would work as well - one where there is a BGP session between B and D and C is not running BGP at all.

Results

Check for routes on PE routers:

```
/routing bgp vpn vpnv4-route print
```

and

```
/ip route print where bgp
```

OSPF

On Router A:

```
/routing ospf network add network=10.1.1.0/24 area=backbone
```

On Router B:

```
/routing ospf instance set default routing-table=vrf1 redistribute-bgp=as-type-1
/routing ospf network add network=10.1.1.0/24 area=backbone
```

On Router D:

```
/routing ospf instance set default routing-table=vrf1 redistribute-bgp=as-type-1
/routing ospf network add network=10.4.4.0/24 area=backbone
```

On Router E:

```
/routing ospf network add network=10.4.4.0/24 area=backbone
/routing ospf network add network=10.7.7.0/24 area=backbone
```

Results

Routing table on CE router A:

```
[admin@A] > /ip route pr
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
```

```
 0 ADC  10.1.1.0/24        10.1.1.1        ether2             0
 1 ADo  10.4.4.0/24                        10.1.1.2 reachab... 110
 2 ADo  10.7.7.0/24                        10.1.1.2 reachab... 110
```

Routing table on CE router E:

```
[admin@E] > /ip route pr
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 0 ADo  10.1.1.0/24                        10.4.4.4 reachab... 110
 1 ADC  10.4.4.0/24        10.4.4.5        ether2             0
 2 ADC  10.7.7.0/24        10.7.7.5        ether3             0
```

Test

On Router A:

Ping from CE1 -> to PE1:

```
[admin@A] > /ping 10.1.1.2
10.1.1.2 64 byte ping: ttl=64 time=8 ms
10.1.1.2 64 byte ping: ttl=64 time=4 ms
10.1.1.2 64 byte ping: ttl=64 time=5 ms
10.1.1.2 64 byte ping: ttl=64 time=5 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 4/5.5/8 ms
```

Ping from CE1 -> to CE2:

```
[admin@A] > /ping 10.4.4.5
10.4.4.5 64 byte ping: ttl=61 time=12 ms
10.4.4.5 64 byte ping: ttl=61 time=5 ms
10.4.4.5 64 byte ping: ttl=61 time=6 ms
10.4.4.5 64 byte ping: ttl=61 time=8 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 5/7.7/12 ms
```

[admin@A] > /ping 10.7.7.5

```
10.7.7.5 64 byte ping: ttl=61 time=14 ms
10.7.7.5 64 byte ping: ttl=61 time=4 ms
10.7.7.5 64 byte ping: ttl=61 time=8 ms
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 4/8.6/14 ms
```

```
[admin@A] > /tool traceroute 10.7.7.5
     ADDRESS                                    STATUS
   1         10.1.1.2 3ms 6ms 2ms
   2         0.0.0.0 timeout timeout timeout
   3         10.3.3.4 4ms 3ms 3ms
   4         10.7.7.5 3ms 3ms 3ms
```

The second hop failure is normal.

To see whole MPLS cloud as one IP hop, configure **propagate-ttl=no**. This setting should be the same on all provider's routers.

On Routers B,C,D:

```
/mpls set propagate-ttl=no
```

```
[admin@A] > /tool traceroute 10.7.7.5
     ADDRESS                                    STATUS
   1         10.1.1.2 6ms 3ms 5ms
   2         10.3.3.4 5ms 3ms 6ms
   3         10.7.7.5 9ms 9ms 6ms
```

No failures here.

Connecting from PE to CE

In this case routing-table must be specified manually.

Ping from PE1 -> to CE1:

```
[admin@B] > ping 10.1.1.1 routing-table=vrf1
10.1.1.1 64 byte ping: ttl=64 time=9 ms
10.1.1.1 64 byte ping: ttl=64 time=6 ms
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 6/7.5/9 ms
```

Manual:LCD TouchScreen

Applies to RouterOS: v5.19+

Summary

Sub-menu: `/lcd`
Package:

RouterBOARD 2011U and CCR series devices are equipped with a resistive touchscreen, for quick access to device stats and simple configuration options. Touch screen requires pressure against the surface to register a touch, therefore light swipes and quick/short taps might not get registered (as opposed to a capacitive touchscreen commonly found on phones). If you find trouble operating the screen with your finger, you can also try a stylus, or opposite end of a pen.

Configuration

LCD Touch screen can be controlled from the `/lcd` console menu.

Available properties:

- `enabled` - Turn the lcd off (backlight) (yes/no);
- `backlight-timer` - time after which lcd display is turned off (value range [5m;2h] / never);
- `reset-calibration` - start the calibration process;

LCD Touch screen Calibration

Before the LCD touch screen can be used, it needs to be calibrated at least once. After the first successful calibration, data is stored on the router. If no calibration values are present, calibration process will start automatically.

During the calibration you must touch 4 points drawn on the screen. Three of the points are used to calculate calibration variables and the 4th point is used to test whether the calibration was successful. If calibration is unsuccessful, calibration variables are not saved. At the end (after touching 4th point) a message is displayed with the calibration result.

LCD screens/modes

Graph Screen

Graph screen shows single interface graphs for RX and TX. Values are updated from right to left (newest to oldest). Info that is shown: RX/TX rate, packets, drops, errors.

Interface name is shown at the bottom right, it is trimmed if it's too long (last characters are cut off). The top right corner shows the time interval for the values. Following time values are available:

- **Min (Minute)** - shows values for the last minute. Unit = second. Vertical line separates first 30 seconds. Total values: 30 + 24;
- **Hour** - shows values for the last hours. Unit = 5 minutes. Vertical lines separate 1 hour. Total values: 12 + 12 + 3;
- **Daily** - shows values for the last days. Unit = hour. Vertical lines separate 1 day. Total values: 12 + 12 + 3;
- **Weekly** - shows values for the last weeks. Unit = day. Vertical lines separate 1 week. Total values: 7 + 7 + 4;

Motions:

- Tap - tapping the finger against the touch screen without moving it too much.
  - If a tap lands into the top right corner of the screen (square box 1/4 of the screen height), info time interval is changed: Min -> Hour -> Daily -> Weekly -> Min...
  - Otherwise a tap cycles through graph info: rate -> packets -> drops -> errors -> rate...
- Swipe/Drag - while holding the finger down, move in any direction. The changes should be highlighted during the drag.
  - Down - Select All Interface graph screen
    - Left - Next interface
    - Right - Previous interface

All interface graph screen

All interface graph screen shows single info (RX or TX Rate, packets, errors, drops) for all interfaces. Values are updated every second. The goal of this screen is to see how values are related to each other for a single interface.

Motions:

- Tap on the top left corner changes TX/RX.
- Swipe/drag changes are highlighted.
  - Up - Back (to Graph screen).
  - Left/Right - select info type: Rate, Packets, Errors, Drops.

See Also

- Other LCD Display Support

[Top | Back to Content]

Manual:Limiting maximum number of prefixes accepted

Misconfiguration of BGP routers occasionaly lead to an injection of large route tables in BGP routing system. To protect your network against such "route leaks", you can limit maximum prefix count accepted.

Configuration settings:

max-prefix-limit (integer, default: unlimited) - maximum number of prefixes accepted from specific peer. After this limit is exceeded, TCP connection between peers is tear down.

max-prefix-restart-time (seconds) - minimum time interval after which peers can reestablish BGP session.
default - infinity, i.e. session is not reestablished without manual intervention.

Example:

```
[admin@A] > routing bgp peer set peer1 max-prefix-limit=10000 max-prefix-restart-time=3600
[admin@A] > routing bgp peer print
 0   name="peer1" instance=default remote-address=159.148.xx.xx remote-as=64550
     tcp-md5-key="" nexthop-choice=default multihop=yes route-reflect=no
     hold-time=3m ttl=100 max-prefix-limit=10000 max-prefix-restart-time=1h
     in-filter="" out-filter=""
```

To observe how many prefixes from a particular peer are installed in the routing table, pay attention to **prefix-count** attribute:

```
[admin@A] > routing bgp peer print status
 0   name="peer1" instance=default remote-address=159.148.xx.xx remote-as=64550
     tcp-md5-key="" nexthop-choice=default multihop=yes route-reflect=no
     hold-time=3m ttl=100 max-prefix-limit=10000 max-prefix-restart-time=1h
     in-filter="" out-filter="" remote-id=10.0.11.155 uptime=1m33s prefix-count=628
     updates-sent=1 updates-received=762 withdrawn-sent=0 withdrawn-received=0
     remote-hold-time=3m used-hold-time=3m used-keepalive-time=1m
     refresh-capability=yes state=established
```

Caveats

Negative effects of this mechanism are complete routing breakdown after BGP session is destroyed. Most probably routing will be broken until manual intervention of network operator that could take hours.

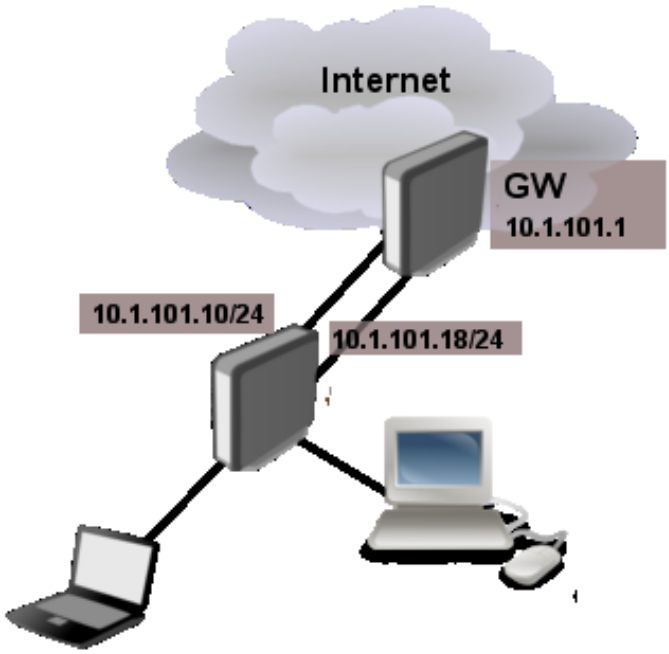
Theoretically this can be imporved using **max-prefix-restart-time**, but obviously until the situation is not fixed at remote peer's end, session will be destroyed repeatedly.

In short, this feature is acceptable as an emergency means. It is not a replacement for full-scale routing filters. ISPs should only accept prefixes which have been assigned or allocated to their downstream peer and filter out everything else.

Manual:Load balancing multiple same subnet links

Applies to RouterOS: v4,v5

This example demonstrates how to set up load balancing if provider is giving IP addresses from the same subnet for all links.



Provider is giving us two links with IP addresses from the same network range (10.1.101.10/24 and 10.1.101.18/24). Gateway for both of these links is the same 10.1.101.1

Here is the whole configuration for those who want to copy&paste

```
/ip address
add address=10.1.101.18/24 interface=ether1
add address=10.1.101.10/24 interface=ether2
add address=192.168.1.1/24 interface=Local
```

```
add address=192.168.2.1/24 interface=Local

/ip route
add gateway=10.1.101.1
add gateway=10.1.101.1%ether1 routing-mark=first
add gateway=10.1.101.1%ether2 routing-mark=other

/ip firewall nat
add action=masquerade chain=srcnat out-interface=ether1
add action=masquerade chain=srcnat out-interface=ether2

/ip firewall mangle
add action=mark-routing chain=prerouting src-address=192.168.1.0/24 new-routing-mark=first
add action=mark-routing chain=prerouting src-address=192.168.2.0/24 new-routing-mark=other
```

In previous RouterOS version multiple IP addresses from the same subnet on different interfaces were not allowed. Fortunately v4 allows such configurations.

In this example our provider assigned two upstream links, one connected to **ether1** and other to **ether2**. Our local network has two subnets 192.168.1.0/24 and 192.168.2.0/24

```
/ip address
add address=10.1.101.18/24 interface=ether1
add address=10.1.101.10/24 interface=ether2
add address=192.168.1.1/24 interface=Local
add address=192.168.2.1/24 interface=Local
```

After IP address is set up, connected route will be installed as ECMP route

```
[admin@MikroTik] /ip route> print detail
 0 ADC  dst-address=10.1.101.0/24 pref-src=10.1.101.18 gateway=ether1,ether2
        gateway-status=ether1 reachable,ether2 reachable distance=0 scope=10
```

Note: Routing filters can be used to adjust preferred source if needed

In our example very simple policy routing is used. Clients from 192.168.1.0/24 subnet is marked to use **"first"** routing table and 192.168.2.0/24 to use **"other"** subnet.

Note: The same can be achieved by setting up route rules instead of mangle.

```
/ip firewall mangle
add action=mark-routing chain=prerouting src-address=192.168.1.0/24 new-routing-mark=first
add action=mark-routing chain=prerouting src-address=192.168.2.0/24 new-routing-mark=other
```

And masquerade our local networks

```
/ip firewall nat
add action=masquerade chain=srcnat out-interface=ether1
add action=masquerade chain=srcnat out-interface=ether2
```

Warning: You will also have to deal with traffic coming to and from the router itself. For explanations look at PCC configuration example.

We are adding two gateways, one to resolve in **"first"** routing table and another to **"other"** routing table.

```
/ip route
add gateway=10.1.101.1%ether1 routing-mark=first
add gateway=10.1.101.1%ether2 routing-mark=other
```

Interesting part of these routes is how we set gateway. `gateway=10.1.101.1%ether1` means that gateway 10.1.101.1 will be explicitly reachable over ether1

```
[admin@MikroTik] /ip route> print detail
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 0 A S  dst-address=0.0.0.0/0 gateway=10.1.101.1%ether2
        gateway-status=10.1.101.1 reachable ether2 distance=1 scope=30
        target-scope=10 routing-mark=other

 1 A S  dst-address=0.0.0.0/0 gateway=10.1.101.1%ether1
        gateway-status=10.1.101.1 reachable ether1 distance=1 scope=30
        target-scope=10 routing-mark=first
```

Finally, we have one additional entry specifying that traffic from the router itself (the traffic without any routing marks) will be resolved in main routing table.

```
/ip route
add gateway=10.1.101.1
```

MAC access

Applies to RouterOS: 2.9, v3, v4

MAC telnet is used to provide access to a router that has no IP address set. It works just like IP telnet. MAC telnet is possible between two MikroTik RouterOS routers only.

Specifications

- Packages required: system
- License required: Level1
- Submenu level: /tool, /tool mac-server
- Standards and Technologies: MAC Telnet
- Hardware usage: Not significant

MAC Telnet Server

- **Submenu level**: */tool mac-server*

Property Description

- **interface** (name | all; default: all) - interface name to which the mac-server clients will connect

Notes

There is an interface list in this submenu level. If you add some interfaces to this list, you allow MAC telnet to that interface. Disabled (disabled=yes) item means that interface is not allowed to accept MAC telnet sessions on that interface. all interfaces iss the default setting to allow MAC teltet on any interface. Example

To enable MAC telnet server on ether1 interface only:

```
[admin@MikroTik] tool mac-server> print
Flags: X - disabled
 #   INTERFACE
 0   all
[admin@MikroTik] tool mac-server> remove 0
[admin@MikroTik] tool mac-server> add interface=ether1 disabled=no
[admin@MikroTik] tool mac-server> print
Flags: X - disabled
 #   INTERFACE
 0   ether1
[admin@MikroTik] tool mac-server>
```

MAC WinBox Server

- **Submenu level**: */tool mac-server mac-winbox*

Property Description

- **interface** (name | all; default: all) - interface name to which it is alowed to connect with Winbox using MAC-based protocol

Notes

There is an interface list in this submenu level. If you add some interfaces to this list, you allow MAC Winbox to that interface. Disabled (disabled=yes) item means that interface is not allowed to accept MAC Winbox sessions on that interface. Example

To enable MAC Winbox server on ether1 interface only:

```
[admin@MikroTik] tool mac-server mac-winbox> print
Flags: X - disabled
 #   INTERFACE
 0   all
[admin@MikroTik] tool mac-server mac-winbox> remove 0
[admin@MikroTik] tool mac-server mac-winbox> add interface=ether1 disabled=no
[admin@MikroTik] tool mac-server mac-winbox> print
Flags: X - disabled
 #   INTERFACE
 0   ether1
[admin@MikroTik] tool mac-server mac-winbox>
```

Monitoring Active Session List

- **Submenu level**: */tool mac-server sessions*

Property Description

- **interface** (read-only: name) - interface to which the client is connected to
- **src-address** (read-only: MAC address) - client's MAC address
- **uptime** (read-only: time) - how long the client is connected to the server

Example

To see active MAC Telnet sessions:

```
[admin@MikroTik] tool mac-server sessions> print
 # INTERFACE SRC-ADDRESS       UPTIME
 0 wlan1     00:0B:6B:31:08:22 00:03:01
[admin@MikroTik] tool mac-server sessions>
```

MAC Scan

- **Command name**: */tool mac-scan*

This command discovers all devices, which support MAC telnet protocol on the given network.

Property Description

- **(name)** - interface name to perform the scan on

MAC Telnet Client

Command name: /tool mac-telnet Property Description (MAC address) - MAC address of a compatible device

Example

```
[admin@MikroTik] > /tool mac-telnet 00:02:6F:06:59:42
Login: admin
Password:
Trying 00:02:6F:06:59:42...
Connected to 00:02:6F:06:59:42

  MMM      MMM       KKK                          TTTTTTTTTT      KKK
  MMMM    MMMM       KKK                          TTTTTTTTTT      KKK
  MMM MMMM MMM  III  KKK  KKK  RRRRRR     OOOOOO      TTT     III  KKK  KKK
  MMM  MM  MMM  III  KKKKK     RRR  RRR  OOO  OOO     TTT     III  KKKKK
  MMM      MMM  III  KKK KKK   RRRRRR    OOO  OOO     TTT     III  KKK KKK
  MMM      MMM  III  KKK  KKK  RRR  RRR   OOOOOO      TTT     III  KKK  KKK

  MikroTik RouterOS 3.0beta10 (c) 1999-2007       http://www.mikrotik.com/

Terminal linux detected, using multiline input mode
[admin@MikroTik] >
```

Manual:Making a simple wireless AP

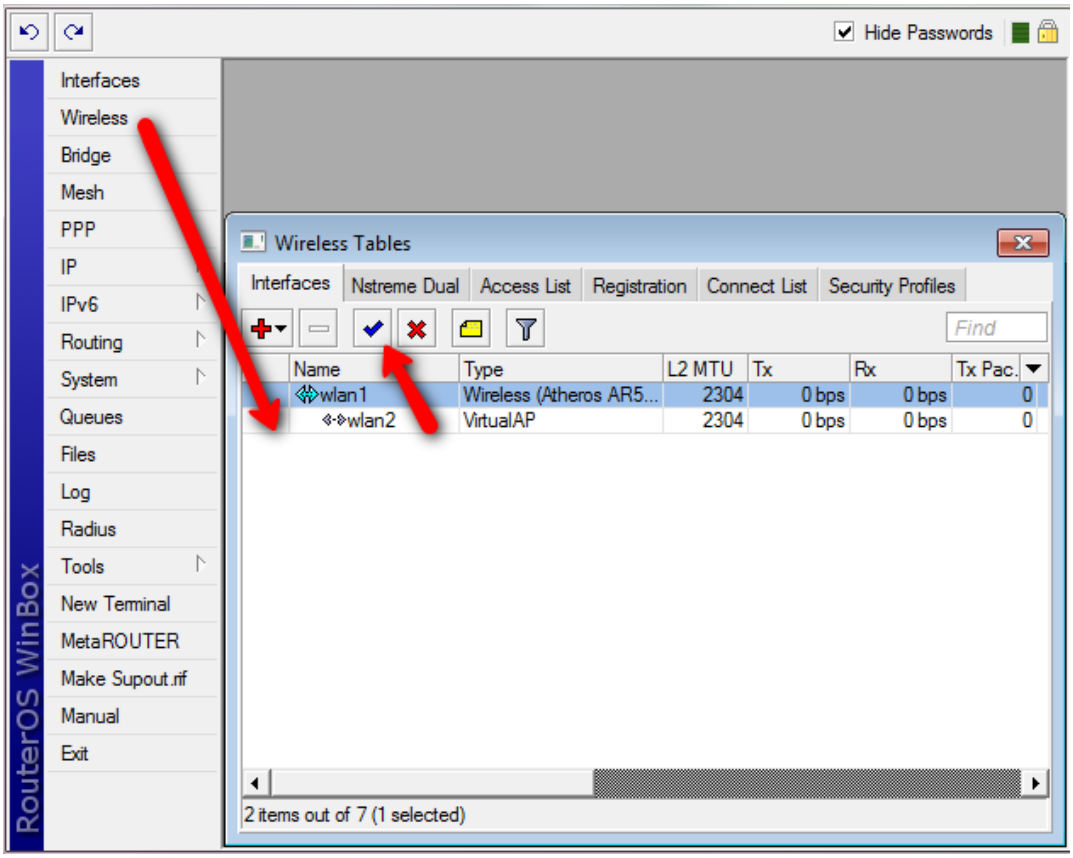
This article will show a very quick overview for beginners on setting up a Wireless Access Point in RouterOS Winbox graphical configuration tool.

Requirements

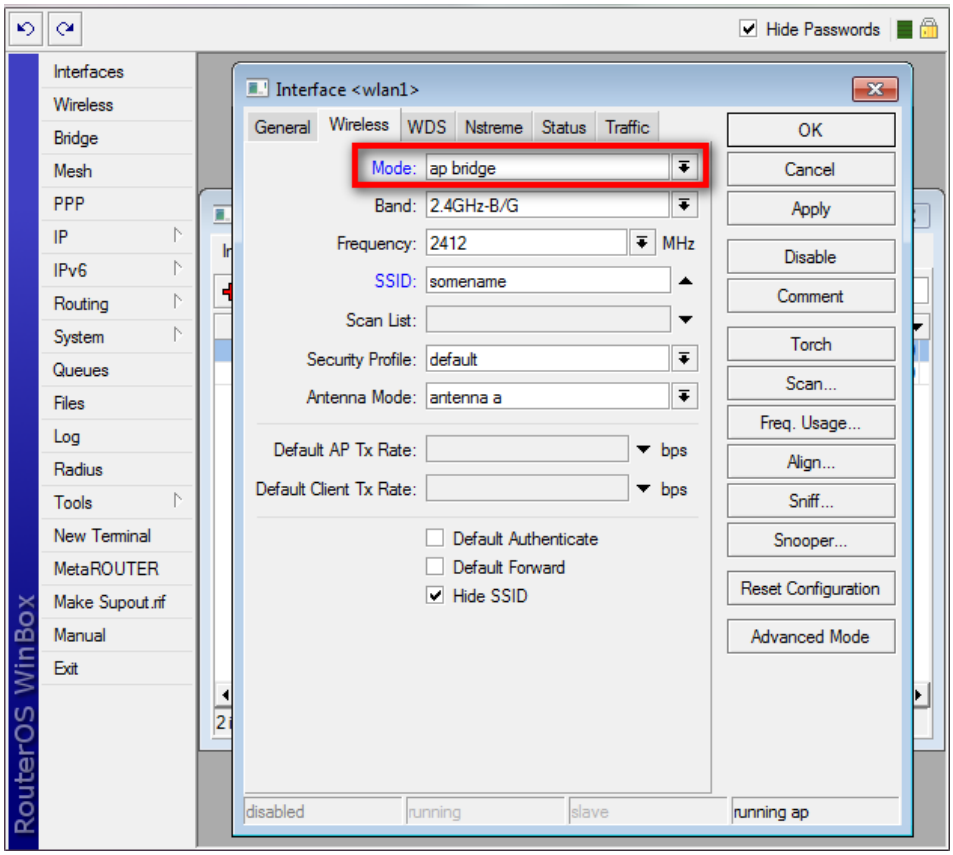
- a router running RouterOS loaded with supported miniPCI wireless cards
- a connection to the router via the Winbox utility

Instructions

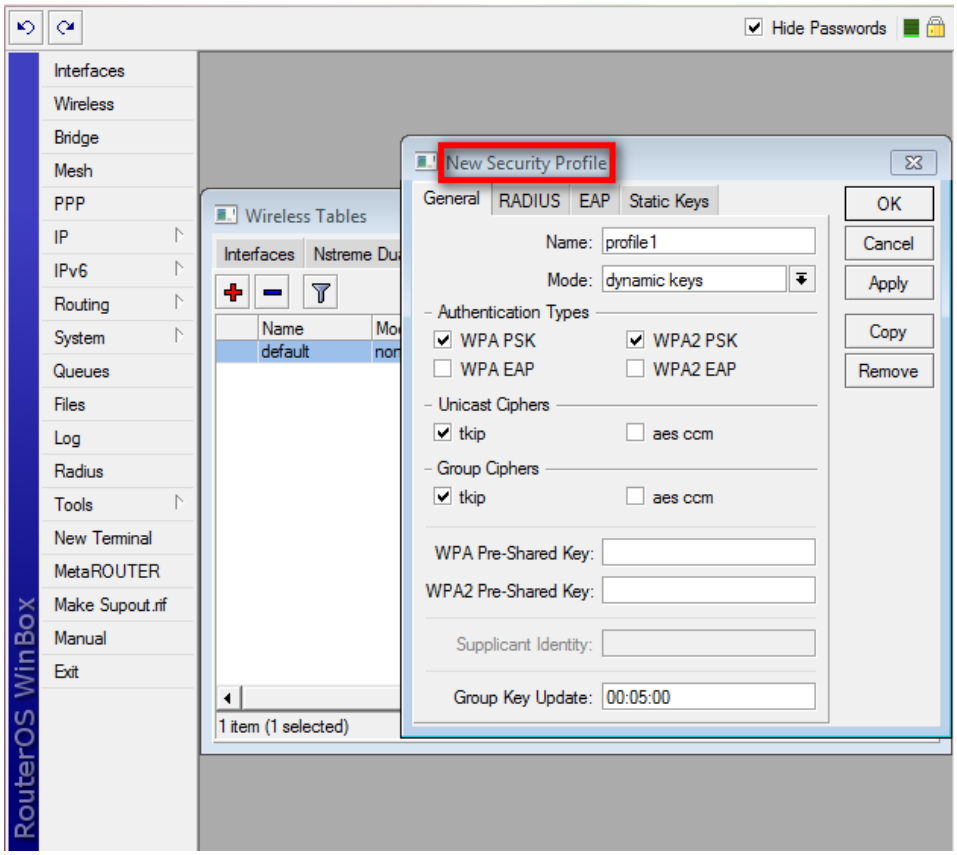
Start by opening the *Wireless Interface window* in Winbox. You will see some wireless cards listed here, they might be disabled - to turn them on, click on the blue **Enable** button. Make sure that the interface is configured and the antennas are connected *before* you enable an interface.



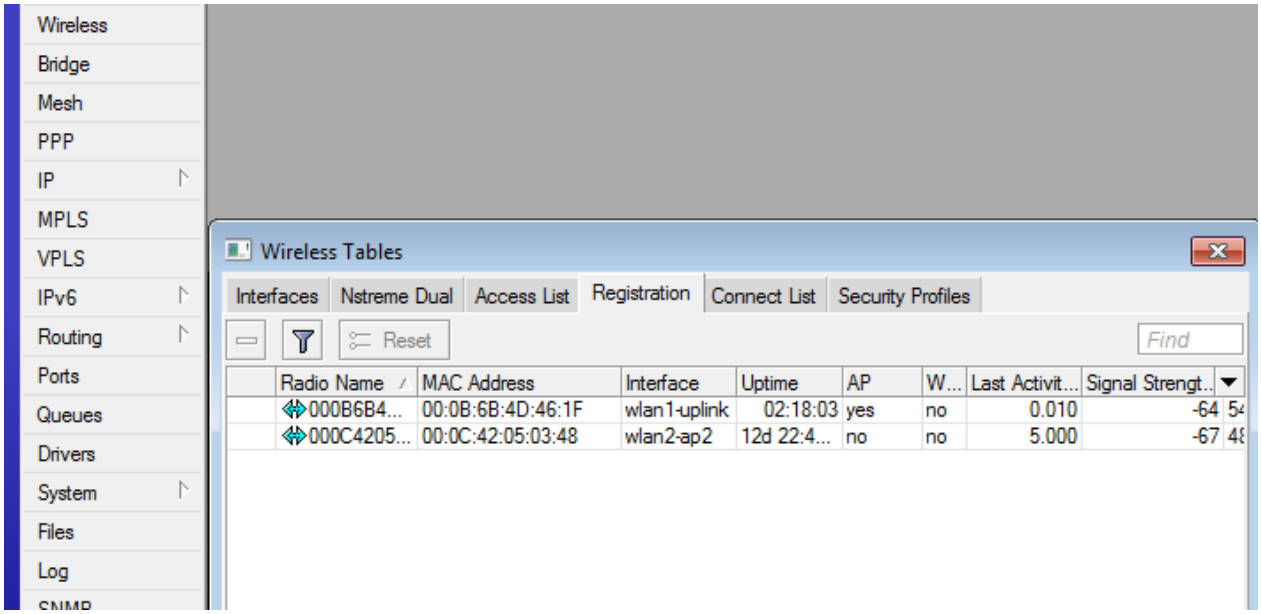
To configure an interface, **double-click it's name**, and the config window will appear. To set the device as an AP, choose "**ap bridge**" mode. You can also set other things, like the desired band, frequency, SSID (the AP identifier) and the security profile.



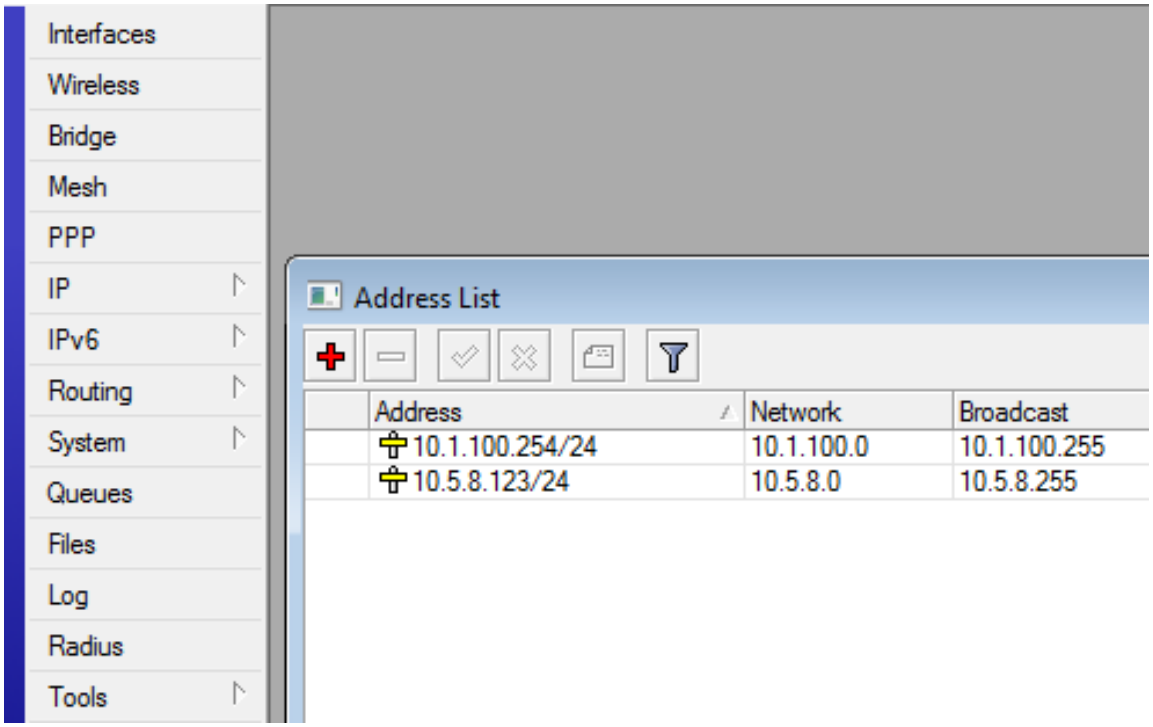
You probably want your AP to be secure, so you need to configure WPA2 security. Close the wireless setting window with OK if you are done, and move to the **Security Profiles** tab of the *Wireless interface window*. There, make a *new profile* with the **Add** button and set desired WPA2 settings. You can choose this new security profile back in the *Interface configuration*.



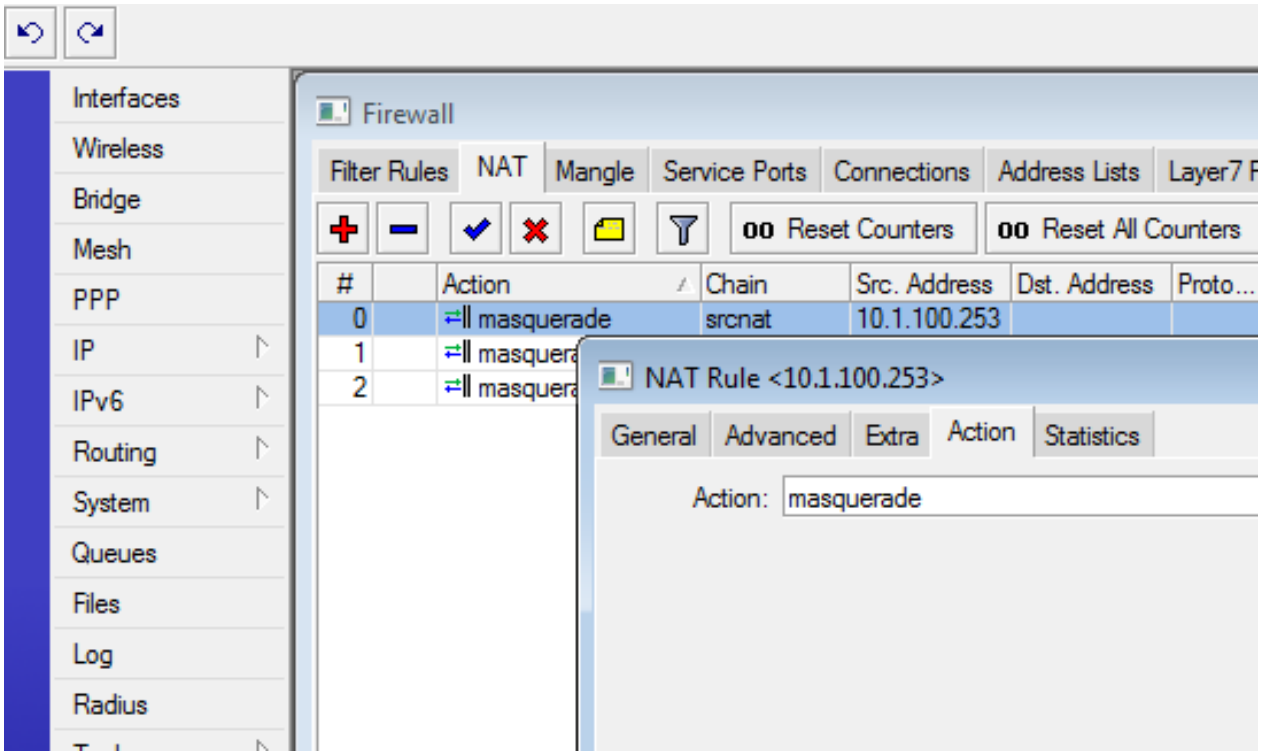
To see if any stations are connected to your AP, go to the **Registration Table** tab in the *Wireless Interface window*.

- 

Just connecting is probaly not enough, as your AP needs an IP address. This can be configured in the **IP** menu. Make sure that your stations also have IP addresses from the same subnet, or set up a DHCP server in this Router (not covered in this tutorial).

- 

If your ISP doesn't know about your new local network and hasn't set up proper routes to it, you need to configure **SRC-NAT** so that your stations have access to the internet via their private IP addresses. They will be **masqueraded** by the router's NAT functionality (not covered in this tutorial)



Manual:Maximum Transmission Unit on RouterBoards

Background

It is sole responsibility of administrator to configure MTUs such that intended services and applications can be successfully implemented in network. In other words - administrator must make sure that MTUs are configured in a way that packet sizes does not exceed the capabilities of network equipment.

Originally MTU was introduced because of the high error rates and low speed of communications. Fragmentation of the data stream gives ability to correct corruption errors only by resending corrupted fragment, not the whole stream. Also on low speed connections such as modems it can take too much time to send a big fragment, so in this case communication is possible only with smaller fragments.

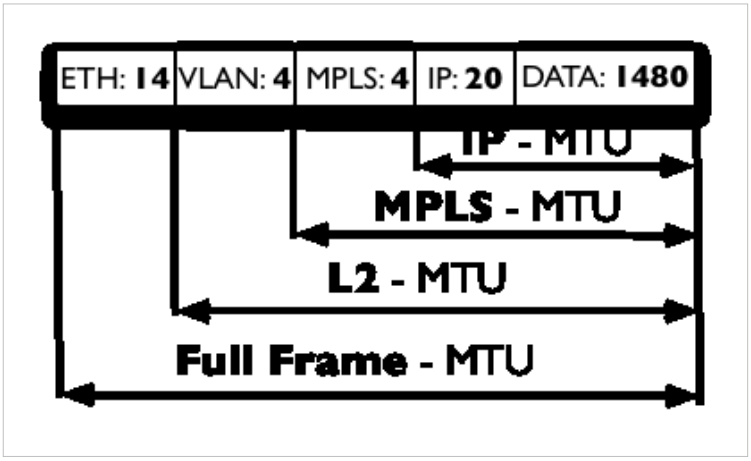
But in present days we have much lower error rates and higher speed of communication, this opens a possibility to increase the value of MTU. By increasing value of MTU we will result in less protocol overhead and reduce CPU utilization mostly due to interrupt reduction.

This way some non-standard frames started to emerge:

- **Giant** or **Jumbo** frames - frames that are bigger than standard (IEEE) Ethernet MTU
- **Baby Giant** or **Baby Jumbo** frames - frames that are just slightly bigger that standard (IEEE) Ethernet MTU

It is common now for Ethernet interfaces to support physical MTU above standard, but this can not be taken for granted. Abilities of other network equipment must be taken into account as well - for example, if 2 routers with Ethernet interfaces supporting physical MTU 1526 are connected through Ethernet switch, in order to successfully implement some application that will produce this big Ethernet frames, switch must also support forwarding such frames.

MTU on RouterOS



Mikrotik RouterOS recognizes several types of MTU:

- IP/Layer-3/L3 MTU
- MPLS/Layer-2.5/L2.5 MTU
- MAC/Layer-2/L2 MTU
- Full frame MTU

Full frame MTU

Full frame MTU indicates the actual size of the frame that are sent by particular interface. Frame Checksum is not included as it is removed by Ethernet driver as soon as frame reach its destination.

MAC/Layer-2/L2 MTU

L2MTU indicates the maximum size of the frame without MAC header that can be sent by this interface.

Starting from the RouterOS v3.25 L2MTU values can be seen in "/interface" menu. L2MTU support is added for all Routerboard related Ethernet interfaces, VLANs, Bridge, VPLS and wireless interfaces. Some of them support configuration of L2MTU value. All other Ethernet interfaces might indicate L2MTU only if the chip set is the same as Routerboard Ethernets.

This will allow users to check if desired setup is possible. Users will be able to utilize additional bytes for VLAN and MPLS tags, or simple increase of interface MTU to get rid of the some unnecessary fragmentation.

This table shows max-l2mtu supported by Mikrotik RouterBoards (Starting from the RouterOS v5.3 also available in "/interface print" menu as value of read-only "max-l2mtu" option):

Integrated Solutions

RouterBoard	ether1	ether2	ether3	ether4	ether5	ether6	ether7	ether8	ether9	ether10	ether11	ether12-13
Groove A-5Hn, Groove 5Hn, SXT 5HnD	2028											
RB750, RB750UP, RB751U-2HnD, OmniTik U-5HnD, OmniTik UPA-5HnD	4076	2028	2028	2028	2028							
RB750GL, RB751G-2HnD	4074	4074	4074	4074	4074							
RB1200	4078	4078	4078	4078	4078	4080	4080	4080	9116	9116		
RB1100AH	9498	9498	9498	9498	9498	9498	9498	9498	9498	9498	9116	9116
RB1100AHx2	9498	9498	9498	9498	9498	9498	9498	9498	9498	9498	9500	9116

RouterBOARD

RouterBoard	ether1	ether2	ether3	ether4	ether5	ether6	ether7	ether8	ether9	ether10	ether11	ether12-13
RB411, RB411U, RB411AR, RB411AH, RB411UAHR	1526											
RB433, RB433AH, RB433UAH, RB433L, RB450, RB493, RB493AH	1526	1522	1522	1522	1522	1522	1522	1522	1522			
RB411GL, RB433GL	1524	1524	1524									
RB435G, RB450G, RB493G	1520	1520	1520	1520	1520	1520	1520	1520	1520			
RB711 series	2028											
RB711G series	4076											
RB800	9500	9500	9116									
RB2011 series	4074	4074	4074	4074	4074	2028	2028	2028	2028	2028	[sfp1] 4047	

Old Products

RouterBoard	ether1	ether2	ether3	ether4	ether5	ether6	ether7	ether8	ether9	ether10	ether11	ether12-13
RB600, RB600A, RB1000	9500	9500	9500	9500								
RB1100	9498	9498	9498	9498	9498	9498	9498	9498	9498	9498	9116	9116
RB750G	1524	1524	1524	1524	1524							
RB333	1632	1632	1632									
RB1xx	1518	1518	1518	1518	1518	1514	1514	1514	1514			
RB532, CrossRoads	1600	1600	1600									
RB44G	7200	7200	7200	7200								
RB44GV	9000	9000	9000	9000								

All wireless interfaces in RouterOS (including Nstreme2)supports 2290 byte L2MTU

MPLS/Layer-2.5/L2.5 MTU

Configured in "/mpls interface" menu, specifies maximal size of packet, including MPLS labels, that is allowed to send out by the particular interface (default is 1508).

Make sure that MPLS MTU is smaller or equal to L2MTU

MPLS MTU affects packets depending on what action MPLS router is performing. It is strongly recommended that MPLS MTU is configured to the same value on all routers forming MPLS cloud because of effects MPLS MTU has on MPLS switched packets. This requirement means that all interfaces participating in MPLS cloud must be configured to the smallest MPLS MTU values among participating interfaces, therefore care must be taken to properly select hardware to be used.

MPLS Switching

If packet with labels included is bigger than MPLS MTU, MPLS tries to guess protocol that is carried inside MPLS frame.

If this is IP packet, MPLS produces ICMP Need Fragment error. This behavior mimics IP protocol behavior. Note that this ICMP error is not routed back to originator of packet but is switched towards end of LSP, so that egress router can route it back.

If this is not IP packet, MPLS simply drops it, because it does not know how to interpret the contents of packet. This feature is very important in situations where MPLS applications such as VPLS are used (where frames that are MPLS tagged are not IP packets, but e.g. encapsulated Ethernet frames as in case of VPLS) - if somewhere along the LSP MPLS MTU will be less than packet size prepared by ingress router, frames will simply get dropped.

IP ingress

When router first introduces label (or labels) on IP packet, and resulting packet size including MPLS labels exceeds MPLS MTU, router behaves as if interface MTU was exceeded - either fragments packet in fragments that does not exceed MPLS MTU when labels are attached (if IP Dont Fragment is not set), or generates ICMP Need Fragmentation error that is sent back to originator.

VPLS ingress

When router encapsulates Ethernet frame for forwarding over VPLS pseudowire, it checks if packet size with VPLS Control Word (4 bytes) and any necessary labels (usually 2 labels - 8 bytes), exceeds MPLS MTU of outgoing interface. If it does, VPLS fragments packet so that it honours MPLS MTU of outgoing interface. Packet is defragmented at egress point of VPLS pseudowire.

IP/Layer-3/L3 MTU

Configured as interface MTU setting (/interface <type> <name> set mtu=X). Specifies how big IP packets router is allowed to send out the particular interface.

If router receives IP packet of size 1500, but MTU for outgoing interface is set to 1400, router will either fragment the packet (if "Don't Fragment" bit is not set in IP header) or drop the packet and send ICMP "Need Fragmentation" error back to originator (this is essential for Path MTU Discovery to work).

Sometimes it can be bad idea to change IP MTU from its default 1500 bytes on router interfaces if complete path end-to-end is not in administrators control. Although IP fragmentation and end-to-end Path MTU Discovery is intended to handle this situation, if ICMP Need Fragmentation errors are filtered somewhere along the path, Path MTU Discovery will not work.

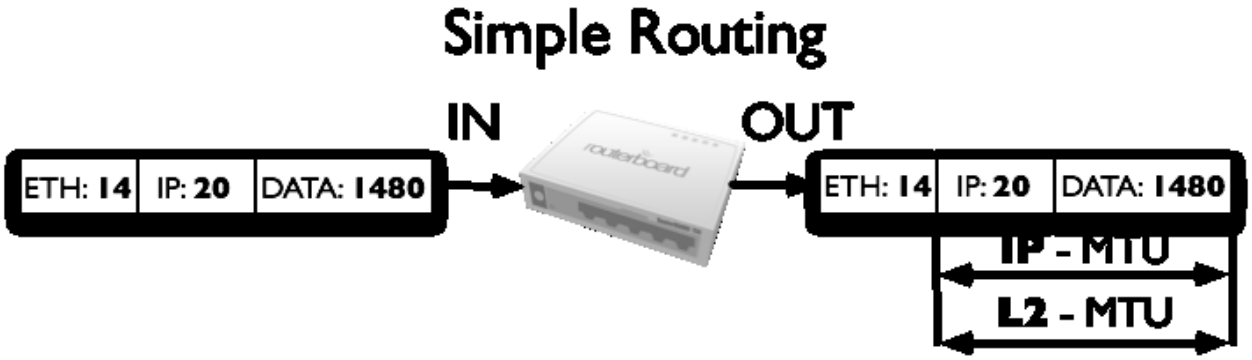
There are several features in MikroTik RouterOS that can benefit from possibility to exceed standard MTU

Simple Examples

In these examples we will take a look at frames entering and leaving router via Ethernet interfaces.

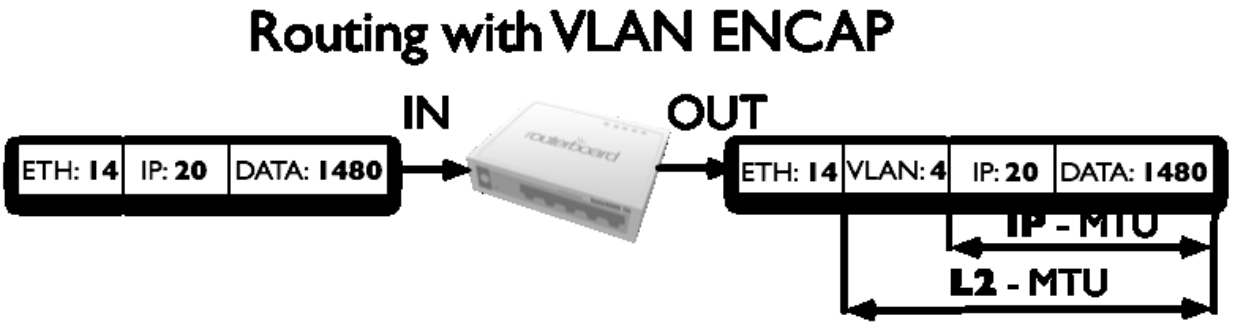
Simple Routing

The image shows the packet MTU size for simple routing, packets size is not modified.



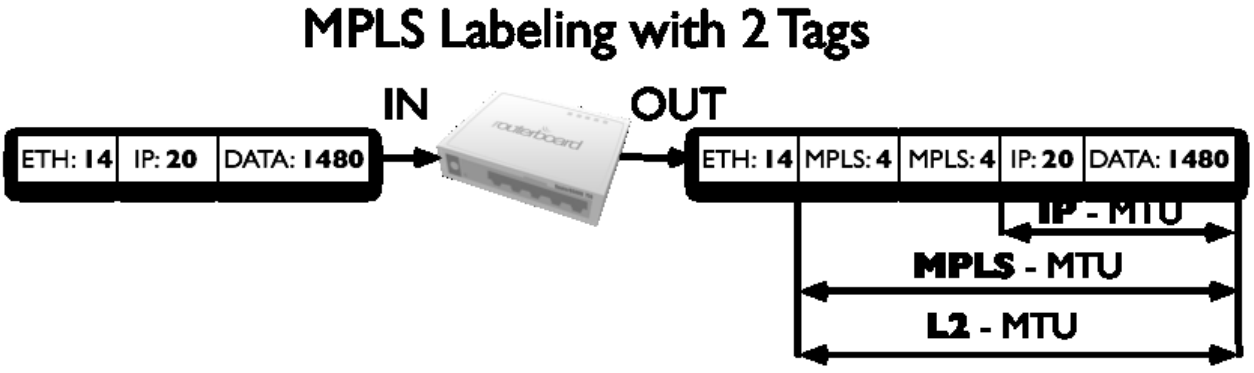
Routing with VLAN Encap

Each VLAN tag is 4 bytes long, VLAN tag is added by router. L2-MTU is increased by 4 bytes.



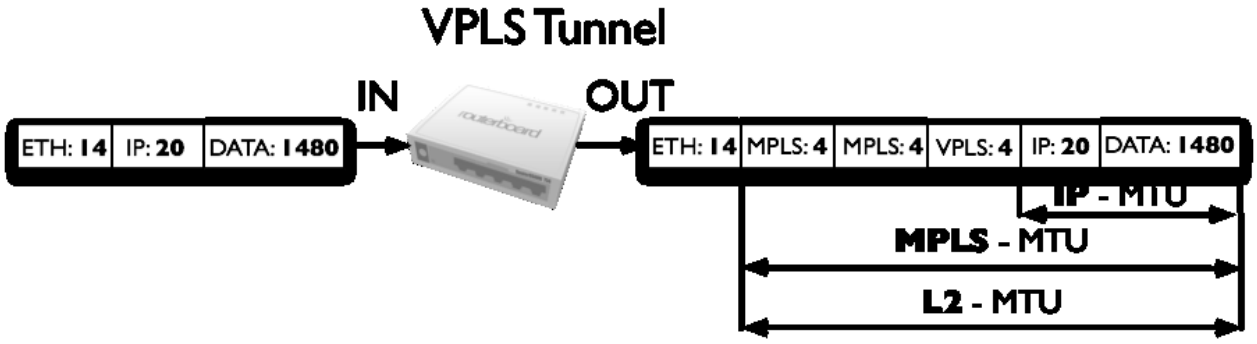
Simple MPLS with tags

When MPLS is used as plain replacement for IP routing, only one label is attached to every packet, therefore packet size increases by 4 bytes, we have the situation with two MPLS labels. In order to be able to forward standard size (1500 bytes) IP packet without fragmentation, MPLS MTU must be set to at least 1508 for two MPLS labels.



VPLS Tunnel

Two MPLS labels are present, when remote endpoint is not directly attached. One MPLS label is used to get to remote endpoint, second label is used to identify VPLS tunnel.

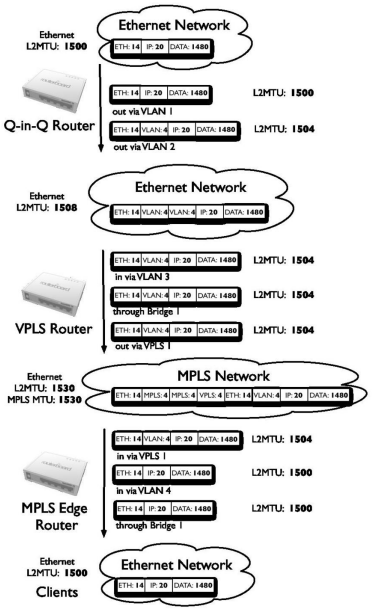


L2MTU advanced example

In this example we will take a closer look at required L2MTU of all Ethernet like interfaces including Bridge, VLAN, VPLS interfaces.

In this setup we will have 3 routers:

- Q-in-Q router - this router will receive standard 1500 byte Ethernet frame and will add two VLAN tags to the packet. Then packet will be sent out via Ethernet network to the second router
- VPLS router - this router will remove outer VLAN tag and will bridge packet with the remaining VLAN tag with VPLS tunnel. VPLS tunnel will take packet through the MPLS network to the third router.
- MPLS Edge router - will remove VPLS and VLAN tags and bridge packet to the client Ethernet network.



[Top | Back to Content]

Manual:Metarouter

Applies to RouterOS: v3, v4

Overview

MetaRouter is a new feature in RouterOS 4.0 beta 1 and RouterOS v3.21

Currently MetaRouter can be used on

- **RB400**, **RB700 series**, **RB900 series**, **RB2011** boards
- Listed PPC boards: **RB1000, RB1100, RB1100AH and RB800**.

Requirements

Each Metarouter instance uses the same amount of resources as a stand-alone RouterOS installation. It means that you need a minimum of 16MB of RAM for each RouterOS virtual machine plus memory for the MetaROUTER host itself. It is suggested to have more than 16MB memory available for each Metarouter. Upcoming RouterOS versions will have ability to run virtual machines with less than 16MB per machine.

Note: *It is possible to run other virtual machines with less than 16MB RAM per machine if the virtual operating system is OpenWRT. The 16MB limitation is only for virtual RouterOS installations.*

Currently on one host you can create up to **8 virtual machines** and up to **8 virtual interfaces**. Workaround to have more than 8 interfaces in total is to use VLANs. In future versions it will be possible to add up to 16 virtual machines.

Also it is not possible to use external storage devices (Store) in the metarouter virtual devices.

Where it can be used?

The MetaRouter function is useful for allowing clients or lower-privilege users access to their own 'router' and config to configure as they like, without the need for a complete second router, or giving them access to the main router configuration.

For example; a WISP can create a virtual router for the clients ethernet port allowing them to define their own firewall settings, while leaving the WISP's wireless settings untouched.

Creating a Metarouter

```
[admin@RB_Meta] /metarouter> add name=mr0 memory-size=32 disk-size=32000  disabled=no
[admin@RB_Meta] /metarouter> print
Flags: X - disabled
#    NAME                 MEMORY-SIZE DISK-SIZE     USED-DISK     STATE
0    mr0                  16MiB       0kiB          377kiB        running
```

As you can see, creating virtual router is quite easy, you just have to specify name of the router, how many RAM will be allocated for it and disk size that will be used by virtual router. Explanations of all other properties are available in reference manual.

*Note: * be careful when using dynamic HDD size for metarouters, a proxy could fill up all your hosts storage!*

Example with no settings

If you will add a new metarouter without specifying any parameters, it will be added with Dynamic HDD size, and 16MiB of RAM:

```
[admin@RB_Meta] /metarouter> add name=mr1
[admin@RB_Meta] /metarouter> print
Flags: X - disabled
 #   NAME                 MEMORY-SIZE DISK-SIZE     USED-DISK     STATE
 1   mr1                  16MiB       0kiB          3kiB          running
```

OpenWRT as virtual machine

Starting from v3.24 and v4.0beta3 MetaROUTER has the ability to import custom built images. As an example we will show how to patch and use OpenWRT as the virtual machine.

Importing image

If you don't have any specific needs, you can import our prebuilt OpenWRT image, which is downloadable MIPS image [1], PPC image [2]. Upload openwrt image to the router and import it by *import-image* command:

```
[admin@MikroTik] /metarouter> import-image file-name=openwrt-mr-mips-rootfs.tgz
 imported: 100%
```

```
[admin@MikroTik] /metarouter> print
Flags: X - disabled
#   NAME      MEMORY-SIZE DISK-SIZE     USED-DISK     STATE
0   mr1       16MiB       unlimited     7383kiB       running
```

As you can see OpenWRT is running, now you can start configuration process, which is explained in sections below.

Building your own OpenWRT image

If you are not satisfied with our prebuilt version of OpenWRT, then you can build and use your own image.

First step is to install *svn* and get the latest source code from openwrt.org

```
svn co svn://svn.openwrt.org/openwrt/trunk
```

Now you have to path downloaded source with our patch [3]

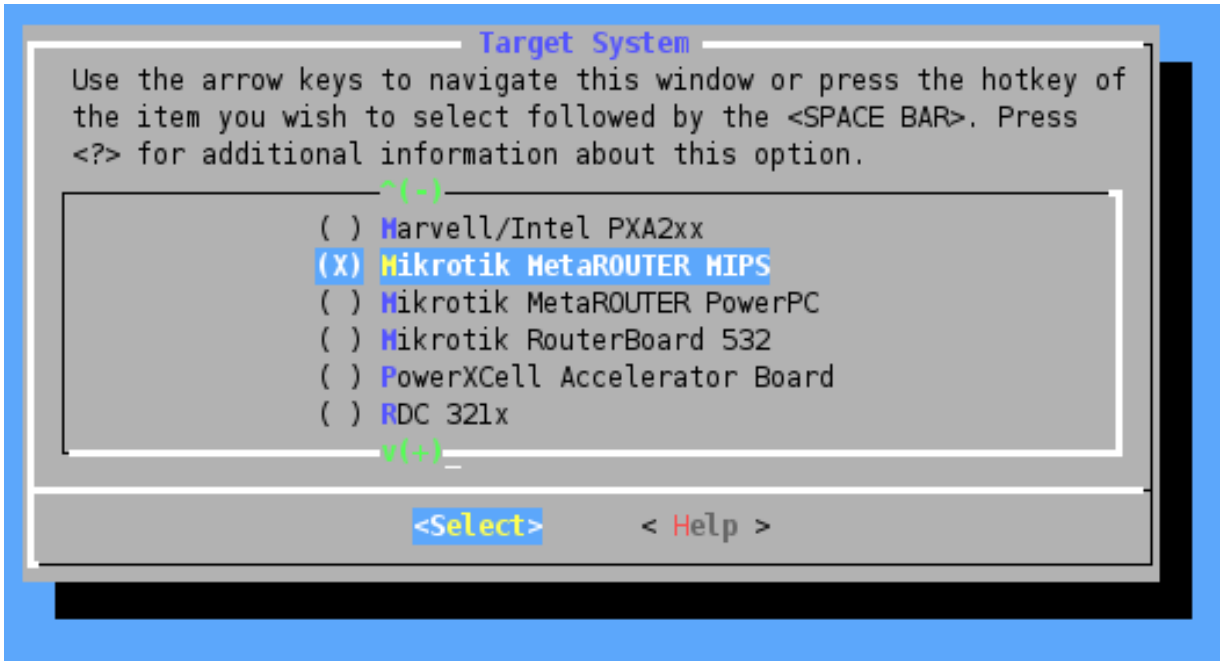
Note: Patch v1.2 adds newer kernel support which makes it possible to compile with latest OpenWRT revisions. This patch also adds PowerPC support (ability to run OpenWRT image on RB1000 and RB1100).

```
 cd trunk/
 wget http://www.mikrotik.com/download/metarouter/openwrt-metarouter-1.2.patch
 patch -p0 <openwrt-metarouter-1.2.patch
```

When source is patched, you have to set up configuration options

```
make menuconfig
```

Go to Target System menu and choose *Mikrotik MetaROUTER MIPS* or *Mikrotik MetaROUTER PowerPC* from the list depending for which platform you are building the image.



Other options depends on what is your requirements (include for example IPv6 and ppp support or not), you can also stick with defaults.

If you see any error messages while trying to launch *menuconfig*, like

```
Build dependency: Please install ncurses. (Missing libncurses.so or ncurses.h)
```

It means that required libraries are not installed, check the output and install all required libraries.

When you are done with build configuration, type

```
make
```

It will take a while to build everything so you can go and have a cup of tea.

After the build process is done, upload newly built image to the router and import it as described in section above.

For more options and build instructions look in OpenWRT's documentation [4]

Adding Interfaces

First, you need to add a new interface to your virtual router. This is done in the interface menu.

The interface command has the following options:

```
[admin@MikroTik] /metarouter> interface add
comment     disabled         dynamic-mac-address  type             virtual-machine
copy-from   dynamic-bridge   static-interface     vm-mac-address
```

Description of each option can be found in reference manual.

Let's add one interface:

```
[admin@MikroTik] /metarouter> interface add virtual-machine=mr1 type=dynamic
```

On the **host** physical router the interface appears as a virtual interface:

```
 [admin@MikroTik] > /interface print
 Flags: D - dynamic, X - disabled, R - running, S - slave
 #     NAME                                                TYPE              MTU
 8  R  ether9                                              ether             1500
 9  R  test                                                bridge            1500
10 DR  vif1                                               vif               1500
```

Connecting to the virtual machine

To connect to your virtual machine, use the console command:

```
/metarouter console 0
```

You will see your newly added virtual interface here:

```
[admin@mr0] > interface print
 Flags: D - dynamic, X - disabled, R - running, S - slave
 #     NAME                                                TYPE              MTU
 0  R  ether1                                              ether             1500
```

To disconnect from the metarouter virtual machine console, hit **CTRL + A** and then **Q** to Quit back to your **Host** console (if you are using minicom, hit **CTRL + A** twice):

```
[admin@MikroTik] >
[Q - quit connection]      [B - send break]
[A - send Ctrl-A prefix]   [R - autoconfigure rate]

 Q

 Welcome back!
```

Configuring a virtual network

Right now you saw that the virtual interface is visible in the **Host** *Interfaces* menu as *vif1* and also in the **metarouter** *interfaces* menu as *ether1*. You can add an IP address on both interfaces, and set up networking. Creating a bridge between the virtual interface and a physical interface allows traffic to pass.

Configuration examples

Creating isolated Metarouter for client

This Example will show how to use Metarouter feature to create a isolated router on top of the WISP client site router. The setup for the example is shown on the diagram below:

1. Adding a Metarouter for client:

```
[admin@RouterGW] /metarouter> add name=client1 memory-size=32
[admin@RouterGW] /metarouter> print
Flags: X - disabled
 #   NAME                          MEMORY-SIZE DISK-SIZE     USED-DISK     STATE
 0   client1                       32MiB       0kiB          189kiB        running
[admin@RouterGW] /metarouter>
```

2. Adding Metarouter Interfaces for the new created Metarouter:

```
[admin@RouterGW] /metarouter interface> add virtual-machine=client1
[admin@RouterGW] /metarouter interface> add virtual-machine=client1
[admin@RouterGW] /metarouter interface> print
Flags: X - disabled, A - active
 #   VIRTUAL-MACHINE                                          TYPE    VM-MAC-ADDRESS
 0 A client1                                                  dynamic 02:49:E8:55:8E:E8
 1 A client1                                                  dynamic 02:16:16:90:EF:0E
[admin@RouterGW] /metarouter interface>
```

3. Creating a Bridge Interface for bridging metarouter interface together with ethernet interface where the client is physically connected:

```
[admin@RouterGW] /interface bridge> add
[admin@RouterGW] /interface bridge> print
Flags: X - disabled, R - running
 0  R name="bridge1" mtu=1500 arp=enabled mac-address=00:00:00:00:00:00 protocol-mode=none
      priority=0x8000 auto-mac=yes admin-mac=00:00:00:00:00:00 max-message-age=20s
      forward-delay=15s transmit-hold-count=6 ageing-time=5m

[admin@RouterGW] /interface bridge port> add interface=ether2 bridge=bridge1
[admin@RouterGW] /interface bridge port> add interface=vif2 bridge=bridge1
[admin@RouterGW] /interface bridge port> print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE                   BRIDGE                   PRIORITY PATH-COST  HORIZON
 0    ether2                      bridge1                  0x80     10         none
 1    vif2                        bridge1                  0x80     10         none
```

4. Adding IP configuration for the new Metarouter interface which will be used for connecting between Metarouter and Metarouter Host system:

```
[admin@RouterGW] /ip address> add address=10.0.1.1/24 interface=vif1
[admin@RouterGW] /ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0 D 10.5.8.68/24       10.5.8.0        10.5.8.255      ether1
 1   10.0.1.1/24        10.0.1.0        10.0.1.255      vif1
[admin@RouterGW] /ip address>
```

5. Connecting to Metarouter using the Console

```
[admin@RouterGW] /metarouter> console client1

[Ctrl-A is the prefix key]

Starting...
Starting services...

MikroTik 3.21
MikroTik Login: admin
```

```
Password:

[admin@MikroTik] > /sys identity set name=Client1
```

6. Configuring Metarouter to make it easy for client to understand the configuration:

```
[admin@Client1] /interface ethernet> p
Flags: X - disabled, R - running, S - slave
 #    NAME                                                MTU   MAC-ADDRESS       ARP
 0 R  ether1                                              1500  02:49:E8:55:8E:E8 enabled
 1 R  ether2                                              1500  02:16:16:90:EF:0E enabled
[admin@Client1] /interface ethernet> set 0 name=public
[admin@Client1] /interface ethernet> set 1 name=local
[admin@Client1] /interface ethernet> print
Flags: X - disabled, R - running, S - slave
 #    NAME                                                MTU   MAC-ADDRESS       ARP
 0 R  public                                              1500  02:49:E8:55:8E:E8 enabled
 1 R  local                                               1500  02:16:16:90:EF:0E enabled
[admin@Client1] /interface ethernet>

[admin@Client1] /ip address> add address=10.0.1.2/24 interfae=public
[admin@Client1] /ip address> add address=10.0.2.1/24 interface=local
[admin@Client1] /ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   10.0.1.2/24        10.0.1.0        10.0.1.255      public
 1   10.0.2.1/24        10.0.2.0        10.0.2.255      local

[admin@Client1] /ip route> add gateway=10.0.1.1
[admin@Client1] /ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY                   DISTANCE INTERFACE
 0 A S  0.0.0.0/0                          r 10.0.1.1                  1        public
 1 ADC  10.0.1.0/24        10.0.1.2                                    0        public
 2 ADC  10.0.2.0/24        10.0.2.1                                    0        local
[admin@Client1] /ip route>

[admin@Client1] /ip firewall nat> add action=masquerade out-interface=public chain=srcnat
```

Reference

General

Sub-menu: `/metarouter`

Menu specific commands:

Property	Description
console (console <vm-id>)	connect to specified virtual machine's console
import-image (import-image file-name=<image-file>)	import custom built image (available starting from v3.24 and v4.0b3)
reboot (reboot <vm-id>)	reboot specified virtual machine
shut-down (shut-down <vm-id>)	shut down specified virtual machine
start (start <vm-id>)	boot up specified virtual machine

Configurable properties:

Property	Description
disk-size (*unlimited|0..4294967295[kiB]* ; Default: **unlimited**)	Disk size that will be allocated by virtual router.
memory-size (*16..256[MiB]* ; Default: **16**)	Amount of memory that will be allocated by virtual router.
name (*string* ;)	Name of the virtual machine.

Read only properties:

Property	Description
used-disk (*integer[kiB]* ;)	currently used disk space by virtual router.
disk-reads (*integer*;)	number of disk reads
disk-writes (*integer*;)	number of disk writes
state (*booting|running|rebooting|shutting-down|stopped|disabled*;)	current state of virtual machine

Interface

Sub-menu: `/metarouter interface`

Configurable properties:

Property	Description
dynamic-bridge (*string*;)	name of the bridge where to assign virtual interface as a port. Useful if interface type is dynamic
dynamic-mac-address (*mac*;)	mac address of dynamically created interface
static-interface (*none|name-of-iface*;)	static interface that virtual interface will be bound to
type (*dynamic|static*;)	specifies whether interface is static or dynamic
virtual-machine (*string*;)	specifies to which virtual machine this interface will be bound
vm-mac-address (*mac*;)	interface mac address that appears in VM

Known Issues

MIPS-BE

Issues and possible workarounds for MetaROUTER feature on RouterBOARDs with MIPS-BE architecture

Random freezing

Only listed routers are affected by this issue: RB450G, RB750G, RB435G, RB493G

Certain RouterBOARD products tend to become unresponsive for some amount of time, after a while becoming available on the network. Similar problem is - that only guest becomes unresponsive and after a while continues to perform as expected. If watchdog is enabled on the router - router will be restarted by it instead of becoming available on its own.

Other routers from this architecture are believed to not to suffer from this issue.

Other issues that do not fit the description most probably are caused by RouterOS misconfiguration and does not have common denominator and have to be checked case by case.

As alternatives listed boards can be used instead: RB2011, RB433AH or any other from these series

PPC

Issues and possible workarounds for MetaROUTER feature on RouterBOARDs with PPC architecture

Not enough resources

Only listed routers are affected: RB1100AH

When attempt is made to create MetaROUTER guest on the router error message is given that there is not enough resources on the router to create guest.

This problem is resolved in 5.12 and later RouterOS releases. If you are using newer release and still encounter problem on the router you have to reinstall router using Netinstall tool.

RouterBOARD RB1100AHx2 reports similar message, but MetaROUTER feature is not currently supported on this router.

All other routers form this architecture that support MetaROUTER feature are not affected.

Other issues that do not fit the description most probably are caused by RouterOS misconfiguration and does not have common denominator and have to be checked case by case.

References

[1] http://www.mikrotik.com/download/metarouter/openwrt-mr-mips-rootfs.tgz
[2] http://www.mikrotik.com/download/metarouter/openwrt-mr-ppc-rootfs.tgz
[3] http://www.mikrotik.com/download/metarouter/openwrt-metarouter-1.2.patch
[4] http://kamikaze.openwrt.org/docs/openwrt.html#x1-410002.1.1

Manual:MLPPP over single and multiple links

Summary

Standards: RFC 1990

Package: ppp

Multi-Link Point to Point Protocol (MP, Multi-Link PPP, MultiPPP or MLPPP) is a method of splitting, recombining, and sequencing data across multiple logical data links.

In a situation where we have multiple DSL links a pair of devices, performance by “widening the pipe” between two devices can be increased by using Multi-Link PPP, without going to a newer, more expensive technology.

Large packets are actually split into bits and sent evenly over ALL logical data links. This is done instantaneously with NO loss of bandwidth. It is important to understand that other end of the link needs to use the same protocol to recombine your data.

Multilink is based on an LCP option negotiation that allows to indicate to its peer that it is capable of combining multiple physical links.

MLPPP over single link

Typically size of the packet sent over PPP link is reduced due to overhead. MP can be used to transmit and receive full frame over single ppp link. To make it work the Multilink Protocol uses additional LCP configuration options **Multilink Maximum Received Reconstructed Unit (MRRU)**

To enable Multi-link PPP over single link you must specify MRRU (Maximum Receive Reconstructed Unit) option. If both sides support this feature there are no need for MSS adjustment (in firewall mangle). Study shows that MRRU is less CPU expensive that 2 mangle rules per client. MRRU allows to divide packet to multiple channels therefore increasing possible MTU and MRU (up to 65535 bytes)

Under Windows it can be enabled in Networking tag, Settings button, "Negotiate multi-link for single link connections". Their MRRU is hard coded to 1614.

Configuration Example

Let's configure pppoe server compatible with Windows clients and MRRU enabled.

```
[admin@RB800] /interface pppoe-server server> add service-name=myPPP interface=ether1 mrru=1614
[admin@RB800] /interface pppoe-server server> print
Flags: X - disabled
 0   service-name="myPPP" interface=ether1 max-mtu=1480 max-mru=1480 mrru=1614
     authentication=pap,chap,mschap1,mschap2 keepalive-timeout=10 one-session-per-host=no
     max-sessions=0 default-profile=default
```

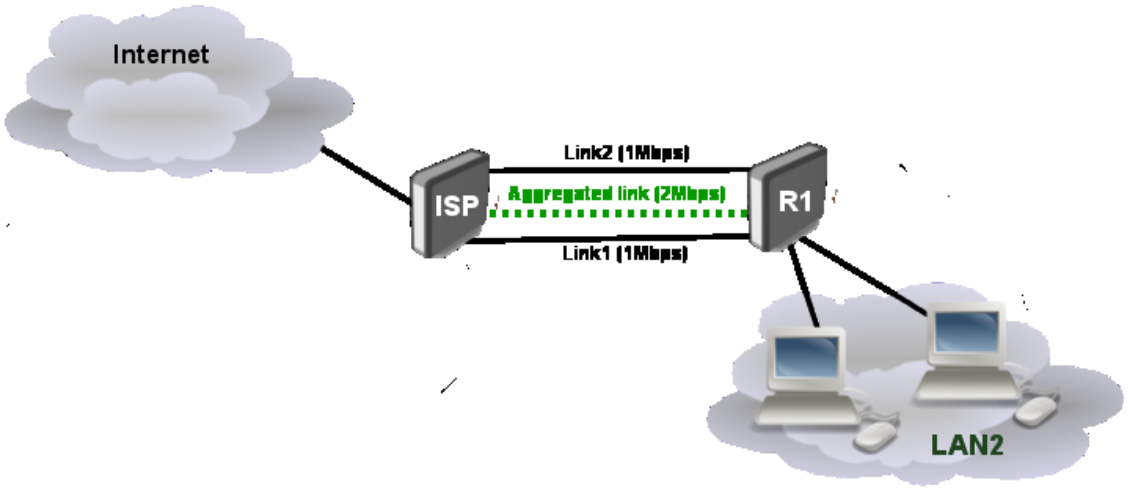
In short - standard PPP link - just specify MRRU in both sides :)

MLPPP over multiple links

MLPPP over multiple links allow to create a single ppp link over multiple physical connections. All PPP links must come from the same server (server must have MLPPP over multiple links support) and all PPP links must have same user name and password.

And to enable MLPPP you just need to create PPP client and specify multiple interfaces instead of single interface. Mikrotik RouterOS have MLPPP clent support starting from version 3.10. Presently there are no MLPPP server support available.

Configuration Example



ISP gives to its client two physical links (DSL lines) 1Mbps each. To get aggregated 2Mbps pipe we have to set up MLPPP. Consider ISP router is pre-configured to support MLPPP.

Configuration on Mikorotik router (R1) is:

```
/interface pppoe-client
   add service-name=ISP interface=ether1,ether2 user=xxx password=yyy disabled=no \
   add-default-route=yes use-peer-dns=yes
```

```
[admin@RB800] /interface pppoe-client> print
Flags: X - disabled, R - running
 0    name="pppoe-out1" max-mtu=1480 max-mru=1480 mrru=disabled interface=ether1,ether2
      user="xxx" password="yyy" profile=default service-name="ISP" ac-name="" add-default-route=yes
      dial-on-demand=no use-peer-dns=yes allow=pap,chap,mschap1,mschap2
```

Now when pppoe client is connected we can set up rest of configuration, local network address, enable dns requests, set up masquerade and firewall

```
/ip address add address=192.168.88.1/24 interface=local

/ip dns set allow-remote-request=yes

/ip firewall nat
add chain=src-nat action=masquerade out-interface=pppoe-out1
```

```
/ip firewall filter
add chain=input connection-state=invalid action=drop \
      comment="Drop Invalid connections"
add chain=input connection-state=established action=accept \
      comment="Allow Established connections"
add chain=input protocol=icmp action=accept \
      comment="Allow ICMP"
add chain=input src-address=192.168.88.0/24 action=accept \
      in-interface=!pppoe-out1
add chain=input action=drop comment="Drop everything else"
```

For more advanced router and customer protection check firewall examples.

See Also

- PPPOE
- Firewall and NAT

[Top | Back to Content]

Manual:MME wireless routing protocol

See also MME command reference

Note that MME is not a replacement for OSPF or RIP. It is meant to be used in mesh networks, and is best suited for wireless nodes with one logical interface. When used in traditional networks, the protocol overhead will be greater than even that of RIP.

Overview

MME (Mesh Made Easy) is a MikroTik routing protocol suited for IP level routing in wireless mesh networks. It is based on ideas from B.A.T.M.A.N. (Better Approach To Mobile Ad-hoc Networking) routing protocol. See https://www.open-mesh.net for more information about B.A.T.M.A.N.

MME works by periodically broadcasting so called originator messages. Routing information contained in a message consists of IP address of it's originator and optional list of IP prefixes - network announcements. If a node receives an originator message it hasn't seen before, it rebroadcasts that message. (There also are some other cases when the message can be rebroadcasted - see below.)

Unlike OLSR or other "traditional" proactive routing protocols, MME does not maintain network topology information. Consequently, MME is not able to calculate routing table, and does not need to. Instead, it keeps tracks of packets received and their sequence numbers - to tell how many packets were lost. This way, from message loss statistics for all combinations of originators and single-hop neighbors, MME is able to find the best gateway to a particular destination.

The main ideas behind MME are based on these observations made in mobile mesh networks:

- it can be impossible to know the exact topology of all network, because it is rapidly changing;
- if topology changes trigger routing table recalulation for all nodes in the network; and for embedded systems, the routing table calculation CPU overhead can be significant.

To avoid these problems, a MME node:

- cares only about the best single-hop neighbor in path to a particular destination;
- avoids routing table calculations.

Secondary functions of the MME protocol are: to carry information about gateways to the Internet, and to dynamically setup default routes. The part of MME responsible for that is dubbed "the gateway protocol".

MME protocol is using UDP port 1966 for originator message traffic. The gateway protocol is using TCP port 1968.

It is assumed in a normal operation of the protocol, a large number of these messages will get lost due to bad link quality. This assumption is important if we are talking about protocol overhead. Theoretically protocol's own traffic consumption is at least as big as for RIP, and obvioulsy worse than that of link state routing protocols (OSPF, OLSR) unless the topology is constantly changing.

Technical side

Basic principles of the main protocol

The main functions of the MME protocol are:

- automatic neighbor MME router (so called "originator") discovery (including multihop neighbors);
- originator message origination and flooding on each interface in every **origination-interval** seconds;
- originator message rebroadcasting based on a few simple rules;
- best gateway selection for each originator and the routes it has advertised.

Originator message rebroadcasting rules:

- do not rebroadcast self originated messages;
- do not rebroadcast messages that has *unidirectional* flag set;
- rebroadcast messages from single-hop neighbors; rebroadcast with *unidirectional* flag set if and only if:
  - the neighbor relation is not bidirectional;
  - OR the neighbor is not the best gateway to himself (i.e. there exists a better multihop path towards this node).
- rebroadcast messages that are not duplicate; a message is considered duplicate if message with this sequence number already was received before;
- rebroadcast duplicate messages if and only if:
  - they came from a neighbor that is the gateway for the originator;
  - the TTL in the packet is equal to last TTL for this neighbor and originator combination.

MME makes routing decisions based no more than last 64 messages received, but this number can be significantly less in case of packet loss. The node can tell that some packets were lost based on their sequence numbers. The more originator messages are received from a node, the better the statistics of that node is.

The MME protocol does not incorporate best route selection logic. If the same network information is configured in two different nodes, there currently is no way how to tell which one to prefer. Both routes will be installed in routing table and one of the selected in a random fashion. Obviously, such configuration is not recommended.

Basic principles of the gateway protocol

Second part of the MME is a default gateway selection protocol. Here two roles for a router are possible. A *gateway server* is node that is willing to serve as internet gateway for other routers. Usually it means it has an ethernet connection or some other way "out of the mesh".

A *gateway client* is a node that is willing to use this dynamic information to about gateways out of the mesh cloud. If there are multiple gateways reachable, client selects the best one based on packet statistics, advertised gateway class, and **gateway-selection** and **preferred-gateway** configuration values. After selecting the best gateway server the client makes a TCP connection to the server. This connection is used for periodic keep-alive message sending. After the connection is established, both the client and the server add dynamic IPIP tunnel interface. The client also adds

default route through this interface.

If the server stops announcing it's gateway capability, or becomes unreachable, the TCP connection and all tunnel state is teared down on both sides. Client also removes the default route.

Note that it's not recommended to have a default route (i.e. prefix 0.0.0.0/0) in MME network announcement configuration.

Packet format

The one and only packet type used in MME is *originator message*. The message contains:

- originator IP;
- current ttl value;
- sequence number;
- gateway class;
- protocol version;
- host and network announcements (0..n IP prefixes).

Gateway protocol clients and servers also exchange *keep-alive messages*, but they contain no information and have undefined format. At the moment, however, a keep-alive message is considered invalid, it if contains fewer than 1 or more than 6 octets.

Configuration examples

Starting the protocol on a single interface:

```
[admin@I] > /routing mme interface add interface=wlan1
```

To change some attributes for routes learned via MME you can use the **mme-in** routing filter. Example:

```
[admin@MikroTik] > routing filter add chain=mme-in set-routing-mark=mark1
```

If you want to redistribute some routes via MME, add them to MME networks. Example:

```
[admin@MikroTik] /routing mme> network add network=1.2.3.0/24
[admin@MikroTik] /routing mme> network p
Flags: X - disabled
 #   NETWORK
 0   1.2.3.0/24
```

Using the gateway protocol

Setup gateway server:

```
[admin@I] /routing mme> set gateway-class=11
```

Setup gateway client:

```
[admin@MikroTik] /routing mme> set gateway-selection=best-statistic
```

Observe the results (on client). Dynamic IPIP interface should be added automatically:

```
[admin@MikroTik] > /interface print
Flags: X - disabled, D - dynamic, R - running
 #    NAME                                                TYPE            MTU
 0  R ether1                                              ether           1500
 1  R ether2                                              ether           1500
```

```
 2 DR ipip1                                                      ipip            1480
```

Default route that goes through this tunnel should be added added automatically:

```
[admin@MikroTik] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE INTERFACE
 0 ADm  0.0.0.0/0                          r ipip1           130      ipip1
```

Manual:MPLS

Sub Categories

List of reference sub-pages

- **Interface**
  - vpls
  - traffic-eng
- **MPLS**
  - ldp
  - traffic-eng

Case studies

General

- MPLS Overview and RouterOS MPLS Implementation Status
- EXP bit behaviour
- L2MTU

Layer2 VPN

- LDP and LDP based VPLS
- BGP based VPLS
- Cisco style VPLS
- VPLS Control Word

Layer3 VPN

- Virtual Routing and Forwarding (VRF)
- OSPF as PE-CE routing protocol
- EBGP as PE-CE routing protocol

Traffic Engineering

- TE Tunnels
- TE Tunnel Bandwidth Control

List of examples

General

- MPLS over PPPoE

Layer2 VPN

- P2P L2VPN to Juniper router

Layer3 VPN

- A complete Layer-3 MPLS VPN example
- VRF Route Leaking
- Internet access from VRF
- Internet access from VRF with NAT

Traffic Engineering

- Simple TE configuration
- TE tunnels for VPLS

Summary

MikroTik RouterOS [1] **supports MPLS.** All MikroTik RouterBOARD [1] hardware products support MPLS.

General Porperties

Property	Description
dynamic-label-range (*range of integer[16..1048575]*; Default: **16-1048575**)	Range of Label numbers used for dynamic allocation. First 16 labels are reserved for special purposes (as defined in RFC). If you intend to configure labels statically then adjust dynamic default range not to include numbers that will be used in static configuration.
propagate-ttl (*yes* | *no*; Default: **yes**)	Whether to copy TTL values from IP header to MPLS header. If this option is set to **no** then hops inside MPLS cloud will be invisible from traceroutes.

Forwarding Table

Sub-menu: `/mpls forwarding-table`

Entries in this sub-menu shows label bindings for specific routes that will be used in MPLS label switching. Properties in this menu are read-only

Property	Description
bytes (*integer*)	Total number of packet bytes matched by this entry
destination (*IP/Mask*)	Destination prefix for which labels are assigned
in-label (*integer*)	Label number for incoming packet
interface (*string*)	
ldp (*yes* | *no*)	Whether labels are LDP signaled
nexthop (*IP*)	IP address of the nexthop
out-label (*integer*)	Label number which is added or switched to for outgoing packet.
packets (*integer*)	Number of packets matched by this entry
traffic-eng (*yes* | *no*)	Shows whether entry is signaled by RSVP-TE (Traffic Engineering)
vpls (*yes* | *no*)	Shows whether entry is used for VPLS tunnels.

For example we have forwarding table as shown below.

```
[admin@RB493G] /mpls forwarding-table> print
Flags: L - ldp, V - vpls, T - traffic-eng
 #   IN-LABEL      OUT-LABELS   DESTINATION                    IN NEXTHOP
 0   expl-null
 1 L 105                        10.255.255.36/32               lo 10.5.101.36
 2 L 120           112          3.3.3.1/32                     lo 10.5.101.3
 3 L 121           113          3.3.3.2/32                     lo 10.5.101.3
[admin@RB493G] /mpls forwarding-table>
```

You can see that all labels are LDP signaled. Note that for entry #1 there is no out-label, it means that MPLS label switching will not occur, packet will be sent out as regular IP packet. In the other hand entry #2 has in-label and out-label, which means that during packet forwarding label will be switched from 120 to 112.

Interface

Sub-menu: `/mpls interface`

This menu allows to configure MTUs including MPLS headers that interface can forward without fragmentation.

Note: If Ethernet card does not support Jumbo frames, then MPLS MTU for all interfaces on all devices participating in LSP should be set to 1500

Properties

Property	Description
comment (*string*; Default:)	Short description of the interface
disabled (*yes* | *no*; Default: **no**)	If set to **yes** then this configuration is ignored.
interface (*string* | *all*; Default: **all**)	Interface name to which apply settings. If set to **all** then the same config will be used for every interface if there is no specific configuration for the interface.
mpls-mtu (*integer [512..65535]*; Default: **1508**)	Option represents how big packets can be carried over the interface with added MPLS labels. `Read More >>`

In RouterOS by default have entry which sets MS MTU to 1508 for all interfaces.

```
[admin@RB493G] /mpls interface> print
Flags: X - disabled
 #    INTERFACE                                                        MPLS-MTU
 0    all                                                                  1508
```

Local Bindings

Sub-menu: `/mpls local-bindings`

This sub-menu shows labels bound to the routes locally in the router. In this menu also static bindings can be configured if there is no intention to use any of dynamic protocols (like LDP).

Properties

Property	Description
comments (*string*; Default:)	Short description of the entry
disabled (*yes* | *no*; Default: **no**)	
dst-address (*IP/Mask*; Default:)	Destination prefix for which label is assigned
label (*integer[0..1048576]* | *alert* | *expl-null* | *expl-null6* | *impl-null* | *none*; Default:)	Label number assigned to destination.

Read-only Properties

Property	Description
adv-path ()	
advertised (*yes* | *no*)	Whether binding was advertised to the neigbors
dynamic (*yes* | *no*)	Whether entry was dynamically added
egress (*yes* | *no*)	
gateway-route (*yes* | *no*)	Whether destination is reachable through the gateway.
local-route (*yes* | *no*)	Whether destination is locally reachable on the router
peers (*IP:label_space*)	IP address and label space of the peer to which this entry was advertised.

Remote Bindings

Sub-menu: `/mpls remote-bindings`

Sub-menu shows label bindings for routes received from other routers. This table is used to build Forwarding Table

[Top | Back to Content]

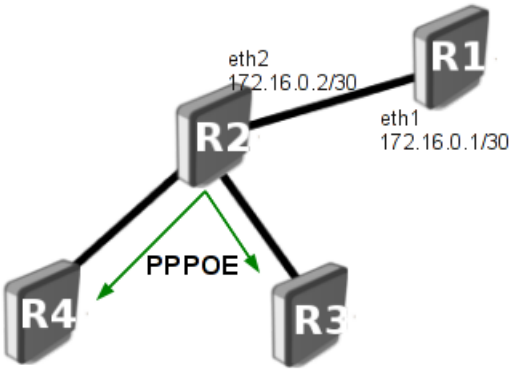
References

[1] http://mikrotik.com/software.html

Manual:MPLS over PPPoE

Applies to RouterOS: v3, v4, v5

This example shows how to set up MPLS network over PPPoE interfaces.



As you ca see from illustration above, router R2 is pppoe server and routers R3 and R4 are pppoe clients. Our goal is to run MPLS on this network.

When running MPLS over PPPoE or other tunnels you have to deal with MTU issues. Tunnels add more overhead (in our case PPPoE adds 8 more bytes). To be able to forward 1500 byte IP packet without fragmentation we will need interface that supports

1500 (IP frame)

+ 8 (PPPoE header)

+ 4 (MPLS header)

= 1512bytes

From RouterBoard MTU table you can check if RouterBoard supports 1512 L2MTU.

Lets say that R2 is RB433 and pppoe clients are connected to ether2. From the table you can see that max supported l2MTU for this interface is 1522.

It means that router will be able to forward packets without fragmentations.

Note: Since v5.0 is added proper support for MPLS over PPP. Now by default MPLS is disabled, to enable it go to

`/ppp profile` menu and set `use-mpls=`**`yes`**

```
/system identity set name=R1

# add loopback interface
/interface bridge
add name=loopback
/ip address
add address=10.255.255.1/32 interface=loopback
add address=172.16.0.1/30 interface=ether1

#set up ospf
/routing ospf instance
set default redistribute-connected=as-type-1
/routing ospf network
add network=172.16.0.0/30 area=backbone

# set up MPLS/LDP
/mpls interface set 0 mpls-mtu=1512
/mpls ldp
set enabled=yes lsr-id=10.255.255.1 transport-address=10.255.255.1
/mpls ldp interface
add interface=ether1
```

Note that we have to add static interface for each PPPoE clients, because later on these interfaces will be added to LDP configuration.

```
/system identity set name=R2

# add loopback interface
/interface bridge
add name=loopback
/ip address
add address=10.255.255.2/32 interface=loopback
add address=172.16.0.2/30 interface=ether1

# set up pppoe
/interface pppoe-server server
```

```
add interface=ether2 service-name=mpls max-mru=1500 max-mtu=1500

/ppp secret
add name=mplsR3 service=pppoe remote-address=192.168.0.2 local-address=192.168.0.1
add name=mplsR4 service=pppoe remote-address=192.168.0.3 local-address=192.168.0.1

/interface pppoe-server
add name="mplsR3" user="mplsR3" service="mpls"
add name="mplsR4" user="mplsR4" service="mpls"

#set up ospf
/routing ospf instance
set default redistribute-connected=as-type-1
/routing ospf network
add network=172.16.0.0/30 area=backbone
add network=192.168.0.2/32 area=backbone
add network=192.168.0.3/32 area=backbone

# set up MPLS/LDP
/mpls interface set 0 mpls-mtu=1512
/mpls ldp
set enabled=yes lsr-id=10.255.255.2 transport-address=10.255.255.2
/mpls ldp interface
add interface=ether1
add interface=mplsR3
add interface=mplsR4
```

```
/system identity set name=R3

# add loopback interface
/interface bridge
add name=loopback
/ip address
add address=10.255.255.3/32 interface=loopback

# set up pppoe
/interface pppoe-client
add name="mplsR3" max-mtu=1500max-mru=1500 interface=ether2 user="mplsR3" service-name=mpls

#set up ospf
/routing ospf instance
set default redistribute-connected=as-type-1
/routing ospf network
add network=192.168.0.1/32 area=backbone

# set up MPLS/LDP
/mpls interface set 0 mpls-mtu=1512
/mpls ldp
```

```
set enabled=yes lsr-id=10.255.255.3 transport-address=10.255.255.3
/mpls ldp interface
add interface=mplsR3
```

```
/system identity set name=R4

# add loopback interface
/interface bridge
add name=loopback
/ip address
add address=10.255.255.4/32 interface=loopback

# set up pppoe
/interface pppoe-client
add name="mplsR4" max-mtu=1500 max-mru=1500 interface=ether2 user="mplsR4" service-name=mpls

#set up ospf
/routing ospf instance
set default redistribute-connected=as-type-1
/routing ospf network
add network=192.168.0.1/32 area=backbone

# set up MPLS/LDP
/mpls interface set 0 mpls-mtu=1512
/mpls ldp
set enabled=yes lsr-id=10.255.255.4 transport-address=10.255.255.4
/mpls ldp interface
add interface=mplsR4
```

At first make sure pppoe clients are connected successfully

```
[admin@R2] /ppp active> print
Flags: R - radius
 #   NAME          SERVICE CALLER-ID          ADDRESS         UPTIME   ENCODING
 0   mplsR3        pppoe   00:0C:42:21:F1:EA 192.168.0.2     46m
 1   mplsR4        pppoe   00:0C:42:21:F1:ED 192.168.0.3     46m55s
```

Check if OSPF is running properly

```
[admin@R2] /routing ospf neighbor> print
 0 router-id=10.255.255.1 address=172.16.0.1 interface=wlan1 priority=1
   dr-address=172.16.0.2 backup-dr-address=172.16.0.1 state="Full"
   state-changes=5 ls-retransmits=0 ls-requests=0 db-summaries=0
   adjacency=5m19s

 1 router-id=10.255.255.3 address=192.168.0.2 interface=mplsR3 priority=1
   dr-address=0.0.0.0 backup-dr-address=0.0.0.0 state="Full" state-changes=4
   ls-retransmits=0 ls-requests=0 db-summaries=0 adjacency=49m33s

 2 router-id=10.255.255.4 address=192.168.0.3 interface=mplsR4 priority=1
```

```
   dr-address=0.0.0.0 backup-dr-address=0.0.0.0 state="Full" state-changes=4
   ls-retransmits=0 ls-requests=0 db-summaries=0 adjacency=50m31s
```

Ensure LDP is running

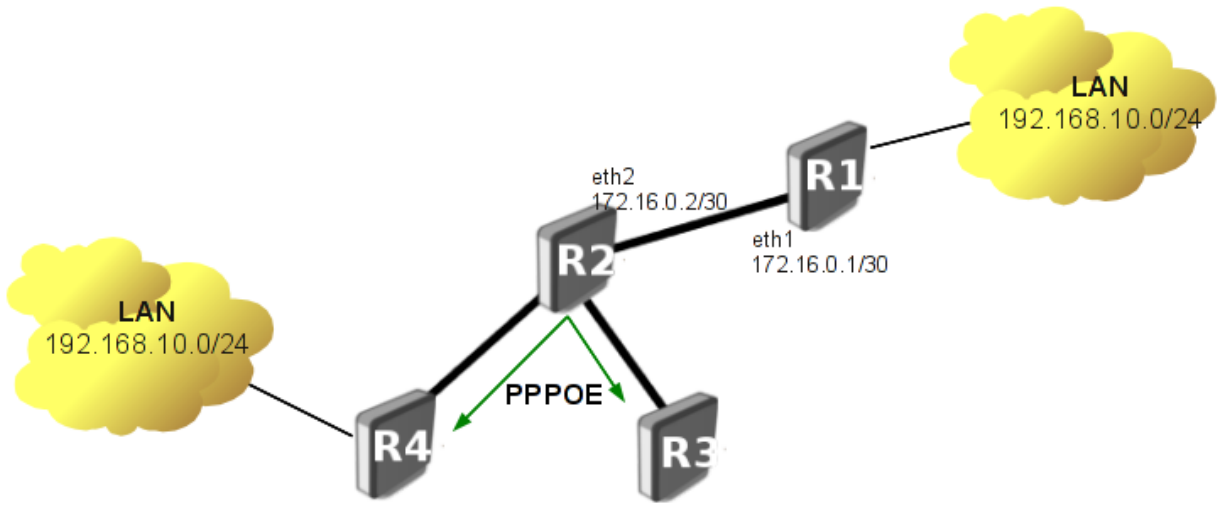
```
[admin@R2] /mpls ldp neighbor> print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello,
V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                      SEN
 0 DO   10.255.255.3    10.255.255.2    10.255.255.3:0            no
 1 DO   10.255.255.4    10.255.255.2    10.255.255.4:0            no
 2 DO   10.255.255.1    10.255.255.2    10.255.255.1:0            no
```

```
[admin@R2] /mpls forwarding-table> print
Flags: L - ldp, V - vpls, T - traffic-eng
 #    IN-LABEL      OUT-LABELS  DESTINATION                   I NEXTHOP
 0    expl-null
 1 L 20                         192.168.0.1/32                m 192.168.0.3
 2 L 21                         10.255.255.4/32               m 192.168.0.3
 3 L 22                         10.255.255.3/32               m 192.168.0.2
 4 L 23                         10.255.255.1/32               w 172.16.0.1
 5 L 24                         192.168.88.0/24               w 172.16.0.1
```

Now we can check if packet switching is working as expected

```
[admin@R4] /mpls ldp neighbor> /tool traceroute 10.255.255.1 src-address=10.255.255.4
     ADDRESS                                    STATUS
   1     192.168.0.1 13ms 19ms 143ms
                       mpls-label=23
   2    10.255.255.1 38ms 15ms 14ms
```

This example extends previous setup by connecting two local networks using VPLS tunnel



Manual:MPLS/EXP bit behaviour

MPLS label EXP field overview

When MPLS label is attached to packet, it increases packet length by 32 bits (4 bytes). These 32 bits are broken down as follows:

- label value itself (20 bits)
- EXP ("experimental") field (3 bits)
- time to live field (8 bits)
- bottom of stack field (1 bit)

Use of "experimental" bits is not specified by MPLS standards, but most common use is to carry QoS information, similar to 802.1q priority in VLAN tag. Note that EXP field is 3 bits only therefore it can carry values from 0 to 7 only, which allows to have 8 traffic classes.

EXP field treatment in RouterOS

When RouterOS receives MPLS packet, it sets "ingress priority" value for packet to that carried inside top label. Note that "ingress priority" **is not** a field inside packet headers - it can be thought of like additional mark assigned to packet while being processed by router. When RouterOS labels MPLS packet, it sets EXP bits to "priority" (not "ingress priority"!) assigned to packet. When RouterOS switches MPLS packet, "ingress priority" is automatically copied to "priority", this way regular MPLS switching communicates priority info over whole label switched path.

Additional info on "ingress priority" and "priority" handling is also in WMM.

Therefore what happens to EXP field depends based on what action is taken on packet:

- if packet is MPLS switched (by popping label off packet and pushing on new one), EXP field in new label will be the same as in received label, because:
  - RouterOS sets "ingress priority" to EXP bits in received label
  - Switching automatically sets "priority" to "ingress priority"
  - RouterOS labels packet with new label and sets its EXP bits to value in "priority".
- if packet is MPLS switched by using penultimate-hop-popping (received label is popped off and no new one is pushed on), EXP field of received priority stays in "priority" field of packet and may be used by some other MAC protocol, e.g. WMM or 802.1q VLAN, for example:
  - RouterOS sets "ingress priority" to EXP bits in received label
  - Switching automatically sets "priority" to "ingress priority"
  - RouterOS switches packet to next hop (without pushing on label) and that happens over VLAN interface
  - VLAN interface sets 802.1q priority in VLAN header to "priority" value of packet.

Note that penultimate-hop-popping can therefore loose QoS information carried over label switched path at the last hop. In cases where this is not desirable, penultimate-hop-popping behaviour should be disabled by using Explicit NULL label instead of Implicit NULL label for last hop in label switched path. Using Explicit NULL label for last hop is default behaviour for MPLS TE tunnels.

- if packet is supposed to be sent over label switched path (first label will get pushed on packet), EXP bits will be set to value in "priority", which in turn can be set up properly using firewall rules or other means (e.g. from DSCP field in IP header)
- if packet is received for local processing, "ingress priority" is set to EXP field of received packet and can therefore be used to update DSCP field of packet or set "priority" from "ingress priority" using firewall rules

Manual:MPLS/LDP

Applies to RouterOS: v3, v4 +

Summary

MikroTik RouterOS implements Label Distribution Protocol (RFC 3036, RFC 5036) for IPv4. LDP is a protocol defined for distributing labels. It is the set of procedures and messages by which Label Switched Routers (LSRs) establish Label Switched Paths (LSPs) through a network by mapping network-layer routing information directly to data-link layer switched paths.

General

Sub-menu: `/mpls ldp`

General LDP settings.

Properties:

Property	Description
distribute-for-default-route (*yes* | *no*; Default: **no**)	Defines whether to distribute label for default route or not.
enabled (*yes* | *no*; Default: **yes**)	Starts LDP protocol.
hoplimit (*integer*; Default: **255**)	Max hop limit used for loop detection.
loop-detect (*yes* | *no*; Default: **no**)	Defines whether to run LSP loop detection or not. Will not work correctly if not enabled on all LSRs. Should be used only on non-TTL networks such as ATM.
lsr-id (*IP*; Default: **0.0.0.0**)	Unique label switching router's ID. If set to **0.0.0.0** highest IP address on the router is used.
path-vector-limit (*integer*; Default: **255**)	Max path vector limit used for loop detection.
transport-address (*IP*; Default: **0.0.0.0**)	Specifies LDP session connections origin address and also advertise this address as transport address to LDP neighbors. If set to **0.0.0.0** highest IP address on the router is used.
use-explicit-null (*yes* | *no*; Default: **no**)	Whether to distribute explicit-null label bindings.

Interface

Sub-menu: `/mpls ldp interface`

List of interfaces that connects Label Switching routers.

Properties:

Property	Description
accept-dynamic-neighbors (*yes* | *no*; Default: **yes**)	Defines whether to discover neighbors dynamically or use only statically configured in LDP neighbors menu
comment (*string*; Default:)	Short description of the item
disabled (*yes* | *no*; Default: **no**)	Defines whether item is ignored or used
hello-interval (*time*; Default: **5s**)	The interval between hello packets that the router sends out this interface.
hold-time (*time*; Default: **15s**)	Specifies the interval after which a neighbor is declared as not reachable.
transport-address (*IP*; Default: **0.0.0.0**)	Used transport address if differs from general settings. If set to **0.0.0.0** transport address from general settings is used.
interface (*string*; Default:)	Name of the interface on which to run LDP

Neighbors

Sub-menu: `/mpls ldp neighbor`

Properties

Property	Description
comment (*string*; Default:)	Short description of the item
disabled (*yes* | *no*; Default: **no**)	Defines whether item is ignored or used
send-targeted (*yes* | *no*; Default: **yes**)	Specifies whether to send targeted hellos, used for targeted (not directly connected) LDP sessions.
transport (*IP*; Default:)	Transport address used by remote neighbor.

Read-only properties

Property	Description
addresses (*IP[,IP[,...]]*)	List of all IP addresses on LDP neighbor
dynamic (*yes* | *no*)	Shows whether item is dynamically created
local-transport (*IP*)	Transport address used to send messages to the neighbor.
operational (*yes* | *no*)	Shows whether item is running
peer (*LSR-ID:integer*)	Shows remote neighbor's LSR ID and label space.
sending-targeted-hello (*yes* | *no*)	Shows whether targeted hellos are sent to the neighbor.
vpls (*yes* | *no*)	Shows whether the neighbor is used for VPLS tunnel.

Accept Filters

Sub-menu: `/mpls ldp accept-filter`

List of label bindings which should be accepted from LDP neighbors.

Properties:

Property	Description
accept (*yes* | *no*; Default: **yes**)	Whether to accept label bindings from the neighbors for specified prefix.
comment (*string*; Default:)	Short description of the item
disabled (*yes* | *no*; Default: **no**)	Defines whether item is ignored or used
neighbor (*all* | *IP*; Default: **all**)	Neighbor to which this filter applies.
prefix (*IP/Mask*; Default: **0.0.0.0/0**)	

Advertise Filters

Sub-menu: `/mpls ldp advertise-filter`

List of label bindings which should be advertised to LDP neighbors.

Properties:

Property	Description
accept (*yes* | *no*; Default: **yes**)	Whether to advertise label bindings to the neighbors for specified prefix.
comment (*string*; Default:)	Short description of the item
disabled (*yes* | *no*; Default: **no**)	Defines whether item is ignored or used
neighbor (*all* | *IP*; Default: **all**)	Neighbor to which this filter applies.
prefix (*IP/Mask*; Default: **0.0.0.0/0**)	

[Top | Back to Content]

Manual:MPLS/Overview

MPLS Overview

MPLS stands for MultiProtocol Label Switching. It kind of replaces IP routing - packet forwarding decision (outgoing interface and next hop router) is no longer based on fields in IP header (usually destination address) and routing table, but on labels that are attached to packet. This approach speeds up forwarding process because next hop lookup becomes very simple compared to routing lookup (finding longest matching prefix).

Efficiency of forwarding process is the main benefit of MPLS, but it must be taken into account that MPLS forwarding disables processing of network layer (e.g. IP) headers, therefore no network layer based actions like NAT and filtering can be applied to MPLS forwarded packets. Any network layer based actions should be taken on ingress or egress of MPLS cloud, with preferred way being ingress - this way, e.g. traffic that is going to be dropped anyway does not travel through MPLS backbone.

In the simplest form MPLS can be thought of like improved routing - labels are distributed by means of LDP protocol for routes that are active and labeled packet takes the same path it would take if it was not labeled. Router that routes unlabeled packet using some route for which it has received label from next hop, imposes label on packet and send it to next hop - it gets MPLS switched further along its path. Router that receives packet with label it has assigned to some route changes packet label with one received from next hop of particular route and sends packet to next hop. Label switched path ensures delivery of data to the MPLS cloud egress point. Applications of MPLS are based on this basic MPLS concept of label switched paths.

Another way of establishing label switching path is traffic engineering tunnels (TE tunnels) by means of RSVP TE protocol. Traffic engineering tunnels allow explicitly routed LSPs and constraint based path selection (where constraints are interface properties and available bandwidth).

Taking into account complexity, new protocols and applications that MPLS introduces and differences of concepts that MPLS adds to routed/bridged network, it is recommended to have in depth understanding of MPLS concepts before implementing MPLS in production network. Some suggested reading material:

- Multiprotocol Label Switching http://en.wikipedia.org/wiki/Multiprotocol_Label_Switching
- RFC3031 Multiprotocol Label Switching Architecture http://www.ietf.org/rfc/rfc3031.txt
- MPLS Fundamentals by Luc De Ghein http://www.amazon.com/MPLS-Fundamentals-Luc-Ghein/dp/1587051974

RouterOS MPLS features

As of version 3.8 MPLS feature development for RouterOS continues in **mpls-test** package that requires **routing-test** package. Currently RouterOS (by means of mpls-test and routing-test packages) supports the following MPLS related features:

- MPLS switching with penultimate hop popping support
- static local label bindings for IPv4
- static remote label bindings for IPv4
- Label Distribution Protocol (RFC 3036, RFC 5036) for IPv4
  - downstream unsolicited label advertisement
  - independent label distribution control
  - liberal label retention
  - targeted session establishment
  - optional loop detection
- Virtual Private Lan Service
  - VPLS LDP signaling (RFC 4762)
  - VPLS pseudowire fragmentation and reassembly (RFC 4623)
  - VPLS MP-BGP based autodiscovery and signaling (RFC 4761), see BGP based VPLS
- RSVP TE Tunnels
  - tunnel head-end
  - explicit paths
  - OSPF extensions for TE tunnels
  - CSPF path selection
  - forwarding of VPLS and MPLS IP VPN traffic on TE tunnels
- MP-BGP based MPLS IP VPN
- OSPF extensions for MPLS TE

Features since version 3.17:

- support for OSPF as CE-PE protocol
- ping and traceroute for specified VRF
- control over network layer TTL propagation in MPLS

Features since version 3.20 (note that this version changes configuration syntax and adds new parameters!):

- Cisco style static VPLS pseudowires (RFC 4447 FEC type 0x80)
- Cisco VPLS BGP-based auto-discovery (draft-ietf-l2vpn-signaling-08)
- support for multiple import/export route target extended communities for BGP based VPLS (both, RFC 4761 and draft-ietf-l2vpn-signaling-08)

Features since version 3.23

- Ingress TE tunnel rate limit and automatic reserved bandwidth adjustment, see TE Tunnel Bandwidth Control
- all tunnel bandwidth settings are specified and displayed in bits per second
- complete support for OSPF as PE-CE routing protocol (including sham links)

Features since version 3.24

- RIP as CE-PE protocol
- per-VRF BGP instance redistribution settings

MPLS features that RouterOS DOES NOT HAVE yet:

- IPv6 support
- LDP features:
  - downstream on demand label advertisement
  - ordered label distribution control
  - conservative label retention
- TE features
  - fast reroute
  - link/node protection
- Support for BGP as label distribution protocol

To ensure compatibility with other manufacturer equipment ensure that required features match, if uncertain, consult with Mikrotik support. RouterOS LDP and TE implementation has been tested with Cisco IOS.

Manual:MPLS/Traffic-eng

Applies to RouterOS: v3, v4 +

Interface

Sub-menu: `/mpls traffic-eng interface`

Properties:

Property	Description
bandwidth (*integer[bps]*; Default: **0bps**)	Total bandwidth that can be allocated on an interface by TE tunnels.
blockade-k-factor (*integer*; Default: **3**)	Value used to calculate blockade state timeout.
comment (*string*; Default:)	Short description of the item
disabled (*yes* | *no*; Default: **yes**)	Defines whether item is ignored or used. By default VPLS interface is disabled.
down-flood-thresholds (*integer[0..100],interer[0..100],...*; Default: **15,30,45,60,75,80,85,90,95,97,98,99,100**)	
igp-flood-period (*time*; Default: **3m**)	
interface (*string*; Default:)	Name of an interface on which to run RSVP.

k-factor (*integer*; Default: **3**) — Value used to calculate RSVP timeout. Timeout is calculated using following formula: `(K + 0.5)*1.5*R`, where K is `k-factor`, R is `refresh-time`. Read more >>

refresh-time (*time*; Default: **30s**) — Interval in which RSVP Path messages are sent out.

resource-class (*integer[0..FFFFFFFF]*; Default: **0**)

te-metric (*integer*; Default: **1**)

up-flood-thresholds (*integer[0..100],interer[0..100],...*; Default: **15,30,45,60,75,80,85,90,95,97,98,99,100**)

use-udp (*yes* | *no*; Default: **no**) — An RSVP implementation generally requires the ability to perform "raw" network I/O, i.e., to send and receive IP datagrams using protocol 46. Some systems may not support raw network I/O, in such cases RSVP messages can be encapsulated in UDP datagrams. Ports 1698 and 1699 will be used.

Read-only properties:

Property	Description
remaining-bw (*integer[bps]*)	Shows currently unallocated bandwidth.

Tunnel Path

Sub-menu: `/mpls traffic-eng tunnel-path`

Properties:

Property	Description
affinity-exclude (*integer*; Default:)	Do not use the path if `resource-class` matches any of specified bits.
affinity-include-all (*integer*; Default:)	Use the path only if `resource-class` matches all of specified bits.
affinity-include-any (*integer*; Default:)	Use the path if `resource-class` matches any of specified bits.
comment (*string*; Default:)	Short description of the item
disabled (*yes* | *no*; Default: **yes**)	Defines whether item is ignored or used. By default VPLS interface is disabled.
holding-priority (*integer[0..7]*; Default:)	Is used to decide whether this path can be preempted by another path. 0 sets the highest priority.
hops (*Address:[strict|loose] [, Address:[strinct|loose]]*; Default:)	List of hops that path traverses. Used if `use-cspf` is not enabled. It is possible to specify **strict** hop or **loose** hop: • `strict` - defines that there must not be any other hops between previous hop and "strict" hop (fully specified path). • `loose` - there are acceptable other hops between previous hop and defined hop (not fully specified path). Read more >>
name (*string*; Default:)	Descriptive name of tunnel path
record-route (*yes* | *no*; Default:)	If enabled, the sender node will receive information about the actual route that the LSP tunnel traverses. Record Route is analogous to a path vector, and hence can be used for loop detection.
reoptimize-interval (*time*; Default:)	Interval in which tunnel path will be re-optimized. Useful if `use-cspf` is set to **yes**.
setup-priority (*integer[0..7]*; Default:)	Parameter is used to decide whether this path can preempt another path. 0 sets the highest priority.
use-cspf (*yes* | *no*; Default: **yes**)	Whether to use CSPF to create dynamic tunnel path.

Monitoring TE Status

Path State

Sub-menu: `/mpls traffic-eng path-state`

Available read only properties:

Property	Description
bandwidth (*integer[bps]*)	Bandwidth required for the path
dst (*address:integer*)	Shows TE path destination address and tunnel ID.
egress (*yes* | *no*)	Shows if router is egress router of the path
forwarding (*yes* | *no*)	Shows if router is forwarding router of the path
in-interface (*string*)	Interface on which path message is received.
in-previous-hop (*IP*)	Recorded previous hop
label (*integer*)	
locally-originated (*yes* | *no*)	Shows if router is ingress router of the path
out-interface (*string*)	Interface through which path message is sent out.
out-label (*integer*)	
out-next-hop (*IP*)	
path-in-explicit-route ()	
path-in-record-route (*List of IPs*)	Received recorded routes along the path.
path-out-explicit-route ()	
path-out-record-route ()	List of recorded routes along the path that is sent out to next hop.
resv-bandwidth (*integer[bps]*)	bandwidth that TE path is reserving.
resv-out-record-route ()	
sending-path (*yes* | *no*)	Whether path messages are being sent
sending-resv (*yes* | *no*)	Whether resv messages are being sent
src (*Address:ID*)	Shows source address and LSP ID number

Resv State

Sub-menu: `/mpls traffic-eng resv-state`

Available read only properties:

Property	Description
active (*yes* | *no*)	Shows whether reservation is active.
bandwidth (*integer[bps]*)	Bandwidth that RSVP session is allocating.
dst (*address:ID*)	Shows TE destination address and tunnel ID from RSVP session.
egress (*yes* | *no*)	Shows if router is egress router of the path
interface (*string*)	Shows an interface on which bandwidth is reserved
label (*integer*)	
next-hop ()	
non-output (*yes* | *no*)	
recorded-route (*IP[label]*)	Shows recorded routes and labels along LSP.
shared (*yes* | *no*)	Whether LSP tunnels can share resources, so that the new LSP tunnel can be set up without having to wait for the old LSP tunnel to be cleared. `Read more >>`
src (*address:ID*)	Shows TE source address and LSP ID from RSVP session.

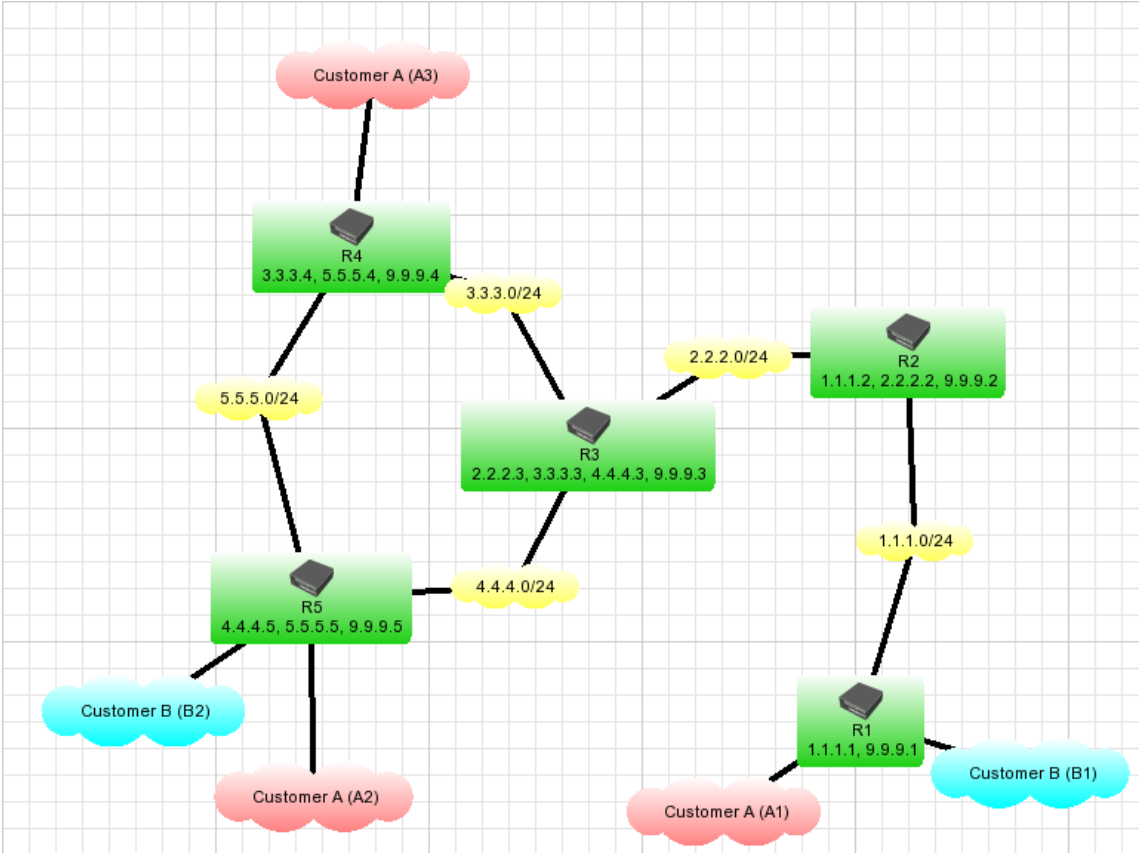
Manual:MPLSVPLS

MPLS Overview

For MPLS overview and MPLS features that RouterOS supports see MPLS Overview

Example network

Consider network service provider that is connecting 3 remote sites of Customer A (A1,A2 and A3) and 2 remote sites of Customer B (B1 and B2) using its routed IP network core, consisting of routers (R1-R5):



Customers require transparent ethernet segment connection between sites. So far it has been implemented by means of bridging EoIP tunnels with physical ethernet interfaces.

Note that there are no IP addresses configured on R1, R4 and R5 interfaces that face customer networks.

Enabling MPLS forwarding can speed up packet forwarding process in such network. Using one of MPLS applications - VPLS can further increase efficency of ethernet frame forwarding by not having to encapsulate ethernet frames in IP frames, thus removing IP header overhead.

This guide gives step by step instructions that will lead to implementation of VPLS to achieve necessary service.

Prerequisites for MPLS

"Loopback" IP address

Although not a strict requirement, it is advisable to configure routers participating in MPLS network with "loopback" IP addresses (not attached to any real network interface) to be used by LDP to establish sessions.

This serves 2 purposes:

- as there is only one LDP session between any 2 routers, no matter how many links connect them, loopback IP address ensures that LDP session is not affected by interface state or address changes
- use of loopback address as LDP transport address ensures proper penultimate hop popping behaviour when multiple labels are attached to packet as in case of VPLS

In RouterOS "loopback" IP address can be configured by creating dummy bridge interface without any ports and adding address to it. For example, on R1 it is done with the following commands:

```
/interface bridge add name=lobridge
/ip address add address=9.9.9.1/32 interface=lobridge
```

The rest of routers are configured similar way.

IP connectivity

As LDP distributes labels for active routes, essential requirement is properly configured IP routing. LDP by default distributes labels for active IGP routes (that is - connected, static, and routing protocol learned routes, except BGP).

In given example setup OSPF is used to distribute routes. For example, on R5 OSPF is configured with the following commands:

```
/routing ospf set redistribute-connected=as-type-1
/routing ospf network add area=backbone network=4.4.4.0/24
/routing ospf network add area=backbone network=5.5.5.0/24
```

On other routers OSPF is configured in similar way.

This yields routing table on R5 like this:

```
[admin@R5] > ip route print
Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY-STATE GATEWAY         DISTANCE             INTERFACE
 0 ADo  1.1.1.0/24                         reachable     4.4.4.3         110                  ether1
 1 ADo  2.2.2.0/24                         reachable     4.4.4.3         110                  ether1
 2 ADo  3.3.3.0/24                         reachable     4.4.4.3         110                  ether1
                                           reachable     5.5.5.4                              ether2
 3 ADC  4.4.4.0/24         4.4.4.5                                       0                    ether1
 4 ADC  5.5.5.0/24         5.5.5.5                                       0                    ether2
 5 ADo  9.9.9.1/32                         reachable     4.4.4.3         110                  ether1
 6 ADo  9.9.9.2/32                         reachable     4.4.4.3         110                  ether1
 7 ADo  9.9.9.3/32                         reachable     4.4.4.3         110                  ether1
 8 ADo  9.9.9.4/32                         reachable     5.5.5.4         110                  ether2
 9 ADC  9.9.9.5/32         9.9.9.5                                       0                    lobridge
```

and traceroute from R5 to R1 like this:

```
[admin@R5] > /tool traceroute 9.9.9.1
     ADDRESS                                    STATUS
   1          4.4.4.3 11ms 1ms 4ms
   2          2.2.2.2 23ms 3ms 2ms
   3          9.9.9.1 26ms 3ms 3ms
```

Configuring LDP

In order to distribute labels for routes, LDP should get enabled. On R1 this is done by commands (interface ether3 is facing network 1.1.1.0/24):

```
/mpls ldp set enabled=yes transport-address=9.9.9.1 lsr-id=9.9.9.1
/mpls ldp interface add interface=ether3
```

Note that transport-address gets set to 9.9.9.1. This makes router originate LDP session connections with this address and also advertise this address as transport address to LDP neighbors.

Other routers are configured in similar way - LDP is enabled on interfaces that connect routers and not enabled on interfaces that connect customer networks. For example, on R5:

```
[admin@R5] > /ip address print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   4.4.4.5/24         4.4.4.0         4.4.4.255       ether1
 1   5.5.5.5/24         5.5.5.0         5.5.5.255       ether2
 2   9.9.9.5/32         9.9.9.5         9.9.9.5         lobridge
[admin@R5] > /mpls ldp interface print
Flags: I - invalid, X - disabled
 #   INTERFACE                                      HELLO-INTERVAL       HOLD-TIME
 0   ether1                                         5s                   15s
 1   ether2                                         5s                   15s
```

After LDP sessions are established, on R5 there are 2 LDP neighbors:

```
[admin@R5] > /mpls ldp neighbor print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                          SEND-TARGETED ADDRESSES
 0 DO   9.9.9.4         9.9.9.5         9.9.9.4:0                     no            3.3.3.4
                                                                                      5.5.5.4
                                                                                      9.9.9.4
 1 DO   9.9.9.3         9.9.9.5         9.9.9.3:0                     no            2.2.2.3
                                                                                      3.3.3.3
                                                                                      4.4.4.3
                                                                                      9.9.9.3
```

/mpls local-bindings shows labels that this router has assigned to routes and peers it has distributed the label to. It shows that R5 has distributed labels for all its routes to both of its neighbors - R3 and R4

```
[admin@R5] > /mpls local-bindings print
Flags: X - disabled, A - advertised, D - dynamic, L - local-route, G - gateway-route, e - egress
 #      DST-ADDRESS        LABEL      PEERS
 0 ADLe 4.4.4.0/24         impl-null  9.9.9.4:0
```

```
                                    9.9.9.3:0
 1 ADLe 9.9.9.5/32         impl-null  9.9.9.4:0
                                    9.9.9.3:0
 2 ADG  9.9.9.4/32         17         9.9.9.4:0
                                    9.9.9.3:0
 3 ADLe 5.5.5.0/24         impl-null  9.9.9.4:0
                                    9.9.9.3:0
 4 ADG  1.1.1.0/24         18         9.9.9.4:0
                                    9.9.9.3:0
 5 ADG  2.2.2.0/24         19         9.9.9.4:0
                                    9.9.9.3:0
 6 ADG  9.9.9.1/32         20         9.9.9.4:0
                                    9.9.9.3:0
 7 ADG  9.9.9.2/32         21         9.9.9.4:0
                                    9.9.9.3:0
 8 ADG  9.9.9.3/32         22         9.9.9.4:0
                                    9.9.9.3:0
 9 ADG  3.3.3.0/24         23         9.9.9.4:0
                                    9.9.9.3:0
```

/mpls remote-bindings shows labels that are allocated for routes by neighboring routers and advertised to this router:

```
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP          LABEL      PEER
 0  D 4.4.4.0/24                          16         9.9.9.4:0
 1 AD 3.3.3.0/24         5.5.5.4          impl-null  9.9.9.4:0
 2  D 9.9.9.5/32                          17         9.9.9.4:0
 3 AD 9.9.9.4/32         5.5.5.4          impl-null  9.9.9.4:0
 4  D 5.5.5.0/24                          impl-null  9.9.9.4:0
 5  D 1.1.1.0/24                          18         9.9.9.4:0
 6  D 2.2.2.0/24                          19         9.9.9.4:0
 7  D 9.9.9.1/32                          20         9.9.9.4:0
 8  D 9.9.9.2/32                          21         9.9.9.4:0
 9  D 9.9.9.3/32                          22         9.9.9.4:0
10 AD 1.1.1.0/24         4.4.4.3          16         9.9.9.3:0
11 AD 2.2.2.0/24         4.4.4.3          impl-null  9.9.9.3:0
12  D 4.4.4.0/24                          impl-null  9.9.9.3:0
13  D 3.3.3.0/24                          impl-null  9.9.9.3:0
14 AD 9.9.9.1/32         4.4.4.3          17         9.9.9.3:0
15 AD 9.9.9.3/32         4.4.4.3          impl-null  9.9.9.3:0
16  D 9.9.9.4/32                          18         9.9.9.3:0
17  D 5.5.5.0/24                          19         9.9.9.3:0
18 AD 9.9.9.2/32         4.4.4.3          20         9.9.9.3:0
19  D 9.9.9.5/32                          21         9.9.9.3:0
```

Here we can observe that R5 has received label bindings for all routes from both its neighbors - R3 and R4, but only the ones for whom particular neighbor is next hop are active. For example:

```
[admin@R5] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE             INTERFACE
...
 5 ADo  9.9.9.1/32                         r 4.4.4.3         110                  ether1
...
```

```
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS         NEXTHOP          LABEL      PEER
...
 7  D 9.9.9.1/32                          20         9.9.9.4:0
...
14 AD 9.9.9.1/32          4.4.4.3          17         9.9.9.3:0
...
```

From the above we see that R3, which is next hop for network 9.9.9.1/32 from R5 perspective, has assigned label 17 for traffic going to 9.9.9.1/32. This implies that when R5 will be routing traffic to this network, will impose label 17.

Label switching rules can be seen in /mpls forwarding-table. For example, on R3 it looks like this:

```
[admin@R3] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION        INTERFACE            NEXTHOP
 ...
 2 17                   17                   9.9.9.1/32         ether1               2.2.2.2
 ...
```

This rule says that R3 receiving packet with label 17 will change it to label 17 assigned by R2 for network 9.9.9.1/32 (R2 is next hop for 9.9.9.1/32 from R3 perspective):

```
[admin@R2] > /mpls local-bindings print
Flags: X - disabled, A - advertised, D - dynamic, L - local-route, G - gateway-route, e - egress
 #      DST-ADDRESS        LABEL      PEERS
 ...
 3 ADG  9.9.9.1/32         17         9.9.9.1:0
                                      9.9.9.3:0
 ...
```

R2 MPLS forwarding table tells:

```
[admin@R2] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION        INTERFACE            NEXTHOP
 ...
 1 17                                        9.9.9.1/32         ether1               1.1.1.1
 ...
```

Notice, that forwarding rule does not have any out-labels. The reason for this is that R2 is doing penultimate hop popping for this network. R1 does not assign any real label for 9.9.9.1/32 network, because it is known that R1 is egress point for 9.9.9.1/32 network (router is egress point for networks that are directly connected to it, because next hop for traffic is not MPLS router), therefore is advertises "implicit null" label for this route:

```
[admin@R2] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 ...
13 AD 9.9.9.1/32         1.1.1.1         impl-null  9.9.9.1:0
 ...
```

This tells R2 to forward traffic for 9.9.9.1/32 to R1 unlabelled, which is exactly what R2 mpls forwarding-table entry tells. Penultimate hop popping ensures that routers do not have to do unnecessary label lookup when it is known in advance that router will have to route packet.

Using traceroute in MPLS networks

RFC4950 introduces extensions to ICMP protocol for MPLS. The basic idea is that some ICMP messages may carry MPLS "label stack object" (list of labels that were on packet when it caused particular ICMP message). ICMP messages of interest for MPLS are Time Exceeded and Need Fragment.

MPLS label carries not only label value, but also TTL field. When imposing label on IP packet, MPLS TTL is set to value in IP header, when last label is removed from IP packet, IP TTL is set to value in MPLS TTL. Therefore MPLS switching network can be diagnosed by means of traceroute tool that supports MPLS extension.

For example, traceroute from R5 to R1 looks like this:

```
[admin@R5] > /tool traceroute 9.9.9.1 src-address=9.9.9.5
     ADDRESS                                    STATUS
   1         4.4.4.3 15ms 5ms 5ms
                      mpls-label=17
   2         2.2.2.2 5ms 3ms 6ms
                      mpls-label=17
   3         9.9.9.1 6ms 3ms 3ms
```

Traceroute results show MPLS labels on packet when it produced ICMP Time Exceeded. The above means: when R3 received packet with MPLS TTL 1, it had label 17 on. This matches label advertised by R3 for 9.9.9.1/32. The same way R2 observed label 17 on packet on next traceroute iteration - R3 switched label 17 to label 17, as explaned above. R1 received packet without labels - R2 did penultimate hop popping as explaned above.

Drawbacks of using traceroute in MPLS network

Label switching ICMP errors

One of drawbacks of using traceroute in MPLS networks is the way MPLS handles produced ICMP errors. In IP networks ICMP errors are simply routed back to source of packet that caused the error. In MPLS network it is possible that router that produces error message does not even have route to source of IP packet (for example in case of assymetric label switching paths or some kind of MPLS tunneling, e.g. to transport MPLS VPN traffic).

Due to this produced ICMP errors are not routed to the source of packet that caused the error, but switched further along label switching path, assuming that when label switching path endpoint will receive ICMP error, it will know how to properly route it back to source.

This causes the situation, that traceroute in MPLS network can not be used the same way as in IP network - to determine failure point in the network. If label switched path is broken anywhere in the middle, no ICMP replies will come back, because they will not make it to the far endpoint of label switching path.

Penultimate hop popping and traceroute source address

Thorough understanding of pen ultimate hop behaviour and routing is necessary to understand and avoid problems that penultimate hop popping causes to traceroute.

In the example setup, regular traceroute from R5 to R1 would yield the following results:

```
[admin@R5] > /tool traceroute 9.9.9.1
     ADDRESS                                  STATUS
   1         0.0.0.0 timeout timeout timeout
   2         2.2.2.2 37ms 4ms 4ms
                      mpls-label=17
   3         9.9.9.1 4ms 2ms 11ms
```

compared to:

```
[admin@R5] > /tool traceroute 9.9.9.1 src-address=9.9.9.5
     ADDRESS                                  STATUS
   1         4.4.4.3 15ms 5ms 5ms
                      mpls-label=17
   2         2.2.2.2 5ms 3ms 6ms
                      mpls-label=17
   3         9.9.9.1 6ms 3ms 3ms
```

The reason why first traceroute does not get response from R3 is that by default traceroute on R5 uses source address 4.4.4.5 for its probes, because it is preferred source for route over which next hop to 9.9.9.1/32 is reachable:

```
[admin@R5] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE             INTERFACE
 ...
 3 ADC  4.4.4.0/24         4.4.4.5                           0                    ether1
 ...
 5 ADo  9.9.9.1/32                         r 4.4.4.3         110                  ether1
 ...
```

When first traceroute probe is transmitted (source: 4.4.4.5, destination 9.9.9.1), R3 drops it and produces ICMP error message (source 4.4.4.3 destination 4.4.4.5) that is switched all the way to R1. R1 then sends ICMP error back - it gets switched along label switching path to 4.4.4.5.

R2 is penultimate hop popping router for network 4.4.4.0/24, because 4.4.4.0/24 is directly connected to R3. Therefore R2 removes last label and sends ICMP error to R3 unlabelled:

```
[admin@R2] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION        INTERFACE            NEXTHOP
 ...
 3 19                                        4.4.4.0/24         ether2               2.2.2.3
 ...
```

R3 drops received IP packet, because it receives packet with its own address as source address. ICMP errors produced by following probes come back correctly, because R3 receives unlabelled packets with source addresses 2.2.2.2 and 9.9.9.1, which are acceptable to route.

Command:

```
[admin@R5] > /tool traceroute 9.9.9.1 src-address=9.9.9.5
 ...
```

produces expected results, because source address of traceroute probes is 9.9.9.5. When ICMP errors are travelling back from R1 to R5, penultimate hop popping for 9.9.9.5/32 network happens at R3, therefore it never gets to route packet with its own address as source address.

Configuring VPLS

Configuring VPLS interfaces

VPLS interface can be considered tunnel interface just like EoIP interface. To achieve transparent ethernet segment forwarding between customer sites the following tunnels need to be established:

- R1-R5 (customer A)
- R1-R4 (customer A)
- R4-R5 (customer A)
- R1-R5 (customer B)

Note that each tunnel setup involves creating VPLS interfaces on both endpoints of tunnel.

Negotiation of VPLS tunnels is done by LDP protocol - both endpoints of tunnel exchange labels they are going to use for tunnel. Data forwarding in tunnel then happens by imposing 2 labels on packets: tunnel label and transport label - label that ensures traffic delivery to the other endpoint of tunnel.

VPLS tunnels are configured in /interface vpls menu. vpls-id parameter identifies VPLS tunnel and must be unique for every tunnel between this and remote peer.

The necessary setup:

- on R1:

```
/interface vpls add name=A1toA2 remote-peer=9.9.9.5 mac-address=00:00:00:00:00:a1 vpls-id=10 disabled=no
/interface vpls add name=A1toA3 remote-peer=9.9.9.4 mac-address=00:00:00:00:00:a1 vpls-id=10 disabled=no
/interface vpls add name=B1toB2 remote-peer=9.9.9.5 mac-address=00:00:00:00:00:b1 vpls-id=11 disabled=no
```

- on R4:

```
/interface vpls add name=A3toA1 remote-peer=9.9.9.1 mac-address=00:00:00:00:00:a3 vpls-id=10 disabled=no
/interface vpls add name=A3toA2 remote-peer=9.9.9.5 mac-address=00:00:00:00:00:a3 vpls-id=10 disabled=no
```

- on R5:

```
/interface vpls add name=A2toA1 remote-peer=9.9.9.1 mac-address=00:00:00:00:00:a2 vpls-id=10 disabled=no
/interface vpls add name=A2toA3 remote-peer=9.9.9.4 mac-address=00:00:00:00:00:a2 vpls-id=10 disabled=no
/interface vpls add name=B2toB1 remote-peer=9.9.9.1 mac-address=00:00:00:00:00:b2 vpls-id=11 disabled=no
```

Configuring VPLS tunnel causes dynamic LDP neighbor to be created and "targeted" LDP session to be established. Targeted LDP session is session that is established between two routers that are not direct neighbors. After this setup R1 LDP neighbors are:

```
[admin@R1] > mpls ldp neighbor pr
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                      SEND-TARGETED ADDRESSES
 0 DO   9.9.9.2         9.9.9.1         9.9.9.2:0                 no            1.1.1.2
```

```
                                                                          2.2.2.2
                                                                          9.9.9.2
 1 DOTV 9.9.9.5         9.9.9.1         9.9.9.5:0                 yes    4.4.4.5
                                                                          5.5.5.5
                                                                          9.9.9.5
 2 DOTV 9.9.9.4         9.9.9.1         9.9.9.4:0                 yes    3.3.3.4
                                                                          5.5.5.4
                                                                          9.9.9.4
```

Note that labels for IP routes are also exchanged between VPLS peers, although there is no chance any of them will be used. For example, without adding additional links, R4 will not become next hop for any route on R1, so labels learned from R4 are not likely to be ever used. Still routers maintain all labels exchanged so that they are ready for use immediately if needed. This default behaviour can be overridden by filtering which is discussed later.

By monitoring state of VPLS interface its related information can be viewed:

```
[admin@R1] /interface vpls> monitor A1toA3 once
    remote-label: 24
     local-label: 27
   remote-status:
      igp-prefix: 9.9.9.4/32
     igp-nexthop: 1.1.1.2
  imposed-labels: 21,24
```

Here we can see that R1 has assigned label 27 for tunnel between R1 and R4. MPLS forwarding table shows that this label is recognized and instead of fowarding to some next hop, received over this tunnel:

```
[admin@R1] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION        INTERFACE          NEXTHOP
 ...
11 27                                        A1toA3
 ...
```

In turn remote endpoint (R4) has assigned label 24.

igp-prefix shows route that is used to get to remote endpoint of tunnel. This implies that when forwarding traffic to remote endpoint of tunnel this router will impose transport label - label distributed by next hop (which is shown as igp-nexthop) to 9.9.9.4/32 for 9.9.9.4/32 route. This can be confirmed on R2:

```
[admin@R2] > /mpls forwarding-table print
 # IN-LABEL             OUT-LABELS           DESTINATION        INTERFACE          NEXTHOP
 ...
 5 21                   18                   9.9.9.4/32         ether2             2.2.2.3
 ...
```

Tunnel label imposed on packets will be as assigned by remote router (R4) for this tunnel.

imposed-labels reflect this setup: packets produced by tunnel will have 2 labels on them: 21 and 24.

Penultimate hop popping effects on VPLS tunnels

Penultimate hop popping of transport label causes packets to arrive at VPLS tunnel endpoint with just one tag - tunnel tag. This makes VPLS tunnel endpoint do just one label lookup to find out what to do with packet. Transport label behaviour can be observed by traceroute tool between tunnel endpoints. For example, traceroute from R1 to R4 looks like this:

```
[admin@R1] > /tool traceroute 9.9.9.4 src-address=9.9.9.1
     ADDRESS                                    STATUS
   1         1.1.1.2 7ms 5ms 3ms
                      mpls-label=21
   2         2.2.2.3 5ms 4ms 18ms
                      mpls-label=18
   3         9.9.9.4 4ms 5ms 3ms
```

traceroute output shows, that endpoint of tunnel is receiving probe without label. The same happens with VPLS tunnel traffic - at R3 transport label (18) is popped and packet is switched with just tunnel label on.

The requirement to deliver packet with tunnel label to endpoint of tunnel explains configuration advice to use "loopback" IP addresses as tunnel endpoints. If in this case R4 was establishing LDP sessions with its address 3.3.3.4, penultimate hop popping would happen not at R3, but at R2, because R3 has network 3.3.3.0/24 as its connected network (and therefore advertises implicit null label). This would cause R3 (and not R4) to receive packet with just tunnel label on, yielding unpredicted results - either dropping frame if R3 does not recognize the packet or forwarding it the wrong way.

Another issue is having VPLS tunnel endpoints directly connected, as in case of R4 and R5. There are no transport labels they can use between themselves, because they both instruct other one to be penultimate hop popping router for their tunnel endpoint address. For example on R5:

```
[admin@R5] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 ...
 3 AD 9.9.9.4/32         5.5.5.4         impl-null  9.9.9.4:0
 ...
```

This causes VPLS tunnel to use only tunnel label when sending packets:

```
[admin@R5] > /int vpls monitor A2toA3 once
    remote-label: 23
     local-label: 27
   remote-status:
      igp-prefix: 9.9.9.4/32
     igp-nexthop: 5.5.5.4
  imposed-labels: 23
```

Bridging ethernet segments with VPLS

VPLS tunnels provide virtual ethernet link between routers. To transparrently connect two physical ethernet segments, they must be bridged with VPLS tunnel. In general it gets done the same way as with EoIP interfaces.

So to transparently bridge customer B networks connected to R1 and R5 the following commands are used on R1:

```
/interface bridge add name=B
/interface bridge port add bridge=B interface=ether1
/interface bridge port add bridge=B interface=B1toB2
```

and on R5:

```
/interface bridge add name=B
/interface bridge port add bridge=B interface=ether3
/interface bridge port add bridge=B interface=B2toB1
```

Note that there is not need to run (R)STP protocol on bridge as there are links between segments B1 and B2 except single VPLS tunnel between R1 and R5.

Split horizon bridging

In the example setup there are 3 tunnels set up to connect segments A1, A2 and A3, establishing so called "full mesh" of tunnels between involved segments. If bridging without (R)STP was enabled, traffic loop would occur. There are a few solutions to this:

- enabling (R)STP to eliminate the loop. This approach has a drawback - (R)STP protocol would disable forwarding through one of tunnels and keep it just for backup purposes. That way traffic between 2 segments would have to go through 2 tunnels, making setup inefficent
- using bridge firewall to make sure that traffic does not get looped - involves firewall rule setup making bridging less efficent
- using bridge horizon feature

The basic idea of split horizon bridging is to make traffic arriving over some port never be sent out some set of ports. For VPLS purposes this would mean never sending packet that arrived over one VPLS tunnel over to other VPLS tunnel, as it is known in advance, that sender of packet has connection to target network itself.

For example, if device in A1 sent packet to broadcast or unknown MAC address (which causes bridges to flood all interfaces), it would get sent to both, R5 and R4 over VPLS tunnels. In regular setup e.g. R5 when receiving such packet over VPLS tunnel would send it in A2 connected to it and also over VPLS tunnel to R4. This way R4 would get 2 copies of the same packet and further cause traffic to loop.

Bridge horizon feature allows to configure bridge ports with horizon setting so that packet received over port with horizon value X is not forwarded or flooded to any port with the same horizon value X. So in case of full mesh of VPLS tunnels, each router must be configured with the same horizon value for VPLS tunnels that are bridged together.

For example, configuration commands for R1 to enable bridging for customer A are:

```
/interface bridge add name=A
/interface bridge port add bridge=A interface=A1toA2 horizon=1
/interface bridge port add bridge=A interface=A1toA3 horizon=1
```

In similar way bridge should be configured on R4 and R5. Note that physical ethernet port is not configured with horizon value. If it was, that would disabled bridge forwarding data at all.

Note that horizon value has meaning only locally - it does not get transmitted over network, therefore it does not matter if the same value is configured in all routers participating in bridged network.

Optimizing label distribution

Label binding filtering

During implementation of given example setup, it has become clear that not all label bindings are necessary. For example, there is no need to exchange IP route label bindings between R1 and R5 or R1 and R4, as there is no chance they will ever be used. Also, if given network core is providing connectivity only for mentioned customer ethernet segments, there is no real use to distribute labels for networks that connect routers between themselves, the only routes that matter are /32 routes to endpoints of VPLS tunnels.

Label binding filtering can be used to distribute only specified sets of labels to reduce resource usage and network load.

There are 2 types of label binding filters:

- which label bindings should be advertised to LDP neighbors, configured in /mpls ldp advertise-filter
- which label bindings should be accepted from LDP neighbors, configured in /mpls ldp accept-filter

Filters are organized in ordered list, specifying prefix that must include the prefix that is tested against filter and neighbor (or wildcard).

In given example setup all routers can be configured so that they advertise labels only for routes that allow to reach endpoints of tunnels. For this 2 advertise filters need to be configured on all routers:

```
/mpls ldp advertise-filter add prefix=9.9.9.0/24 advertise=yes
/mpls ldp advertise-filter add prefix=0.0.0.0/0 advertise=no
```

This filter causes routers to advertise only bindings for routes that are included by 9.9.9.0/24 prefix which covers tunnel endpoints (9.9.9.1/32, 9.9.9.4/32, 9.9.9.5/32). The second rule is necessary because default filter result, when no rule matches is to allow action in question.

In given setup there is no need to set up accept filter, because by convention introduced by 2 abovementioned rules no LDP router will distribute unnecessary bindings.

Note that filter changes do not affect existing mappings, so to take filter into effect, connections between neighbors need to be reset. This can get done by removing them:

```
[admin@R1] /mpls ldp neighbor> print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                      SEND-TARGETED ADDRESSES
 0 DO   9.9.9.2         9.9.9.1         9.9.9.2:0                 no            1.1.1.2
                                                                                2.2.2.2
                                                                                9.9.9.2
 1 DOTV 9.9.9.5         9.9.9.1         9.9.9.5:0                 yes           4.4.4.5
                                                                                5.5.5.5
                                                                                9.9.9.5
 2 DOTV 9.9.9.4         9.9.9.1         9.9.9.4:0                 yes           3.3.3.4
                                                                                5.5.5.4
                                                                                9.9.9.4
[admin@R1] /mpls ldp neighbor> remove [find]
```

So on R1, for example we get:

```
[admin@R1] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 0  D 9.9.9.1/32                         30         9.9.9.5:0
 1  D 9.9.9.5/32                         impl-null  9.9.9.5:0
 2  D 9.9.9.4/32                         31         9.9.9.5:0
 3  D 9.9.9.2/32                         32         9.9.9.5:0
 4  D 9.9.9.3/32                         33         9.9.9.5:0
 5 AD 9.9.9.2/32         1.1.1.2         impl-null  9.9.9.2:0
 6  D 9.9.9.1/32                         24         9.9.9.2:0
 7 AD 9.9.9.3/32         1.1.1.2         25         9.9.9.2:0
 8 AD 9.9.9.4/32         1.1.1.2         26         9.9.9.2:0
 9 AD 9.9.9.5/32         1.1.1.2         27         9.9.9.2:0
10  D 9.9.9.1/32                         27         9.9.9.4:0
11  D 9.9.9.5/32                         28         9.9.9.4:0
12  D 9.9.9.4/32                         impl-null  9.9.9.4:0
13  D 9.9.9.2/32                         29         9.9.9.4:0
14  D 9.9.9.3/32                         30         9.9.9.4:0
```

There still are unnecessary bindings, this time - the bindings distributed due to establishing targeted LDP session with remote endpoints of VPLS tunnels (bindings from 9.9.9.5 and 9.9.9.4)

To filter out those, we configure routers to not distribute any IP bindings to any of tunnel endpoint routers. For example on R1, filter table should look like this:

```
[admin@R1] /mpls ldp advertise-filter> print
Flags: X - disabled
 #   PREFIX             NEIGHBOR        ADVERTISE
 0   0.0.0.0/0          9.9.9.4         no
 1   0.0.0.0/0          9.9.9.5         no
 2   9.9.9.0/24         all             yes
 3   0.0.0.0/0          all             no
```

This causes routers to have minimal label binding tables, for example on R1:

```
[admin@R1] > /mpls local-bindings print
Flags: X - disabled, A - advertised, D - dynamic, L - local-route, G - gateway-route, e - egress
 #      DST-ADDRESS        LABEL      PEERS
 0 ADLe 9.9.9.1/32         impl-null  9.9.9.2:0
 1 ADG  9.9.9.3/32         40         9.9.9.2:0
 2 ADG  9.9.9.4/32         41         9.9.9.2:0
 3 ADG  9.9.9.2/32         42         9.9.9.2:0
 4 ADG  9.9.9.5/32         43         9.9.9.2:0
[admin@R1] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS        NEXTHOP         LABEL      PEER
 0 AD 9.9.9.2/32         1.1.1.2         impl-null  9.9.9.2:0
 1  D 9.9.9.1/32                         24         9.9.9.2:0
 2 AD 9.9.9.3/32         1.1.1.2         25         9.9.9.2:0
 3 AD 9.9.9.4/32         1.1.1.2         26         9.9.9.2:0
```

```
 4 AD 9.9.9.5/32         1.1.1.2         27         9.9.9.2:0
```

Note that IP binding distribution should not be disabled between R4 and R5 although they are tunnel endpoints. Doing so would not harm regular case, because R4 and R5 does not need IP bindings to VPLS tunnel data, but in case link between R3 and R5 would go down, all traffic to R5 from R1 would have to be rerouted through R4. In such case R5 not distributing IP bindings to R4 would cause R4 to not be able to forward MPLS traffic to R5.

Effects of label binding filtering on data forwarding in network

Note the traceroute results after these changes. Traceroute from R1 to R5 using R1 loopback address as source address still behaves the same - each hop reports received labels:

```
[admin@R1] > tool traceroute 9.9.9.5 src-address=9.9.9.1
     ADDRESS                                    STATUS
   1         1.1.1.2 11ms 4ms 5ms
                      mpls-label=27
   2         2.2.2.3 4ms 4ms 4ms
                      mpls-label=25
   3         9.9.9.5 12ms 3ms 3ms
```

But in case of traceroute using R1 address that faces network:

```
[admin@R1] > tool traceroute 9.9.9.5 src-address=1.1.1.1
     ADDRESS                                    STATUS
   1         0.0.0.0 timeout timeout timeout
   2         0.0.0.0 timeout timeout timeout
   3         9.9.9.5 3ms 3ms 3ms
```

Now all hops except the last one do not respond. The reason for this is the fact, that there is no label switching path back from R5 to R1 (which this time uses address 1.1.1.1) because there are no label bindings distributed - so ICMP response is routed and routers on the way back (R3 and R2) receive packets with their own source address and drop them right away without routing.

On the other hand, traceroute from R1 to R5 using their non-loopback addresses:

```
[admin@R1] > tool traceroute 4.4.4.5 src-address=1.1.1.1
     ADDRESS                                    STATUS
   1         1.1.1.2 13ms 1ms 1ms
   2         2.2.2.3 3ms 2ms 2ms
   3         4.4.4.5 3ms 3ms 23ms
```

There is no label switching involved doing this traceroute and it works just like in network without MPLS at all.

Manual:Routing/Multicast

Applies to RouterOS: v3.x, v4.x

Summary

Protocol Independent Multicast - Sparse Mode (PIM-SM or PIM) enables RouterOS to support multicast streaming over network area where routers have PIM set up. Several configured PIM routers together will make multicast cloud where client devices can use IGMP to manage subscriptions to streams. PIM should be used when network topology is complex or stream sources are connected to multicast cloud. Continuous cloud must have configured unique rendezvous point for multicast group or groups used in it and other participants should know how to reach rendezvous point. In simple case when in part of cloud reside only potential clients and no stream sources IGMP proxy can be used instead to conserve resources.

Requirements

Multicast is available on all architectures supported by RouterOS. Packages required:

- *system*
- *multicast*

Note: v3.x *routing-test* and *multicast* packages are incompatible. In case when both are present one of them will be disabled

Note: To get the package you have to download all-packages archive and upload/install multicast package separately on the router

Protocol independent multicast (PIM)

Menu: */routing pim*

General PIM protcol settings.

Property	Desciption
switch-to-spt (*yes|no, default: yes*)	whether to switch to Shortest Path Tree (SPT) phase if multicast data bandwidth threshold is reached. For routers upstream from RP, if this option is disabled, it means that the router will not proceed from protocol phase one (register encapsulation) to native multicast traffic flow. It is recommended to enable this option.
switch-to-spt-bytes (*integer, default: 0*)	multicast data bandwidth threshold. If this threshold is reached in the specified time interval, switching to Shortest Path Tree (SPT) happens. If value 0 is configured, switching will happen immediately.
switch-to-spt-interval (*time, default: 100s*)	time interval in which to account multicast data bandwidth, used in conjunction with **switch-to-spt-bytes** to determine if switching threshold is reached.

Interfaces

Menu: */routing pim interface*

Since RouterOS v4.6 it is possible to specify source address interface will use to participate in multicast cloud. Previously one of interface addresses was chosen without any particular order.

Configuration of interface of the router that will participate in multicast network. Interfaces that are not configured here (or in IGMP-Proxy) will discard multicast packets.

When deploying multicast configuration over wireless links one should be cautious how and what works. For details about multicast and wireless links.

Note: There is no interface count limitation in this menu other than how much hardware can handle

Property	Desciption
alternative-subnets (*IP address/mask Default: nil*) :	if router can receive multicast streams over groups that are not in standard Class-D section then you have to set up this field, so these groups are recognised as multicast groups and will not be discarded.
assert-override-interval (*time Default: 3s*)	time that is subtracted by assert winner from assert-time field, to ensure, that assert winner will always send its assert messages before everyone else.
assert-time (*time Default: 3m*)	Time interval when assert-winner will send out repeated assert.
comment (*text*)	text information for the entry
copy-from (*number*)	use other, already configured entry as stencil for this new one
disabled (*yes | no; Default: no*)	state of the entry
dr-priority (*integer Default: 1*)	if for stream source more than one router with multicast support is available, then one with highest priority will become Designated router of that multicast stream and will handle stream delivery to RP. Higher value means higher priority.
hello-holdtime (*time Default: 1m45s*)	how long consider sender of hello packet received on interface in neighbour list. (usually 3.5 times of hello-period)
hello-period (*time Default: 30s*)	how often hello packet will be sent over this interface.
hello-trigerred-delay (*time Default: 5s*)	when interface starts to participate in multicast cloud then this value is max time interface will wait before sending hello packet. That period of waiting is random from 0 to value set in this field.
igmp-version (*IGMPv1 | IGMPv2 | IGMPv3 Default: IGMPv2*)	what IGMP protocol version to support on the interface.
interface (*interface name*)	interface name that will participate in multicast cloud with these settings.
join-prune-holdtime (*time Default: 3m30s*)	how long save join or prune status before discard it
join-prune-period (*time Default: 1m*)	time interval between sending out join or prune messages
preferred-source-address (*IP address Default: 0.0.0.0*)	address that should be used to send out IGMP/PIM packets. Address used should be already assigned to interface. (introduced in 4.6)
propagation-delay (*integer in milliseconds Default: 50*)	expected propagation delay between PIM routers on this network or link.
protocols (*pim, igmp; Default: pim,igmp*)	what protocols to support on the interface
require-hello (*yes|no Default:yes*)	if sending PIM router have to be neighbour to receiving (this) router to work with those packets.

tracking-support (*yes* | *no Default: yes*)	if propagation-delay is not negotiated or is not set then that value will be suppressed, if one of PIM neighbours has set false in this field, then propagation-delay will be suppressed.
override-interval (*integer in milliseconds Default: 250*)	will override propagation-delay negotiated value if set delay time is smaller than this.

Rendezvous point

Menu: */routing pim rp*

Rendezvous point configuration. Rendezvous point (RP) is a distribution point for multicast group, source provides its data to it, and if there are any subscribers, then RP will provide data to client. Note, that RP will always receive data stream if that exists.

Property	Desciption
comment (text)	add comment to static RP entry
copy-from (number)	creates another RP just like one you pointed to with number you used.
disabled (yes, no)	used to change status of RP entry effectively disabling or enabling it.
group (multicas group address Default: *224.0.0.0/4*)	sets what group this RP will be assigned to. Values accepted are class D ip addresses with mask, thus effectively marking multiple groups to this RP entry e.g. 224.10.10.0/24 will add 256 groups starting with 224.10.10.0 till 224.10.10.255.
hash-mask-length (number 4..32 Default: *30*)	when multicast group have multiple RPs, and they are same scope and same priority, then this value is compared. and so you can load balance this way.
priority (number Default: *192*)	if several RPs are available for multicast group, and they are both with same scope, then RP with highest priority is chosen. Smaller non-negative value is considered of higher priority. Example: priority of 100 is higher than priority of 101.
address (IP address)	at what address you have to look for RP for multicast group specified in group field. If group is set to one of routers interfaces, it should be reachable through whole multicast network, if it not, you will have to set up rules in MRIB (multicast routing information base).

Rendezvous point candidates

Menu: */routing pim rp-candidates*

Rendezvous point candidate configuration, when RP is elected for multicast group

Property	Desciption
comment (text)	additional textual information about entry
copy-from (number)	create this entry using other entry as a stencil
disabled (*yes* | *no Default: no*)	state of entry
group (*multicast group address Default: 224.0.0.0/4*)	routes with will be chosen to be a group RP if no other RP will not participate with higher priority
holdtime (*time Default: 2m30s*)	after what time next election will be initiated
is-scope-zone (*yes|no Default: no*)	if set to yes, scope-zone setting is obeyed, if set to no, then scope-zone just represents range of groups that it will function as RP
priority (*number Default: 192*)	value is used when RP is elected, lower value mean higher priority
interface (interface)	to what interface to bind to if this router is elected as multicast groups RP

Bootstrap router candidates

Menu: */routing pim bsr-candidates*

bootstrap router candidate configuration

Property	Desciption
comment (*text*)	set text describing bsr-candidate list entry
disabled (*yes|no Default: no*)	set state of the list entry
hash-mask-length (*number 4..32 Default: 30*)	to how much first bits of multicast group should be hashed to reduce protocol overhead
is-scope-zone (*yes|no Default: no*)	if set to yes, scope-zone setting is obeyed, if set to no, then scope-zone just represents range of groups that it will function as BSR
priority (*number Default: 1*)	priority of the router in bsr election
scope-zone (*IP address/mask Default: 224.0.0.0/4*)	multicast group range that this router will function as BSR
interface (*interface*)	interface of the router that bsr-candidate will be attached to and if elected BSR

Multicast route information base

Menu: */routing pim mrib*

MRIB routes are used for reverse path forwarding check. In a way, they perform opposite function that FIB (*Forwarding Information Base*) routes: FIB is used to find the right By default, MRIB is populated by FIB routes. Use "multicast" routing filter chanin to control that or set specific parameters for imported FIB routes (e.g. you can change the distance of the route). In addition, you can specify static MRIB routes. This is useful only if you are using multihoming and multicast packet flow will be different from unicast packet flow.

Active MRIB entries that are imported from FIB are shown with "dynamic" flag.

Property	Desciption
width="300px"|**comment** (*text*)	textual note to entry can be added to static entries only.
copy-from (*number*)	use other entry as a template to create new one.
destination (*IP address/mask Default: 0.0.0.0/0*)	hosts that will be reachable through gateway
disabled (*yes|no Default: no*)	status of entry
metric (*integer Default: 1*)	value of cost of the route. Route with least weight will be used if available.
gateway (IP address)	address through where hosts listed in destination field will be reachable.

IGMP group status

Menu: */routing pim igmp-group*

Property	Desciption
interface (*interface*)	group join request received/sent on this interface
group (*IP address*)	IGMP group of interest
source (*IP address*)	source of IGMP request
state (*exclude* | *forward* | *don't forward*)	state of IGMP group membership
version (*IGMPv1*|*IGMPv2*|*IGMPv3*)	version of IGMP protocol used
timeout (*time*)	time when entry will expire if not refreshed

Multicast neighbors

Menu: */routing pim neighbors*

This menu only allows to see information about multicast routers that are reachable within one Ethernet from all interfaces participating in multicast routing. This list is created and updated automatically according to state of multicast network.

Property	Desciption
address (*ip address*)	IP address of neighbour multicast router that router have received hello packet.
interface (*text*)	on what interface hello packet was received
priority (*number 1..255*)	priority of the neighbour router
holdtime (*time*)	how long entry will be held in neighbour list (configured in interface menu *hello-holdtime*)
timeout (*time*)	how much time left when entry will be dropped from list if no hello packets are received. Every time hello packet is received this entry will be refreshed.

Bootstrap router status

Menu: */routing pim bsr*

Property	Desciption
zone-type (*active* | *expiring* | *configured*)	type of the zone
valign="top"|**bsr-address** (*IP address*)	address of BSR router
scope-zone (*IP address/mask*)	multicast group range this router is a BSR
bsr-priority (*integer*)	priority of BSR router
local-address (*IP address*)	local BSR candidate address in scope zone
local-priority (*integer*)	local BSR candidate priority in scope zone
state (*init* | *candidate* | *pending* | *elected* | *no-info* | *accept-any* | *accept-preferred*)	state of BSR router
timeout (*time* | *-1*)	time-out when this entry will be removed • -1 : never expire • time value : time remaining to expiry
sz-timeout (*time* | *-1*)	in what time when sope zone will time out. • -1 : never expire • time value : time remaining to expiry

Multicast forwarding cache status

Menu: */routing pim mfc*

Multicast forwarding cache - this section only provides information about current state of multicast cloud at given router, showing states of joins for multicast groups.

Property	Desciption
group (*IP address*)	name of multicast group
source (*IP address*)	source of multicast group data
rp (*IP address*)	address of RP for that multicast group
incoming-interface (*interface*)	interface that is used to receive multicast data stream
outgoing-interface (*interface*)	interface that is used to transmit multicast data stream

Multicast group joins status

Menu: */routing pim join*

Group join list of all the group joins that are registered by PIM-SM Multicast forwarding cache - this section only provides information about current state of multicast cloud at given router, showing states of joins for multicast groups.

Property	Desciption
group (*IP address*)	multicast group that has at least one registered join request
source (*IP address*)	data provider for that group
rp (*IP address*)	rendezvous point for multicast group
upstream-interface-source (*interface*)	router interface receives data stream of the multicast group
upstream-interface-rp (*interface*)	router interface that is toward the rendezvous point
upstream-mrib-nexthop (*IP address*)	address of the next router towards RP
upstream-pim-nexthop (*IP address*)	address of next router towards RP according to PIM RP tree
join-state (*joined* | *not-joined* | *rpt-not-joined* | *rpt-pruned* | *rpt-not-pruned*)	state of this entry towards RP
join-register-state (*joined* | *pruned* | *join-pending* | *unknown*)	join status on register interface
timeout (*time*)	time-out when entry will be removed from the list.
keepalive-timer (*yes|no*)	how long entry will be kept in the list
i-am-designated-router (*interface list*)	interface name list on which router is chosen as designated router
local-receivers (*interface list*)	interfaces where are clients registered with (*.G) join
joined-rp (*interface list*)	list of interfaces that have clients that originated (*,*,RP) join
joined-wc (*interface list*)	list of interfaces that have clients that originated (*.G) join
joined (*interface list*)	list of interfaces that are in joined state
pruned (*interface list*)	list of interfaces that are in prune state
prune-pending ()	list of interfaces that are in prune-pending state
assert-winner (*interface list*)	list of interfaces that are in assert-winner state
assert-looser (*interface list*)	list of interfaces that are assert-lost state
assert-winner-wc (*interface list*)	list of interfaces that have (*,G) join and have assert-winner state

assert-looser-wc (*interface list*)	list of interfaces that have (*,G) join and have assert-lost state
assert-tracking-wc (*interface list*)	list of interfaces that have (*,G) join and will track assert
could-assert-wc (*interface list*)	list of interfaces that have (*,G) join and could trigger assert
immediate-rp (*interface list*)	list of interfaces that are included in the immediate outgoing interfaces for the corresponding (*,*,RP) entry.
immediate-wc (*interface list*)	list of interfaces that are included in the immediate outgoing interfaces for the corresponding (*,RP) entry.
immediate-sg (*interface list*)	list of interfaces that are included in the immediate outgoing interfaces for the corresponding (S,G) entry.
immediate-sg-rpt (*interface list*)	list of interfaces that are included in the immediate outgoing interfaces for the corresponding (S,G,rpt) entry.
include-wc (*interface list*)	list of interfaces to which traffic might be forwarded because of hosts that are local members on that interface.

Manual:Multicast detailed example

MikroTik is using PIM-SM protocol implementation from XORP [1], integrated in our own routing program.

Since RouterOS 3.16 there is also support for simpler IGMP proxy based multicast routing.

Multicast Routing Overview

IP Multicast is a technology that allows one-to-many and many-to-many distribution of data on the Internet. Senders send their data to a multicast IP destination address, and receives express an interest in receiving traffic destined for such an address. The network then figures out how to get the data from senders to receivers.

If both the sender and receiver for a multicast group are on the same local broadcast subnet, then the routers do not need to be involved in the process, and communication can take place directly. If, however, the sender and receiver are on different subnets, then a multicast routing protocol needs to be involved in setting up multicast forwarding state on the tree between the sender and the receivers.

MikroTik supports PIM-SM multicast routing protocol. PIM means "platform independent multicast" - i.e. this protocol is not tied to any particular unicast routing IGP. SM means "sparse-mode"; as opposed to dense-mode, in sparse-mode protocols explicit control messages are used to ensure that traffic is only delivered to the subnets where there are receivers that requested to receive it.

In addition to the routing protocols used to set up forwarding state between subnets, a way is needed for the routers to discover that there are local receivers on a directly attached subnet. For IPv4 this role is served by the Internet Group Management Protocol (IGMP).

Service Models: ASM vs SSM

There are two different models for IP multicast:

- Any Source Multicast (ASM), in which a receiver joins a multicast group, and receives traffic from any senders that send to that group.
- Source-Specific Multicast (SSM), in which a receiver explicitly joins to a (source, group) pairing.

Multicast Addressing

For IPv4, multicast addresses are in the range 224.0.0.0 to 239.255.255.255 inclusive. Addresses within 232.0.0.0/8 are reserved for SSM usage. Addresses in 239.0.0.0/8 are ASM addresses defined for varying sizes of limited scope. Addresses within 224.0.0.0/24 are considered link-local and are forwarded between subnets. Mostly these addresses are used by applications that do not require communication to other networks. Here are some assigned hostgroup addresses by the internet assigned numbers authority (IANA):

- 224.0.0.1 - All systems on the subnet
- 224.0.0.2 - All routers on the subnet
- 224.0.0.9 - For RIPv2
- 224.0.0.14 - For VRRP
- 224.0.1.1 - Network time protocol (NTP)

The internet assigned numbers authority (IANA) allocates ethernet addresses from 01:00:5E:00:00:00 through 01:00:5E:7F:FF:FF for multicasting, therefore leaving only 23 bits available for the multicast group ID.

IGMP

When a receiver joins a multicast group, the multicast routers serving that receiver's subnet need to know that the receiver has joined so that they can arrange for multicast traffic destined for that group to reach this subnet. The Internet Group Management Protocol (IGMP) is a link-local protocol for IPv4 that communicates this information between receivers and routers. The same role for IPv6 is performed by the Multicast Listener Discovery protocol (MLD).

The basic IGMP mechanism works as follows. When a multicast receiver joins a multicast group it multicasts an IGMP Join message onto the subnet on which it is joining. The local routers receive this join, and cause multicast traffic destined for the group to reach this subnet. Periodically one of the local routers sends a IGMP Query message onto the subnet. If there are multiple multicast routers on the subnet, then one of them is elected as the sole querier for that subnet. In response to an IGMP query, receivers respond by refreshing their IGMP Join. If the join is not refreshed in response to queries, then the state is removed, and multicast traffic for this group ceases to reach this subnet.

There are three different versions of IGMP:

- IGMP version 1 functions as described above.
- IGMP version 2 adds support for IGMP Leave messages to allow fast leave from a multicast group.
- IGMP version 3 adds support for source include and exclude lists, to allow a receiver in indicate that it only wants to hear traffic from certain sources, or not receive traffic from certain sources.

PIM-SM Protocol Overview *(From the PIM-SM specification RFC 4601)*

PIM-SM relies on an underlying topology-gathering protocol to populate a routing table with routes. This routing table is called the MRIB or Multicast Routing Information Base. The routes in this table may be taken directly from the unicast routing table, or it may be different and provided by a separate routing protocol such as Multi-protocol BGP.

Regardless of how it is created, the primary role of the MRIB in the PIM-SM protocol is to provide the next-hop router along a multicast-capable path to each destination subnet. The MRIB is used to determine the next-hop neighbor to which any PIM Join/Prune message is sent. Data flows along the reverse path of the Join messages. Thus, in contrast to the unicast RIB which specifies the next-hop that a data packet would take to get to some subnet, the MRIB gives reverse-path information, and indicates the path that a multicast data packet would take from its origin subnet to the router that has the MRIB.

Phase One: RP Tree

In phase one, a multicast receiver expresses its interest in receiving traffic destined for a multicast group. Typically it does this using IGMP or MLD. One of the receiver's local PIM routers is elected as the Designated Router (DR) for that subnet. On receiving the receiver's expression of interest, the DR then sends a PIM Join message towards the Rendezvous Point (RP) for that multicast group. The RP is a PIM-SM router that has been configured to serve a bootstrapping role for certain multicast groups. This Join message is known as a (*,G) Join because it joins group G for all sources to that group. The (*,G) Join travels hop-by-hop towards the RP for the group, and in each router it passes through, multicast tree state for group G is instantiated. Eventually the (*,G) Join either reaches the RP, or reaches a router that already has (*,G) Join state for that group. When many receivers join the group, their Join messages converge on the RP, and form a distribution tree for group G that is rooted at the RP. This is known as the RP Tree (RPT), and is also known as the shared tree because it is shared by all sources sending to that group. Join messages are resent periodically so long a the receiver remains in the group. When all receivers on a leaf-network leave the group, the DR will send a PIM (*,G) Prune message towards the RP for that multicast group. However if the Prune message is not sent for any reason, the state will eventually time out.

A multicast data sender just starts sending data destined for a multicast group. The sender's local router (DR) takes those data packets, unicast-encapsulates them, and sends them directly to the RP. The RP receives these encapsulated data packets, decapsulates them, and forwards them onto the shared tree. The packets then follow the (*,G) multicast tree state in the routers on the RP Tree, being replicated wherever the RP Tree branches, and eventually reaching all the receivers for that multicast group. The process of encapsulating data packets to the RP is called registering, and the encapsulation packets are known as PIM Register packets. At the end of phase one, multicast traffic is flowing encapsulated to the RP, and then natively over the RP tree to the multicast receivers.

Phase Two: Register-Stop

Register-encapsulation of data packets is inefficient for two reasons:

- Encapsulation and decapsulation may be relatively expensive operations for a router to perform, depending on whether or not the router has appropriate hardware for these tasks.
- Traveling all the way to the RP, and then back down the shared tree may entail the packets traveling a relatively long distance to reach receivers that are close to the sender. For some applications, this increased latency is undesirable.

Although Register-encapsulation may continue indefinitely, for the reasons above, the RP will normally choose to switch to native forwarding. To do this, when the RP receives a register-encapsulated data packet from source S on group G, it will normally initiate an (S,G) source-specific Join towards S. This Join message travels hop-by-hop towards S, instantiating (S,G) multicast tree state in the routers along the path. (S,G) multicast tree state is used only

to forward packets for group G if those packets come from source S. Eventually the Join message reaches S's subnet or a router that already has (S,G) multicast tree state, and then packets from S start to flow following the (S,G) tree state towards the RP. These data packets may also reach routers with (*,G) state along the path towards the RP - if so, they can short-cut onto the RP tree at this point.

While the RP is in the process of joining the source-specific tree for S, the data packets will continue being encapsulated to the RP. When packets from S also start to arrive natively at the the RP, the RP will be receiving two copies of each of these packets. At this point, the RP starts to discard the encapsulated copy of these packets, and it sends a Register-Stop message back to S's DR to prevent the DR unnecessarily encapsulating the packets.

At the end of phase 2, traffic will be flowing natively from S along a source-specific tree to the RP, and from there along the shared tree to the receivers. Where the two trees intersect, traffic may transfer from the source-specific tree to the RP tree, and so avoid taking a long detour via the RP. It should be noted that a sender may start sending before or after a receiver joins the group, and thus phase two may happen before the shared tree to the receiver is built.

Phase 3: Shortest-Path Tree

Although having the RP join back towards the source removes the encapsulation overhead, it does not completely optimize the forwarding paths. For many receivers the route via the RP may involve a significant detour when compared with the shortest path from the source to the receiver. To obtain lower latencies, a router on the receiver's LAN, typically the DR, may optionally initiate a transfer from the shared tree to a source-specific shortest-path tree (SPT). To do this, it issues an (S,G) Join towards S. This instantiates state in the routers along the path to S. Eventually this join either reaches S's subnet, or reaches a router that already has (S,G) state. When this happens, data packets from S start to flow following the (S,G) state until they reach the receiver.

At this point the receiver (or a router upstream of the receiver) will be receiving two copies of the data - one from the SPT and one from the RPT. When the first traffic starts to arrive from the SPT, the DR or upstream router starts to drop the packets for G from S that arrive via the RP tree. In addition, it sends an (S,G) Prune message towards the RP. This is known as an (S,G,rpt) Prune. The Prune message travels hop-by-hop, instantiating state along the path towards the RP indicating that traffic from S for G should NOT be forwarded in this direction. The prune is propagated until it reaches the RP or a router that still needs the traffic from S for other receivers. By now, the receiver will be receiving traffic from S along the shortest-path tree between the receiver and S. In addition, the RP is receiving the traffic from S, but this traffic is no longer reaching the receiver along the RP tree. As far as the receiver is concerned, this is the final distribution tree.

Multi-access Transit LANs

In contrast to PtP interfaces, multi-access LANs present a new problem. Now there can be more than one router that is can connected both to upstream and downstream networks. If all of these routers were forwarding multicast traffic to clients, that would be a waste of bandwidth. To avoid this problem, only one router is elected as traffic forwarder. The election is done by using PIM Assert mesages.

RP Discovery

PIM-SM routers need to know the address of the RP for each group for which they have (*,G) state. This address is obtained either through a bootstrap mechanism or through static configuration. One dynamic way to do this is to use the Bootstrap Router (BSR) mechanism. One router in each PIM-SM domain is elected the Bootstrap Router through a simple election process. All the routers in the domain that are configured to be candidates to be RPs periodically unicast their candidacy to the BSR. From the candidates, the BSR picks an RP-set, and periodically announces this set in a Bootstrap message. Bootstrap messages are flooded hop-by-hop throughout the domain until all routers in the domain know the RP-Set. To map a group to an RP, a router hashes the group address into the RP-set using an order-preserving hash function (one that minimizes changes if the RP-Set changes). The resulting RP is the one that

it uses as the RP for that group.

Multicast and Wireless

Multicast (and broadcast) over wireless works over wireless depending on who is transmitting multicast packet:

If AP transmits multicast packet, packet is transmitted over air with multicast receiver address (and yes - only single copy of packet is transmitted no matter how many clients are registered). As there is no single recipient of packet, it does not get acked - therefore delivery is not reliable (no retransmissions in case somebody does not receive packet). Due to this unreliable delivery, lowest basic rate is used to ensure as reliable delivery as possible. So even if you can send unicast stream between AP and Station using 54Mpbs air rate, multicasts are sent only using 6Mbps air rate (assuming that 6Mbps is lowest basic rate).

If Station transmits multicast packet, it is transmitted over air with unicast AP receiver address (as Stations always transmit all packets to AP, and always using unicast receiver address). Due to this, delivery of multicast packets from Station to AP is reliable and subject to normal rate selection process - max throughput can be used. What complicates the matter - when AP receives multicast packet from Station, it processes it locally, but it also must send it back over the air so that the rest of stations in wireless network see it (think of AP as switch in ethernet, except that is does not have to send multiple copies of packet for all clients to receive). This causes AP to execute multicast transmission procedure as described above, therefore every multicast packet gets transmitted over air twice - first time by Station (using its transmit rate) and second time - by AP using lowest basic rate. This behaviour can be altered by disabling "forwarding" option for particular Station on AP (or for all clients using "default-forwarding" option) - disabling forwarding causes AP not to forward traffic between clients and also disables "sending back" multicasts and broadcasts received from client, because that can be considered special case of forwarding traffic between clients.

If multicast traffic is forwarded across WDS link, it is transmitted over air with unicast receiver address (remote end of WDS link) and therefore is reliable and subject to normal rate selection.

From above we can draw some conclusions, how to increase wireless network throughput:

- Ensure that unicast receiver address is used when transmitting multicast
- Increase lowest basic rate if feasible.

Some ideas on how to apply above conclusions:

- In case multicast traffic is to be delivered over point-to-point link (e.g. some backbone link), you should ensure by some means that multicasts are delivered over wireless using unicast receiver address. This can be achieved by using tunnels (as previously discussed) or by using WDS links (either AP-to-AP WDS or AP-to-StationWDS WDS link). If WDS links are used, care must be taken not to inject multicasts in regular wireless interface (to avoid regular AP multicast transmission procedure).
- In case WDS can not be used, network should be planned so that Station is transmitting multicasts and AP should have forwarding disabled.
- In case multicasts are to be delivered to multiple destinations in wireless network, it is best to organize network such that AP is transmitting multicasts (because AP will transmit just one copy) and increase lowest basic rate, if throughput is not enough.

Note: Starting from v5.15 it is possible to enable **multicast-mode** on the access point. This mode uses unicast receiver MAC address to forward multicast traffic, similar as in case of WDS. On the AP set multicast-helper=enabled and on the station set mode=station-bridge

Example

Almost minimal setup where multicast routing is necessary:

- multicast sender (server);
- multicast receiver (client);
- two routers running PIM between them.

Multicast traffic in this example will be destined to address 224.0.1.20

Traffic flow:

```
 Sender -- (subnet I) --> Router A -- (subnet II) --> Router B -- (subnet III) --> Receiver
```

Router A will be configured as Rendezvous Point.

Enable PIM and IGMP router A:

```
[admin@A] > routing pim interface add
[admin@A] > routing pim interface p
Flags: X - disabled, I - inactive, D - dynamic, R - designated-router,
v1 - IGMPv1, v2 - IGMPv2, v3 - IGMPv3
 #      INTERFACE    PROTOCOLS
 0   v2 all          pim
                     igmp
 1 DRv2 ether3       pim
                     igmp
 2 DR   register     pim
```

Configure static Rendezvous Point:

```
[admin@A] > routing pim rp add address=<IP of router A>
```

You may also need to configure **alternative-subnets** on upstream interface - in case if the multicast sender address is in an IP subnet that is not directly reachable from the local router.

```
[admin@MikroTik] > routing pim interface set <upstream-interface> alternative-subnets=1.2.3.0/24,2.3.4.0/24
```

Enable PIM and IGMP router B:

```
[admin@B] > routing pim interface add interface=ether1
[admin@B] > routing pim interface p
Flags: X - disabled, I - inactive, D - dynamic, R - designated-router,
v1 - IGMPv1, v2 - IGMPv2, v3 - IGMPv3
 #      INTERFACE    PROTOCOLS
 0  Rv2 ether1       pim
                     igmp
 1 DR   register     pim
```

Configure static Rendezvous Point:

```
[admin@B] > routing pim rp add address=<IP of router A>
```

Add route on multicast sender:

```
# ip route add 224.0.1.20/32 via <IP of router A>
```

Start sender and receiver programs. You can either write simple programs yourself, or use any of these:

- MGEN is a program that can be used to send or receive multicast traffic, etc: http://cs.itd.nrl.navy.mil/work/mgen/index.php
- Alternatively, mtest is a bare-bones sender/receiver program available from: http://netweb.usc.edu/pim/pimd/

And hey, it works! Client should receive data now.

Caveats

- Route metric cannot be configured, 0xffff is always used instead (important for PIM asserts). Route distance (from FIB or static MRIB) is used as "metric preference" and can be only in range 0..255.
- Scope zones are not supported.
- It is unclear whether Linux kernel fully supports IGMPv3.

FAQ

Q. Does MT support Source Specific Multicast (SSM)?

A. Yes, SSM is a part of PIM-SM specification and we support it.

Q. Is support for PIM-DM planned?

A. No, as PIM-SM performs good in almost every setup, both sparse and dense.

References

1. XORP User Manual, chapters 13 - 16 [2]
2. Multicast tutorial. Deals with multicast addressing IGMP, PIM-SM / SSM, MSDP and MBGP [3]
3. RFC 2236: Internet Group Management Protocol, Version 2 [4]
4. RFC 3376: Internet Group Management Protocol, Version 3 [5]
5. RFC 4601: Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised) [6]
6. RFC 5059: Bootstrap Router (BSR) Mechanism for PIM [7]

References

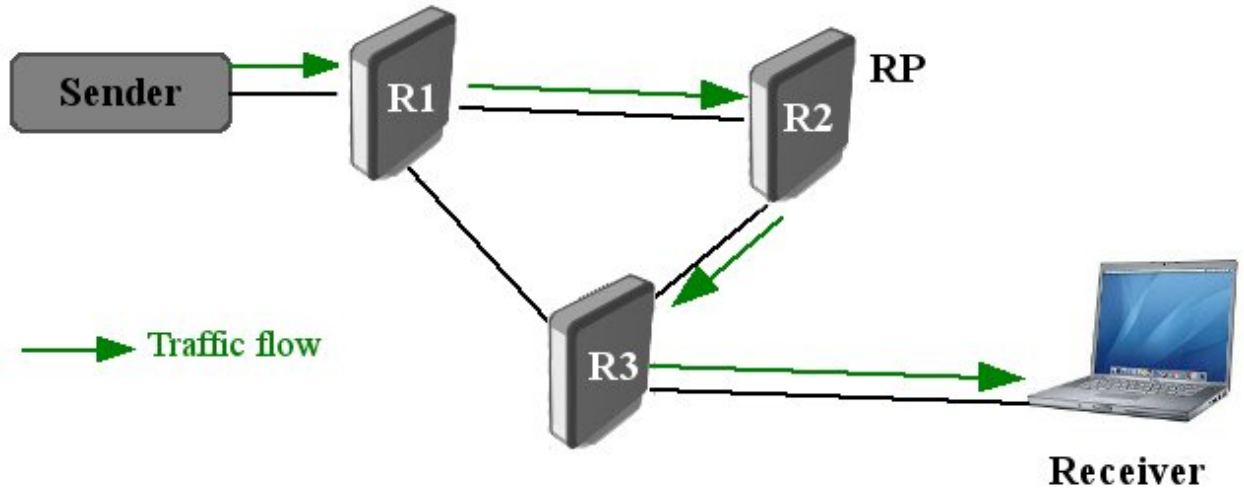
[1] http://www.xorp.org
[2] http://www.xorp.org/releases/current/docs/user_manual/user_manual.pdf
[3] http://www.garr.it/emc_training/tutorials/mcast_tutorial.pdf
[4] http://www.ietf.org/rfc/rfc2236.txt
[5] http://www.ietf.org/rfc/rfc3376.txt
[6] http://www.ietf.org/rfc/rfc4601.txt
[7] http://www.ietf.org/rfc/rfc5059.txt

Manual:Multicast SPT Switchover

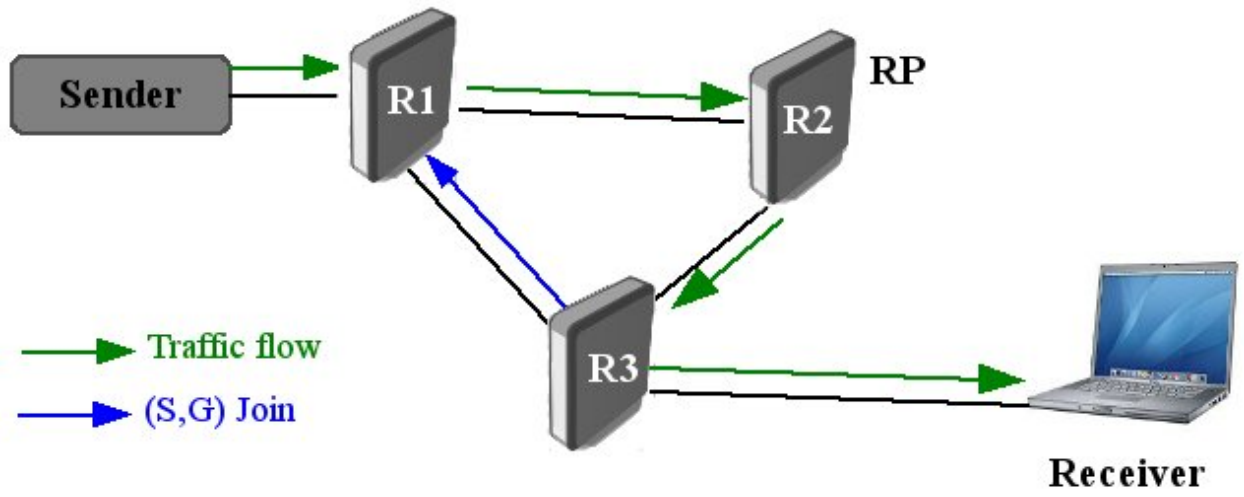
Applies to RouterOS: v3, v4

Overview

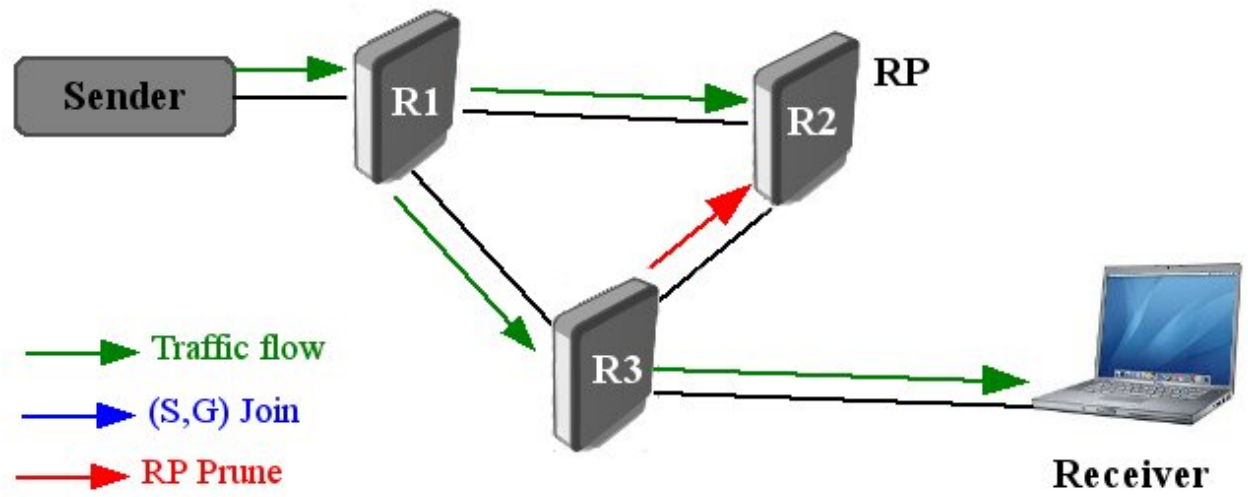
For many receivers the route via RP may not be the best path to source. To obtain lower latencies, it is possible to initiate a transfer from the shared tree to a shortest-path tree (SPT).



From diagram above it is obvious that traffic flow from sender to receiver through RP is not the best path. In this situation shortest-path tree (SPT) switchover comes to help. To initiate switchover, last-hop router sends an (S,G) join towards S and additional (S,G) state is created.



When join reaches S subnet or router that already has (S,G) state, data packets from S start to flow following the (S,G) state until they reach the receiver. Now receiver will be receiving two copies of the data - one from the SPT and one from the RPT. At this point additional (S,G) state is created along shared tree to prune off traffic through RP.



This is known as an (S,G,rpt) Prune. The Prune message travels hop-by-hop, instantiating state along the path towards the RP indicating that traffic from S for G should not be forwarded in this direction. By now, the receiver will be receiving traffic from S along the shortest-path tree between the receiver and S.

Configuration

For this configuration three routers will be used - R1,R2 and R3. Router R2 will be used as RP and multicast traffic will be send to address 224.1.1.10.

Configuration for all three routers are the same and is quite simple.

- Enable switchover option:

```
/routing pim switch-to-stp=yes switch-to-spt-bytes=102400
```

- Add interfaces and RP:

```
/routing pim interface add all
/rputing pim rp add address=<R2_IP_address>
```

There are two main options for SPT threshold - interval and bytes. From those two values bitrate threshold is calculated for switching from RP Tree to SP tree.

- "switch-to-spt-interval" - measurement interval in seconds for measuring bitrate of traffic from multicast sender. Values greater than 10seconds are recommended (default value is 100s).
- "switch-to-spt-bytes" - specifies maximum number of bytes from multicast sender that can be receibed in interval seconds. When threshold is exceeded, router will attempt to switch to SPT. If bytes are set to 0, then switch should happen immediately.

In our example threshold is set to 1024 bytes/s

Testing

To test configuration, you can use VLC to launch multicast stream Download it from [1] Simple streaming how-to: [2]

After the stream is launched you can check all PIM created join states.

```
/routing pim join print
Flags: RP - (*,*,RP), WC - (*,G), SG - (S,G), SG_rpt - (S,G,rpt)
       GROUP            SOURCE           RP
    WC 224.0.0.0        10.55.2.1        10.55.2.1
    SG 224.1.1.10       0.0.0.0          10.55.2.1
       224.1.1.10       10.1.101.225     10.55.2.1
SG_rpt 224.1.1.10       10.1.101.225     10.55.2.1
```

References

[1] http://www.videolan.org/
[2] http://www.videolan.org/doc/streaming-howto/en/ch02.html

Manual:My First IPv6 Network

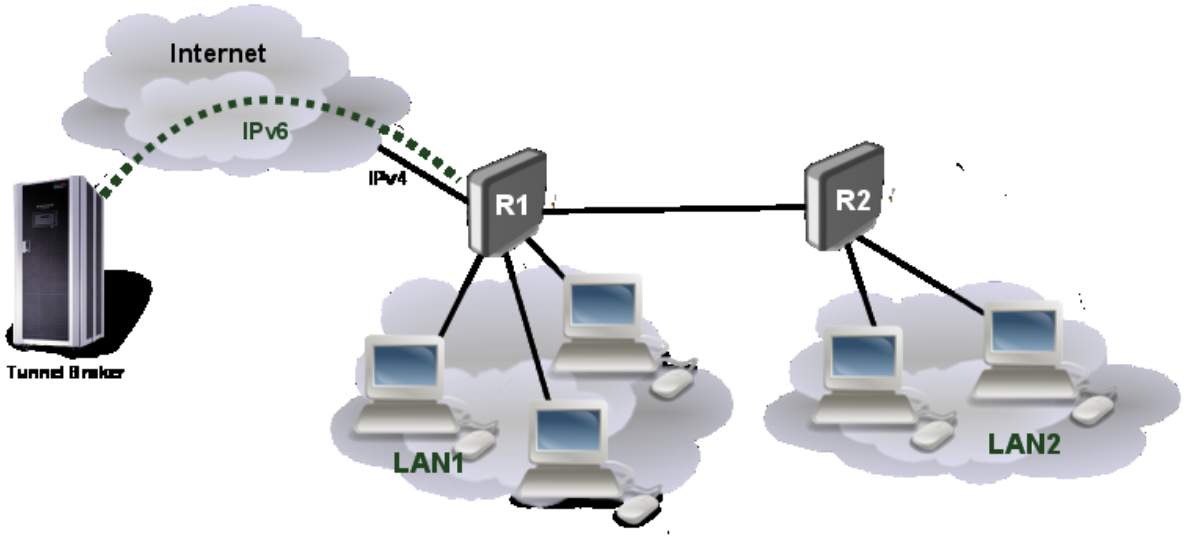
Applies to RouterOS: v3, v4 +

Summary

This example demonstrates how to set up your first IPv6 network using tunnel broker's provided service.

Application Example

Consider following network setup:



Our main gateway (R1) has only IPv4 internet connectivity and ISP is not providing IPv6 services. Our network consists of two isolated network segments Lan1 and Lan2.

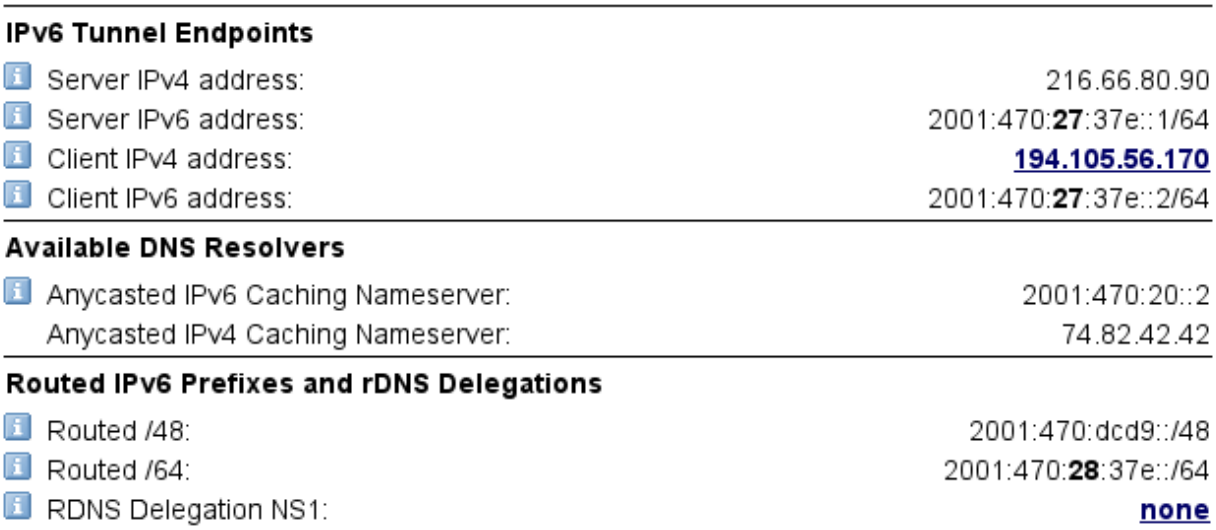
To enable IPv6 we will need to create a tunnel to IPv6 tunnel broker which will transit our IPv6 traffic over IPv4 network.

Tunnel broker

In this example we will use Hurricane Electric tunnel broker services [1].

After registration click on "Create regular tunnel", enter your IP address and choose closest server to your location. That's it tunnel is now allocated.

Now go to tunnel details, where you will see all the parameters for successful tunnel creation and allocated IPv6 address block. As we have two separate lan segments we will need /48 address block, allocate it by clicking on "allocate".



IPv6 Tunnel Endpoints	
Server IPv4 address:	216.66.80.90
Server IPv6 address:	2001:470:**27**:37e::1/64
Client IPv4 address:	**194.105.56.170**
Client IPv6 address:	2001:470:**27**:37e::2/64
Available DNS Resolvers	
Anycasted IPv6 Caching Nameserver:	2001:470:20::2
Anycasted IPv4 Caching Nameserver:	74.82.42.42
Routed IPv6 Prefixes and rDNS Delegations	
Routed /48:	2001:470:dcd9::/48
Routed /64:	2001:470:**28**:37e::/64
RDNS Delegation NS1:	**none**

Configuration

Here is whole configurations for those who want to copy&paste.

R1:

```
# ipv4 connectivity to ISP

/ip address
  add address=194.105.56.170/24 interface=ether1
/ip route
  add gateway=194.105.56.1

# ipv6 service

/interface 6to4
  add comment="HE IPv6" local-address=194.105.56.170 mtu=1280 name=sit1 remote-address=\
    216.66.80.90

/ipv6 address
  add address=2001:470:27:37e::2/64 advertise=no eui-64=no interface=sit1
```

```
/ipv6 route
  add dst-address=::/0 gateway=2001:470:27:37e::1

#Lan1
/ipv6 address
  add address=2001:470:dcd9:1::1/64 advertise=yes interface=ether3

# routing between segments
/routing ospf-v3 instance
  set default router-id=10.10.10.1 distribute-default=if-installed-as-type-1 \
  redistribute-connected=as-type-1

/routing ospf-v3 interface
  add area=backbone interface=ether2
```

R2:

```
#Lan2
/ipv6 address
  add address=2001:470:dcd9:2::1/64 advertise=yes interface=ether3

# routing between segments
/routing ospf-v3 instance
  set default router-id=10.10.10.2 redistribute-connected=as-type-1

/routing ospf-v3 interface
  add area=backbone interface=ether1
```

IPv4 connectivity

IPv4 connectivity is needed only between ISP and our main gateway (R1), as our home network is going to be purely IPv6.

Set up ip address and route on R1:

```
/ip address
  add address=194.105.56.170/24 interface=ether1
/ip route
  add gateway=194.105.56.1
```

IPv6 tunnel service

Lets create 6to4 tunnel using parameters from HE provided tunnel details:

```
/interface 6to4
  add comment="HE IPv6" local-address=194.105.56.170 mtu=1280 name=sit1 remote-address=\
    216.66.80.90
```

Add provided IPv6 address and default route to tunnel broker.

```
/ipv6 address
  add address=2001:470:27:37e::2/64 advertise=no eui-64=no interface=sit1

/ipv6 route
  add dst-address=::/0 gateway=2001:470:27:37e::1
```

At this point router should be capable of reaching any IPv6 destination.

Lan segment address blocks

Next, we need to assign a subnet address from the **/48** address block to two of our ethernet segments. Since the prefix length for IPv6 subnet is always **/64**, we have **65536** subnets available from /48 address block! Let's just assign **2001:470:dcd9:1::/64** to Lan1, and **2001:470:dcd9:2::/64** to Lan2.

R1:

```
#Lan1
/ipv6 address
  add address=2001:470:dcd9:1::1/64 advertise=yes interface=ether3
```

R2:

```
#Lan2
/ipv6 address
  add address=2001:470:dcd9:2::1/64 advertise=yes interface=ether3
```

Notice, that **advertise** flag is enabled. It means that Stateless auto configuration is enabled and absolutely no address configuration is required on client side.

Routing between segments

We will use OSPF as the routing protocol between both routers. Notice that in IPv6 network additional addresses between routers are not required. Link-local addresses are used for connectivity between routers.

R1:

```
/routing ospf-v3 instance
  set default router-id=10.10.10.1 distribute-default=if-installed-as-type-1 \
  redistribute-connected=as-type-1

/routing ospf-v3 interface
  add area=backbone interface=ether2
```

R2:

```
/routing ospf-v3 instance
  set default router-id=10.10.10.2 redistribute-connected=as-type-1
```

```
/routing ospf-v3 interface
  add area=backbone interface=ether1
```

When configuring OSPF on a network without configured IPv4, important configuration part is to set up **router-id**. Wen this parameter is not set, OSPF will try to get it from configured IPv4 addresses, if IPv4 address are missing process will fail and OSPF will not work.

At this point both LAN segments can reach Ipv6 Global network routed over 6to4 tunnel.

See Also

- Simple IPv6 routing example

[Top | Back to Content]

References

[1] http://www.tunnelbroker.net/

Manual:IP/Neighbor discovery

Applies to RouterOS: v5 +

Overview

MikroTik Neighbor Discovery protocol (MNDP) allows to "find" other devices compatible with MNDP or CDP (Cisco Discovery Protocol) in Layer2 broadcast domain.

Neigbors

Sub-menu: `/ip neighbor`

This sub-menu lists all discovered neighbors in Layer-2 broadcast domain. It shows to which interface neighbor is connected, shows it's ip/MAC addresses and several Mikrotik related parameters. List is read-only.

As an example, you can see several RouterBoards and two Cisco routers:

```
[admin@MikroTik] /ip neighbor> print
 # INTERFACE ADDRESS         MAC-ADDRESS       IDENTITY   VERSION    BOARD
 0 ether13   192.168.33.2    00:0C:42:00:38:9F MikroTik   5.99       RB1100AHx2
 1 ether11   1.1.1.4         00:0C:42:40:94:25 test-host  5.8        RB1000
 2 Local     10.0.11.203     00:02:B9:3E:AD:E0 c2611-r1   Cisco I...
 3 Local     10.0.11.47      00:0C:42:84:25:BA 11.47-750  5.7        RB750
 4 Local     10.0.11.254     00:0C:42:70:04:83 tsys-sw1   5.8        RB750G
 5 Local     10.0.11.202     00:17:5A:90:66:08 c7200      Cisco I...
```

Properties

Property	Description
address (*IP*)	Highest IP address configured on a discovered device
address6 (*IPv6*)	IPv6 address configured on a discovered device
age (*time*)	Time interval since last discovery packet
board (*string*)	RouterBoard model. Displayed only to devices with installed RouterOS
identity (*string*)	Configured system identity
interface (*string*)	Interface name to which discovered device is connected
interface-name (*string*)	Interface name on the neighbor device connected to the L2 broadcast domain. Applies to CDP.
ipv6 (*yes* | *no*)	Shows whether device has IPv6 enabled.
mac-address (*MAC*)	Mac address of remote device. Can be used to connect with mac-telnet.
platform (*string*)	Name of the platform. For example "MikroTik", "cisco" ... etc.
software-id (*string*)	RouterOS software ID on a remote device. Applies only to devices installed with RouterOS.
unpack (*none|simple|uncompressed-headers|uncompressed-all*)	Shows discovery packet compression type.
uptime (*time*)	Uptime of remote device. Shown only to devices installed with RouterOS.
version (*string*)	Version number of installed software on a remote device

Discovery configuration

Sub-menu: `/ip neighbor discovery`

In this menu is possible to change state of the interface whether it participates in neighbor discovery or not. If it does, it will send out basic information about system and process received discovery packets broadcasted in Layer-2 network. List of interfaces is automatically managed by RouterOS. Items in the list cannot be removed nor added. Default settings depend on interface type and current state.

Property	Description
comment (*string*; Default:)	Short description of an entry
disabled (*yes* | *no*; Default:)	Whether item is disabled and do not participate in sending/receiving of discovery information. Added in v5.x
discover (*yes* | *no*; Default:)	Whether to participate in sending/receiving of discovery information. Since v5.x left for compatibility with older scripts.

[Top | Back to Content]

Manual:Netinstall

Applies to RouterOS: 2.9, v3, v4

NetInstall Description

NetInstall is a program that runs on Windows computer that allows you to install MikroTiK RouterOS onto a PC or onto a RouterBoard via an Ethernet network.

You can download Netinstall on our download page [1].

NetInstall is also used to re-install RouterOS in cases where the the previous install failed, became damaged or access passwords were lost.

- Your device must support booting from ethernet, and there must be a direct ethernet link from the Netinstall computer to the target device. All RouterBOARDs support PXE network booting, it must be either enabled inside RouterOS "routerboard" menu if RouterOS is operable, or in the bootloader settings. For this you will need a serial cable.

Note: For RouterBOARD devices with no serial port, and no RouterOS access, the reset button can also start PXE booting mode. See your RouterBOARD manual PDF for details. For example RB750 PDF [1]

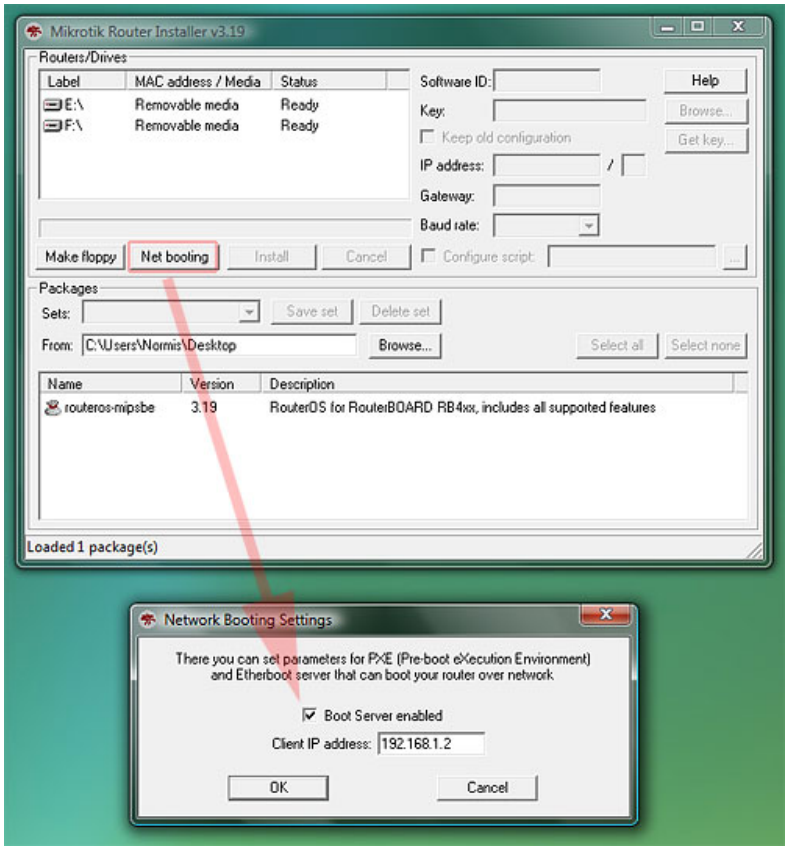
- Netinstall can also directly install RouterOS on a disk (USB/CF/IDE) that is connected to the Netinstall Windows machine. After installation just move the disk to the Router machine and boot from it.

Interface

The following options are available in the Netinstall window:

- **Routers/Drives** - list of PC drives, and in the routers that were detected near the Netinstall PC
- **Make floppy** - used to create a bootable 1.44" floppy disk for PCs which don't have Etherboot support
- **Net booting** - used to enable PXE booting over network (your default choice)
- **Install/Cancel** - after selecting the router and selecting the RouterOS packages below, use this to start install
- **SoftID** - the SoftID that was generated on the router. Use this to purchase your key
- **Key / Browse** - apply the purchased key here, or leave blank to install a 24h trial
- **Get key** - get the key from your mikrotik.com account directly
- **Flashfig** - launch Flashfig - the mass config utility which works on brand new devices
- **Keep old configuration** - keeps the configuration that was on the router, just reinstalls software (no reset)
- **IP address /** "Netmask **- enter IP address and netmask in CIDR notation to preconfigure in the router**
- **Gateway** - default gateway to preconfigure in the router
- **Baud rate** - default serial port baud-rate to preconfigure in the router
- **Configure script** File that contains RouterOS CLI commands that directly configure router (e.g. commands produced by export command). Used to apply default configuration

Screenshot



- for installation over network, don't forget to enable the PXE server, and make sure Netinstall is not blocked by your firewall or antivirus. The connection should be directly from your Windows PC to the Router PC (or RouterBOARD), or at least through a switch/hub.

NetInstall Example

This is a step by step example of how to install RouterOS on a RouterBoard 532 from a typical notebook computer.

Requirements

The Notebook computer must be equiped with the following ports and contain the following files:

- Ethernet port.
- Serial port.
- Serial communications program (such as Hyper Terminal)
- The .npk RouterOS file(s) (not .zip file) of the RouterOS version that you wish to install onto the Routerboard.
- The NetInstall program available from the Downloads page at www.mikrotik.com

Connection process

1. Connect the routerboard to a switch, a hub or directly to the Notebook computer via Ethernet. The notebook computer Ethernet port will need to be configured with a usable IP address and subnet. For example: 10.1.1.10/24
2. Connect the routerboard to the notebook computer via serial, and establish a serial communication session with the RouterBoard. Serial configuration example in in the Serial console manual
3. Run the NetInstall program on your notebook computer.
4. Press the NetInstall "Net Booting" button, **enable the Boot Server**, and enter a valid, usable IP address (within the same subnet of the IP address of the Notebook) that the NetInstall program will assign to the RouterBoard to enable communication with the Notebook computer. For example: 10.1.1.5/24
5. Set the RouterBoard BIOS to boot from the Ethernet interface.

Configuring RouterBOARD

Configuring RouterBOARD without COM port

- To boot RouterBOARD withtout COM port from Network, you can use reset button. Consult RouterBOARD.com and specific RouterBOARD User Guide to find reset button location and usage instructions. For example RB751U-2HnD etherboot instructions,

RouterBOARD 751U-2HnD RouterBOOT reset button (RES, front panel) has two functions to reset RouterOS configuration and boot it from Etherboot: - Connect Netinstall PC to "ether1" port and hold this button during boot time longer, until LED turns off, then release it to make the RouterBOARD look for Netinstall servers.

- As well Etherboot can be configured by RouterOS (when you have access to it),

```
system routerboard settings set boot-device=try-ethernet-once-then-nand
```

Configuring RouterBOARD with COM port

To access Routerboard BIOS configuration: reboot the Routerboard while observing the activity on the Serial Console. You will see the following prompt on the Serial Console “Press any key within 2 seconds to enter setup” indicating that you have a 1 or 2 second window of time when pressing any key will give you access to Routerboard BIOS configuration options.

(press any key when prompted):

You will see the following list of available BIOS Configuration commands. To set up the boot device, press the 'o' key:

```
What do you want to configure?
  d - boot delay
  k - boot key
  s - serial console
  l - debug level
  o - boot device
  b - beep on boot
  v - vga to serial
  t - ata translation
  p - memory settings
  m - memory test
  u - cpu mode
  f - pci back-off
  r - reset configuration
  g - bios upgrade through serial port
```

```
  c - bios license information
  x - exit setup
```

Next Selection: Press the 'e' key to make the RouterBoard to boot from Ethernet interface:

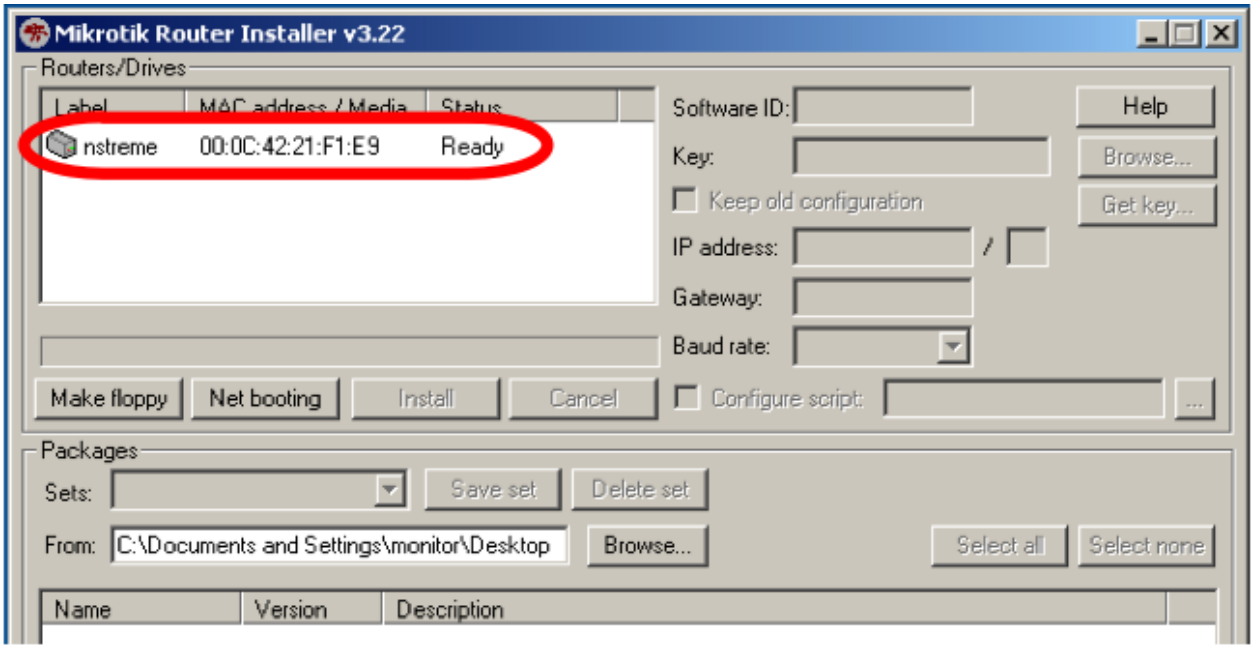
```
Select boot device:
* i - IDE
  e - Etherboot
  1 - Etherboot (timeout 15s), IDE
  2 - Etherboot (timeout  1m), IDE
  3 - Etherboot (timeout  5m), IDE
  4 - Etherboot (timeout 30m), IDE
  5 - IDE, try Etherboot first on next boot (15s)
  6 - IDE, try Etherboot first on next boot (1m)
  7 - IDE, try Etherboot first on next boot (5m)
  8 - IDE, try Etherboot first on next boot (30m)
```

The RouterBoard BIOS will return to the first menu. Press the 'x' key to exit from BIOS. The router will reboot.

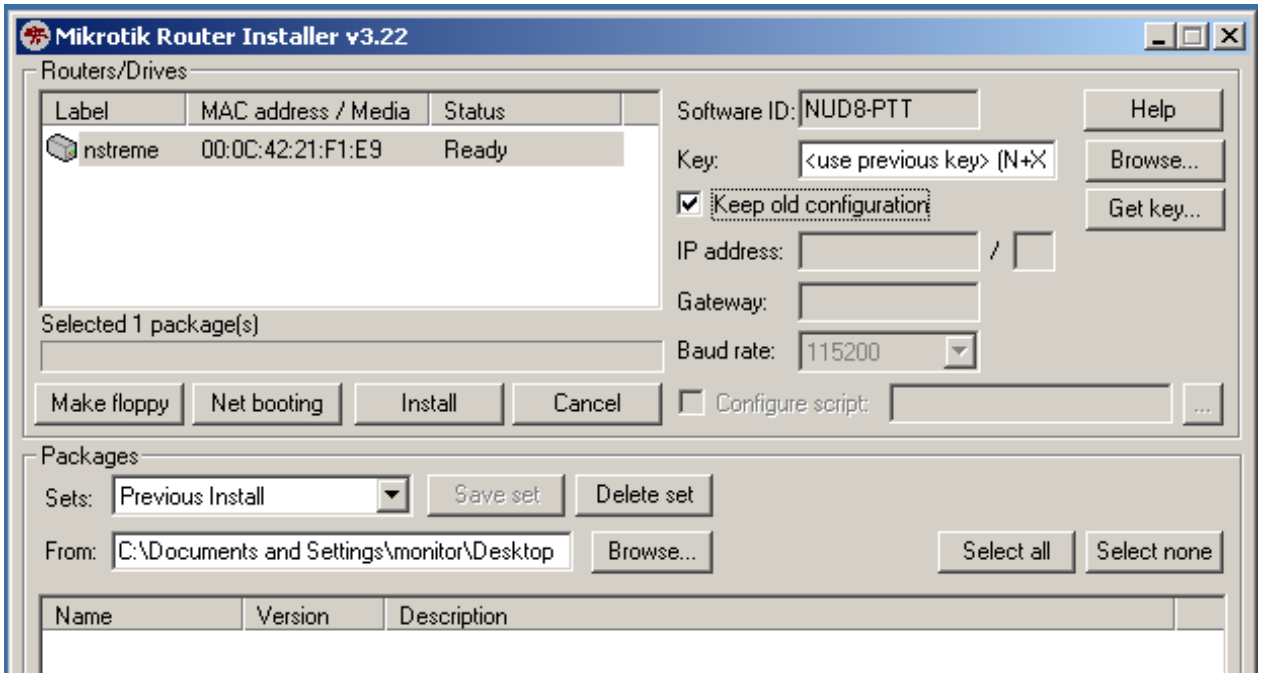
- Make sure boot-protocol is **bootp**.

Installation

Watch the serial console as the RouterBoard reboots, it will indicate that the RouterBoard is attempting to boot to the NetInstall program. The NetInstall program will give the RouterBoard the IP address you entered at Step 4 (above), and the RouterBoard will be ready for software installation. Now you should see the MAC Address of the RouterBoard appear in the Routers/Drives list of the NetInstall program.

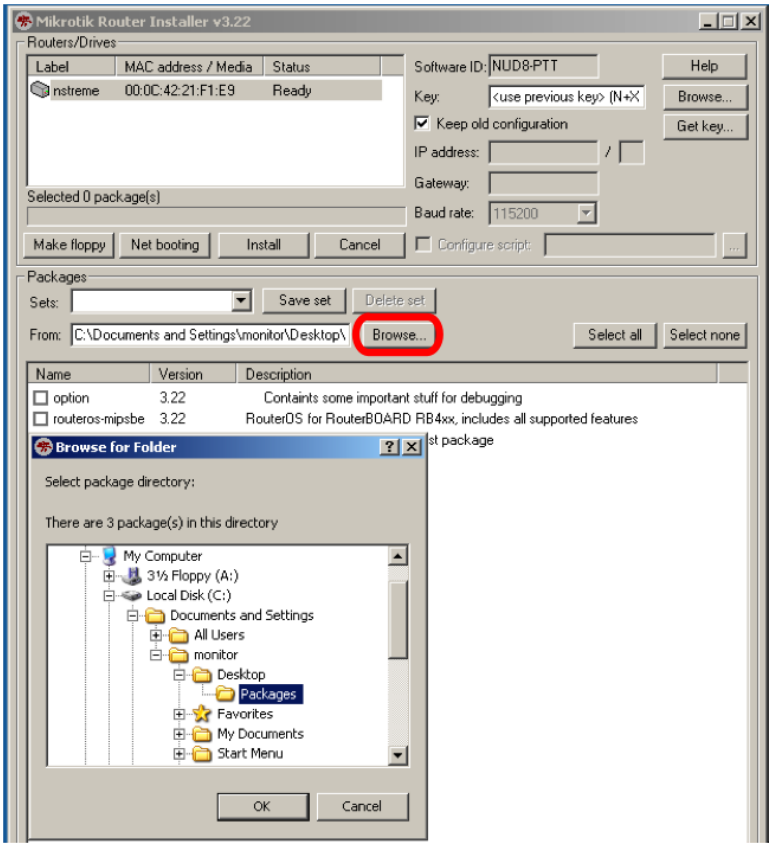


Click on the desired Router/Drive entry and you will be able to configure various installation parameters associated with that Router/Drive entry.

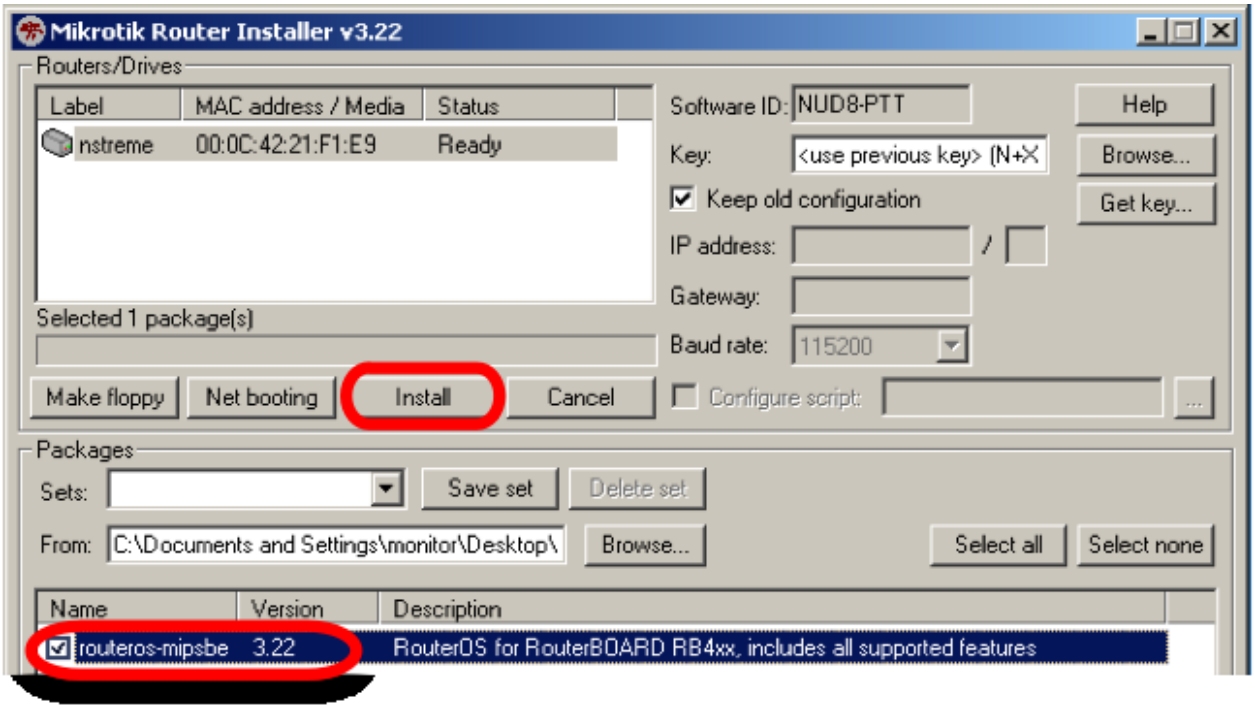


For most Re-Installations of RouterOS on RouterBoards you will only need to set the following parameter:

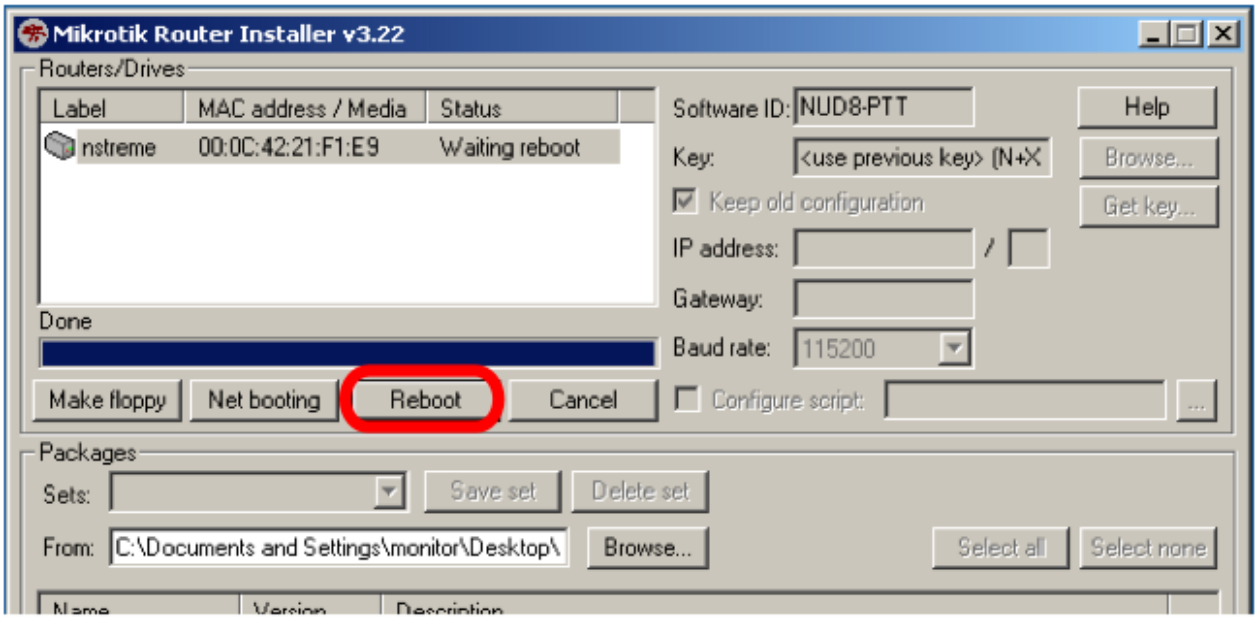
Press the "Browse" button on the NetInstall program screen. Browse to the folder containing the .npk RouterOS file(s) of the RouterOS version that you wish to install onto the Routerboard.



When you have finalized the installation parameters, press the "Install" button to install RouterOS.

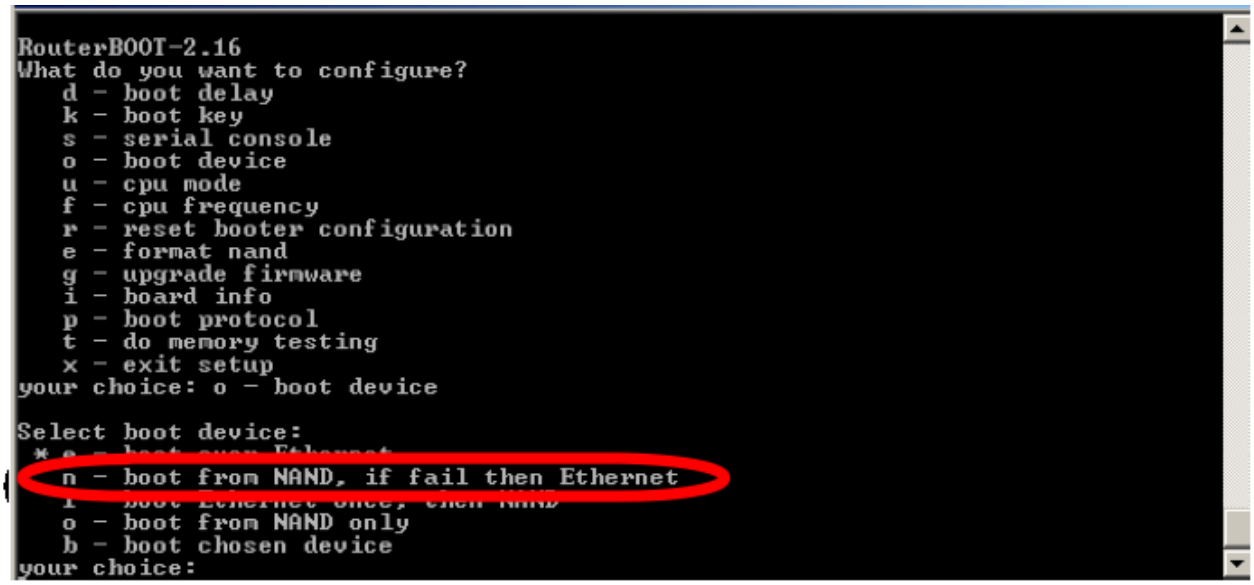


When the installation process has finished, press 'Enter' on the console or 'Reboot' button in the NetInstall program.



Cleanup

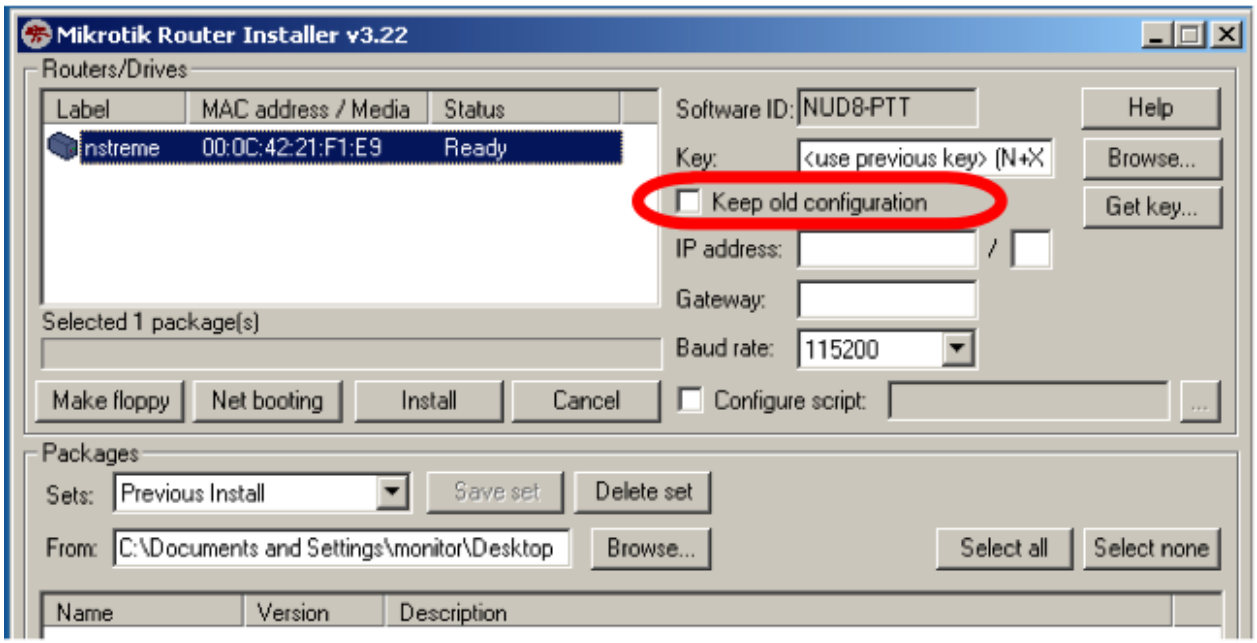
1. Reset the BIOS Configuration of the RouterBoard to boot from its own memory.



2. Reboot the RouterBoard.

Reset RouterOS Password

Netinstall can be used to reset password of RouterOS by erasing all configuration from the router. Uncheck 'Keep Old Configuration' during Netinstall and proceed with standard procedure,



[Top | Back to Content]

References

[1] http://www.routerboard.com/pricelist/download_file.php?file_id=118

Manual:Tools/Netwatch

Applies to RouterOS: v3, v4, v5 +

Summary

Netwatch monitors state of hosts on the network. It does so by sending ICMP pings to the list of specified IP addresses. For each entry in netwatch table you can specify IP address, ping interval and console scripts. The main advantage of netwatch is it's ability to issue arbitrary console commands on host state changes.

Warning: Netwatch executes scripts as *sys user, so any defined global variable in netwatch script will not be readable by scheduler or other users

Properties

Sub-menu: `/tool netwatch`

Property	Description
down-script (*string*; Default:)	Console script that is executed once when state of a host changes to **down**
host (*IP*; Default: **0.0.0.0**)	IP address of the host that should be monitored
interval (*time*; Default: **1m**)	Time interval between pings. Lowering this will make state changes more responsive, but can create unnecessary traffic and consume system resources.
timeout (*time*; Default: **1s**)	Timeout in seconds after which host is considered down
up-script (*string*; Default:)	Console script that is executed once when state of a host changes to **up**

Status

Command `/tool netwatch print` will show current status of netwatch and **read-only** properties listed in table below:

Property	Description
since (*time*)	Indicates when state of the host changed last time
status (*up* | *down* | *unknown*)	Shows the current status of the host

Basic examples

This example will run the scripts gw_1 or gw_2 which change the default gateway depending on the status of one of the gateways:

```
[admin@MikroTik] system script> add name=gw_1 source={/ip route set
{... [/ip route find dst 0.0.0.0] gateway 10.0.0.1}
[admin@MikroTik] system script> add name=gw_2 source={/ip route set
{.. [/ip route find dst 0.0.0.0] gateway 10.0.0.217}
[admin@MikroTik] system script> /tool netwatch
[admin@MikroTik] tool netwatch> add host=10.0.0.217 interval=10s timeout=998ms \
\... up-script=gw_2 down-script=gw_1
[admin@MikroTik] tool netwatch> print
Flags: X - disabled
  #   HOST            TIMEOUT               INTERVAL            STATUS
  0   10.0.0.217      997ms               10s                 up
[admin@MikroTik] tool netwatch> print detail
Flags: X - disabled
  0   host=10.0.0.217 timeout=997ms interval=10s since=feb/27/2003 14:01:03
      status=up up-script=gw_2 down-script=gw_1

[admin@MikroTik] tool netwatch>
```

Without scripts, netwatch can be used just as an information tool to see which links are up, or which specific hosts are running at the moment.

Let's look at the example above - it changes default route if gateway becomes unreachable. How it's done? There are two scripts. The script "gw_2" is executed once when status of host changes to up. In our case, it's equivalent to entering this console command:

```
[admin@MikroTik] > /ip route set [find dst-address="0.0.0.0/0"] gateway=10.0.0.217
```

The **find** command returns list of all routes whose dst-address value is 0.0.0.0/0. Usually, that is the default route. It is substituted as first argument to /ip route set command, which changes gateway of this route to 10.0.0.217

The script "gw_1" is executed once when status of host becomes down. It does the following:

```
[admin@MikroTik] > /ip route set [find dst-address="0.0.0.0/0"] gateway=10.0.0.1
```

It changes the default gateway if 10.0.0.217 address has become unreachable.

Here is another example, that sends e-mail notification whenever the 10.0.0.215 host goes down:

```
[admin@MikroTik] system script> add name=e-down source={/tool e-mail send
{... from="rieks@mt.lv" server="159.148.147.198" body="Router down"
{... subject="Router at second floor is down" to="rieks@latnet.lv"}
[admin@MikroTik] system script> add name=e-up source={/tool e-mail send
{... from="rieks@mt.lv" server="159.148.147.198" body="Router up"
{.. subject="Router at second floor is up" to="rieks@latnet.lv"}
[admin@MikroTik] system script>
[admin@MikroTik] system script> /tool netwatch
[admin@MikroTik] system netwatch> add host=10.0.0.215 timeout=999ms \
\... interval=20s up-script=e-up down-script=e-down
[admin@MikroTik] tool netwatch> print detail
```

```
Flags: X - disabled
  0   host=10.0.0.215 timeout=998ms interval=20s since=feb/27/2003 14:15:36
      status=up up-script=e-up down-script=e-down

[admin@MikroTik] tool netwatch>
```

[Top | Back to Content]

Manual:NTH in RouterOS 3.x

In v3.0 it is a little different implementation of NTH. It has only two parameters 'every' and 'packet'.

How it works in v3.0

Every rule has its own counter. When rule receives packet counter for current rule is increased by one. If counter matches value of 'every' packet will be matched and counter will be set to zero.

If passthrough is not set then packets will be marked as follows:

- first rule nth=2,1 rule will match every first packet of 2, hence, 50% of all the traffic that is matched by the rules
- second rule if passthrough=no will match ONLY 25% of traffic because in 3.0 you need only one rule to catch traffic not like 2.9

Example

Now it is possible to match 50% of all traffic only with one rule:

```
/ip firewall mangle
add action=mark-packet chain=prerouting new-packet-mark=AAA nth=2,1;
```

If more than one rule is needed, then there are two ways to match packets:

- first rule sees all packets and matches 1/3 of all, second rule sees 2/3 of packets and matches 1/2, third rule sees and matches all packets that passed through first two rules (1/3 of all packets).

```
/ip firewall mangle
add action=mark-packet chain=prerouting new-packet-mark=AAA nth=3,1 passthrough=no;
add action=mark-packet chain=prerouting new-packet-mark=BBB nth=2,1 passthrough=no;
add action=mark-packet chain=prerouting new-packet-mark=CCC ;
```

- all rules can see all packets and each rule matches every 3-rd packet.

```
/ip firewall mangle
add action=mark-packet chain=prerouting new-packet-mark=AAA nth=3,1 passthrough=yes;
add action=mark-packet chain=prerouting new-packet-mark=BBB nth=3,2 passthrough=yes;
add action=mark-packet chain=prerouting new-packet-mark=CCC nth=3,3 passthrough=yes;
```

Manual:Nv2

Overview of Nv2 protocol

Nv2 protocol is proprietary wireless protocol developed by MikroTik for use with Atheros 802.11 wireless chips. Nv2 is based on TDMA (Time Division Multiple Access) media access technology instead of CSMA (Carrier Sense Multiple Access) media access technology used in regular 802.11 devices.

TDMA media access technology solves hidden node problem and improves media usage, thus improving throughput and latency, especially in PtMP networks.

Nv2 is supported for Atheros 802.11n chips and legacy 802.11a/b/g chips starting from AR5212, but not supported on older AR5211 and AR5210 chips. This means that both - 11n and legacy devices can participate in the same network and it is not required to upgrade hardware to implement Nv2 in network.

Media access in Nv2 network is controlled by Nv2 Access Point. Nv2 AP divides time in fixed size "periods" which are dynamically divided in downlink (data sent from AP to clients) and uplink (data sent from clients to AP) portions, based on queue state on AP and clients. Uplink time is further divided between connected clients based on their requirements for bandwidth. At the begginning of each period AP broadcasts schedule that tells clients when they should transmit and the amount of time they can use.

In order to allow new clients to connect, Nv2 AP periodically assigns uplink time for "unspecified" client - this time interval is then used by fresh client to initiate registration to AP. Then AP estimates propagation delay between AP and client and starts periodically scheduling uplink time for this client in order to complete registration and receive data from client.

Nv2 implements dynamic rate selection on per-client basis and ARQ for data transmissions. This enables reliable communications across Nv2 links.

For QoS Nv2 implements variable number of priority queues with builtin default QoS scheduler that can be accompanied with fine grained QoS policy based on firewall rules or priority information propagated across network using VLAN priority or MPLS EXP bits.

Nv2 protocol limit is 511 clients.

Nv2 protocol implementation status

As of version 5.0rc1 Nv2 has the following features:

- TDMA media access
- WDS support
- QoS support with variable number or priority queues

As of version 5.0rc2:

- data encryption

As of version 5.0rc3

- RADIUS authentication features

As of version 5.0rc4

- added missing statistics fields

Features that Nv2 **DOES NOT HAVE YET**:

- administrator controlled media access policy
- synchronization between Nv2 APs
- some other features...

Compatibility and coexistence with other wireless protocols

Nv2 protocol is not compatible to or based on any other available wireless protocols or implementations, either TDMA based or any other kind. This implies that **only Nv2 supporting and enabled devices can participate in Nv2 network**.

Regular 802.11 devices will not recognize and will not be able to connect to Nv2 AP. RouterOS devices that have Nv2 support (that is - have RouterOS version 5.0rc1 or higher) will see Nv2 APs when issuing scan command, but will only connect to Nv2 AP if properly configured.

As Nv2 does not use CSMA technology it may disturb any other network in the same channel. In the same way other networks may disturb Nv2 network, because every other signal is considered noise.

The key points regarding compatibility and coexistence:

- only RouterOS devices will be able to participate in Nv2 network
- only RouterOS devices will see Nv2 AP when scanning
- Nv2 network will disturb other networks in the same channel
- Nv2 network may be affected by any (Nv2 or not) other networks in the same channel
- Nv2 enabled device will not connect to any other TDMA based network

How Nv2 compares with Nstreme and 802.11

Nv2 vs 802.11

The key differences between Nv2 and 802.11:

- Media access is scheduled by AP - this eliminates hidden node problem and allows to implement centralized media access policy - AP controls how much time is used by every client and can assign time to clients according to some policy instead of every device contending for media access.
- Reduced propagation delay overhead - There are no per-frame ACKs in Nv2 - this significantly improves throughput, especially on long distance links where data frame and following ACK frame propagation delay significantly reduces the effectiveness of media usage.
- Reduced per frame overhead - Nv2 implements frame aggregation and fragmentation to maximize assigned media usage and reduce per-frame overhead (interframe spaces, preambles).

Nv2 vs Nstreme

The key differences between Nv2 and Nstreme:

- Reduced polling overhead - instead of polling each client, Nv2 AP broadcasts uplink schedule that assigns time to multiple clients, this can be considered "group polling" - no time is wasted for polling each client individually, leaving more time for actual data transmission. This improves throughput, especially in PtMP configurations.
- Reduced propagation delay overhead - Nv2 must not poll each client individually, this allows to create uplink schedule based on estimated distance (propagation delay) to clients such that media usage is most effective. This improves throughput, especially in PtMP configurations.
- More control over latency - reduced overhead, adjustable period size and QoS features allows for more control over latency in the network.

Configuring Nv2

As of version 5.0rc1 new wireless interface setting **wireless-protocol** has been introduced. This setting controls which wireless protocol selects and uses. Note that meaning of this setting depends on interface role (either it is AP or client) that depends on interface **mode** setting. Find possible values of **wireless-protocol** and their meaning in table below.

value	AP	client
unspecified	establish nstreme or 802.11 network based on old **nstreme** setting	connect to nstreme or 802.11 network based on old **nstreme** setting
any	same as **unspecified**	scan for all matching networks, no matter what protocol, connect using protocol of chosen network
802.11	establish 802.11 network	connect to 802.11 networks only
nstreme	establish Nstreme network	connect to Nstreme networks only
Nv2	establish Nv2 network	connect to Nv2 networks only
Nv2-nstreme-802.11	establish Nv2 network	scan for Nv2 networks, if suitable network found - connect, otherwise scan for Nstreme networks, if suitable network found - connect, otherwise scan for 802.11 network and if suitable network found - connect.
Nv2-nstreme	establish Nv2 network	scan for Nv2 networks, if suitable network found - connect, otherwise scan for Nstreme networks and if suitable network found - connect

Note that **wireless-protocol** values **Nv2-nstreme-802.11** and **Nv2-nstreme DO NOT** specify some hybrid or special kind of protocol - these values are implemented to simplify client configuration when protocol of network that client must connect to can change. Using these values can help in migrating network to Nv2 protocol.

Most of Nv2 settings are significant only to Nv2 AP - Nv2 client automatically adapts necessary settings from AP. The following settings are relevant to Nv2 AP:

- **Nv2-queue-count** - specifies how many priority queues are used in Nv2 network. For more details see Manual:Nv2#QoS_in_Nv2_network
- **Nv2-qos** - controls frame to priority queue mapping policy. For more details see Manual:Nv2#QoS_in_Nv2_network
- **Nv2-cell-radius** - specifies distance to farthest client in Nv2 network in km. This setting affects the size of contention time slot that AP allocates for clients to initiate connection and also size of time slots used for estimating distance to client. If this setting is too small, clients that are farther away may have trouble connecting and/or disconnect with "ranging timeout" error. Although during normal operation the effect of this setting should be negligible, in order to maintain maximum performance, it is advised to not increase this setting if not necessary, so AP is not reserving time that is actually never used, but instead allocates it for actual data transfer.
- **tdma-period-size** - specifies size in ms of time periods that Nv2 AP uses for media access scheduling. Smaller period can potentially decrease latency (because AP can assign time for client sooner), but will increase protocol overhead and therefore decrease throughput. On the other hand - increasing period will increase throughput but also increase latency. It may be required to increase this value for especially long links to get acceptable throughput. This necessity can be caused by the fact that there is "propagation gap" between downlink (from AP to clients) and uplink (from clients to AP) data during which no data transfer is happening. This gap is necessary because client must receive last frame from AP - this happens after propagation delay after AP's transmission, and only then client can transmit - as a result frame from client arrives at AP after propagation delay after client's transmission (so the gap is propagation delay times two). The longer the distance, the bigger is necessary propagation gap in every period. If propagation gap takes significant portion of period, actual throughput may become unacceptable and period size should get increased at the expense of increased latency. Basically value of

this setting must be carefully chosen to maximize throughput but also to keep latency at acceptable levels.

The follwing settings are significant on both - Nv2 AP and Nv2 client:

- **Nv2-security** - specifies Nv2 security mode, for more details see Manual:Nv2#Security_in_Nv2_network
- **Nv2-preshared-key** - specifies preshared key to be used, for more details see Manual:Nv2#Security_in_Nv2_network

Migrating to Nv2

Using **wireless-protocol** setting aids in migration or evaluating Nv2 protocol in existing networks really simple and reduce downtime as much as possible. These are the recommended steps:

- upgrade AP to version that supports Nv2, but do not enable Nv2 on AP yet.
- upgrade clients to version that supports Nv2
- configure all clients with **wireless-protocol=Nv2-nstreme-802.11**. Clients will still connect to AP using protocol that was used previously, because AP is not changed over to Nv2 yet
- configure Nv2 related settings on AP
- if it is necessary to use data encryption and secure authentication, configure Nv2 security related settings on AP and clients (refer to Manual:Nv2#Security_in_Nv2_network).
- set **wireless-protocol=Nv2** on AP. This will make AP to change to Nv2 protocol. Clients should now connect using Nv2 protocol.
- in case of some trouble you can easily switch back to previous protocol by simply changing it back to whatever was used before on AP.
- fine tune Nv2 related settings to get acceptable latency and throughput
- implement QoS policy for maximum performance.

The basic troubleshooting guide:

- clients have trouble connecting or disconnect with "ranging timeout" error - check that **Nv2-cell-radius** setting is set appropriately
- unexpectedly low throughput on long distance links although signal and rate is fine - try to increase **tdma-period-size** setting

QoS in Nv2 network

QoS in Nv2 is implemented by means of variable number of priority queues. Queue is considered for transmission based on rule recommended by 802.1D-2004 - only if all higher priority queues are empty. In practice this means that at first all frames from queue with higher priority will be sent, and only then next queue is considered. Therefore QoS policy must be designed with care so that higher priority queues do not make lower priority queues starve.

QoS policy in Nv2 network is controlled by AP, clients adapt policy from AP. On AP QoS policy is configured with **Nv2-queue-count** and **Nv2-qos** parameters. **Nv2-queue-count** parameter specifies number of priority queues used. Mapping of frames to queues is controlled by **Nv2-qos** parameter.

Nv2-qos=default

In this mode outgoing frame at first is inspected by built-in QoS policy algorithm that selects queue based on packet type and size. If built-in rules do not match, queue is selected based on frame priority field, as in **Nv2-qos=frame-priority** mode.

Nv2-qos=frame-priority

In this mode QoS queue is selected based on frame priority field. Note that frame priority field is not some field in headers and therefore it is valid only while packet is processed by given device. Frame priority field must be set either explicitly by firewall rules or implicitly from ingress priority by frame forwarding process, for example, from MPLS EXP bits. For more information on frame priority field see:

- Manual:MPLS/EXP_bit_behaviour
- Manual:WMM

Queue is selected based on frame priority according to 802.1D recommended user priority to traffic class mapping. Mapping depends on number of available queues (**Nv2-queue-count** parameter). For example, if number of queues is 4, mapping is as follows (pay attention how this mapping resembles mapping used by WMM):

- priority 0,1 -> queue 0
- priority 2,3 -> queue 1
- priority 4,5 -> queue 2
- priority 6,7 -> queue 3

If number of queues is 2 (default), mapping is as follows:

- priority 0,1,2,3 -> queue 0
- priority 4,5,6,7 -> queue 1

If number of queues is 8 (maximum possible), mapping is as follows:

- priority 0 -> queue 0
- priority 1 -> queue 1
- priority 2 -> queue 2
- priority 3 -> queue 3
- priority 4 -> queue 4
- priority 5 -> queue 5
- priority 6 -> queue 6
- priority 7 -> queue 7

For other mappings, discussion on rationale for these mappings and recommended practices please see 802.1D-2004.

Security in Nv2 network

Nv2 security implementation has the following features:

- hardware accelerated data encryption using AES-CCM with 128 bit keys;
- 4-way handshake for key management (similar to that of 802.11i);
- preshared key authentication method (similar to that of 802.11i);
- periodically updated group keys (used for broadcast and multicast data).

Being proprietary protocol Nv2 does not use security mechanisms of 802.11, therefore security configuration is different. Interface using Nv2 protocol ignores **security-profile** setting. Instead, security is configured by the following interface settings:

- **Nv2-security** - this setting enables/disables use of security in Nv2 network. Note that when security is enabled on AP, it will not accept clients with disabled security. In the same way clients with enabled security will not connect

to unsecure APs.

- **Nv2-preshared-key** - preshared key to use for authentication. Data encryption keys are derived from preshared key during 4-way handshake. Preshared key must be the same in order for 2 devices to establish connection. If preshared key will differ, connection will time out because remote party will not be able to correctly interpret key exchange messages.

NV2 skin

WebFig skin that has all wireless options removed but ones that has any impact on NV2 configuration. nv2 wireless skin [1]

References

[1] http://www.mikrotik.com/download/nv2.json

Manual:Interface/OVPN

Applies to RouterOS: v5+

Summary

Standards:

Package: `ppp`

Note: RouterOS supports only TCP mode. LZO compression is not supported and username/password authentication is required

OVPN Client

Sub-menu: `/interface ovpn-client`

Properties

Property	Description
`add-default-route` (*yes* | *no*; Default: **no**)	Whether to add OVPN remote address as a default route.
`auth` (*md5* | *none* | *sha1*; Default: **sha1**)	Allowed authentication methods.
`certificate` (*string* | *none*; Default: **none**)	Name of the client certificate imported into certificate list.
`cipher` (*aes128* | *aes192* | *aes256* | *blowfish128* | *none*; Default: **blowfish128**)	Allowed cipher.
`comment` (*string*; Default:)	Descriptive name of an item
`connect-to` (*IP*; Default: **0.0.0.0**)	Remote address of the OVPN server.
`disabled` (*yes* | *no*; Default: **yes**)	Whether interface is disabled or not. By default it is disabled.
`mac-address` (*MAC*; Default:)	Mac address of OVPN interface. Will be auto generated if not specified.

max-mtu (*integer*; Default: **1500**)	Maximum Transmission Unit. Max packet size that OVPN interface will be able to send without packet fragmentation.
mode (*ip* | *ethernet*; Default: **ip**)	Layer3 or layer2 tunnel mode (alternatively tun, tap)
name (*string*; Default:)	Descriptive name of the interface.
password (*string*; Default: "")	Password used for authentication.
port (*integer*; Default: **1194**)	Port to connect to.
profile (*name*; Default: **default**)	Used PPP profile.
user (*string*; Default:)	User name used for authentication.

Quick example

This example demonstrates how to set up OVPN client with username "test", password "123" and server 10.1.101.1

```
[admin@bumba] /interface ovpn-client> add connect-to=10.1.101.1 user=test password=123 disabled=no
[admin@bumba] /interface ovpn-client> print
Flags: X - disabled, R - running
 0    name="ovpn-out1" mac-address=FE:7B:9C:F9:59:D0 max-mtu=1500 connect-to=10.1.101.1
      port=1194 mode=ip user="test" password="123" profile=default certificate=none auth=sha1
      cipher=blowfish128 add-default-route=no
```

OVPN Server

Sub-menu: `/interface ovpn-server`

This sub-menu shows interfaces for each connected OVPN clients.

An interface is created for each tunnel established to the given server. There are two types of interfaces in OVPN server's configuration

- Static interfaces are added administratively if there is a need to reference the particular interface name (in firewall rules or elsewhere) created for the particular user.
- Dynamic interfaces are added to this list automatically whenever a user is connected and its username does not match any existing static entry (or in case the entry is active already, as there can not be two separate tunnel interfaces referenced by the same name).

Dynamic interfaces appear when a user connects and disappear once the user disconnects, so it is impossible to reference the tunnel created for that use in router configuration (for example, in firewall), so if you need a persistent rules for that user, create a static entry for him/her. Otherwise it is safe to use dynamic configuration.

Note: in both cases PPP users must be configured properly - static entries do not replace PPP configuration.

Server configuration

Sub-menu: `/interface ovpn-server server`

Properties:

Property	Description
auth (; Default: **sha1,md5**)	Authentication methods that server will accept.
certificate (*name* | *none*; Default: **none**)	Name of the certificate that OVPN server will use.
cipher (*aes128* | *none*; Default: **aes128,blowfish128**)	
default-profile (*name*; Default: **default**)	
enabled (*yes* | *no*; Default: **no**)	Defines whether OVPN server is enabled or not.
keepalive-timeout (*integer* | *disabled*; Default: **60**)	Defines the time period (in seconds) after which the router is starting to send keepalive packets every second. If no traffic and no keepalive responses has came for that period of time (i.e. 2 * keepalive-timeout), not responding client is proclaimed disconnected
mac-address (*MAC*; Default:)	Auto Generated MAC address of the server.
max-mtu (*integer*; Default: **1500**)	Maximum Transmission Unit. Max packet size that OVPN interface will be able to send without packet fragmentation.
mode (*ip* | *ethernet*; Default: **ip**)	
netmask (*integer*; Default: **24**)	
require-client-certificate (*yes* | *no*; Default: **no**)	If set to yes, then server checks whether client's certificate belongs to the same certificate chain.

```
[admin@bumba] /interface ovpn-server server> set enabled=yes
[admin@bumba] /interface ovpn-server server> set certificate=server
[admin@bumba] /interface ovpn-server server> print
                     enabled: yes
                        port: 1194
                        mode: ip
                     netmask: 24
                 mac-address: FE:A5:57:72:9D:EC
                     max-mtu: 1500
           keepalive-timeout: 60
             default-profile: default
                 certificate: server
  require-client-certificate: no
                        auth: sha1,md5
                      cipher: blowfish128,aes128
```

Warning: It is very important that date on the router is in the range of certificate's date of expiration . To overcome any certificate verification problems, enable **NTP** date synchronization on both server and client.

Monitoring

Monitor command can be used to monitor status of the tunnel on both client and server.

```
[admin@dzeltenais_burkaans] /interface ovpn-server> monitor 0
     status: "connected"
     uptime: 17m47s
  idle-time: 17m47s
       user: "test"
```

```
  caller-id: "10.1.101.18:43886"
        mtu: 1500
```

Read-only properties

Property	Description
status ()	Current status. Value other than "connected" indicates that there are some problems estabising tunnel.
uptime (*time*)	Elapsed time since tunnel was established.
idle-time (*time*)	Elapsed time since last activity on the tunnel.
user (*string*)	Username used to establish the tunnel.
mtu (*integer*)	Negotiated and used MTU
caller-id (*IP:ID*)	

Application Examples

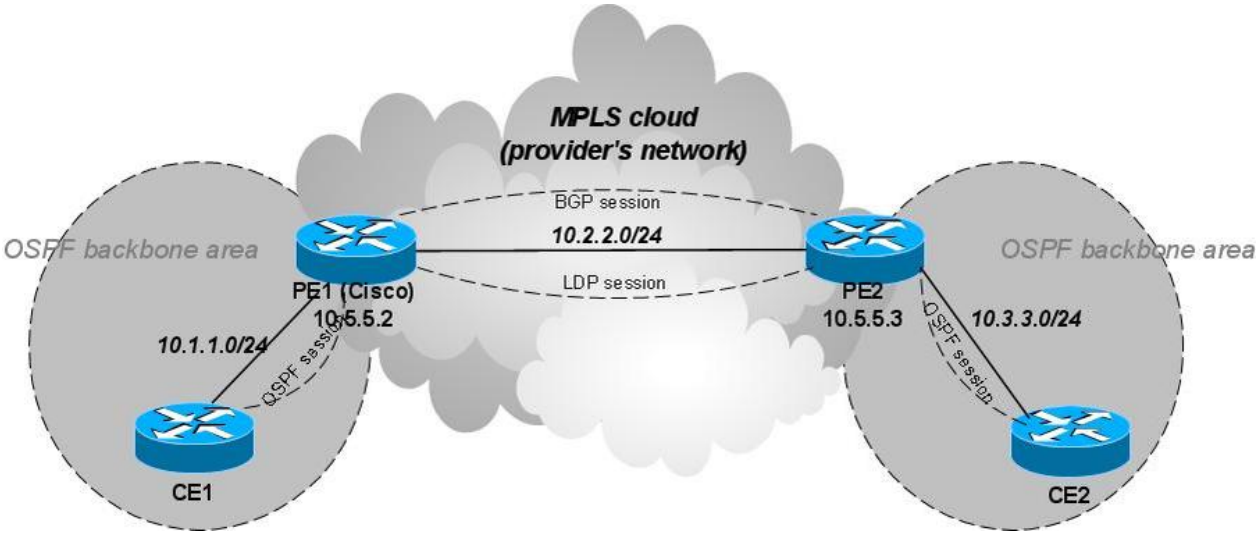
[Top | Back to Content]

Manual:OSPF as PE-CE routing protocol

Software:

- PE1 router is Cisco 7200
- PE2 is MT and has RouterOS 3.23 with **routing-test** and **mpls-test** packages.
- CE1 and CE2 have any RouterOS version.

Configuration with inter-area routing



CE1

```
/ip address add address=10.1.1.1/24 interface=ether1
# static route to redistribute
/ip route add dst-address=10.10.1.0/24 gateway=x.x.x.x
/routing ospf instance set default redistribute-static=as-type-1 router-id=0.0.0.1
/routing ospf network add area=backbone network=1.1.1.0/24
```

CE2

```
/ip address add address=10.3.3.4/24 interface=ether1
# static route to redistribute
/ip route add dst-address=10.10.4.0/24 gateway=y.y.y.y
/routing ospf instance set default redistribute-static=as-type-1 router-id=0.0.0.4
/routing ospf network add area=backbone network=10.3.3.0/24
```

PE1 (Cisco)

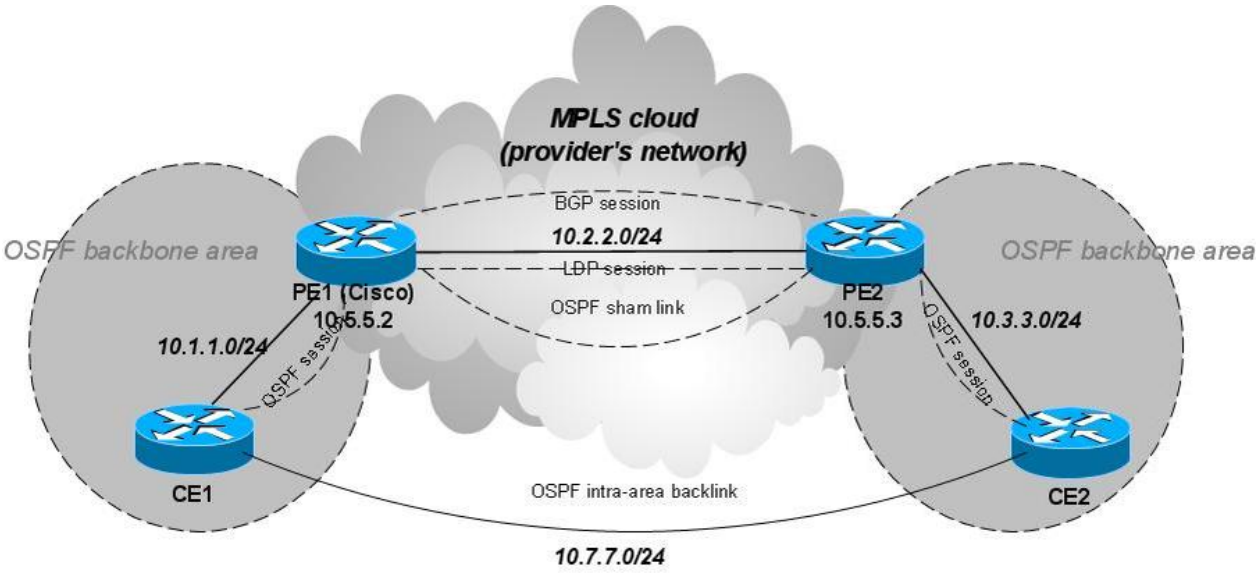
```
ip vrf vrf1
 rd 1.1.1.1:111
 route-target export 1.1.1.1:111
 route-target import 1.1.1.1:111
 exit

interface Loopback0
 ip address 10.5.5.2 255.255.255.255

mpls ldp router-id Loopback0 force
mpls label protocol ldp

interface FastEthernet0/0
 ip vrf forwarding vrf1
 ip address 10.1.1.2 255.255.255.0

interface FastEthernet1/0
 ip address 10.2.2.2 255.255.255.0
 mpls ip

router ospf 1 vrf vrf1
 router-id 10.5.5.2
 network 10.1.1.0 0.0.0.255 area 0
 redistribute bgp 65000 subnets
 domain-id 0.0.0.1
 domain-tag 2222

router bgp 65000
 neighbor 10.5.5.3 remote-as 65000
 neighbor 10.5.5.3 update-source Loopback0
 address-family vpnv4
```

```
  neighbor 10.5.5.3 activate
  neighbor 10.5.5.3 send-community both
  exit-address-family
 address-family ipv4 vrf vrf1
  redistribute connected
  redistribute ospf 1 vrf vrf1 match internal external
  exit-address-family

ip route 10.5.5.3 255.255.255.255 10.2.2.3
```

PE2

```
/interface bridge add name=lobridge


/ip address
 add address=10.2.2.3/24 interface=ether1
 add address=10.3.3.3/24 interface=ether2
 add address=10.5.5.3/32 interface=lobridge


/ip route vrf add disabled=no export-route-targets=1.1.1.1:111 import-route-targets=1.1.1.1:111 \
    interfaces=ether2,vrf-lobridge route-distinguisher=1.1.1.1:111 routing-mark=vrf1
/ip route add dst-address=10.5.5.2/32 gateway=10.2.2.2


/routing bgp instance set default as=65000
/routing bgp instance vrf add instance=default routing-mark=vrf1 redistribute-connected=yes redistribute-ospf=yes
/routing bgp peer add instance=default remote-as=65000 remote-address=10.5.5.2 \
       address-families=vpnv4 update-source=lobridge


/routing ospf instance redistribute-bgp=as-type-1 router-id=10.5.5.3 routing-table=vrf1 \
       domain-id=0.0.0.1 domain-tag=3333
/routing ospf network add area=backbone network=10.3.3.0/24


/mpls ldp set enabled=yes transport-address=10.5.5.3
/mpls ldp interface add interface=ether1
```

Configuration with intra-area routing (including a sham link)



CE1 additional backlink

```
/ip address add address=10.7.7.1/24 interface=backlink
/routing ospf network add area=backbone network=10.7.7.0/24
/routing ospf interface add interface=backlink cost=1000 network-type=point-to-point
```

CE2 additional backlink

```
/ip address add address=10.7.7.4/24 interface=backlink
/routing ospf network add area=backbone network=10.7.7.0/24
/routing ospf interface add interface=backlink cost=1000 network-type=point-to-point
```

PE1 (Cisco) with a sham link

```
interface Loopback1
 ip vrf forwarding vrf1
 ip address 10.6.6.2 255.255.255.255

router ospf 1 vrf vrf1
 area 0 sham-link 10.6.6.2 10.6.6.3 cost 10

ip route 10.6.6.3 255.255.255.255 10.2.2.3

! all the rest of settings remain unchanged
```

PE2 with a sham-link

```
/interface bridge add name=vrf-lobridge

/ip address add address=10.6.6.3/32 interface=vrf-lobridge
# change the VRF to include vrf-lobridge interface
/ip route vrf add disabled=no export-route-targets=1.1.1.1:111 import-route-targets=1.1.1.1:111 \
    interfaces=ether2,vrf-lobridge route-distinguisher=1.1.1.1:111 routing-mark=vrf1
```

```
# configure the sham link
/routing ospf sham-link add area=backbone src-address=10.6.6.3 dst-address=10.6.6.2
# add route to sham link's remote address
/ip route add dst-address=10.6.6.2 gateway=10.2.2.2
```

Manual:OSPF Case Studies

Applies to RouterOS: v3, v4

Summary

This chapter describes the Open Shortest Path First (OSPF) routing protocol support in RouterOS.

OSPF is Interior Gateway Protocol (IGP) and distributes routing information only between routers belonging to the same Autonomous System (AS).

OSPF is based on link-state technology that has several advantages over distance-vector protocols such as RIP:

- no hop count limitations;
- multicast addressing is used to send routing information updates;
- updates are sent only when network topology changes occur;
- logical definition of networks where routers are divided into areas
- transfers and tags external routes injected into AS.

However there are few disadvantages:

- OSPF is quite CPU and memory intensive due to SPF algorithm and maintenance of multiple copies of routing information;
- more complex protocol to implement compared to RIP;

MikroTik RouterOS implements OSPF version 2 (RFC 2328) and version 3 (RFC 5340, OSPF for IPv6).

OSPF Terminology

Term definitions related to OSPF operations.

- **Neighbor** - connected (adjacent) router that is running OSPF with the adjacent interface assigned to the same area. Neighbors are found by Hello packets.
- **Adjacency** - logical connection between router and its corresponding DR and BDR. No routing information is exchanged unless adjacencies are formed.
- **Link** - link refers to a network or router interface assigned to any given network.
- **Interface** - physical interface on the router. Interface is considered as link, when it is added to OSPF. Used to build link database.
- **LSA** - Link State Advertisement, data packet contains link-state and routing information, that is shared among OSPF neighbors.
- **DR** - Designated Router, chosen router to minimize the number of adjacencies formed. Option is used in broadcast networks.

- **BDR** -Backup Designated Router, hot standby for the DR. BDR receives all routing updates from adjacent routers, but it does not flood LSA updates.
- **Area** - areas are used to establish a hierarchical network.
- **ABR** - Area Border Router, router connected to multiple areas.
- **ASBR** - Autonomous System Boundary Router, router connected to an external network (in a different AS).
- **NBMA** - Non-broadcast multi-access, networks allow multi-access but have no broadcast capability (for example X.25, Frame Relay). Additional OSPF neighbor configuration is required for those networks.
- **Broadcast** - Network that allows broadcasting, for example Ethernet.
- **Point-to-point** - Network type eliminates the need for DRs and BDRs
- **Router-ID** - IP address used to identify OSPF router. If the OSPF Router-ID is not configured manually, router uses one of the IP addresses assigned to the router as its Router-ID.
- **Link State** - The term link state refers to the status of a link between two routers. It defines the relationship between a router's interface and its neighboring routers.
- **Cost** - Link-state protocols assign a value to each link called cost. the cost value is depend to speed of media. A cost is associated with the outside of each router interface. This is referred to as interface output cost.
- **Autonomous System** - An autonomous system is a group of routers that use a common routing protocol to exchange routing information.

All of these terms are important for understanding the operation of the OSPF and they are used throughout the article.

OSPF Operation

OSPF is a link-state protocol. Interface of the router is considered an OSPF link and state of all the links are stored in **link-state database**.

Link-state routing protocols are distributing, replicating database that describes the routing topology. Each router in routing domain collects local routing topology and sends this information via **link-state advertisements (LSAs)**. LSAs are flooded to all other routers in routing domain and each router generates **link-state database** from received LSAs. The link-state protocol's flooding algorithm ensures that each router has identical link-state database. Each router is calculating routing table based on this link-state database.

OSPF defines several LSA types:

- **type 1** - (*Router LSA*) Sent by routers within the Area, including the list of directly attached links. Does not cross the ABR or ASBR.
- **type 2** - (*Network LSA*) Generated for every “transit network” within an area. A transit network has at least two directly attached OSPF routers. Ethernet is an example of a Transit Network. A Type 2 LSA lists each of the attached routers that make up the transit network and is generated by the DR.
- **type 3** - (*Summary LSA*) The ABR sends Type 3 Summary LSAs. A Type 3 LSA advertises any networks owned by an area to the rest of the areas in the OSPF AS. By default, OSPF advertises Type 3 LSAs for every subnet defined in the originating area, which can cause flooding problems, so it´s a good idea to use a manual summarization at the ABR.
- **type 4** - (*ASBR-Summary LSA*) It announces the ASBR address, it shows “where” the ASBR is located, announcing it´s address instead of it´s routing table.
- **type 5** - (*External LSA*) Announces the Routes learned through the ASBR. External LSAs are flooded to all areas except Stub areas. These LSAs divides in two types: external type 1 and external type2.
- **type 6** - (*Group Membership LSA*) This was defined for Multicast extensions to OSPF and is not used by ROuterOS.
- **type 7** - type 7 LSAs are used to tell the ABRs about these external routes imorted in NSSA area. Area Border Router then translates these LSAs to type 5 external LSAs and floods as normal to the rest of the OSPF

network

- **type 8** - (*Link-local only LSA for OSPFv3*)
- **type 9** -
- **type 10** -
- **type 11** -

Note: If we do not have any ASBR, there´s no LSA Types 4 and 5 in the network.

Looking at the link-state database each routing domain router knows how many other routers are in the network, how many interfaces routers have, what networks link between router connects, cost of each link and so on.

There are several steps before OSPF network becomes fully functional:

- Neighbor discovery
- Database Synchronization
- Routing calculation

Communication between OSPF routers

OSPF runs directly over the IP network layer using protocol number 89.
Destination IP address is set to neighbor's IP address or to one of the OSPF multicast addresses AllSPFRouters (224.0.0.5) or AllDRRouters (224.0.0.6). Use of these addresses are described later in this article.
Every OSPF packet begins with standard 24-byte header.

Field	Description
`Packet type`	There are several types of OSPF packets: Hello packet, Database Description (DD) packet, Link state request packet, link State Update packet and Link State Acknowledgment packet. All of these packets except Hello packet are used in link-state database synchronization
`Router ID`	one of router's IP addresses unless configured manually
`Area ID`	Allows OSPF router to associate the packet to the proper OSPF area.
`Checksum`	Allows receiving router to determine if packet was damaged in transit.
`Authentication fields`	These fields allow the receiving router to verify that the packet's contents was not modified and that packet really came from OSPF router which Router ID appears in the packet.

There are five different OSPF packet types used to ensure proper LSA flooding over the OSPF network.

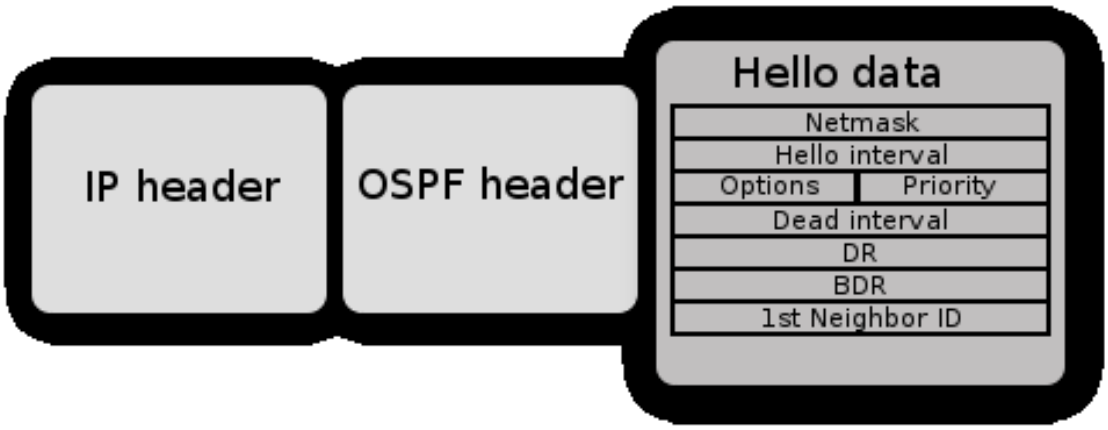
- **Hello packet** - used to discover OSPF neighbors and build adjacencies.
- **Database Description (DD)** - check for Database synchronization between routers. Exchanged after adjacencies are built.

- **Link-State Request (LSR)** - used to request up to date pieces of the neighbor's database. Out of date parts of routes database are determined after DD exchange.
- **Link-State Update (LSU)** - carries a collection of specifically requested link-state records.
- **Link-State Acknowledgment (LSack)** - is used to acknowledge other packet types that way introducing reliable communication.

Neighbor discovery

Neighbors are discovered by periodically sending OSPF Hello packets out of configured interfaces. By default *Hello packets* are sent out with 10 second interval. This interval can be changed by setting hello interval. Router learns the existence of a neighboring router when it receives the neighbor's Hello in return.

The transmission and reception of Hello packets also allows router to detect failure of the neighbor. If Hello packets are not received within Dead interval (which by default is 40s) router starts to route packets around the failure. Hello protocol ensures that the neighboring routers agree on the Hello interval and Dead interval parameters, preventing situations when not in time received Hello packets mistakenly bring the link down.



Field	Description
`network mask`	The IP mask of the originating router's interface IP address.
`hello interval`	period between Hello packets (default 10s)
`options`	OSPF options for neighbor information
`router priority`	an 8-bit value used to aid in the election of the DR and BDR. (Not set in p2p links)
`router dead interval`	time interval has to be received before consider the neighbor is down. (By default four times bigger than Hello interval)
`DR`	the router-id of the current DR
`BDR`	the router-id of the current BDR
`Neighbor router IDs`	a list of router-ids for all the originating router's neighbors

On each type of network segment Hello protocol works a little different. It is clear that on point-to-point segments only one neighbor is possible and no additional actions are required. However if more than one neighbor can be on the segment additional actions are taken to make OSPF functionality even more efficient.

Note: Network mask, Priority, DR and BDR fields are used only when the neighbors are connected by a broadcast or NBMA network segment.

Two routers do not become neighbors unless the following conditions are met.

- Two way communication between routers is possible. Determined by flooding Hello packets.
- Interface should belong to the same area;

- Interface should belong to the same subnet and have the same network mask, unless it has network-type configured as **point-to-point**;
- Routers should have the same authentication options, and have to exchange same password (if any);
- Hello and Dead intervals should be the same in Hello packets;
- External routing and NSSA flags should be the same in Hello packets.

Discovery on Broadcast Subnets

Attached node to the broadcast subnet can send single packet and that packet is received by all other attached nodes. This is very useful for auto-configuration and information replication. Another useful capability in broadcast subnets is multicast. This capability allows to send single packet which will be received by nodes configured to receive multicast packet. OSPF is using this capability to find OSPF neighbors and detect bidirectional connectivity.

Consider Ethernet network illustrated in image below.

Each OSPF router joins the IP multicast group **AllSPFRouters** (224.0.0.5), then router periodically multicasts its Hello packets to the IP address 224.0.0.5. All other routers that joined the same group will receive multicasted Hello packet. In that way OSPF routers maintain relationships with all other OSPF routers by sending single packet instead of sending separate packet to each neighbor on the segment.

This approach has several advantages:

- Automatic neighbor discovery by multicasting or broadcasting Hello packets.
- Less bandwidth usage compared to other subnet types. On broadcast segment there are n*(n-1)/2 neighbor relations, but those relations are maintained by sending only n Hellos.
- If broadcast has multicast capability, then OSPF operates without disturbing non-OSPF nodes on the broadcast segment. If multicast capability is not supported all routers will receive broadcasted Hello packet even if node is not OSPF router.

Discovery on NBMA Subnets

Nonbroadcast multiaccess (NBMA) segments similar to broadcast supports more than two routers, only difference is that NBMA do not support data-link broadcast capability. Due to this limitation OSPF neighbors must be discovered initially through configuration. On RouterOS NBMA configuration is possible in `/routig ospf nbma-neighbor` menu. To reduce the amount of Hello traffic, most routers attached to NBMA subnet should be assigned Router Priority of 0 (set by default in RouterOS). Routers that are eligible to become Designated Routers should have priority values other than 0. It ensures that during election of DR and BDR Hellos are sent only to eligible routers.

Discovery on PTMP Subnets

Point-to-MultiPoint treats the network as a collection of point-to-point links.

On PTMP subnets Hello protocol is used only to detect active OSPF neighbors and to detect bidirectional communication between neighbors. Routers on PTMP subnets send Hello packets to all other routers that are directly connected to them. Designated Routers and Backup Designated routers are not elected on Point-to-multipoint subnets.

Database Synchronization

Link-state Database synchronization between OSPF routers are very important.
There are two types of database synchronizations:

- initial database synchronization

- reliable flooding.

When the connection between two neighbors first come up, *initial database synchronization* will happen. Unsynchronized databases may lead to calculation of incorrect routing table, resulting in routing loops or black holes.

OSPF is using explicit database download when neighbor connections first come up. This procedure is called **Database exchange**. Instead of sending the entire database, OSPF router sends only its LSA headers in a sequence of OSPF **Database Description (DD)** packets. Router will send next DD packet only when previous packet is acknowledged. When entire sequence of DD packets has been received, router knows which LSAs it does not have and which LSAs are more recent. The router then sends **Link-State Request (LSR)** packets requesting desired LSAs, and the neighbor responds by flooding LSAs in **Link-State Update (LSU)** packets. After all updates are received neighbors are said to be **fully adjacent**.

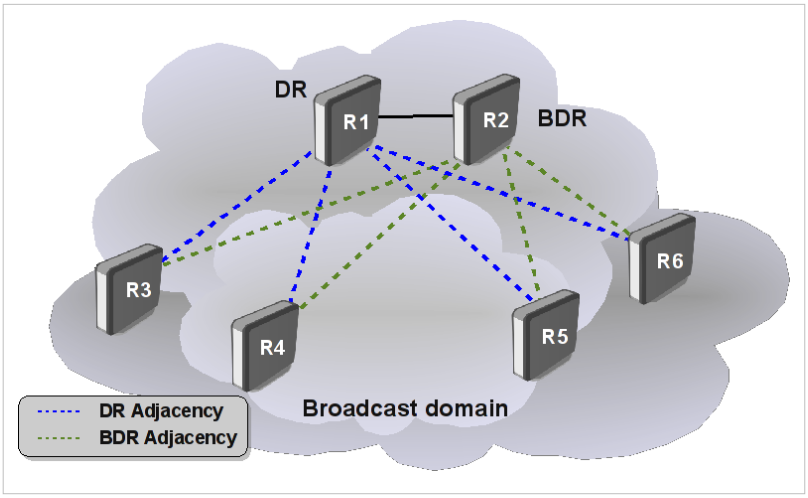
Reliable flooding is another database synchronization method. It is used when adjacencies are already established and OSPF router wants to inform other routers about LSA changes. When OSPF router receives such Link State Update, it installs new LSA in link-state database, sends an acknowledgement packet back to sender, repackages LSA in new LSU and sends it out all interfaces except the one that received the LSA in the first place.

OSPF determines if LSAs are up to date by comparing sequence numbers. Sequence numbers start with 0×80000001, the larger the number, the more recent the LSA is. Sequence number is incremented each time the record is flooded and neighbor receiving update resets Maximum age timer. LSAs are refreshed every 30 minutes, but without a refresh LSA remains in the database for maximum age of 60 minutes.

Databases are not always synchronized between all OSPF neighbors, OSPF decides whether databases needs to be synchronized depending on network segment, for example, on point-to-point links databases are always synchronized between routers, but on ethernet networks databases are synchronized between certain neighbor pairs.

Synchronization on Broadcast Subnets

On broadcast segment there are n*(n-1)/2 neighbor relations, it will be huge amount of Link State Updates and Acknowledgements sent over the subnet if OSPF router will try to synchronize with each OSPF router on the subnet.



This problem is solved by electing one **Designated Router** and one **Backup Designated Router** for each broadcast subnet. All other routers are synchronizing and forming adjacencies only with those two elected routers. This approach reduces amount of adjacencies from n*(n-1)/2 to only 2n-1.

Image on the right illustrates adjacency formations on broadcast subnets. Routers R1 and R2 are Designated Router and Backup Designated router respectively. For example, R3 wants to flood Link State Update (LSU) to both R1 and R2, router sends LSU to IP multicast address **AllDRouters** (224.0.0.6) and only DR and BDR listens to this multicast address. Then Designated Router sends LSU addressed to AllSPFRouters, updating the rest of the routers.

DR election

DR and BDR routers are elected from data received in Hello packet. The first OSPF router on a subnet is always elected as Designated Router, when second router is added it becomes Backup Designated Router. When existing DR or BDR fails new DR or BDR is elected taking into account configured router priority. Router with the highest priority becomes the new DR or BDR.

Being Designated Router or Backup Designated Router consumes additional resources. If Router Priority is set to 0, then router is not participating in the election process. This is very useful if certain slower routers are not capable of being DR or BDR.

Synchronization on NBMA Subnets

Database synchronization on NBMA networks are similar as on broadcast networks. DR and BDR are elected, databases initially are exchanged only with DR and BDR routers and flooding always goes through the DR. The only difference is that Link State Updates must be replicated and sent to each adjacent router separately.

Synchronization on PTMP Subnets

On PTMP subnets OSPF router becomes adjacent to all other routes with which it can communicate directly.

Routing table calculation

When link-state databases are synchronized OSPF routers are able to calculate routing table.
Link state database describes the routers and links that interconnect them and are appropriate for forwarding. It also contains the cost (metric) of each link. This metric is used to calculate shortest path to destination network.
Each router can advertise a different cost for the router's own link direction, making it possible to have asymmetric links (packets to destination travels over one path, but response travels different path). Asymmetric paths are not very popular, because it makes harder to find routing problems.
The Cost in RouterOS is set to 10 on all interfaces by default. Value can be changed in ospf interface configuration menu, for example to add ether2 interface with cost of 100:

```
/routing ospf interface add interface=ether2 cost=100
```

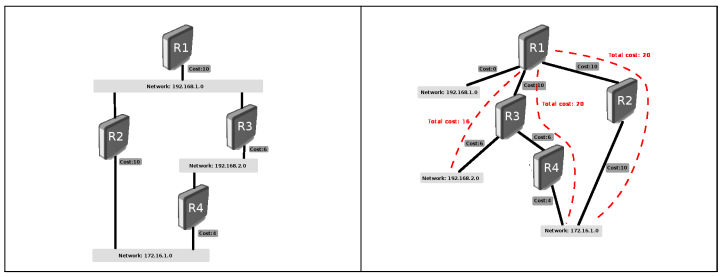
The cost of an interface on Cisco routers is inversely proportional to the bandwidth of that interface. Higher bandwidth indicates lower cost. If similar costs are necessary on RouterOS, then use following formula:

```
Cost = 100000000/bw in bps.
```

OSPF router is using Dijkstra's Shortest Path First (SPF) algorithm to calculate shortest path. The algorithm places router at the root of a tree and calculates shortest path to each destination based on the cumulative cost required to reach the destination. Each router calculates own tree even though all routers are using the same link-state database.

SPT calculation

Assume we have the following network. Network consists of 4(four) routers. OSPF costs for outgoing interfaces are shown near the line that represents the link. In order to build shortest path tree for router R1, we need to make R1 the root and calculate the smallest cost for each destination.



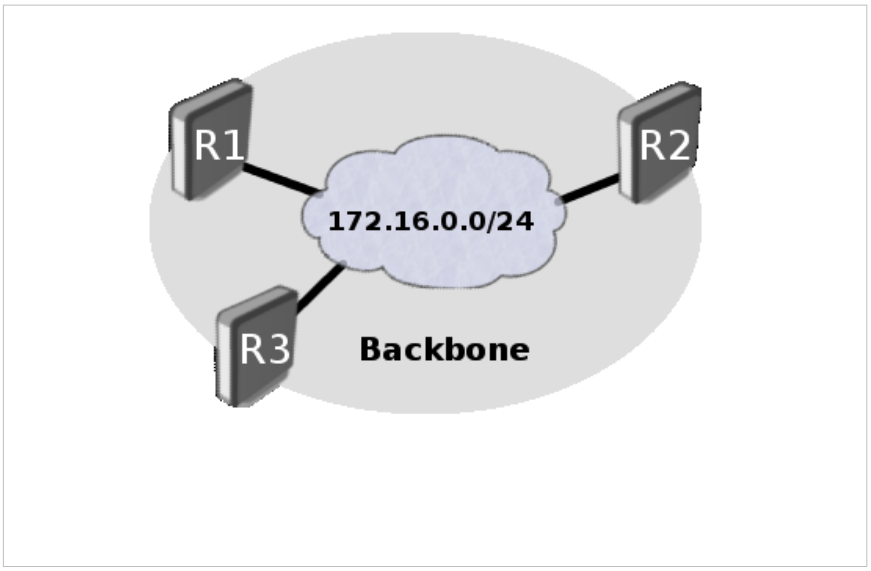
As you can see from image above multiple shortest paths have been found to 172.16.1.0 network, allowing load balancing of the traffic to that destination called equal-cost multipath (ECMP). After the shortest path tree is built, router starts to build the routing table accordingly. Networks are reached consequently to the cost calculated in the tree.

Routing table calculation looks quite simple, however when some of the OSPF extensions are used or OSPF areas are calculated, routing calculation gets more complicated.

Configuring OSPF

Let's look how to configure single-area OSPF network.

One command is required to start OSPF on MikroTik RouterOS - add network in ospf network menu.
Let's assume we have the following network.



It has only one area with three routers connected to the same network 172.16.0.0/24. Backbone area is created during RouterOS installation and additional configuration is not required for area settings.
R1 configuration:

```
/ip address add address=172.16.0.1/24 interface=ether1
/routing ospf network add network=172.16.0.0/24 area=backbone
```

R2 configuration:

```
/ip address add address=172.16.0.2/24 interface=ether1
/routing ospf network add network=172.16.0.0/24 area=backbone
```

R3 configuration:

```
/ip address add address=172.16.0.3/24 interface=ether1
/routing ospf network add network=172.16.0.0/24 area=backbone
```

To verify if OSPF instance is running on router:

```
[admin@MikroTik] /routing ospf> monitor once
             state: running
         router-id: 172.16.0.1
         dijkstras: 6
      db-exchanges: 0
   db-remote-inits: 0
    db-local-inits: 0
  external-imports: 0
```

As you can see OSPF is up and running, notice that **router-id** is set the same as IP address of the router. It was done automatically, because **router-id** was not specified during OSPF configuration.
Add a network to assign interface to the certain area. Look at the OSPF interface menu to verify that dynamic entry was created and correct network type was detected.

```
[admin@MikroTik] /routing ospf interface> print
Flags: X - disabled, I - inactive, D - dynamic, P - passive
 #    INTERFACE         COST  PRIORITY NETWORK-TYPE   AUTHENTICATION AUTHENTICATION-KEY
 0 D  ether1            10    1        broadcast      none
```

Next step is to verify, that both neighbors are found, DR and BDR is elected and adjacencies are established:

```
[admin@MikroTik] /routing ospf neighbor> print
 0 router-id=172.16.0.2 address=172.16.0.2 interface=ether1 priority=1
   dr-address=172.16.0.3 backup-dr-address=172.16.0.2 state="Full" state-changes=5
   ls-retransmits=0 ls-requests=0 db-summaries=0 adjacency=9m2s

 1 router-id=172.16.0.3 address=172.16.0.3 interface=ether1 priority=1
   dr-address=172.16.0.3 backup-dr-address=172.16.0.2 state="Full" state-changes=5
   ls-retransmits=0 ls-requests=0 db-summaries=0 adjacency=6m42s
```

Most of the properties are self explanatory, but if something is unclear, description can be found in neighbor reference manual
Last thing to check whether LSA table is generated properly.

```
[admin@MikroTik] /routing ospf lsa> print
AREA              TYPE         ID              ORIGINATOR      SEQUENCE-NUMBER AGE
backbone          router       172.16.0.1      172.16.0.1      0x80000003      587
backbone          router       172.16.0.2      172.16.0.2      0x80000003      588
backbone          router       172.16.0.3      172.16.0.3      0x80000002      592
backbone          network      172.16.0.3      172.16.0.3      0x80000002      587
```

We have three router links and one network link. All properties are explained in LSA reference manual.
Congratulations, we have fully working OSPF network at this point.

Authentication

It is possible to secure OSPF packets exchange, MikroTik RouterOS provides two authentication methods, simple and MD5. OSPF authentication is disabled by default.
Authentication is configured per interface. Add static ospf interface entry and specify authentication properties to secure OSPF information exchange. md5 authentication configuration on ether1 is shown below:

```
/routing ospf interface
  add interface=ether1 authentication=md5 authentication-key=mySampleKey authentication-key-id=2
```

Simple authentication is plain text authentication method. Method is vulnerable to passive attacks, anybody with packet sniffer can easily get password. Method should be used only to protect OSPF from mis-configurations.

MD5 is a cryptographic authentication and is more preferred. Authentication-key, key-id and OSPF packet content is used to generate message digest that is added to the packet. Unlike the simple authentication method, key is not exchanged over the network.

`Authentication-key-id` value is 1, when authentication is not set (even for router that do not allow to set key id at all).

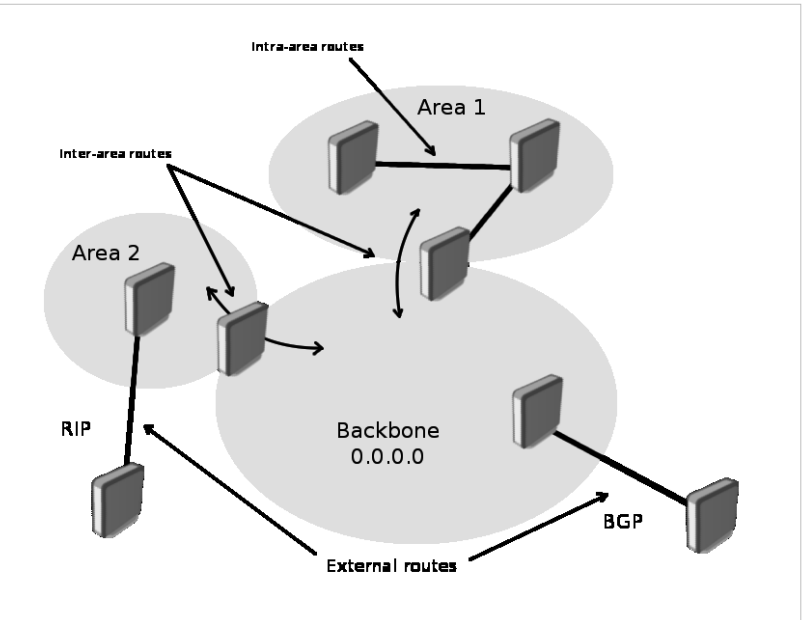
Multi-area networks

Large single area network can produce serious issues:

- Each router recalculates database every time whenever network topology change occurs, the process takes CPU resources.
- Each router holds entire link-state database, which shows the topology of the entire network, it takes memory resources.
- Complete copy of the routing table and number of routing table entries may be significantly greater than the number of networks, that can take even more memory resources.
- Updating large databases require more bandwidth.

To keep routing table size, memory and CPU demands to a manageable levels. OSPF uses a two-layer area hierarchy:

- **backbone (transit) area** - Primary function of this area is the fast and efficient movement of IP packets. Backbone area interconnects other areas and generally, end users are not found within a backbone area.
- **regular area** - Primary function of this area is to connect users and resources. To travel from one are to another, traffic must travel over the backbone, meaning that two regular areas cannot be directly connected. Regular areas have several Subtypes:
  - Standard Area
  - Stub Area
  - Totally Stubby Area
  - Not-so-stubby area (NSSA)



Each area is identified by 32-bit Area ID and has its own link-state database, consisting of router-LSAs and network-LSAs describing how all routers within that area are interconnected. Detailed knowledge of area's topology is hidden from all other areas; router-LSAs and network-LSAs are not flooded beyond the area's borders. Area Border Routers (ABRs) leak addressing information from one area into another in OSPF summary-LSAs. This allows to pick the best area border router when forwarding data to destinations from another area and is called **intra-area routing**.

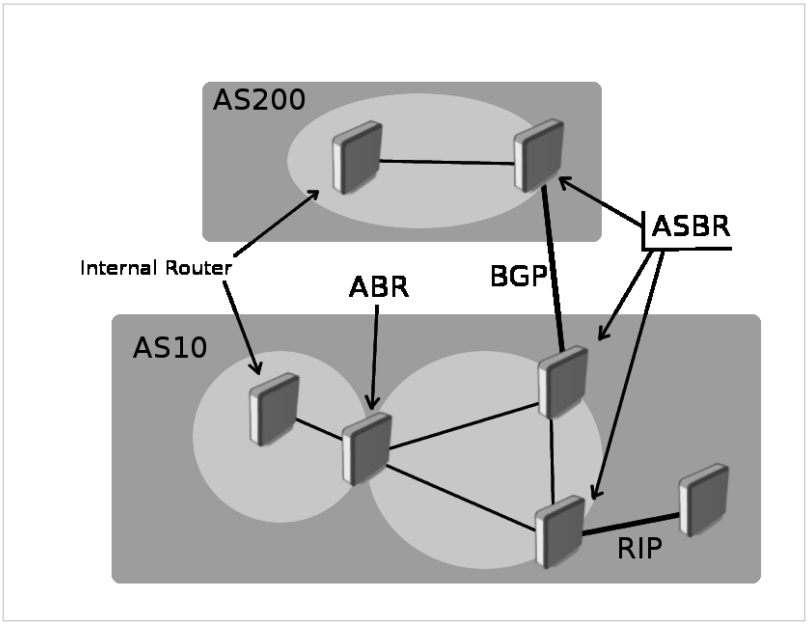
Routing information exchange between areas is essentially Distance Vector algorithm and to prevent algorithm's convergence problems, such as counting to infinity, all areas are required to attach directly to **backbone area** making simple hub-and-spoke topology. Area-ID of backbone area is always 0.0.0.0 and can not be changed.

There are several types of routing information:

- **intra-area routes** - routes generated from within an area (destination belongs to the area).
- **inter-area routes** - routes originated from other areas, also called Summary Routes.
- **external routes** - routes originated from other routing protocols and that are injected into OSPF by redistribution.

External Routing Information

On the edge of an OSPF routing domain, you can find routers called **AS boundary routers (ASBRs)** that run one of other routing protocols. The job of those routers are to import routing information learned from other routing protocols into the OSPF routing domain. External routes can be imported at two separate levels depending on metric type.



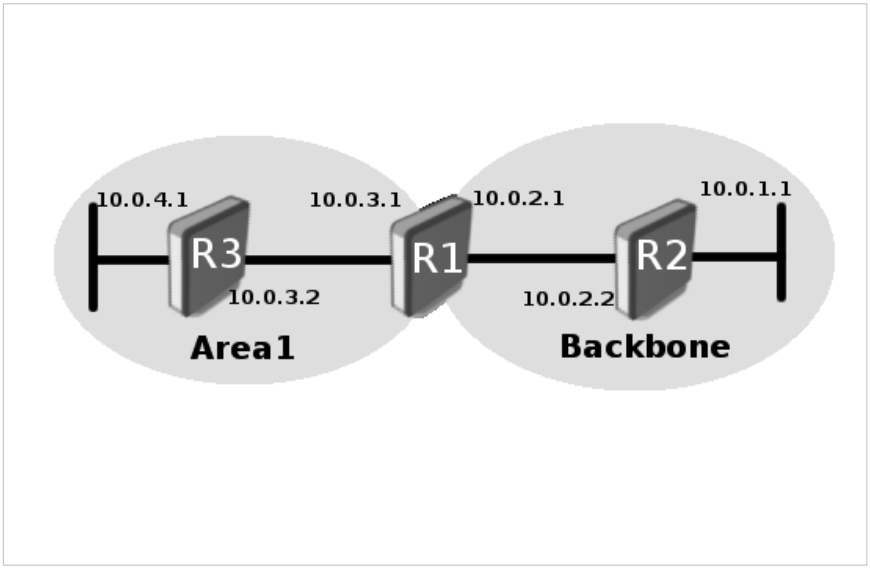
- `type1` - ospf metric is the sum of the internal OSPF cost and the external route cost
- `type2` - ospf metric is equal only to the external route cost.

OSPF provides several area types: backbone area, standard area, stub area and not-so-stubby area. All areas are covered later in the article.

Backbone area is the core of all OSPF network, all areas have to be connected to backbone area. Start configuring OSPF from backbone and then expand network configuration to other areas.

Simple multi-area network

Consider the multi-area network shown below.



R1 configuration:

```
/ip address add address=10.0.3.1/24 interface=ether1
/ip address add address=10.0.2.1/24 interface=ether2
/routing ospf area add name=area1 area-id=1.1.1.1
/routing ospf network add network=10.0.2.0/24 area=backbone
/routing ospf network add network=10.0.3.0/24 area=area1
```

R2 configuration:

```
/ip address add address=10.0.1.1/24 interface=ether2
/ip address add address=10.0.2.2/24 interface=ether1
/routing ospf network add network=10.0.2.0/24 area=backbone
```

R3 configuration:

```
/ip address add address=10.0.3.2/24 interface=ether2
/ip address add address=10.0.4.1/24 interface=ether1
/routing ospf area add name=area1 area-id=1.1.1.1
/routing ospf network add network=10.0.3.0/24 area=area1
```

Route Redistribution

OSPF external routes are routes that are being redistributed from other routing protocols or from static routes.

Remember OSPF configuration setup described in previous section. As you may notice networks 10.0.1.0/24 and 10.0.4.0/24 are not redistributed into OSPF. OSPF protocol does not redistribute external routes by default. Redistribution should be enabled in general OSPF configuration menu to do that. We need to redistribute connected routes in our case, add following configuration to routers R3 and R2:

```
/routing ospf set redistribute-connected=as-type-1
```

Check routing table to see that both networks are redistributed.

```
[admin@MikroTik] /ip route> print
```

Let's add another network to R3:

```
/ip address add address=10.0.5.1/24 interface=ether1
```

10.0.5.0/24 and 10.0.4.0/24 networks are redistributed from R3 over OSPF now. But we do not want other routers to know that 10.0.5.0/24 is reachable over router R3. To achieve it we can add rules in routing filters inside "ospf-out" chain.
Add routing filter to R3

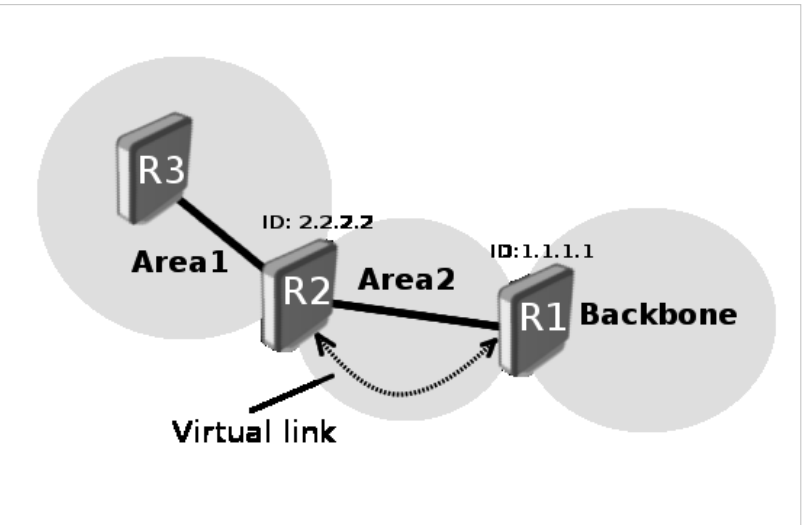
```
/routing filter add chain=ospf-out prefix=10.0.5.0/24 action=discard
```

Routing filters provide two chains to operate with OSPF routes: **ospf-in** and **ospf-out**. Ospf-in chain is used to filter incoming routes and ospf-out is used to filter outgoing routes. More about routing filters can be found in routing filters reference manual.

Virtual Link

All OSPF areas have to be attached to the backbone area, but sometimes physical connection is not possible. In this case areas can be attached logically by using **virtual links**. Also virtual links can be used to glue together fragmented backbone area.

No physical connection to backbone



Area may not have physical connection to backbone, virtual link is used to provide logical path to the backbone of the disconnected area. Link has to be established between two ABRs that have common area with one ABR connected to the backbone.

We can see that both R1 and R2 routers are ABRs and R1 is connected to backbone area. Area2 will be used as **transit area** and R1 is the **entry point** into backbone area. Virtual link has to be configured on both routers.

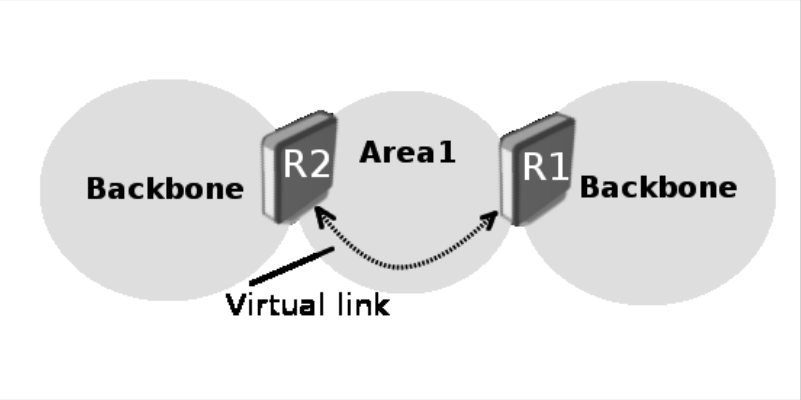
R1 configuration:

```
/routing ospf virtual-link add transit-area=area2 neighbor-id=2.2.2.2
```

R2 configuration:

```
/routing ospf virtual-link add transit-area=area2 neighbor-id=1.1.1.1
```

Partitioned backbone



OSPF allows to link discontinuous parts of the backbone area using virtual links. This might be required when two separate OSPF networks are merged into one large network. Virtual link can be configured between separate ABRs that touch backbone area from each side and have a common area.

Additional area could be created to become transit area, when common area does not exist, it is illustrated in the image above.

Virtual Links are not required for non-backbone areas, when they get partitioned. OSPF does not actively attempt to repair area partitions, each component simply becomes a separate area, when an area becomes partitioned. The backbone performs routing between the new areas. Some destinations are reachable via **intra-area** routing, the area partition requires **inter-area** routing.

However, to maintain full routing after the partition, an address range has not to be split across multiple components of the area partition.

Route Summarization

Route summarization is consolidation of multiple routes into one single advertisement. It is normally done at the area boundaries (Area Border Routers), but summarization can be configured between any two areas.

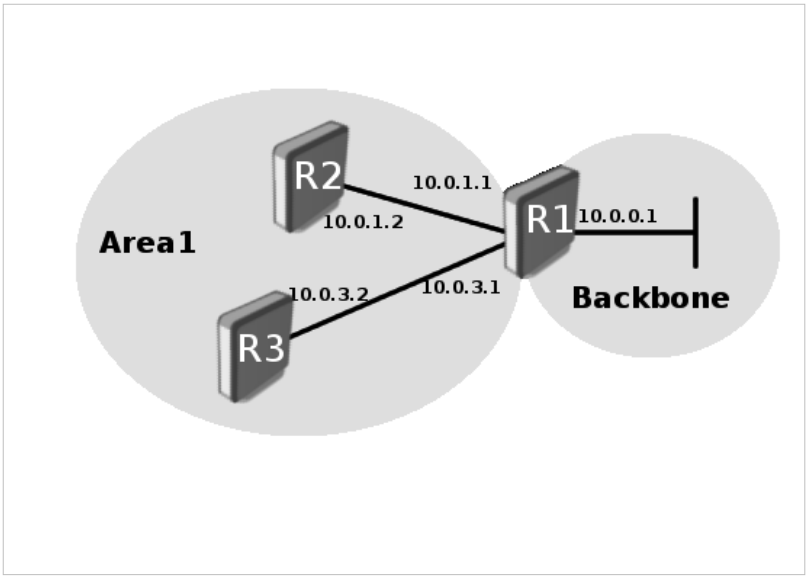
It is better to summarize in the direction to the backbone. Then way the backbone receives all the aggregate addresses and injects them into other areas already summarized. There are two types of summarization: inter-area and external route summarization.

Inter-Area Route Summarization

Inter-area route summarization is done on ABRs, it does not apply to external routes injected into OSPF via redistribution. Summarization configuration is done in OSPF area range menu.

Stub Area

Main purpose of stub areas is to keep such areas from carrying external routes. Routing from these areas to the outside world is based on a default route. Stub area reduces the database size inside an area and reduces memory requirements of routers in the area.



Stub area has few restrictions, ASBR routers cannot be internal to the area, stub area cannot be used as transit area for virtual links. The restrictions are made because stub area is mainly configured not to carry external routes.

Totally stubby area is an extension for stub area. A totally stubby area blocks external routes and summarized (inter-area) routes from going into the area. Only intra-area routes are injected into the area. inject-summary-lsa=no is used to configure totally stubby area in the RouterOS.

Let's consider the example above. Area1 is configured as stub area meaning that routers R2 and R3 will not receive any routing information from backbone area except default route.

R1 configuration:

```
/routing ospf area add name=area1 area-id=1.1.1.1 type=stub inject-summary-lsa=yes
/routing ospf network
add network=10.0.0.0/24 area=backbone
add network=10.0.1.0/24 area=area1
add network=10.0.3.0/24 area=area1
```

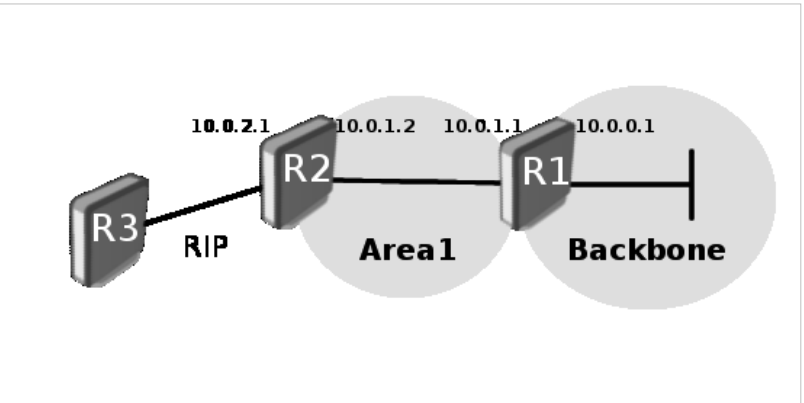
R2 configuration:

```
/routing ospf area add name=area1 area-id=1.1.1.1 type=stub inject-summary-lsa=yes
/routing ospf network
add network=10.0.1.0/24 area=area1
```

R3 configuration:

```
/routing ospf area add name=area1 area-id=1.1.1.1 type=stub inject-summary-lsa=yes
/routing ospf network
add network=10.0.3.0/24 area=area1
```

NSSA



Not-so-stubby area (NSSA) is useful when it is required to inject external routes, but injection of type 5 LSA routes is not required.

Look at the image above. There are two areas (backbone and area1) and RIP connection to area1. We need Area1 to be configured as stub area, but it is also required to inject external routes from RIP protocol. Area1 should be configured as NSSA in this case.

Configuration example does not cover RIP configuration.

R1 configuration:

```
/routing ospf area add name=area1 area-id=1.1.1.1 type=nssa
/routing ospf network
add network=10.0.0.0/24 area=backbone
add network=10.0.1.0/24 area=area1
```

R2 configuration:

```
/routing ospf set redistribute-rip=as-type-1
/routing ospf area add name=area1 area-id=1.1.1.1 type=nssa
/routing ospf network
add network=10.0.1.0/24 area=area1
```

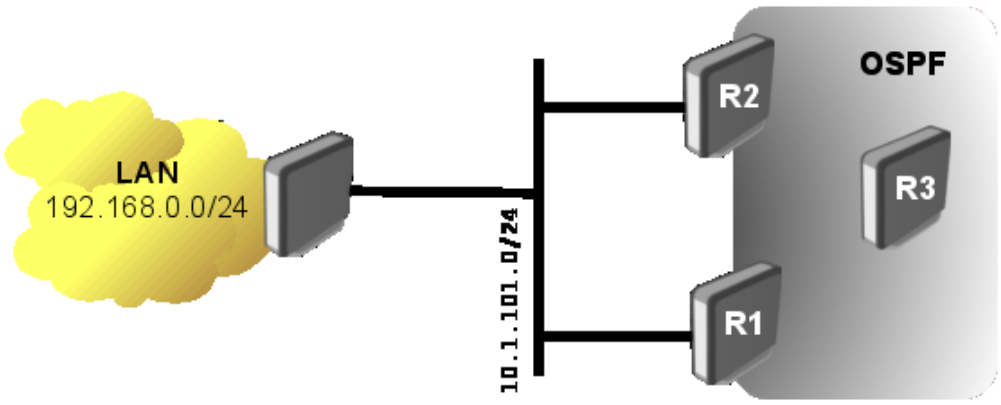
NSSA areas have one another limitation: virtual links cannot be used over such area type.

Related Links

- OSPF Configuration Examples
- OSPF Reference Manual

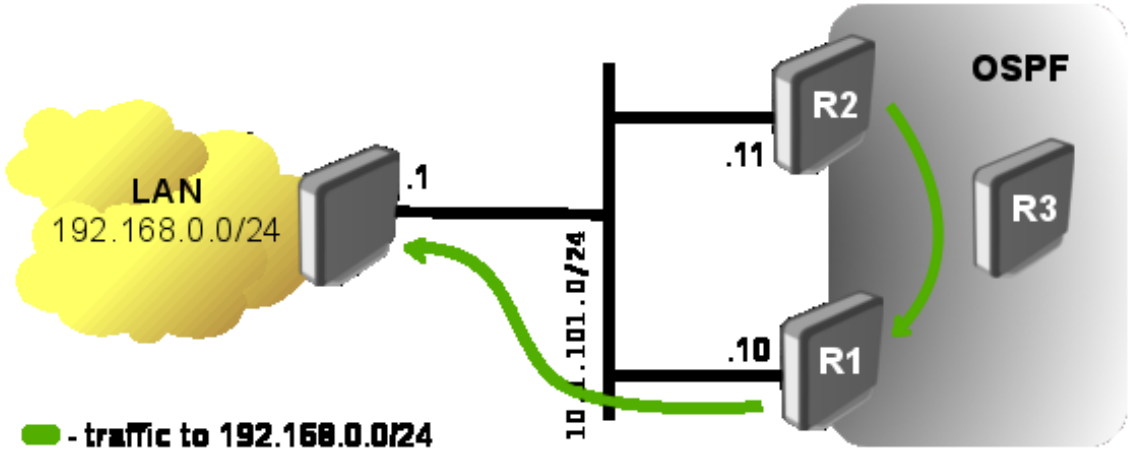
Manual:OSPF Forwarding Address

Applies to RouterOS: v3, v4, v5 +



Lets assume that router **R1** has static route to external network *192.168.0.0/24*. OSPF is running between R1,R2 and R3 and static route is distributed across the OSPF network.

The problem in such setup is obvious, R2 can not reach external network directly. Traffic from **R2** will be forwarded to router **R1**




```
[admin@R2] /ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
...
 8 ADo  192.168.0.0/24                     10.1.101.10          110
```

Fortunately OSPF has mechanism to solve such situations. OSPF router can set **forwarding-address** to something other than itself which indicates that alternate nexthop is possible. Mostly forwarding address is set to **0.0.0.0** suggesting that the route is reachable only via the advertising router.

Forwarding address is set in LSA, if following conditions are met:

- OSPF must be enabled on next-hop interface

- Interface is not passive
- Interface is not p2p or p2mp
- Next-hop address falls into network provided in ospf networks

So knowing conditions, we can make router **R1** to set forwarding address. We simply need to add 10.1.101.0/24 network to OSPF networks in router's **R1** configuration:

```
/routing ospf network add network=10.1.101.0/24 area=backbone
```

Note: OSPF adjacency between routers in 10.1.101.0/24 network is not required

Now lets verify that forwarding address is actually working:

```
[admin@R2] /ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
...
 8 ADo  192.168.0.0/24                        10.1.101.1          110
```

On all OSPF routers you will see LSA set with forwarding address other than 0.0.0.0

```
[admin@R2] /routing ospf lsa>print detail
 instance=default area=external type=as-external id=192.168.0.0
   originator=10.1.101.1 sequence-number=0x80000073 age=16 checksum=0x8510
   options="E" body=
     netmask=255.255.255.0
     forwarding-address=10.1.101.1
     metric=30
     route-tag=0x0
     type1
```

[Top | Back to Content]

Manual:OSPF-examples

Simple OSPF configuration

The following example illustrates how to configure single-area OSPF network. Let's assume we have the following network.

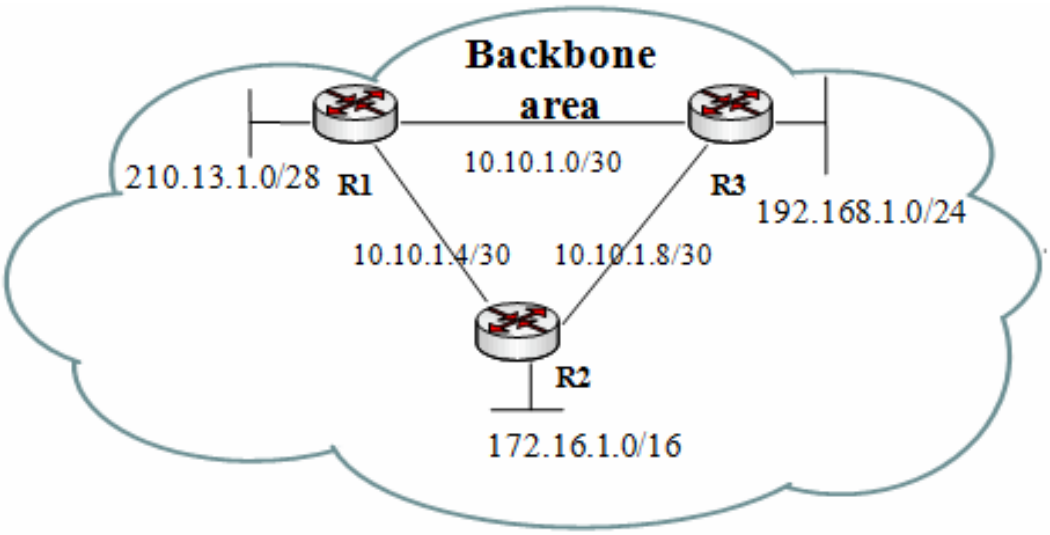


Figure 6.6. Example of single area OSPF network

Example network consists of 3 routers connected together within 10.10.1.0/24 network and each router has also one additional attached network.

In this example following IP addresses are configured:

```
[admin@MikroTikR1]/ip address add address=10.10.1.1/30 interface=ether1
 [admin@MikroTikR1]/ip address add address=10.10.1.5/30 interface=ether2
 [admin@MikroTikR1]/ip address add address=210.13.1.0/28 interface=ether3
```

```
 [admin@MikroTikR2]/ip address add address=10.10.1.6/30 interface=ether1
 [admin@MikroTikR2]/ip address add address=10.10.1.9/30 interface=ether2
 [admin@MikroTikR2]/ip address add address=172.16.1.0/16 interface=ether3
```

```
 [admin@MikroTikR3]/ip address add address=10.10.1.2 /30 interface=ether1
 [admin@MikroTikR3]/ip address add address=10.10.1.10/30 interface=ether2
 [admin@MikroTikR3]/ip address add address=192.168.1.0/24 interface=ether3
```

There are three basic elements of OSPF configuration:

- Enable OSPF instance
- OSPF area configuration
- OSPF network configuration

General information is configured in */routing ospf instance* menu. For advanced OSPF setups, it is possible to run multiple OSPF instances. Default instance configuration is good to start, we just need to enable default instance.

R1:

```
[admin@MikroTikR1] /routing ospf instance> add name=default
```

R2:

```
[admin@MikroTikR2] /routing ospf instance> add name=default
```

R3:

```
[admin@MikroTikR3] /routing ospf instance> add name=default
```

Show OSPF instance information:

```
[admin@MikroTikR1] /routing ospf instance> print
Flags: X - disabled
 0   name="default" router-id=0.0.0.0 distribute-default=never
     redistribute-connected=as-type-1 redistribute-static=as-type-1
     redistribute-rip=no redistribute-bgp=no redistribute-other-ospf=no
     metric-default=1 metric-connected=20 metric-static=20 metric-rip=20
     metric-bgp=auto metric-other-ospf=auto in-filter=ospf-in
out-filter=ospf-out
```

As you can see router-id is 0.0.0.0, it means that router will use one of router's IP addresses as router-id. In most cases it is recommended to set up *loopback* IP address as router-id. ***Loopback*** IP address is virtual, software address that is used for router identification in network. The benefits are that loopback address is always up (active) and can't be down as physical interface. OSPF protocol used it for communication among routers that identified by router-id. Loopback interface are configured as follows:

Create bridge interface named, for example, "loopback":

```
[admin@MikroTikR1] /interface bridge> add name=loopback
```

Add IP address:

```
[admin@MikroTikR1] > ip address add address=10.255.255.1/32 interface=loopback
```

Configure router-id as loopback:

```
[admin@MikroTikR1] /routing ospf instance> set 0 router-id=10.255.255.1
```

This can be done on other routers (R2, R3) as well.

Next step is to configure OSPF area. Backbone area is created during RouterOS installation and additional configuration is not required.

Note: Remember that backbone area-id is always (zero) 0.0.0.0.

And the last step is to add network to the certain OSPF area.

On R1

```
[admin@MikroTikR1] /routing ospf network> add network=210.13.1.0/28 area=backbone
[admin@MikroTikR1] /routing ospf network> add network=10.10.1.0/30 area=backbone
[admin@MikroTikR1] /routing ospf network> add network=10.10.1.4/30 area=backbone
```

Instead of typing in each network, you can aggregate networks using appropriate subnet mask. For example, to aggregate 10.10.1.0/30, 10.10.1.4/30, 10.10.1.8/30 networks, you can set up following ospf network:

```
[admin@MikroTikR1] /routing ospf network> add network=10.10.1.0/'''24''' area=backbone
```

R2:

```
[admin@MikroTikR2] /routing ospf network> add network=172.16.1.0/16 area=backbone
[admin@MikroTikR2] /routing ospf network> add network=10.10.1.0/24 area=backbone
```

R3:

```
[admin@MikroTikR3] /routing ospf network> add network=192.168.1.0/24 area=backbone
[admin@MikroTikR3] /routing ospf network> add network=10.10.1.0/24 area=backbone
```

You can verify your OSPF operation as follows:

- Look at the OSPF interface menu to verify that dynamic entry was created:

```
[admin@MikroTikR1] /routing ospf interface> print
```

- Check your OSPF neighbors, what DR and BDR is elected and adjacencies established:

```
[admin@MikroTikR1] /routing ospf neighbor> print
```

- Check router's routing table (make sure OSPF routes are present):

```
[admin@MikroTik_CE1] > ip route print
```

Simple multi-area configuration

Backbone area is the core of all OSPF network, all areas have to be connected to the backbone area. Start configuring OSPF from backbone and then expand network configuration to other areas.

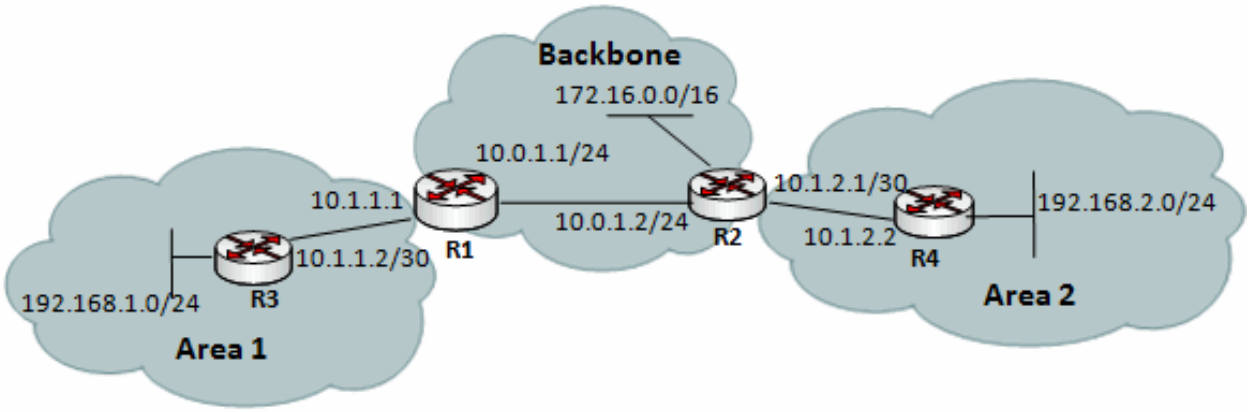


Figure 6.7. Example of multi- area OSPF network

Lets assume that IP addresses are already configured and default OSPF instance is enabled.

All we need to do is:

- create an area
- attach OSPF networks to the area

R1 configuration:

```
/routing ospf> add name=area1 area-id=0.0.0.1
/routing ospf> add network=10.0.1.0/24 area=backbone
/routing ospf> add network=10.1.1.0/30 area=area1
```

R2 configuration:

```
/routing ospf> add name=area2 area-id=0.0.0.2
/routing ospf> add network=10.0.1.0/24 area=backbone
```

```
/routing ospf> add network=10.1.2.0/30 area=area2
```

R3 configuration:

```
/routing ospf> add name=area1 area-id=0.0.0.1
/routing ospf> add network=10.1.1.0/30 area=area1
```

R4 configuration:

```
/routing ospf> add name=area2 area-id=0.0.0.2
/routing ospf> add network=10.1.2.0/30 area=area2
```

Now you can check routing table using command */ip route print*

Routing table on router R3:

```
[admin@R3] > ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 1 ADo  10.0.1.0/24                        10.1.1.1           110
 2 ADC  10.1.1.0/30         10.1.1.2       ether1             110
 3 ADo  10.1.2.0/30                        10.1.1.1           110
 4 ADC  192.168.1.0/24      192.168.1.1    ether2             0
```

As you can see remote networks 172.16.0.0/16 and 192.168.2.0/24 are not in the routing table, because they are not distributed by OSPF. *Redistribution* feature allows different routing protocols to exchange routing information making possible, for example, to redistribute static or connected routes into OSPF. In our setup we need to redistribute connected network. We need to add following configuration on routers R1, R2 and R3.

```
[admin@R3] /routing ospf instance> set 0 redistribute-connected=as-type-1
[admin@R3] /routing ospf instance> print
Flags: X - disabled
 0   name="default" router-id=0.0.0.0 distribute-default=never
     <u>redistribute-connected=as-type-1</u> redistribute-static=no
     redistribute-rip=no redistribute-bgp=no redistribute-other-ospf=no
     metric-default=1 metric-connected=20 metric-static=20 metric-rip=20
     metric-bgp=auto metric-other-ospf=auto in-filter=ospf-in
     out-filter=ospf-out
```

Now check router R3 to see if routes 192.168.2.0/24 and 172.16.0.0/16 are installed in routing table.

```
[admin@R3] > ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 1 ADo  10.0.1.0/24                        10.1.1.1           110
 2 ADC  10.1.1.0/30         10.1.1.2       ether1             110
 3 ADo  10.1.2.0/30                        10.1.1.1           110
 4 ADo  172.16.0.0/16                      10.1.1.1           110
 5 ADC  192.168.1.0/24      192.168.1.1    ether2             0
```

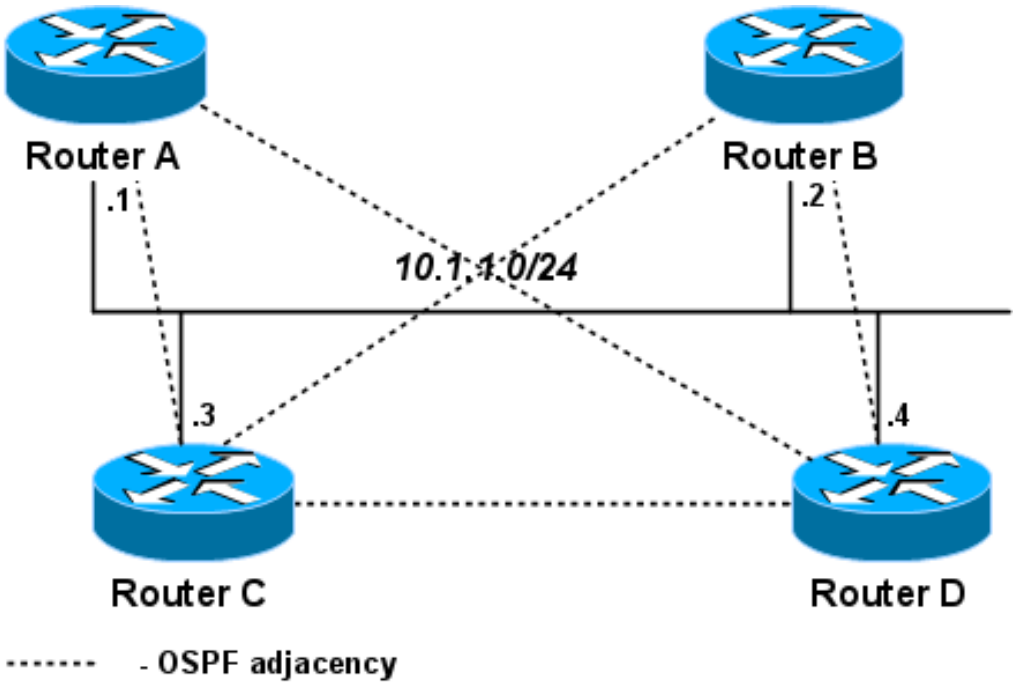
```
 6 ADo  192.168.2.0/24                    10.1.1.1           110
```

NBMA networks

OSPF network type NBMA (Non-Broadcast Multiple Access) uses only unicast communications, so it is the preferred way of OSPF configuration in situations where multicast addressing is not possible or desirable for some reasons. Examples of such situations:

- in 802.11 wireless networks multicast packets are not always reliably delivered (read Multicast_and_Wireless for details); using multicast here can create OSPF stability problems;
- using multicast may be not efficient in bridged or meshed networks (i.e. large layer-2 broadcast domains).

Especially efficient way to configure OSPF is to allow only a few routers on a link to become the designated router. (But be careful - if all routers that are capable of becoming the designated router will be down on some link, OSPF will be down on that link too!) Since a router can become the DR only when priority on it's interface is not zero, this priority can be configured as zero in interface and nbma-neighbor configuration to prevent that from happening.



In this setup only C and D are allowed to become designated routers.

On all routers:

```
routing ospf network add network=10.1.1.0/24 area=backbone
routing ospf nbma-neighbor add address=10.1.1.1 priority=0
routing ospf nbma-neighbor add address=10.1.1.2 priority=0
routing ospf nbma-neighbor add address=10.1.1.3 priority=1
routing ospf nbma-neighbor add address=10.1.1.4 priority=1
```

(For simplicity, to keep configuration the same on all routers, nbma-neighbor to self is also added. Normally you wouldn't do that, but it does not cause any harm either.)

Configure interface priorities. On routers A, B:

```
routing ospf interface add interface=ether1 network-type=nbma priority=0
```

On routers C, D (they can become the designated router):

```
routing ospf interface add interface=ether1 network-type=nbma priority=1
```

Results

On Router A:

```
[admin@A] > routing ospf neighbor print
 0 router-id=10.1.1.5 address=10.1.1.5 interface=ether1 priority=1 dr-address=10.1.1.4
   backup-dr-address=10.1.1.3 state="Full" state-changes=6 ls-retransmits=0
   ls-requests=0 db-summaries=0 adjacency=4m53s

 1 router-id=10.1.1.3 address=10.1.1.3 interface=ether1 priority=1 dr-address=1.1.1.4
   backup-dr-address=10.1.1.3 state="Full" state-changes=6 ls-retransmits=0
   ls-requests=0 db-summaries=0 adjacency=4m43s

 2 address=10.1.1.2 interface=ether1 priority=0 state="Down" state-changes=2

 3 address=10.1.1.1 interface=ether1 priority=0 state="Down" state-changes=2
```

On Router D:

```
[admin@D] > routing ospf neighbor print
 0 address=10.1.1.4 interface=ether1 priority=1 state="Down" state-changes=2

 1 router-id=10.1.1.3 address=10.1.1.3 interface=ether1 priority=1 dr-address=10.1.1.4
   backup-dr-address=10.1.1.3 state="Full" state-changes=6 ls-retransmits=0
   ls-requests=0 db-summaries=0 adjacency=6m8s

 2 router-id=10.1.1.2 address=10.1.1.2 interface=ether1 priority=0 dr-address=10.1.1.4
   backup-dr-address=10.1.1.3 state="Full" state-changes=5 ls-retransmits=0
   ls-requests=0 db-summaries=0 adjacency=6m4s

 3 router-id=10.1.1.1 address=10.1.1.1 interface=ether1 priority=0 dr-address=10.1.1.4
   backup-dr-address=10.1.1.3 state="Full" state-changes=5 ls-retransmits=0
   ls-requests=0 db-summaries=0 adjacency=6m4s
```

OSPF Forwarding Address

OSPF may take extra hops at the boundary between OSPF routing domain and another Autonomous System. By looking at the following illustration you can see that even if router R3 is directly connected, packets will travel through the OSPF network and use router R1 as a gateway to other AS.

To overcome this problem, concept of OSPF *forwarding-address* was introduced. This concept allows to say "Send traffic directly to router R1". This is achieved by setting *forwarding address* other than itself in LSA updates indicating that there is an alternate next-hop. Mostly all the time forwarding address is left 0.0.0.0, suggesting that the route is reachable only through the advertising router.

Sere the full example

[Top | Back to Content **]**

Manual:Routing/OSPF

Applies to RouterOS: v3, v4 +

Summary

MikroTik RouterOS implements OSPF version 2 (RFC 2328). The OSPF protocol is the link-state protocol that takes care of the routes in the dynamic network structure that can employ different paths to its subnetworks. It always chooses shortest path to the subnetwork first.

Instance

Sub-menu: `/routing ospf instance`

Since v3.17 it is possible to run multiple OSPF instances. General OSPF configuration now is moved to instances.

Properties

Property	Description
distribute-default (*never* | *if-installed-as-type-1* | *if-installed-as-type-2* | *always-as-type-1* | *always-as-type-2*; Default: **never**)	specifies how to distribute default route. Should be used for ABR (Area Border router) or ASBR (Autonomous System boundary router) • `never` - do not send own default route to other routers • `if-installed-as-type-1` - send the default route with type 1 metric only if it has been installed (a static default route, or route added by DHCP, PPP, etc.) • `if-installed-as-type-2` - send the default route with type 2 metric only if it has been installed (a static default route, or route added by DHCP, PPP, etc.) • `always-as-type-1` - always send the default route with type 1 metric • `always-as-type-2` - always send the default route with type 2 metric
domain-id (*Hex|Address*;)	MPLS related parameter. Identifies OSPF domain of the instance. This value is attached to OSPF routes redistributed in BGP as VPNv4 routes as BGP extended community attribute, and used when BGP VPNv4 routes are redistributed back OSPF to determine whether to generate inter-area or AS-external LSA for that route. By default Null domain-id is used, as described in RFC 4577.
domain-tag (*integer: 0..4294967295* ;)	if set, then used in route redistribution (as route-tag in all external LSAs generated by this router), and in route calculation (all external LSAs having this route tag are ignored). Needed for interoperability with older Cisco systems. By default not set.
in-filter (*string*;)	name of the routing filter chain used for incoming prefixes
metric-bgp (*integer|auto*; Default: **20**)	routes learned from the BGP protocol are redistributed with this metric. When set to *auto*, MED attribute value from BGP route will be used, if MED is not set then default value 20 is used.
metric-connected (*integer*; Default: **20**)	routes to directly connected networks are distributed with this metric
metric-default (*integer*; Default: **1**)	the default route is distributed with this metric
metric-other-ospf (*integer|auto*; Default: **20**)	routes learned from other OSPF instances are redistributed with this metric. If *auto* is configured, then the cost from previous instance is taken into account, otherwise cost is set to statically configured value.
metric-rip (*integer*; Default: **20**)	routes learned from the RIP protocol are redistributed with this metric

metric-static (*integer*; Default: **20**)	static routes are distributed with this metric
mpls-te-area (*string*;)	the area used for MPLS traffic engineering. TE Opaque LSAs are generated in this area. No more than one OSPF instance can have mpls-te-area configured.
mpls-te-router-id (*ip*;)	loopback interface from which to take IP address used as Router-ID in MPLS TE Opaque LSAs
out-filter (*string*;)	name of the routing filter chain used for outgoing prefixes
redistribute-bgp (*as-type-1* | *as-type-2* | *no*; Default: **no**)	redistribute routes learned by the BGP protocol
redistribute-connected (*as-type-1* | *as-type-2* | *no*; Default: **no**)	redistribute connected routes, i.e. routes to directly reachable networks
redistribute-other-ospf (*as-type-1* | *as-type-2* | *no*; Default: **no**)	redistribute routes learned by other OSPF instances
redistribute-rip (*as-type-1* | *as-type-2* | *no*; Default: **no**)	redistribute routes learned by the RIP protocol
redistribute-static (*as-type-1* | *as-type-2* | *no*; Default: **no**)	redistribute static routes
router-id (*IP address*; Default: **0.0.0.0**)	the OSPF Router ID. If not specified, OSPF use one of router's IP addresses.
routing-table (*name of routing table*;)	the routing table this OSPF instance operates on

Notes

OSPF protocol supports two types of metrics:

- `type1` - ospf metric is the sum of the internal OSPF cost and the external route cost
- `type2` - ospf metric is equal only to the external route cost.

Status

Command `/routing ospf monitor` will display current OSPF status.
For multi instance OSPF you have to use following command: `/routing ospf instance print status`

Available read only properties:

Property	Description
state (*down* | *running*)	shows if OSPF is running or not
effective-router-id (*IP address*)	Router-ID chosen by OSPF.
dijkstras (*integer*)	shows how many times Dijkstra's algorithm was executed (i.e. OSPF routes were recalculated)
db-exchanges (*integer*)	number of OSPF database exchanges currently going on
external-imports (*integer*)	how many external routes were imported into OSPF from this router

Area

Sub-menu: `/routing ospf area`

Description

OSPF allows collections of routers to be grouped together. Such a group is called an area. Each area runs a separate copy of the basic link-state routing algorithm. This means that each area has its own link-state database and corresponding shortest path tree.

The structure of an area is invisible from other areas. This isolation of knowledge makes the protocol more scalable if multiple areas are used; routing table calculation takes less CPU resources and routing traffic is reduced.

However, multi-area setups create additional complexity. It is not recommended separate areas with fewer than 50 routers. The maximum number of routers in one area is mostly dependent on CPU power you have for routing table calculation.

Properties

Property	Description
area-id (*IP address*; Default: **0.0.0.0**)	OSPF area identifier. If the router has networks in more than one area, then an area with area-id=0.0.0.0 (the backbone) must always be present. The backbone always contains all area border routers. The backbone is responsible for distributing routing information between non-backbone areas. The backbone must be contiguous, i.e. there must be no disconnected segments. However, area border routers do not need to be physically connected to the backbone - connection to it may be simulated using a virtual link.
default-cost (*integer*; Default: **1**)	specifies the cost for the default route originated by this stub area ABR. Applicable only for stub areas on ABRs
inject-summary-lsas (*yes* | *no*; Default: **yes**)	specifies whether to flood summary LSAs in this stub area. Applicable only for stub areas on ABRs
name (*string*; Default:)	the name of the area
translator-role (*translate-always* | *translate-candidate* | *translate-never*; Default: **translate-candidate**)	Parameter indicates which ABR will be used as translator from type7 to type5. Applicable only if area type is NSSA • translate-always - router will be always used as translator • translate-never - router will never be used as translator • translate-candidate - ospf ellects one of candidate routers to be a translator
type (*default* | *nssa* | *stub*; Default: **default**)	area type

Status

`/routing ospf area print status` will show additional read-only properties

Property	Description
interfaces (*integer*;)	count of interfaces assigned to this area
active-interfaces (*integer*;)	count of interfaces in operating state assigned to this area
neighbors (*integer*;)	count of OSPF neighbors in this area
adjacent-neighbors (*integer*;)	count of adjacent OSPF neighbors in this area

Area Range

Sub-menu: `/routing ospf area range`

Description

Prefix ranges are used to aggregate routing information on area boundaries. By default, ABR creates a summary LSA for each route in specific area, and advertises it in adjacent areas.

Using ranges allows to create only one summary LSA for multiple routes and send only single advertisement into adjacent areas, or to suppress advertisements altogether.

If a range is configured with 'advertise' parameter, a single summary LSA is advertised for each range if there are any routes under the range is the specific area. Else ('advertise' parameter disabled) no summary LSAs are created and advertised outside area boundaries at all.

Properties

Property	Description
advertise (*yes* | *no*; Default: **yes**)	whether to create summary LSA and advertise it to adjacent areas
area (*string*; Default:)	the OSPF area associated with this range
cost (*integer* | *default*; Default: **default**)	the cost of the summary LSA this range will create `default` - use the largest cost of all routes used (i.e. routes that fall within this range)
range (*IP prefix*; Default:)	the network prefix of this range

Note: For an active range (i.e. one that has at least one OSPF route from the specified area falling under it), a route with type 'unreachable' is created and installed in the routing table.

Network

Sub-menu: `/routing ospf network`

To start the OSPF protocol, you have to define the networks on which OSPF will run and associated area for each of these networks

Property	Description
area (*string*; Default: **backbone**)	the OSPF area to be associated with the specified address range
network (*IP prefix*; Default:)	the network prefix associated with the area. OSPF will be enabled on all interfaces that has at least one address falling within this range. Note that the network prefix of the address is used for this check (i.e. not the local address). For point-to-point interfaces this means the address of the remote endpoint.

Interface

Sub-menu: `/routing ospf interface`

Property	Description
authentication (*none* | *simple* | *md5*; Default: **none**)	specifies authentication method for OSPF protocol messages. • `none` - do not use authentication • `simple` - plain text authentication • `md5` - keyed Message Digest 5 authentication
authentication-key (*string*; Default: "")	authentication key to be used for simple or MD5 authentication
authentication-key-id (*integer*; Default: **1**)	key id is used to calculate message digest (used only when MD5 authentication is enabled). Value should match on all OSPF routers from the same region.
cost (*integer: 1..65535*; Default: **1**)	interface cost expressed as link state metric
dead-interval (*time*; Default: **40s**)	specifies the interval after which a neighbor is declared as dead. This interval is advertised in hello packets. This value must be the same for all routers on a specific network, otherwise adjacency between them will not form
hello-interval (*time*; Default: **10s**)	the interval between hello packets that the router sends out this interface. The smaller this interval is, the faster topological changes will be detected, but more routing traffic will ensue. This value must be the same for all routers on a specific network, otherwise adjacency between them will not form
interface (*string* | *all*; Default: **all**)	the interface name • `all` - for all interfaces without specific configuration
network-type (*broadcast* | *nbma* | *point-to-point* | *ptmp*; Default: **broadcast**)	the OSPF network type on this interface. Note that if interface configuration does not exist, the default network type is 'point-to-point' on PtP interfaces, and 'broadcast' on all other interfaces. • `broadcast` - network type suitable for Ethernet and other multicast capable link layers. Elects designated router • `nbma` - Non-Broadcast Multiple Access. Protocol packets are sent to each neighbors unicast address. Requires manual configuration of neighbors. Elects designated router • `point-to-point` - suitable for networks that consists only of two nodes. Does not elect designed router • `ptmp` - Point-to-Multipoint. Easier to configure than NBMA because it requires no manual configuration of neighbor. Does not elect designed router. This is the most robust network type and as such suitable for wireless networks, if 'broadcast' mode does not works good enough for them
passive (*yes* | *no*; Default: **no**)	if enabled, do not send or receive OSPF traffic on this interface
priority (*integer: 0..255*; Default: **1**)	router's priority. Used to determine the designated router in a broadcast network. The router with highest priority value takes precedence. Priority value 0 means the router is not eligible to become designated or backup designated router at all.
retransmit-interval (*time*; Default: **5s**)	time between retransmitting lost link state advertisements. When a router sends a link state advertisement (LSA) to its neighbor, it keeps the LSA until it receives back the acknowledgment. If it receives no acknowledgment in time, it will retransmit the LSA

transmit-delay (*time*; Default: **1s**)	link state transmit delay is the estimated time it takes to transmit a link state update packet on the interface

Status

`/routing ospf interface print status` will show additional information about used interfaces

Property	Description
ip-address (*IP address*;)	Ip address assigned to this interface
state (*backup* | *designated-router* | *point-to-point* | *passive;*)	current interface state
instance (*instance name*;)	OSPF instance that is using this interface
area (*area name*;)	area to which interface is assigned
neighbors (*integer*;)	count of OSPF neighbors found on this interface
adjacent-neighbors (*integer*;)	count of OSPF neighbors found on this interface that have formed adjacencies
designated-router (*IP address*;)	router-ID of elected designated router (DR)
backup-designated-router (*IP address*;)	router-ID of elected backup designated router (BDR)

NBMA Neighbor

Sub-menu: `/routing ospf nbma-neighbor`

Manual configuration for non-broadcast multi-access neighbors. Required only if interfaces with 'network-type=nbma' are configured.

Property	Description
address (*IP address*; Default:)	the unicast IP address of the neighbor
poll-interval (*time*; Default: **2m**)	how often to send hello messages to neighbors which are in "down" state (i.e. there is no traffic from them)
priority (*integer: 0..255*; Default: **0**)	assumed priority value of neighbors which are in "down" state

Virtual Link

Sub-menu: `/routing ospf virtual-link`

Description

As stated in OSPF RFC, the backbone area must be contiguous. However, it is possible to define areas in such a way that the backbone is no longer contiguous. In this case the system administrator must restore backbone connectivity by configuring virtual links. Virtual link can be configured between two routers through common area called transit area, one of them should have to be connected with backbone. Virtual links belong to the backbone. The protocol treats two routers joined by a virtual link as if they were connected by an unnumbered point-to-point network

Properties

Property	Description
authentication (*none* | *simple* | *md5*; Default: **none**)	specifies authentication method for OSPF protocol messages.
authentication-key (*string*; Default: **""**)	authentication key to be used for simple or MD5 authentication
authentication-key-id (*integer*; Default: **1**)	key id used in MD5 authentication
neighbor-id (*IP address*; Default: **0.0.0.0**)	specifies **router-id** of the neighbour
transit-area (*string*; Default: **(unknown)**)	a non-backbone area the two routers have in common

Note: Virtual link should be configured on both routers. Virtual links can not be established through stub areas.

LSA

Sub-menu: `/routing ospf lsa`

Read only properties:

Property	Description
instance (*string*)	Instance name where LSA is used.
area (*string*)	
type (*string*)	
id (*IP address*)	LSA record ID
originator (*IP address*)	LSA record originator
sequence-number (*string*)	Number of times the LSA for a link has been updated.
age (*integerr*)	How long ago (in seconds) the last update occurred
checksum (*string*)	LSA checksum
options (*string*)	
body (*string*)	

Neighbor

Sub-menu: `/routing ospf Neighbor`

Read only properties:

Property	Description
router-id (*IP address*)	neighbor router's **RouterID**
address (*IP address*)	IP address of neighbor router that is used to form OSPF connection
interface (*string*)	interface that neighbor router is connected to
priority (*integer*)	priority configured on neighbor
dr-address (*IP address*)	IP address of Designated Router
backup-dr-address (*IP address*)	IP address of Backup Designated Router

state (*down* | *attempt* | *init* | *2-way* | *ExStart* | *Exchange* | *Loading* | *full*)	• **Down** - No Hello packets has been received from neighbor. • **Attempt** - Applies only to NBMA clouds. State indicates that no recent information was received from neighbor. • **Init** - Hello packet received from the neighbor, but bidirectional communication is not established (Its own RouterID is not listed in Hello packet). • **2-way** - This state indicates that bi-directional communication is established. DR and BDR election occur during this state, routers build adjacencies based on whether router is DR or BDR, link is point-to-point or a virtual link. • **ExStart** - Routers try to establish the initial sequence number that is used for the packets information exchange. Router with higher ID becomes the master and starts the exchange. • **Exchange** - Routers exchange database description (DD) packets. • **Loading** - In this state actual link state information is exchanged. Link State Request packets are sent to neighbors to request any new LSAs that were found during Exchange state. • **Full** - Adjacency is complete, neighbor routers are fully adjacent. LSA information is synchronized between adjacent routers. Routers achieve the full state with their DR and BDR only, exception is P2P links.
state-changes (*integer*)	Total count of OSPF state changes since neighbor identification
ls-retransmits (*integer*)	
ls-requests (*integer*)	
db-summaries (*integer*)	
adjacency (*time*)	Elapsed time since adjacency was formed

OSPF Router

Sub-menu: `/routing ospf ospf-router`

List of all area border routers (ABRs).

Read only properties:

Property	Description
area (*string*)	
router-id (*IP address*)	
state (*string*)	
gateway (*IP address*)	
cost (*integer*)	

Route

Sub-menu: `/routing ospf route`

Read only properties:

Property	Description
instance (*string*)	Which OSPF instance route belongs to
dst-address (*IP prefix*)	Destination prefix
state (*intra-area* | *inter-area* | *ext-1* | *ext-2* | *imported-ext-1* | *imported-ext-2*)	State representing origin of the route
gateway (*IP address*)	used gateway
interface (*string*)	used interface
cost (*integer*)	Cost of the route
area (*external* | *backbone* | <*other area*>)	Which OSPF area this route belongs to

Sham link

Sub-menu: `/routing ospf sham-link`

Description

A sham-link is required between any two VPN sites that belong to the same OSPF area and share an OSPF backdoor link. If there is no intra-area link between the CE routers, you do not need to configure an OSPF sham link.

Sham link configuration example

Sham link must be configured on both sides.

For a sham link to be active, two conditions must be met:

- src-address is a valid local address with /32 netmask in OSPF instance's routing table.
- there is a valid route to dst-address in the OSPF instance's routing table.

When the sham link is active, hello packets are sent on it only until the neighbor reaches full state. After that, hello packet sending on the sham link is suppressed.

RouterOS does not support periodic LSA refresh suppression on sham-links yet.

Properties

Property	Description
area (*area name*)	name of area that shares an OSPF backdoor link
cost (*integer: 1..65535*)	cost of the link
dst-address (*IP address*)	loopback address of link's remote router
src-address (*IP address*)	loopback address of link's local router

See More

- OSPF case studies
- OSPF Configuration Examples

[Top | Back to Content]

Manual:OSPF and Point-to-Point interfaces

OSPF configuration on PPP interfaces often is a subject to misunderstanding. You need to keep in mind two things:

1. There is no need to explicitly configure an interface in "/routing ospf interface" to start running OSPF on it. Only "routing ospf network" configuration determines whether the interface will be active or not. If it has matching network network, i.e. the address of the interface falls within range of some network, then the interface will be running OSPF. Else it won't participate in the protocol. "/routing ospf interface" is used only if specific configuration for some interface is needed - typically to configure different link cost.
2. In case of PPP interfaces, the interface will be active if **either** local address or the address of remote are matched against some network. See sample configuration for an illustration. This counterintuitive behaviour will be changed in 3.x routing-test package. Only remote address will be considered there.

- Also remember that running OSPF on a big number of (flapping) PPP interfaces is not recommended.

Configuration example: use local address as OSPF network

Assume we have a PPPoE tunnel between two routers 10.0.0.134 and 10.0.0.133. Configure OSPF on the PPPoE interface on the first router:

```
[admin@I] > /ip address p
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   10.0.0.133/24      10.0.0.0        10.0.0.255      ether1
 1 D 10.1.1.254/32      10.1.1.1        0.0.0.0         pppoe-out1
[admin@I] > routing ospf network add network=10.1.1.254/32 area=backbone
```

Do the same on the second router:

```
[admin@II] > /ip address p
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   10.0.0.134/24      10.0.0.0        10.0.0.255      ether1
 1 D 10.1.1.1/32        10.1.1.254      0.0.0.0         <pppoe-atis>
[admin@II] > routing ospf network add network=10.1.1.1/32 area=backbone
```

An OSPF adjacency has been established; neighbor at 10.1.1.1 is in 'Full' state:

```
[admin@I] > routing ospf neighbor pr
 router-id=10.0.0.133 address=10.1.1.254 priority=1 dr-address=0.0.0.0
   backup-dr-address-id=0.0.0.0 state="2-Way" state-changes=0 ls-retransmits=0
   ls-requests=0 db-summaries=0
 router-id=10.0.0.134 address=10.1.1.1 priority=1 dr-address=0.0.0.0
   backup-dr-address-id=0.0.0.0 state="Full" state-changes=5 ls-retransmits=0
   ls-requests=0 db-summaries=0
[admin@I] >
```

External links

- OSPF in MT manual [1]
- OSPF RFC [2]

References

[1] http://www.mikrotik.com/docs/ros/2.9/routing/ospf
[2] http://rfc-ref.org/RFC-TEXTS/2328/contents.html

Manual:Interface/PPPoE

Applies to RouterOS: v3, v4

Summary

The PPPoE (Point to Point Protocol over Ethernet) protocol provides extensive user management, network management and accounting benefits to ISPs and network administrators. Currently PPPoE is used mainly by ISPs to control client connections for xDSL and cable modems as well as plain Ethernet networks. PPPoE is an extension of the standard Point to Point Protocol (PPP). The difference between them is expressed in transport method: PPPoE employs Ethernet instead of serial modem connection.

Generally speaking, PPPoE is used to hand out IP addresses to clients based on the username (and workstation, if desired) authentication as opposed to workstation only authentication, when static IP addresses or DHCP are used. It is adviced not to use static IP addresses or DHCP on the same interfaces as PPPoE for obvious security reasons.

The PPPoE client and server work over any Ethernet level interface on the router - wireless 802.11 (Aironet, Cisco, WaveLan, Prism, Atheros), 10/100/1000 Mbit/s Ethernet, RadioLan and EoIP (Ethernet over IP tunnel).

Feature list

- PPPoE server and client support;
- Multilink PPP (MLPPP);
- MLPPP over single link (ability to transmit full-sized frames);
- BCP (Bridge Control Protocol) support - allows to send raw Ethernet frames over PPP links;
- MPPE 40bit and MPPE 128bit RSA encryption;
- pap, chap, mschap v1/v2 authentication;
- RADIUS support for client authentication and accounting.

Note that when RADIUS server is authenticating a user with CHAP, MS-CHAPv1 or MS-CHAPv2, the RADIUS protocol does not use shared secret, it is used only in authentication reply. So if you have a wrong shared secret, RADIUS server will accept the request. You can use /radius monitor command to see bad-replies parameter. This value should increase whenever a client tries to connect.

Supported connections:

- MikroTik RouterOS PPPoE client to any PPPoE server (access concentrator)
- MikroTik RouterOS server (access concentrator) to multiple PPPoE clients (clients are avaliable for almost all operating systems and most routers)

Specifications

- Packages required: ppp
- License required: Level1 (limited to 1 interface) , Level3 (limited to 200 interfaces) , Level4 (limited to 200 interfaces) , Level5 (limited to 500 interfaces) , Level6 (unlimited)
- Submenu level: /interface pppoe-server, /interface pppoe-client
- Standards and Technologies: PPPoE (RFC 2516)
- Hardware usage: PPPoE server may require additional RAM (uses approx. 9KiB (plus extra 10KiB for packet queue, if data rate limitation is used) for each connection) and CPU power. Maximum of 65535 connections is supported.

Quick Setup Guide

To configure MikroTik RouterOS to be a PPPoE client, just add a pppoe-client:

```
/interface pppoe-client
  add name=pppoe-user-mike user=user password=passwd interface=wlan1 \
  service-name=internet disabled=no
```

To configure MikroTik RouterOS to be an Access Concentrator (PPPoE Server):

- add an address pool for the clients from 10.1.1.62 to 10.1.1.72;
- add ppp profile;
- add ppp secret (username/password);
- add pppoe server itself.

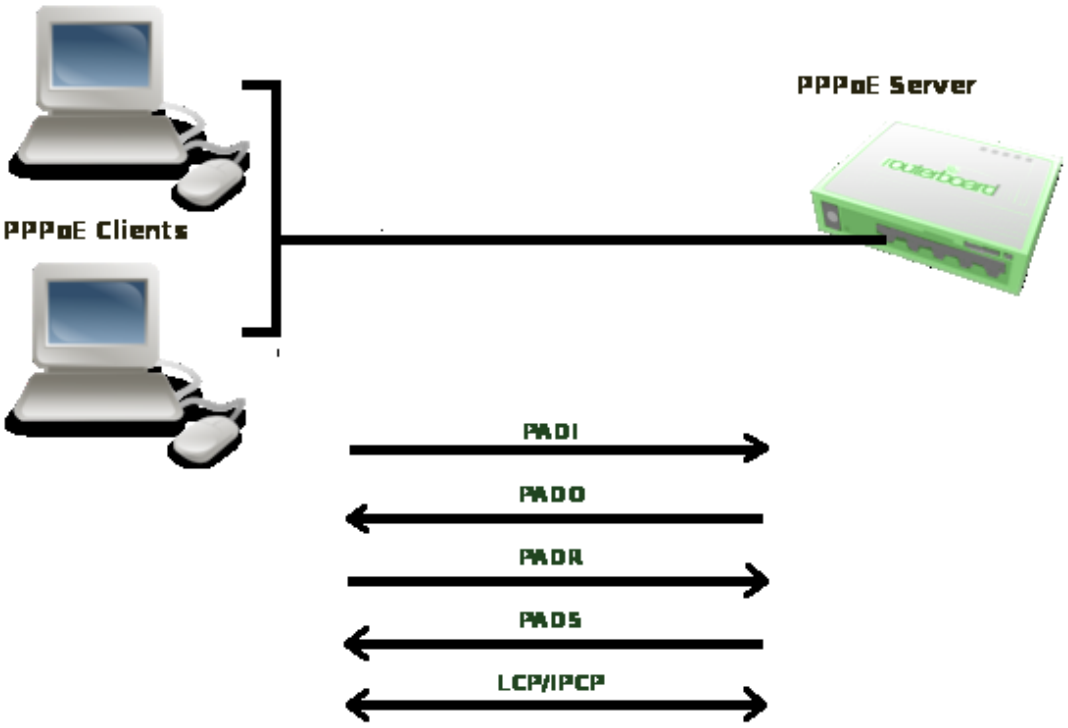
```
/ip pool
add name="pppoe-pool" ranges=10.1.1.62-10.1.1.72

/ppp profile
add name="pppoe-profile" local-address=10.1.1.1 remote-address=pppoe-pool

/ppp secret
add name=user password=passwd service=pppoe profile=pppoe-profile

/interface pppoe-server server
add service-name=internet interface=wlan1 default-profile=pppoe-profile
```

PPPoE Operation

Stages

PPPoE has two stages:

- **Discovery stage** - a client discovers all available access concentrators and selects one of them to establish PPPoE session.This stage has four steps: *initialization, offer, request and session confirmation* . PPPoE Discovery uses special Ethernet frames with their own Ethernet frame type **0x8863**.



To initiate discovery, PPPoE client sends **PADI** frame to the broadcast Ethernet address (FF:FF:FF:FF:FF:FF) and may specify particular *service name*.

When server receives **PADI** frame, it responds with **PADO** frame to Client's unicast Ethernet address. There can be more than one server in broadcast range of the client. In such case client collects PADO frames and picks one (in most cases it picks the server which responded first) to start session.

Client sends **PADR** frame to unicast Ethernet address of the server it chose. If server agrees to set up a session with this particular client, it allocates resources to set up PPP session and assigns *Session ID* number. This number is sent back to client in **PADS** frame. When client receives PADS frame, it knows servers mac address and *Session ID*, it allocates resources and session can begin.

- **Session** - When discovery stage is completed, both peers know *PPPoE Session ID* and other peer's *Etehrnet (MAC) address* which together defines PPPoE session. PPP frames are encapsulated in PPPoE session frames, which have Ethernet frame type **0x8864**.
  When server sends confirmation and client receives it, PPP Session stage is started that consists of following steps:
  - **LCP negotiation**
  - **Authentication**
  - **IPCP negotiation** - client is assigned with an IP address.

PPPoE server sends Echo-Request packets to the client to determine the state of the session, otherwise server will not be able to determine that session is terminated in cases when client terminates session without sending

Terminate-Request packet.

More detailed description of PPPoE protocol can be found in RFC 2516

Used Packet Types

Packet	Description
PADI	**PPPoE Active Discovery Initialization** The PPPoE client sends out a PADI packet to the broadcast address. This packet can also populate the "service-name" field if a service name has been entered on the dial-up networking properties of the PPPoE broadband connectoid. If a service name has not been entered, this field is not populated
PADO	**PPPoE Active Discovery Offer** The PPPoE server, or Access Concentrator, should respond to the PADI with a PADO if the Access Concentrator is able to service the "service-name" field that had been listed in the PADI packet. If no "service-name" field had been listed, the Access Concentrator will respond with a PADO packet that has the "service-name" field populated with the service names that the Access Concentrator can service. The PADO packet is sent to the unicast address of the PPPoE client
PADR	**PPPoE Active Discovery Request** When a PADO packet is received, the PPPoE client responds with a PADR packet. This packet is sent to the unicast address of the Access Concentrator. The client may receive multiple PADO packets, but the client responds to the first valid PADO that the client received. If the initial PADI packet had a blank "service-name" field filed, the client populates the "service-name" field of the PADR packet with the first service name that had been returned in the PADO packet.
PADS	**PPPoE Active Discovery Session confirmation** When the PADR is received, the Access Concentrator generates a unique session identification (ID) for the Point-to-Point Protocol (PPP) session and returns this ID to the PPPoE client in the PADS packet. This packet is sent to the unicast address of the client.
PADT	**PPPoE Active Discovery Terminate** might be sent anytime after a session is established to indicate that a PPPoE session terminated. It can be sent by either server or client.

MTU

Typically largest Ethernet frame that can be transmitted without fragmentation is 1500 bytes. PPPoE adds another 6 bytes of overhead and PPP field adds two more bytes, leaving 1492 bytes for IP datagram. Therefore max PPPoE MRU and MTU values must not be larger than 1492.

TCP stacks try to avoid fragmentation, os they use an MSS (Maximum Segment Size). By default MSS is chosen as MTU of the outgoing interface minus the usual size of the TCP and IP headers (40 bytes), which results in 1460 bytes for an Eternet interface. Unfortunately there may be intermediate links with lower MTU which will cause fragmentation. In such case TCP stack performs path MTU discovery. Routers which cannot forward the datagram without fragmentation are supposed to drop packet and send ICMP-Fragmentation-Required to originating host. When host receives such ICMP, it tries lower MTU. This should work in ideal world, however in real world many routers do not generate fragmentation-required datagrams, also many firewalls drop all ICMP datagrams.

Workaround for this problem is to adjust MSS if it is too big. By default RouterOS adds mangle rules to intercept TCP SYN packets and silently adjust any advertised MSS option so they will be appropriate for the PPPoE link.

Additional information on maximum supported MTUs for routerboards are listed here.

PPPoE Client

Sub-menu: `/interface pppoe-client`

Properties

Property	Description
ac-name (*string*; Default: "")	Access Concentrator name, this may ne left blank and the client will connect to any access concentrator on the broadcast domain
add-default-route (*yes|no*; Default: **no**)	Enable/Disable whether to add default route automatically
allow (*mschap2|mschap1|chap|pap*; Default: **mschap2,mschap1,chap,pap**)	allowed authentication methods, by default all methods are allowed
dial-on-demand (*yes|no*; Default: **no**)	connects to AC only when outbound traffic is generated
interface (*string*; Default:)	interface name on which client will run
max-mru (*integer*; Default: **1460**)	Maximum Receive Unit
max-mtu (*integer*; Default: **1460**)	Maximum Transmission Unit
mrru (*integer: 512..65535|disabled*; Default: **disabled**)	maximum packet size that can be received on the link. If a packet is bigger than tunnel MTU, it will be split into multiple packets, allowing full size IP or Ethernet packets to be sent over the tunnel. `Read more >>`
name (*string*; Default: **pppoe-out[i]**)	name of the PPPoE interface, generated by ROuterOS if not specified
password (*string*; Default:)	password used to authenticate
profile (*string*; Default: **default**)	default profile for the connection defined in /ppp profiles
service-name (*string*; Default: "")	specifies the service name set on the access concentrator, can be left blank to connect to any PPPoE server
use-peer-dns (*yes|no*; Default: **no**)	enable/disable getting DNS settings from the peer
user (*string*; Default: "")	username used for authentication

Status

Command `/interface pppoe-client monitor` will display current PPPoE status.

Available read only properties:

Property	Description
ac-mac (*MAC address*)	MAC address of the access concentrator (AC) the client is connected to
ac-name (*string*)	name of the Access Concentrator
encoding (*string*)	encryption and encoding (if asymmetric, separated with '/') being used in this connection
mru (*integer*)	effective MRU of the link
mtu (*integer*)	effective MTU of the link
service-name (*string*)	used service name
status (*string*)	current link status. Available values are: • dialing, • verifying password..., • connected, • disconnected.
uptime (*time*)	connection time displayed in days, hours, minutes and seconds

Scanner

Starting from v3.21 RouterOS has new tool - PPPoE Scanner. It allows you to scan all active PPPoE servers in broadcast domain. Command to run scanner is as follows`/interface pppoe-client scan <interface>`

Available read only properties:

Property	Description
service (*string*)	Service name configured on server
mac-address (*MAC*)	Mac address of detected server
ac-name (*string*)	name of the Access Concentrator

Notes

Note for Windows. Some connection instructions may use the form where the "phone number", such as "MikroTik_AC\mt1", is specified to indicate that "MikroTik_AC" is the access concentrator name and "mt1" is the service name.

Specifying MRRU means enabling MP (Multilink PPP) over single link. This protocol is used to split big packets into smaller ones. Under Windows it can be enabled in Networking tag, Settings button, "Negotiate multi-link for single link connections". Their MRRU is hardcoded to 1614. This setting is usefull to overcome PathMTU discovery failures. The MP should be enabled on both peers.

Example

To add and enable PPPoE client on the ether1 interface connecting to the AC that provides testSN service using user name user with the password passwd:

```
[admin@RemoteOffice] interface pppoe-client> add interface=ether1 service-name=testSN user=user
 password=passwd disabled=no
[admin@RemoteOffice] interface pppoe-client> print
Flags: X - disabled, R - running
 0  R name="pppoe-out1" max-mtu=1480 max-mru=1480 mrru=disabled interface=ether1
      user="user" password="passwd" profile=default service-name="testSN"
      ac-name="" add-default-route=no dial-on-demand=no use-peer-dns=no
      allow=pap,chap,mschap1,mschap2

[admin@MikroTik] interface pppoe-client> monitor pppoe-out1
        status: "connected"
        uptime: 6s
     idle-time: 6s
      encoding: "MPPE128 stateless"
  service-name: "testSN"
       ac-name: "MikroTik"
        ac-mac: 00:0C:42:04:00:73
           mtu: 1480
           mru: 1480
```

Additional Resources

PPPoE Clients:

- RASPPPoE [1]for Windows 95, 98, 98SE, ME, NT4, 2000, XP, .NET

PPPoE Server Setup (Access Concentrator)

Sub-menu: `/interface pppoe-server server`

The PPPoE server (access concentrator) supports multiple servers for each interface - with differing service names. Currently the throughput of the PPPoE server has been tested to 160 Mb/s on a Celeron 600 CPU. Using higher speed CPUs, throughput should increase proportionately.

The access concentrator name and PPPoE service name are used by clients to identity the access concentrator to register with. The access concentrator name is the same as the identity of the router displayed before the command prompt. The identity may be set within the /system identity submenu.

Note that if no service name is specified in WindowsXP, it will use only service with no name. So if you want to serve WindowsXP clients, leave your service name empty.

Properties

Property	Description
authentication (*mschap2* | *mschap1* | *chap* | *pap*; Default: **"mschap2, mschap1, chap, pap"**)	Authentication algorithm
default-profile (*string*; Default: **"default"**)	Default user profile to use
interface (*string*; Default: **""**)	Interface, which the clients are connected to
keepalive-timeout (*time*; Default: **"10"**)	Defines the time period (in seconds) after which the router is starting to send keepalive packets every second. If no traffic and no keepalive responses came for that period of time (i.e. 2 * keepalive-timeout), not responding client is proclaimed disconnected.
max-mru (*integer*; Default: **"1480"**)	Maximum Receive Unit. The optimal value is the MTU of the interface the tunnel is working over decreased by 20 (so, for 1500-byte Ethernet link, set the MTU to 1480 to avoid fragmentation of packets)
max-mtu (*integer*; Default: **"1480"**)	Maximum Transmission Unit. The optimal value is the MTU of the interface the tunnel is working over decreased by 20 (so, for 1500-byte Ethernet link, set the MTU to 1480 to avoid fragmentation of packets)
max-sessions (*integer*; Default: **"0"**)	Maximum number of clients that the AC can serve. '0'- no limitations.
mrru (*integer: 512..65535* | *disabled*; Default: **"disabled"**)	Maximum packet size that can be received on the link. If a packet is bigger than tunnel MTU, it will be split into multiple packets, allowing full size IP or Ethernet packets to be sent over the tunnel. `Read more >>`
one-session-per-host (*yes* | *no*; Default: **"no"**)	Allow only one session per host (determined by MAC address). If a host will try to establish a new session, the old one will be closed
service-name (*string*; Default: **""**)	The PPPoE service name. Server will accept clients which sends PADI message with service-names that matches this setting or if service-name field in PADI message is not set.

Notes

The default keepalive-timeout value of 10 is OK in most cases. If you set it to 0, the router will not disconnect clients until they explicitly log out or the router is restarted. To resolve this problem, the one-session-per-host property can be used.

Security issue: do not assign an IP address to the interface you will be receiving the PPPoE requests on.

Specifying MRRU means enabling MP (Multilink PPP) over single link. This protocol is used to split big packets into smaller ones. Under Windows it can be enabled in Networking tag, Settings button, "Negotiate multi-link for single link connections". Their MRRU is hardcoded to 1614. This setting is usefull to overcome PathMTU discovery failures. The MP should be enabled on both peers.

Example

To add PPPoE server on ether1 interface providing ex service and allowing only one connection per host:

```
[admin@MikroTik] interface pppoe-server server> add interface=ether1 service-name=ex
  one-session-per-host=yes
[admin@MikroTik] interface pppoe-server server> print
Flags: X - disabled
  0 X service-name="ex" interface=ether1 mtu=1480 mru=1480 mrru=disabled
      authentication=mschap2,mschap,chap,pap keepalive-timeout=10
      one-session-per-host=yes max-sessions=0 default-profile=default
[admin@MikroTik] interface pppoe-server server>
```

PPPoE Server

Sub-menu: `/interface pppoe-server`

There are two types of interface (tunnel) items in PPTP server configuration - static users and dynamic connections. An interface is created for each tunnel established to the given server. Static interfaces are added administratively if there is a need to reference the particular interface name (in firewall rules or elsewhere) created for the particular user. Dynamic interfaces are added to this list automatically whenever a user is connected and its username does not match any existing static entry (or in case the entry is active already, as there can not be two separate tunnel interfaces referenced by the same name). Dynamic interfaces appear when a user connects and disappear once the user disconnects, so it is impossible to reference the tunnel created for that use in router configuration (for example, in firewall), so if you need a persistent rules for that user, create a static entry for him/her. Otherwise it is safe to use dynamic configuration. Note that in both cases PPP users must be configured properly - static entries do not replace PPP configuration.

Property Description

- encoding (read-only: text) - encryption and encoding (if asymmetric, separated with '/') being used in this connection
- mru (read-only: integer) - client's MRU
- mtu (read-only: integer) - client's MTU
- name (name) - interface name
- remote-address (read-only: MAC address) - MAC address of the connected client
- service (name) - name of the service the user is connected to
- uptime (read-only: time) - shows how long the client is connected
- user (name) - the name of the connected user (must be present in the user darabase anyway)

Example

To view the currently connected users:

```
[admin@MikroTik] interface pppoe-server> print
Flags: X - disabled, D - dynamic, R - running
 #     NAME       USER      SERVICE   REMOTE... ENCODING  UPTIME
 0  DR <pppoe-ex> user      ex        00:0C:... MPPE12... 40m45s
[admin@MikroTik] interface pppoe-server>
```

To disconnect the user ex:

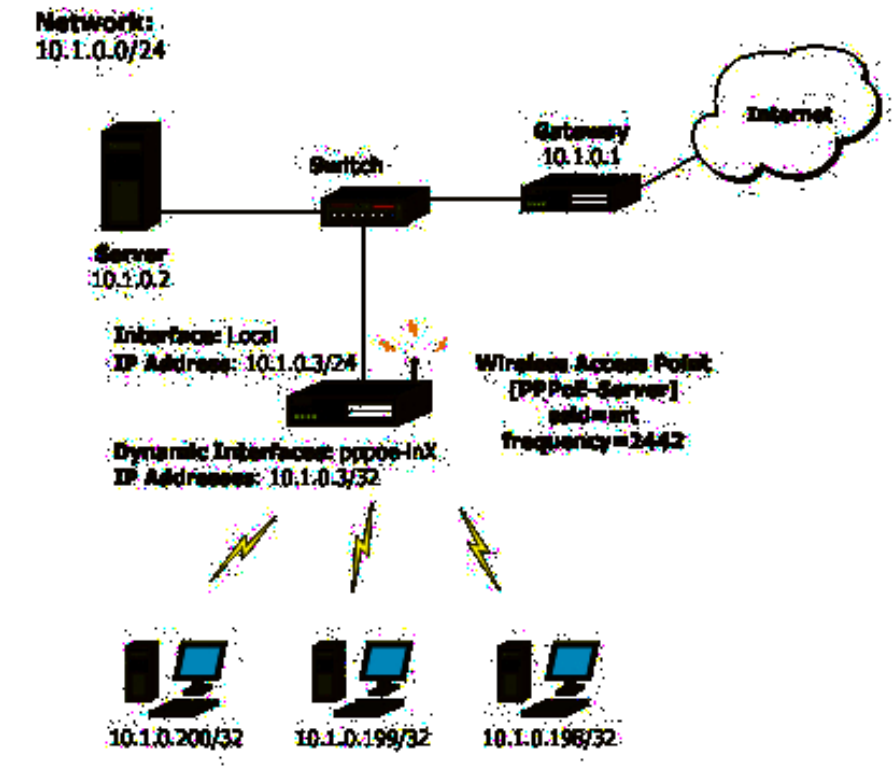
```
[admin@MikroTik] interface pppoe-server> remove [find user=ex]
[admin@MikroTik] interface pppoe-server> print
[admin@MikroTik] interface pppoe-server>
```

Application Examples

PPPoE in a multipoint wireless 802.11g network

In a wireless network, the PPPoE server may be attached to an Access Point (as well as to a regular station of wireless infrastructure). Either our RouterOS client or Windows PPPoE clients may connect to the Access Point for PPPoE authentication. Further, for RouterOS clients, the radio interface may be set to MTU 1600 so that the PPPoE interface may be set to MTU 1500. This optimizes the transmission of 1500 byte packets and avoids any problems associated with MTUs lower than 1500. It has not been determined how to change the MTU of the Windows wireless interface at this moment.

Let us consider the following setup where the MikroTik Wireless AP offers wireless clients transparent access to the local network with authentication:



First of all, the wireless interface should be configured:

```
[admin@PPPoE-Server] interface wireless> set 0 mode=ap-bridge \
   frequency=2442 band=2.4ghz-b/g ssid=mt disabled=no
[admin@PPPoE-Server] interface wireless> print
Flags: X - disabled, R - running
 0 X  name="wlan1" mtu=1500 mac-address=00:0C:42:18:5C:3D arp=enabled
      interface-type=Atheros AR5413 mode=ap-bridge ssid="mt" frequency=2442
      band=2.4ghz-b/g scan-list=default antenna-mode=ant-a wds-mode=disabled
      wds-default-bridge=none wds-ignore-ssid=no default-authentication=yes
      default-forwarding=yes default-ap-tx-limit=0 default-client-tx-limit=0
      hide-ssid=no security-profile=default compression=no
[admin@PPPoE-Server] interface wireless>
```

Now, configure the Ethernet interface, add the IP address and set the default route:

```
[admin@PPPoE-Server] ip address> add address=10.1.0.3/24 interface=Local
[admin@PPPoE-Server] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   10.1.0.3/24        10.1.0.0        10.1.0.255      Local
[admin@PPPoE-Server] ip address> /ip route
[admin@PPPoE-Server] ip route> add gateway=10.1.0.1
[admin@PPPoE-Server] ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE INTER...
 0 ADC  10.1.0.0/24        10.1.0.3                          0        Local
 1 A S  0.0.0.0/0                          r 10.1.0.1        1        Local
[admin@PPPoE-Server] ip route> /interface ethernet
[admin@PPPoE-Server] interface ethernet> set Local arp=proxy-arp
[admin@PPPoE-Server] interface ethernet> print
Flags: X - disabled, R - running
 #    NAME                                  MTU   MAC-ADDRESS       ARP
 0  R Local                                 1500  00:0C:42:03:25:53 proxy-arp
[admin@PPPoE-Server] interface ethernet>
```

We should add PPPoE server to the wireless interface:

```
[admin@PPPoE-Server] interface pppoe-server server> add interface=wlan1 \
   service-name=mt one-session-per-host=yes disabled=no
[admin@PPPoE-Server] interface pppoe-server server> print
Flags: X - disabled
 0   service-name="mt" interface=wlan1 max-mtu=1480 max-mru=1480 mrru=disabled
     authentication=pap,chap,mschap1,mschap2 keepalive-timeout=10
     one-session-per-host=yes max-sessions=0 default-profile=default
[admin@PPPoE-Server] interface pppoe-server server>
```

Finally, we can set up PPPoE clients:

```
[admin@PPPoE-Server] ip pool> add name=pppoe ranges=10.1.0.100-10.1.0.200
[admin@PPPoE-Server] ip pool> print
 # NAME                                      RANGES
 0 pppoe                                     10.1.0.100-10.1.0.200
[admin@PPPoE-Server] ip pool> /ppp profile
[admin@PPPoE-Server] ppp profile> set default use-encryption=yes \
   local-address=10.1.0.3 remote-address=pppoe
[admin@PPPoE-Server] ppp profile> print
Flags: * - default
 0 * name="default" local-address=10.1.0.3 remote-address=pppoe
     use-compression=no use-vj-compression=no use-encryption=yes only-one=no
     change-tcp-mss=yes

 1 * name="default-encryption" use-compression=default
     use-vj-compression=default use-encryption=yes only-one=default
     change-tcp-mss=default
[admin@PPPoE-Server] ppp profile> .. secret
[admin@PPPoE-Server] ppp secret> add name=w password=wkst service=pppoe
[admin@PPPoE-Server] ppp secret> add name=l password=ltp service=pppoe
[admin@PPPoE-Server] ppp secret> print
Flags: X - disabled
 #   NAME        SERVICE CALLER-ID PASSWORD  PROFILE            REMOTE-ADDRESS
 0   w           pppoe             wkst      default            0.0.0.0
 1   l           pppoe             ltp       default            0.0.0.0
[admin@PPPoE-Server] ppp secret>
```

Thus we have completed the configuration and added two users: w and l who are able to connect to Internet, using PPPoE client software.

Note that Windows XP built-in client supports encryption, but RASPPPOE does not. So, if it is planned not to support Windows clients older than Windows XP, it is recommended not to require encryption. In other case, the server will accept clients that do not encrypt data.

Troubleshooting

- **I can connect to my PPPoE server. The ping goes even through it, but I still cannot open web pages**

Make sure that you have specified a valid DNS server in the router (in /ip dns or in /ppp profile the dns-server parameter).

- **The PPPoE server shows more than one active user entry for one client, when the clients disconnect, they are still shown and active**

Set the `keepalive-timeout` parameter (in the PPPoE server configuration) to 10 if You want clients to be considered logged off if they do not respond for 10 seconds.

Note that if the `keepalive-timeout` parameter is set to 0 and the `only-one` parameter (in PPP profile settings) is set to yes then the clients might be able to connect only once. To resolve this problem `one-session-per-host` parameter in PPPoE server configuration should be set to yes

- **My Windows XP client cannot connect to the PPPoE server**

You have to specify the "Service Name" in the properties of the XP PPPoE client. If the service name is not set, or it does not match the service name of the MikroTik PPPoE server, you get the "line is busy" errors, or the system

shows "verifying password - unknown error"

- **I want to have logs for PPPoE connection establishment**

Configure the logging feature under the /system logging facility and enable the PPP type logs. Read more >>

[Top | Back to Content]

References

[1] http://www.raspppoe.com/

Manual:Interface/PPTP

Applies to RouterOS: v3, v4, v5+

Summary

Standards: `RFC 2637`

PPTP is a secure tunnel for transporting IP traffic using PPP. PPTP encapsulates PPP in virtual lines that run over IP. PPTP incorporates PPP and MPPE (Microsoft Point to Point Encryption) to make encrypted links. The purpose of this protocol is to make well-managed secure connections between routers as well as between routers and PPTP clients (clients are available for and/or included in almost all OSs including Windows).

Multilink PPP (MP) is supported in order to provide MRRU (the ability to transmit full-sized 1500 and larger packets) and bridging over PPP links (using Bridge Control Protocol (BCP) that allows to send raw Ethernet frames over PPP links). This way it is possible to setup bridging without EoIP. The bridge should either have an administratively set MAC address or an Ethernet-like interface in it, as PPP links do not have MAC addresses.

PPTP includes PPP authentication and accounting for each PPTP connection. Full authentication and accounting of each connection may be done through a RADIUS client or locally.

MPPE 40bit RC4 and MPPE 128bit RC4 encryption are supported.

PPTP traffic uses TCP port 1723 and IP protocol GRE (Generic Routing Encapsulation, IP protocol ID 47), as assigned by the Internet Assigned Numbers Authority (IANA). PPTP can be used with most firewalls and routers by enabling traffic destined for TCP port 1723 and protocol 47 traffic to be routed through the firewall or router.

PPTP connections may be limited or impossible to setup though a masqueraded/NAT IP connection. Please see the Microsoft and RFC links listed below for more information.

PPTP Client

Sub-menu: `/interface pptp-client`

Properties

Property	Description
add-default-route (*yes* | *no*; Default: **no**)	Whether to add PPTP remote address as a default route.
allow (*mschap2* | *mschap1* | *chap* | *pap*; Default: **mschap2, mschap1, chap, pap**)	Allowed authentication methods.
connect-to (*IP*; Default:)	Remote address of PPTP server
dial-on-demand (*yes* | *no*; Default: **no**)	
disabled (*yes* | *no*; Default: **yes**)	Whether interface is disabled or not. By default it is disabled
max-mru (*integer*; Default: **1460**)	Maximum Receive Unit. Max packet size that PPTP interface will be able to receive without packet fragmentation.
max-mtu (*integer*; Default: **1460**)	Maximum Transmission Unit. Max packet size that PPTP interface will be able to send without packet fragmentation.
mrru (*disabled* | *integer*; Default: **disabled**)	Maximum packet size that can be received on the link. If a packet is bigger than tunnel MTU, it will be split into multiple packets, allowing full size IP or Ethernet packets to be sent over the tunnel. Read more >>
name (*string*; Default:)	Descriptive name of the interface.
password (*string*; Default: "")	Password used for authentication.
profile (*name*; Default: **default-encryption**)	Used PPP profile.
user (*string*; Default:)	User name used for authentication.

Quick example

This example demonstrates how to set up PPTP client with username "pptp-hm", password "123" and server 10.1.101.100

```
[admin@dzeltenais_burkaans] /interface pptp-client>add name=pptp-hm user=pptp-hm password=123 \
\... connect-to=10.1.101.100 disabled=no
[admin@dzeltenais_burkaans] /interface pptp-client> print detail
Flags: X - disabled, R - running
 0    name="pptp-hm" max-mtu=1460 max-mru=1460 mrru=disabled
      connect-to=10.1.101.100 user="pptp-hm" password="123"
      profile=default-encryption add-default-route=no dial-on-demand=no
      allow=pap,chap,mschap1,mschap2
```

PPTP Server

Sub-menu: /interface pptp-server

This sub-menu shows interfaces for each connected PPTP clients.

An interface is created for each tunnel established to the given server. There are two types of interfaces in PPTP server's configuration

- Static interfaces are added administratively if there is a need to reference the particular interface name (in firewall rules or elsewhere) created for the particular user.
- Dynamic interfaces are added to this list automatically whenever a user is connected and its username does not match any existing static entry (or in case the entry is active already, as there can not be two separate tunnel interfaces referenced by the same name).

Dynamic interfaces appear when a user connects and disappear once the user disconnects, so it is impossible to reference the tunnel created for that use in router configuration (for example, in firewall), so if you need a persistent

rules for that user, create a static entry for him/her. Otherwise it is safe to use dynamic configuration.

Note: in both cases PPP users must be configured properly - static entries do not replace PPP configuration.

Server configuration

Sub-menu: `/interface pptp-server server`

Properties:

Property	Description
`authentication` (*pap* | *chap* | *mschap1* | *mschap2*; Default: **mschap1,mschap2**)	Authentication methods that server will accept.
`default-profile` (*name*; Default: **default-encryption**)	
`enabled` (*yes* | *no*; Default: **no**)	Defines whether PPTP server is enabled or not.
`keepalive-timeout` (*time*; Default: **30**)	Defines the time period (in seconds) after which the router is starting to send keepalive packets every second. If no traffic and no keepalive responses has came for that period of time (i.e. 2 * keepalive-timeout), not responding client is proclaimed disconnected
`max-mru` (*integer*; Default: **1460**)	Maximum Receive Unit. Max packet size that PPTP interface will be able to receive without packet fragmentation.
`max-mtu` (*integer*; Default: **1460**)	Maximum Transmission Unit. Max packet size that PPTP interface will be able to send without packet fragmentation.
`mrru` (*disabled* | *integer*; Default: **disabled**)	Maximum packet size that can be received on the link. If a packet is bigger than tunnel MTU, it will be split into multiple packets, allowing full size IP or Ethernet packets to be sent over the tunnel. `Read more >>`

To enable PPTP server:

```
[admin@MikroTik] interface pptp-server server> set enabled=yes
[admin@MikroTik] interface pptp-server server> print
            enabled: yes
            max-mtu: 1460
            max-mru: 1460
               mrru: disabled
     authentication: mschap2,mschap1
  keepalive-timeout: 30
    default-profile: default
[admin@MikroTik] interface pptp-server server>
```

Monitoring

Monitor command can be used to monitor status of the tunnel on both client and server.

```
[admin@dzeltenais_burkaans] /interface pptp-client> monitor 0
     status: "connected"
     uptime: 7h24m18s
  idle-time: 6h21m4s
   encoding: "MPPE128 stateless"
        mtu: 1460
        mru: 1460
```

Read-only properties

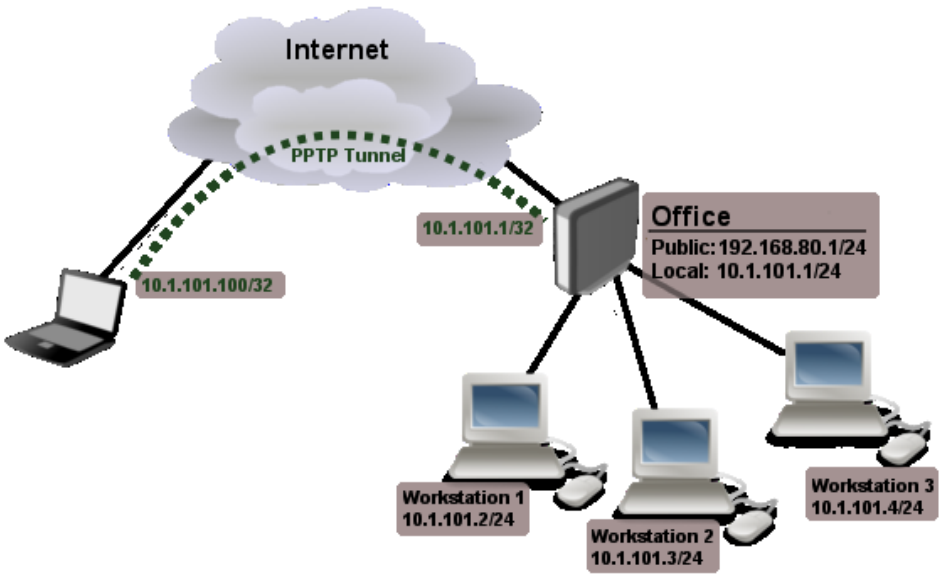
Property	Description
status ()	Current PPTP status. Value other than "connected" indicates that there are some problems estabising tunnel.
uptime (*time*)	Elapsed time since tunnel was established.
idle-time (*time*)	Elapsed time since last activity on the tunnel.
encoding ()	Used encryption method
mtu (*integer*)	Negotiated and used MTU
mru (*integer*)	Negotiated and used MRU

Application Examples

Connecting Remote Client

The following example shows how to connect a computer to a remote office network over PPTP encrypted tunnel giving that computer an IP address from the same network as the remote office has (without need of bridging over EoIP tunnels)

Consider following setup



Office router is connected to internet through **ether1**. Workstations are connected to **ether2**. Laptop is connected to the internet and can reach Office router's public IP (in our example it is 192.168.80.1).

First step is to create a user

```
[admin@RemoteOffice] /ppp secret> add name=Laptop service=pptp password=123
local-address=10.1.101.1 remote-address=10.1.101.100
[admin@RemoteOffice] /ppp secret> print detail
Flags: X - disabled
  0   name="Laptop" service=pptp caller-id="" password="123" profile=default
      local-address=10.1.101.1 remote-address=10.1.101.100 routes==""

[admin@RemoteOffice] /ppp secret>
```

Notice that pptp local address is the same as routers address on local interface and remote address is form the same range as local network (10.1.101.0/24).

Next step is to enable pptp server and pptp client on the laptop.

```
[admin@RemoteOffice] /interface pptp-server server> set enabled=yes
[admin@RemoteOffice] /interface pptp-server server> print
            enabled: yes
            max-mtu: 1460
            max-mru: 1460
               mrru: disabled
     authentication: mschap2
  keepalive-timeout: 30
    default-profile: default
[admin@RemoteOffice] /interface pptp-server server>
```

PPTP client from the laptop should connect to routers public IP which in our example is 192.168.80.1.
Please, consult the respective manual on how to set up a PPTP client with the software You are using.

At this point (when pptp client is successfully connected) if you will try to ping any workstation form the laptop, ping will time out, because Laptop is unable to get ARPs from workstations. Solution is to set up `proxy-arp` on local interface

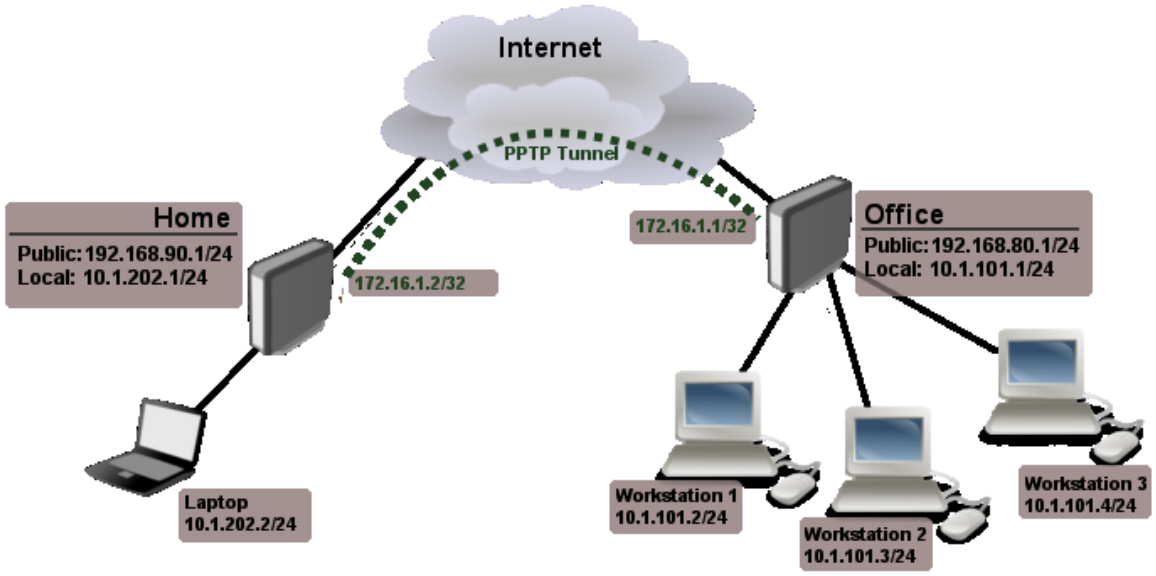
```
[admin@RemoteOffice] /interface ethernet> set Office arp=proxy-arp
[admin@RemoteOffice] /interface ethernet> print
Flags: X - disabled, R - running
  #    NAME                MTU   MAC-ADDRESS         ARP
  0  R ether1              1500  00:30:4F:0B:7B:C1 enabled
  1  R ether2              1500  00:30:4F:06:62:12 proxy-arp
[admin@RemoteOffice] interface ethernet>
```

After `proxy-arp` is enabled client can successfully reach all workstations in local network behind the router.

Site-to-Site PPTP

The following is an example of connecting two Intranets using PPTP tunnel over the Internet.

Consider following setup



Office and Home routers are connected to internet through **ether1**, workstations and laptops are connected to **ether2**. Both local networks are routed through pptp client, thus they are not in the same broadcast domain. If both networks should be in the same broadcast domain then you need to use BCP and bridge pptp tunnel with local interface.

First step is to create a user

```
[admin@RemoteOffice] /ppp secret> add name=Home service=pptp password=123
local-address=172.16.1.1 remote-address=172.16.1.2 routes="10.1.202.0/24 172.16.1.2 1"
[admin@RemoteOffice] /ppp secret> print detail
Flags: X - disabled
  0   name="Home" service=pptp caller-id="" password="123" profile=default
      local-address=172.16.1.1 remote-address=172.16.1.2 routes=="10.1.201.0/24 172.16.1.1 1"

[admin@RemoteOffice] /ppp secret>
```

Notice that we set up pptp to add route whenever client connects. If this option is not set, then you will need static routing configuration on the server to route traffic between sites through pptp tunnel.

Next step is to enable pptp server on the office router and configure pptp client on the Home router.

```
[admin@RemoteOffice] /interface pptp-server server> set enabled=yes
[admin@RemoteOffice] /interface pptp-server server> print
            enabled: yes
            max-mtu: 1460
            max-mru: 1460
               mrru: disabled
     authentication: mschap2
  keepalive-timeout: 30
    default-profile: default
[admin@RemoteOffice] /interface pptp-server server>
```

```
[admin@Home] /interface pptp-client> add user=Home password=123 connect-to=192.168.80.1 disabled=no
[admin@Home] /interface pptp-client> print
Flags: X - disabled, R - running
 0    name="pptp-out1" max-mtu=1460 max-mru=1460 mrru=disabled connect-to=192.168.80.1 user="Home"
       password="123" profile=default-encryption add-default-route=no dial-on-demand=no
       allow=pap,chap,mschap1,mschap2
[admin@Home] /interface pptp-client>
```

Now we need to add route to reach local network behind Home router

```
[admin@RemoteOffice] /ip route> add dst-address=10.1.202.0/24 gateway=172.16.1.2
```

Now after tunnel is established and routes are set, you should be able to ping remote network.

Read More

- BCP (Bridge Control Protocol)
- http://msdn.microsoft.com/library/backgrnd/html/understanding_pptp.htm
- http://support.microsoft.com/support/kb/articles/q162/8/47.asp
- http://www.ietf.org/rfc/rfc2637.txt?number=2637
- http://www.ietf.org/rfc/rfc3078.txt?number=3078
- http://www.ietf.org/rfc/rfc3079.txt?number=3079

[Top | Back to Content]

Manual:IP/Proxy

Applies to RouterOS: v3, v4

Summary

Sub-menu: `/ip proxy`
Standards: `RFC 1945, RFC 2616`

MikroTik RouterOS performs proxying of HTTP and HTTP-proxy (for FTP, HTTP and HTTPS protocols) requests. Proxy server performs Internet object cache function by storing requested Internet objects, i.e., data available via HTTP and FTP protocols on a system positioned closer to the recipient in the form of speeding up customer browsing by delivering them requested file copies from proxy cache at local network speed. MikroTik RouterOS implements the following proxy server features:

- Regular HTTP proxy – customer (itself) specify what is proxy server for him
- Transparent proxy – customer does not know about the proxy being enabled and there isn't need any additional configuration for web browser of client.
- Access list by source, destination, URL and requested method (HTTP firewall)
- Cache access list to specify which objects to cache, and which not.
- Direct Access List – to specify which resources should be accessed directly, and which - through another proxy server
- Logging facility – allows to get and to store information about proxy operation

- Parent proxy support – allows to specify other proxy server, ***('if they don't have the requested object ask their parents, or to the original server.)***

A proxy server usually is placed at various points between users and the destination server (*also known as origin server*) on the Internet. *(see Figure 10.1).*

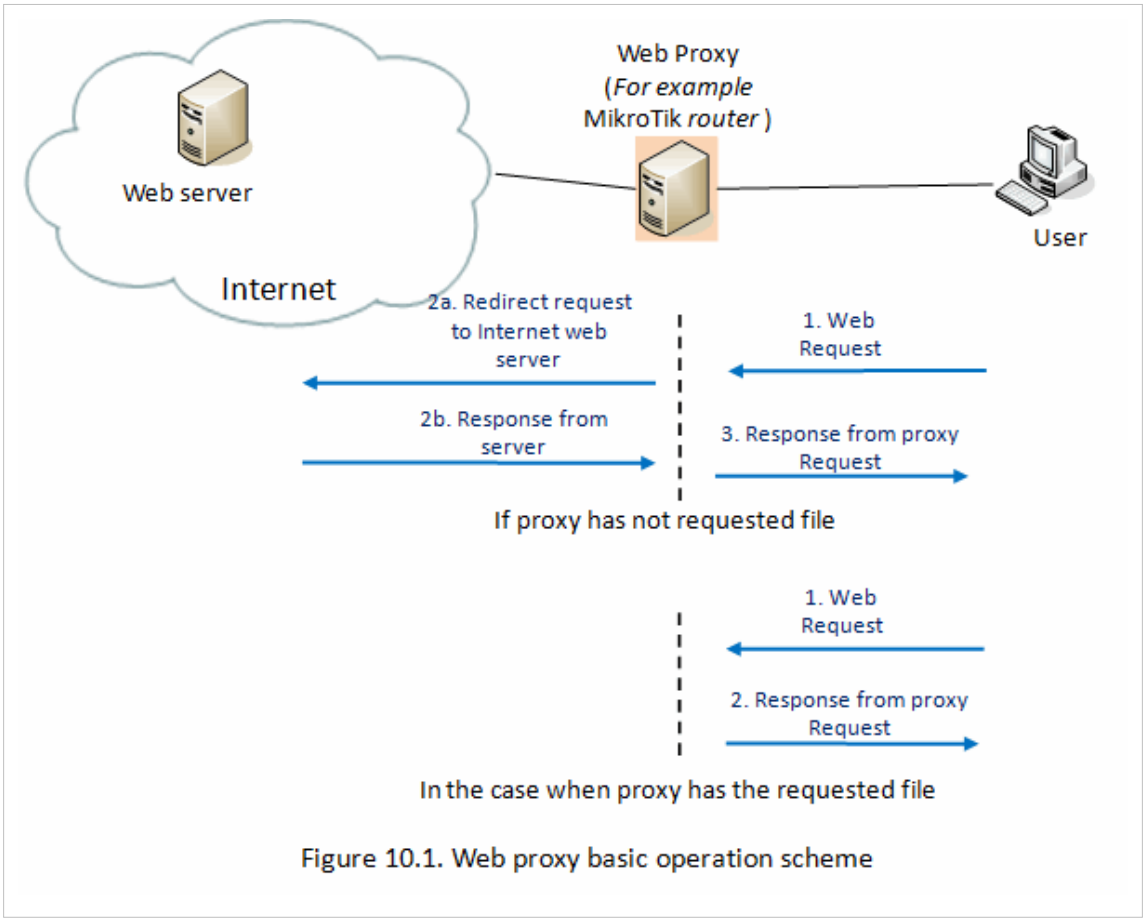


Figure 10.1. Web proxy basic operation scheme

A *Web proxy (cache)* watches requests coming from client, saving copies of the responses for itself. Then, if there is another request for the same URL, it can use the response that it has, instead of asking the origin server for it again. If proxy has not requested file, it downloads that from the original server.

There can be many potential purpose of proxy server:

- To decrease access speed to resources (it takes less time for the client to get the object).
- Works as HTTP firewall (deny access to undesirable web pages),

Allows to filter web content (by specific parameters, like source address, destination address and port, URL, HTTP request method) scan outbound content, e.g., for data leak protection.

Note: it may be useful to have Web proxy running even with no cache when you want to use it only as something like HTTP and FTP firewall (for example, denying access undesired web pages or deny specific type of files e.g. .mp3 files) or to redirect requests to external proxy (possibly, to a proxy with caching functions) transparently.

Proxy configuration example

In MikroTik RouterOS proxy configuration is performed in **/ip proxy** menu. See below how to enable the proxy on port 8080 and set up 195.10.10.1 as proxy source address:

```
[admin@MikroTik] ip proxy> set enabled=yes port=8080 src-address=195.10.10.1

[admin@MikroTik] ip proxy> print
                    enabled: yes
                src-address: 195.10.10.1
                       port: 8080
               parent-proxy: 0.0.0.0:0
                cache-drive: system
         cache-administrator: "admin@mikrotik.com"
         max-disk-cache-size: none
          max-ram-cache-size: 100000KiB
          cache-only-on-disk: yes
  maximal-client-connections: 1000
  maximal-server-connections: 1000
             max-fresh-time: 3d
```

When setting up regular proxy service, make sure it serves only your clients and prevent unauthorised access to it by creating firewall that allow only your clients to use proxy, otherwise it may be used as an open proxy.

Remember that regular proxy require also client's web browser configuration.

For example:

Explorer 8.x	**Firefox 3.x**	**Opera 10.x**
Select *Tools>Internet options.*	Select *Tools>Options.*	Select *Tool>Preferences.*
Click the *Connections tab*.	Click the *Advanced tab.*	Open the *Advanced tab/Network.*
Select the necessary connection and choose *Settings* button.	Open the *Network tab.*	Click the *Proxy servers.*
Configure proxy address and port.	Click the *Connection/Settings*	*Enter proxy address and port.*
	Select *Manual proxy configuration'*	

Transparent proxy configuration example

RouterOS can also act as a Transparent Caching server, with no configuration required in the customer's web browser. Transparent proxy does not modify requested URL or response. RouterOS will take all HTTP requests and redirect them to the local proxy service. This process will be entirely transparent to the user (users may not know anything about proxy server that is located between them and original server), and the only difference to them will be the increased browsing speed.

To enable the transparent mode, firewall rule in destination NAT has to be added, specifying which connections (to which ports) should be transparently redirected to the proxy. Check proxy settings above and redirect us users (192.168.1.0/24) to proxy server.

```
[admin@MikroTik] ip firewall nat> add chain=dstnat protocol=tcp src-address=192.168.1.0/24 \
dst-port=80 action=redirect to-ports=8080

 [admin@MikroTik] ip firewall nat> print
 Flags: X - disabled, I - invalid, D - dynamic
```

```
 0   chain=dstnat protocol=tcp dst-port=80 action=redirect to-ports=8000
[admin@MikroTik] ip firewall nat>
```

The web proxy can be used as transparent and normal web proxy at the same time. In transparent mode it is possible to use it as standard web proxy, too. However, in this case, proxy users may have trouble to reach web pages which are accessed transparently.

Proxy based firewall – Access List

Access list is implemented in the same way as MikroTik firewall rules processed from the top to the bottom. First matching rule specifies decision of what to do with this connection. Connections can be matched by its source address, destination address, destination port, sub-string of requested URL (Uniform Resource Locator) or request method. If none of these parameters is specified, every connection will match this rule.

If connection is matched by a rule, action property of this rule specifies whether connection will be allowed or not (deny). If connection does not match any rule, it will be allowed.

In this example assume that we have configured transparent proxy server as given in example above.

Block particular Websites.

```
/ip proxy access add dst-host=www.facebook.com action=deny
```

It will block website http://www.facebook.com [1], we can always block the same for different networks by giving src-address.

```
/ip proxy access add src-address=192.168.1.0/24 dst-host=www.facebook.com action=deny
```

Users from network 192.168.1.0/24 will not be able to access website www.facebook.com [1].

You can block also websites that contain specific words in URL:

```
/ip proxy access add dst-host=:mail action=deny
```

This statement will block all websites which contain word “mail” in URL. Like www.mail.com [2], www.hotmail.com [3], mail.yahoo.com etc.

We can also stop downloading specific types of files like .flv, .avi, .mp4, .mp3, .exe, .dat, …etc.

```
/ip proxy access
add path=*.flv action=deny
add path=*.avi action=deny
add path=*.mp4 action=deny
add path=*.mp3 action=deny
add path=*.zip action=deny
add path=*.rar action=deny.
```

Here are available also different wildcard characters, to creating specific conditions and to match it by proxy access list.

Wildcard properties (dst-host and dst-path) match a complete string (i.e., they will not match "example.com" if they are set to "example"). Available wildcards are '*' (match any number of any characters) and '?' (match any one character).

Regular expressions are also accepted here, but if the property should be treated as a regular expression, it should start with a colon (':').

To show that no symbols are allowed before the given pattern, we use ^ symbol at the beginning of the pattern.

To specify that no symbols are allowed after the given pattern, we use $ symbol at the end of the pattern.

Reference

List of all available parameters and commands per menu.

General

Sub-menu: `/ip proxy`

Property	Description
always-from-cache (*yes* | *no*; Default: **no**)	
cache-administrator (*string*; Default: **webmaster**)	Administrator's e-mail displayed on proxy error page
cache-hit-dscp (*integer: 0..63*; Default: **4**)	
cache-on-disk (*yes* | *no*; Default: **no**)	
max-cache-size (*none* | *unlimited* | *integer: 0..4294967295*; Default: **none**)	Specifies the maximal cache size, measured in kibibytes
max-client-connections (*integer: 1..5000*; Default: **600**)	Maximal number of connections accepted from clients (any further connections will be rejected)
max-fresh-time (*time*; Default: **3d**)	Maximal time to store a cached object. The validity period of an object is is usually defined by the object itself, but in case it is set too high, you can override the maximal value
max-server-connections (*integer: 1..5000*; Default: **600**)	Maximal number of connections made to servers (any further connections from clients will be put on hold until some server connections will terminate)
parent-proxy (*Ip4* | *ip6*; Default: **0.0.0.0**)	IP address and port of another HTTP proxy to redirect all requests to. If set to **0.0.0.0** parent proxy is not used.
parent-proxy-port (*integer: 0..65535*; Default: **0**)	Port that parent proxy is listening on.
port (*integer: 0..65535*; Default: **8080**)	TCP port the proxy server will be listening on. This port have to be specified on all clients that want to use the server as HTTP proxy. Transparent (with zero configuration for clients) proxy setup can be made by redirecting HTTP requests to this port in IP firewall using destination NAT feature
serialize-connections (*yes* | *no*; Default: **no**)	
src-address (*Ip4* | *Ip6*; Default: **0.0.0.0**)	Proxy will use specified address when connecting to parent proxy or web site. If set to **0.0.0.0** then appropriate IP address will be taken from routing table.

Access List

Sub-menu: `/ip proxy access`

Access list is configured like a regular firewall rules. Rules are processed from the top to the bottom. First matching rule specifies decision of what to do with this connection. There is a total of 6 classifiers that specify matching constraints. If none of these classifiers is specified, the particular rule will match every connection.

If connection is matched by a rule, action property of this rule specifies whether connection will be allowed or not. If the particular connection does not match any rule, it will be allowed.

Property	Description
action (*allow* | *deny*; Default: **allow**)	Specifies whether to pass or deny matched packets
dst-address (*Ip4[-Ip4 | /0..32] | Ip6/0..128*; Default:)	Destination address of the target server.
dst-host (*string*; Default:)	IP address or DNS name used to make connection the target server (this is the string user wrote in browser before specifying port and path to a particular web page
dst-port (*integer[-integer[,integer[,...]]]: 0..65535*; Default:)	List or range of ports the packet is destined to
local-port (*integer: 0..65535*; Default:)	Specifies the port of the web proxy via which the packet was received. This value should match one of the ports web proxy is listening on.
method (*any* | *connect* | *delete* | *get* | *head* | *options* | *post* | *put* | *trace*; Default:)	HTTP method used in the request (see HTTP Methods section in the end of this document)
path (*string*; Default:)	Name of the requested page within the target server (i.e. the name of a particular web page or document without the name of the server it resides on)
redirect-to (*string*; Default:)	In case access is denied by this rule, the user shall be redirected to the URL specified here
src-address (*Ip4[-Ip4 | /0..32] | Ip6/0..128*; Default:)	Source address of the connection originator.

Read only properties:

Property	Description
hits (*integer*)	Count of requests that were matched by this rule

Wildcard properties (dst-host and dst-path) match a complete string (i.e., they will not match "example.com" if they are set to "example"). Available wildcards are '*' (match any number of any characters) and '?' (match any one character). Regular expressions are also accepted here, but if the property should be treated as a regular expression, it should start with a colon (':').

Small hints in using regular expressions:

- \\ symbol sequence is used to enter \ character in console
- \. pattern means . only (in regular expressions single dot in pattern means any symbol)
- to show that no symbols are allowed before the given pattern, we use ^ symbol at the beginning of the pattern
- to specify that no symbols are allowed after the given pattern, we use $ symbol at the end of the pattern
- to enter [or] symbols, you should escape them with backslash \.

It is strongly recommended to deny all IP addresses except those behind the router as the proxy still may be used to access your internal-use-only (intranet) web servers. Also, consult examples in Firewall Manual on how to protect your router.

Direct Access

Sub-menu: `/ip proxy direct`

If **parent-proxy** property is specified, it is possible to tell proxy server whether to try to pass the request to the parent proxy or to resolve it connecting to the requested server directly. Direct Access List is managed just like Proxy Access List described in the previous chapter except the action argument.

Unlike the access list, the direct proxy access list has default action equal to deny. It takes place when no rules are specified or a particular request did not match any rule.

Property	Description
action (*allow* | *deny*; Default: **allow**)	Specifies the action to perform on matched packets: • `allow` - always resolve matched requests directly bypassing the parent router • `deny` - resolve matched requests through the parent proxy. If no one is specified this has the same effect as **allow**.
dst-address (*Ip4[-Ip4 | /0..32] | Ip6/0..128*; Default:)	Destination address of the target server.
dst-host (*string*; Default:)	IP address or DNS name used to make connection the target server (this is the string user wrote in browser before specifying port and path to a particular web page
dst-port (*integer[-integer[,integer[,...]]]: 0..65535*; Default:)	List or range of ports used by connection to target server.
local-port (*integer: 0..65535*; Default:)	Specifies the port of the web proxy via which the packet was received. This value should match one of the ports web proxy is listening on.
method (*any* | *connect* | *delete* | *get* | *head* | *options* | *post* | *put* | *trace*; Default:)	HTTP method used in the request (see HTTP Methods section in the end of this document)
path (*string*; Default:)	Name of the requested page within the target server (i.e. the name of a particular web page or document without the name of the server it resides on)
src-address (*Ip4[-Ip4 | /0..32] | Ip6/0..128*; Default:)	Source address of the connection originator.

Read only properties:

Property	Description
hits (*integer*)	Count of requests that were matched by this rule

Cache Management

Sub-menu: `/ip proxy cache`

Cache access list specifies, which requests (domains, servers, pages) have to be cached locally by web proxy, and which not. This list is implemented exactly the same way as web proxy access list. Default action is to cache object (if no matching rule is found).

Property	Description
action (*allow* | *deny*; Default: **allow**)	Specifies the action to perform on matched packets: • `allow` - cache objects from matched request • `deny` - do not cache objects from matched request
dst-address (*Ip4[-Ip4 | /0..32] | Ip6/0..128*; Default:)	Destination address of the target server
dst-host (*string*; Default:)	IP address or DNS name used to make connection the target server (this is the string user wrote in browser before specifying port and path to a particular web page
dst-port (*integer[-integer[,integer[,...]]]: 0..65535*; Default:)	List or range of ports the packet is destined to.
local-port (*integer: 0..65535*; Default:)	Specifies the port of the web proxy via which the packet was received. This value should match one of the ports web proxy is listening on.
method (*any* | *connect* | *delete* | *get* | *head* | *options* | *post* | *put* | *trace*; Default:)	HTTP method used in the request (see HTTP Methods section in the end of this document)
path (*string*; Default:)	Name of the requested page within the target server (i.e. the name of a particular web page or document without the name of the server it resides on)
src-address (*Ip4[-Ip4 | /0..32] | Ip6/0..128*; Default:)	Source address of the connection originator

Read only properties:

Property	Description
hits (*integer*)	Count of requests that were matched by this rule

Connections

Sub-menu: `/ip proxy connections`

This menu conntains the list of current connections the proxy is serving.

Read only properties:

Property	Description
client ()	
dst-address (*Ip4* | *Ip6*)	IPv4/Ipv6 destination address of the connection
protocol (*string*)	Protocol name
rx-bytes (*integer*)	The amount of bytes received by the client
server ()	
src-address (*Ip4* | *Ip6*)	Ipv4/ipv6 address of the connection originator

state (*closing* | *connecting* | *converting* | *hotspot* | *idle* | *resolving* | *rx-header* | *tx-body* | *tx-eof* | *tx-header* | *waiting*)	Connection state: • `closing` - the data transfer is finished, and the connection is being finalized • `connecting` - establishing toe connection • `converting` - replacing header and footer fields in response or request paket • `hotspot` - check if hotspot authentication allows to continue (for hotspot proxy) • `idle` - staying idle • `resolving` - resolving server's DNS name • `rx-header` - receiving HTTP header • `tx-body` - transmitting HTTP body to the client • `tx-eof` - writing chunk-end (when converting to chunked response) • `tx-header` - transmitting HTTP header to the client • `waiting` - waiting for transmission form a peer
tx-bytes (*integer*)	The amount of bytes sent by the client

Cache Inserts

Sub-menu: `/ip proxy inserts`

This menu shows statistics on objects stored in cache (cache inserts).

Read only properties:

Property	Description
denied (*integer*)	Number of inserts denied by the caching list.
errors (*integer*)	Number of disk or other system-related errors
no-memory (*integer*)	Number of objects not stored because there was not enough memory
successes (*integer*)	Number of successfull cache inserts.
too-large (*integer*)	Number of objects too large to store

Cache Lookups

Sub-menu: `/ip proxy lookup`

This menu shows statistics on objects read from cache (cache lookups).

Read only properties:

Property	Description
denied (*integer*)	Number of requests denied by the access list.
expired (*integer*)	Number of requests found in cache, but expired, and, thus, requested from an external server
no-expiration-info (*integer*)	Conditional request received for a page that does not have the information to compare the request with
non-cacheable (*integer*)	Number of requests requested from the external servers unconditionally (as their caching is denied by the cache access list)
not-found (*integer*)	Number of requests not found in the cache, and, thus, requested from an external server (or parent proxy if configured accordingly)
successes (*integer*)	Number of requests found in the cache.

Cache Contents

Sub-menu: `/ip proxy cache-contents`

This menu shows cached contents.

Read only properties:

Property	Description
file-size (*integer*)	Cached object size
last-accessed (*time*)	
last-accessed-time (*time*)	
last-modified (*time*)	
last-modified-time (*time*)	
uri (*string*)	

HTTP Methods

Options

This method is a request of information about the communication options available on the chain between the client and the server identified by the **Request-URI**. The method allows the client to determine the options and (or) the requirements associated with a resource without initiating any resource retrieval

GET

This method retrieves whatever information identified by the Request-URI. If the Request-URI refers to a data processing process than the response to the GET method should contain data produced by the process, not the source code of the process procedure(-s), unless the source is the result of the process.

The GET method can become a conditional GET if the request message includes an If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match, or If-Range header field. The conditional GET method is used to reduce the network traffic specifying that the transfer of the entity should occur only under circumstances described by conditional header field(-s).

The GET method can become a partial GET if the request message includes a Range header field. The partial GET method intends to reduce unnecessary network usage by requesting only parts of entities without transferring data already held by client.

The response to a GET request is cacheable if and only if it meets the requirements for HTTP caching.

HEAD

This method shares all features of GET method except that the server must not return a message-body in the response. This retrieves the metainformation of the entity implied by the request which leads to a wide usage of it for testing hypertext links for validity, accessibility, and recent modification.

The response to a HEAD request may be cacheable in the way that the information contained in the response may be used to update previously cached entity identified by that Request-URI.

POST

This method requests that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI.

The actual action performed by the POST method is determined by the origin server and usually is Request-URI dependent.

Responses to POST method are not cacheable, unless the response includes appropriate Cache-Control or Expires header fields.

PUT

This method requests that the enclosed entity be stored under the supplied Request-URI. If another entity exists under specified Request-URI, the enclosed entity should be considered as updated (newer) version of that residing on the origin server. If the Request-URI is not pointing to an existing resource, the origin server should create a resource with that URI.

If the request passes through a cache and the Request-URI identifies one or more currently cached entities, those entries should be treated as stale. Responses to this method are not cacheable.

TRACE

This method invokes a remote, application-layer loop-back of the request message. The final recipient of the request should reflect the message received back to the client as the entity-body of a 200 (OK) response. The final recipient is either the origin server or the first proxy or gateway to receive a Max-Forwards value of 0 in the request. A TRACE request must not include an entity.

Responses to this method MUST NOT be cached.

[Top | Back to Content **]**

References

[1] http://www.facebook.com
[2] http://www.mail.com
[3] http://www.hotmail.com

Manual:Packet Flow

Applies to RouterOS: v3, v4, v5+

Overview

MikroTik RouterOS is designed to be easy to operate in various aspects of network configuration. Therefore creating limitation for individual IP or natting internal clients to a public address or Hotspot configuration can be done without the knowledge about how the packets are processed in the router - you just go to corresponding menu and create necessary configuration.

However more complicated tasks, such as traffic prioritization, routing policies, where it is necessary to utilize more than one RouterOS facility, requires knowledge: How these facilities work together? What happens when and why?

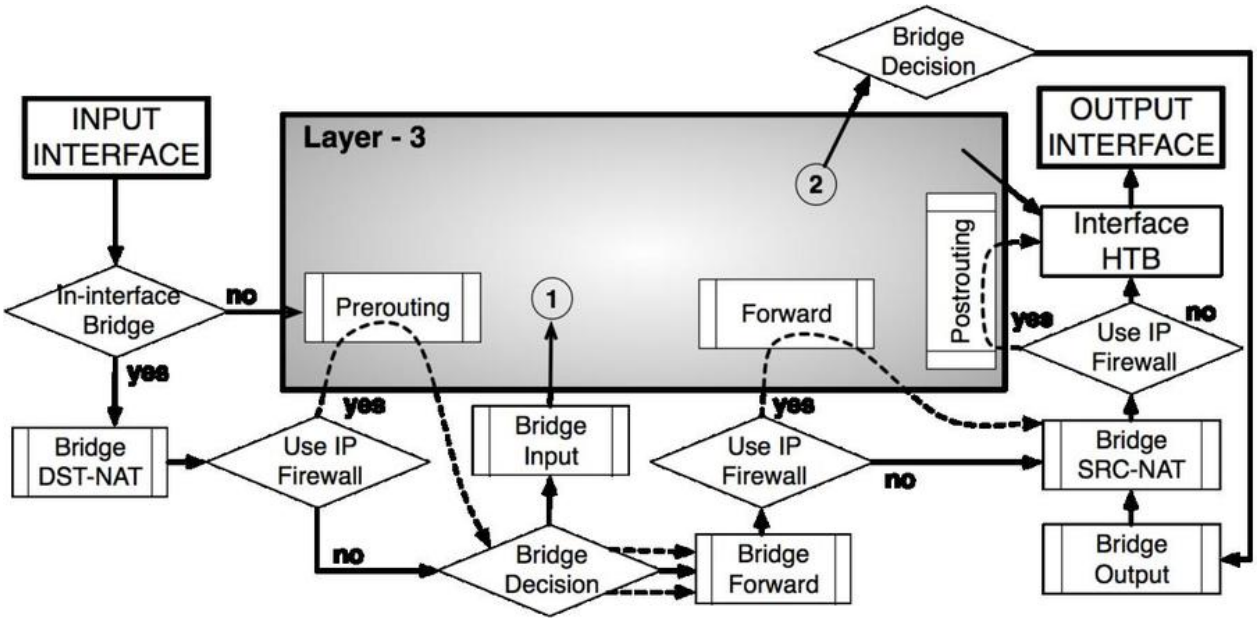
To address these questions we created a packet flow diagram.

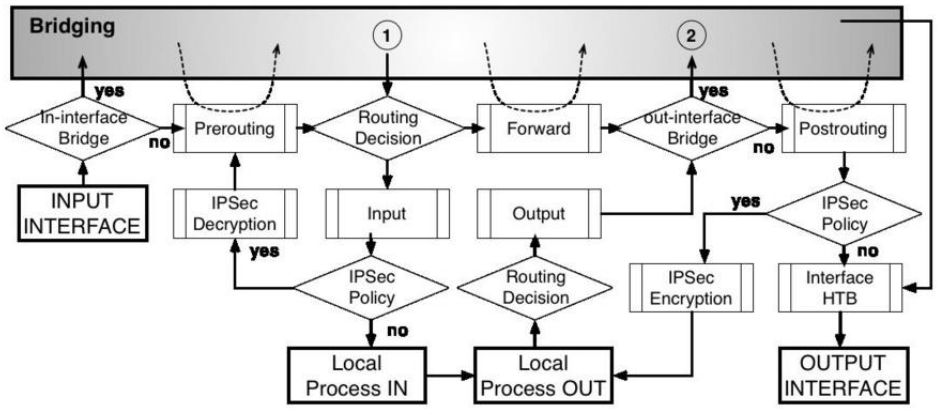
Diagram

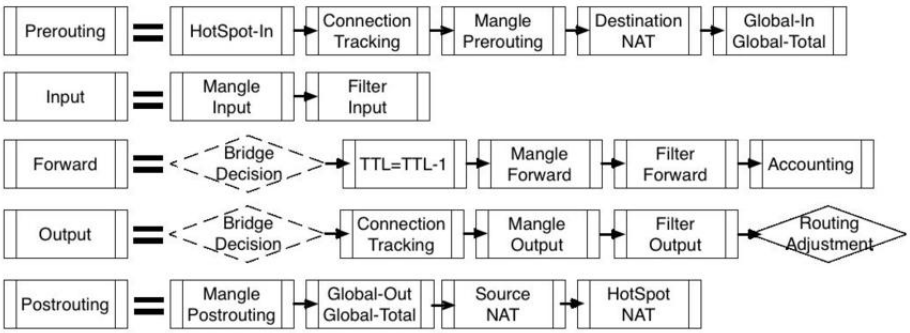
As it was impossible to get everything in one diagram, **Packet flow diagram** for Mikrotik RouterOS v3.x was created in 2 parts:

- **Bridging or Layer-2 (MAC)** where Routing part is simplified to one "Layer-3" box
- **Routing or Layer-3 (IP)** where Bridging part is simplified to one "Bridging" box

The packet flow diagram is also available as a PDF [1].







Analysis

Basic Concepts

INPUT INTERFACE - starting point in packets way through the router facilities. It does not matter what interface (physical or virtual) packet is received it will start its way from here.

OUTPUT INTERFACE - last point in packets way through the router facilities. Just before the packet is actually sent out.

Local Process IN - last point in packets way **to** router itself, after this packet is discarded

Local Process OUT - starting point for packets generated by router itself

Configurable Facilities

Each and every facilities in this section corresponds with one particular menu in RouterOS. Users are able to access those menu and configure these facilities directly

Connection Tracking - **/ip firewall connection tracking**

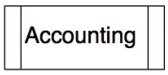
Filter Input | Filter Forward | Filter Output - **/ip firewall filter**

Source NAT | Destination NAT - **/ip firewall nat**

Mangle Prerouting | Mangle Input | Mangle Forward | Mangle Output | Mangle Postrouting - **/ip firewall mangle**

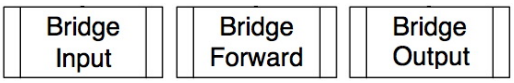
- **/queue simple** and **/queue tree**

- **/ip ipsec policy**

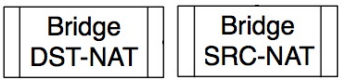


- **/ip accounting**

- **/interface bridge settings** - available only for traffic that go **through** the bridge. For all other traffic default value is **Yes**

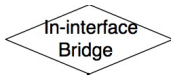


- **/interface bridge filter**



- **/interface bridge nat**

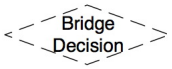
Automated processes and decisions



- check if the *actual input interface* is a port for bridge OR checks if *input interface* is bridge

- allow to capture traffic witch otherwise would be discarded by connection tracking - this way our Hotspot feature are able to provide connectivity even if networks settings are in complete mess

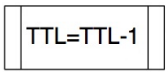
- bridge goes through the MAC address table in order to find a match to destination MAC address of packet. When match is found - packet will be send out via corresponding bridge port. In case of no match - multiple copies of packet will be created and packet will be sent out via all bridge ports



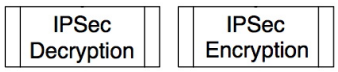
- this is a workaround, allows to use "out-bridge-port" before actual bridge decision.

- router goes through the route n order to find a match to destination IP address of packet. When match is found - packet will be send out via corresponding port or to the router itself . In case of no match - packet will be discarded.

- this is a workaround that allows to set-up policy routing in mangle chain output



- indicates exact place where Time To Live (TTL) of the routed packet is reduced by 1. If it become 0 packet will be discarded



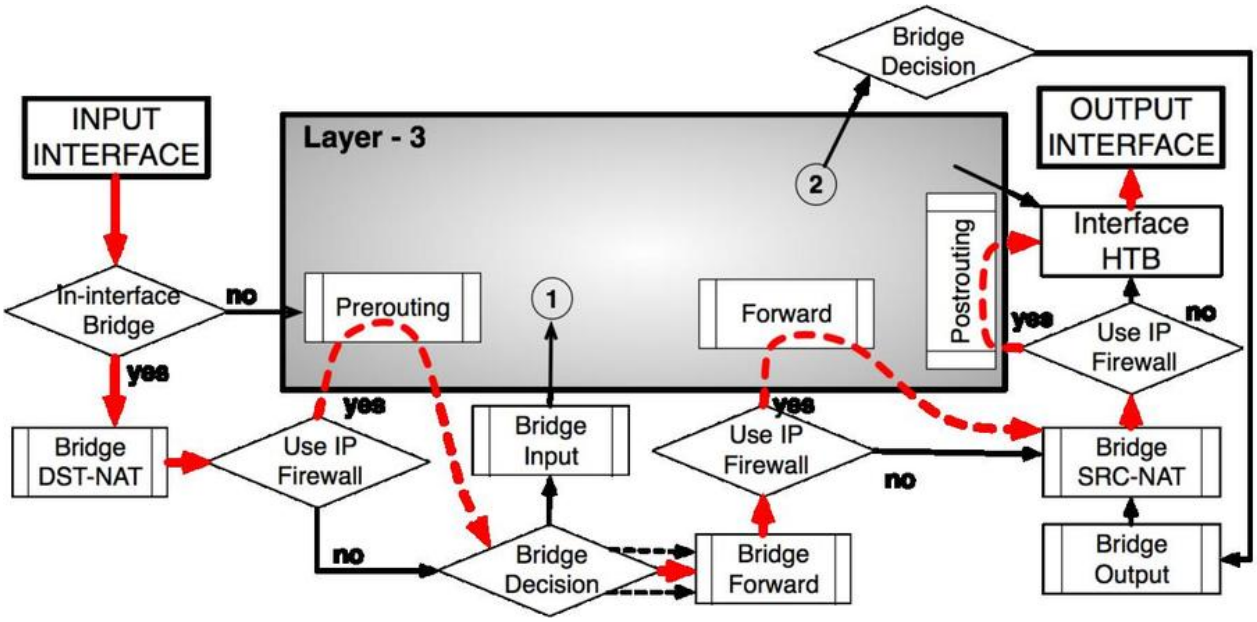
- self explainatory

- check if the *actual output interface* is a port for bridge OR checks if *output interface* is bridge

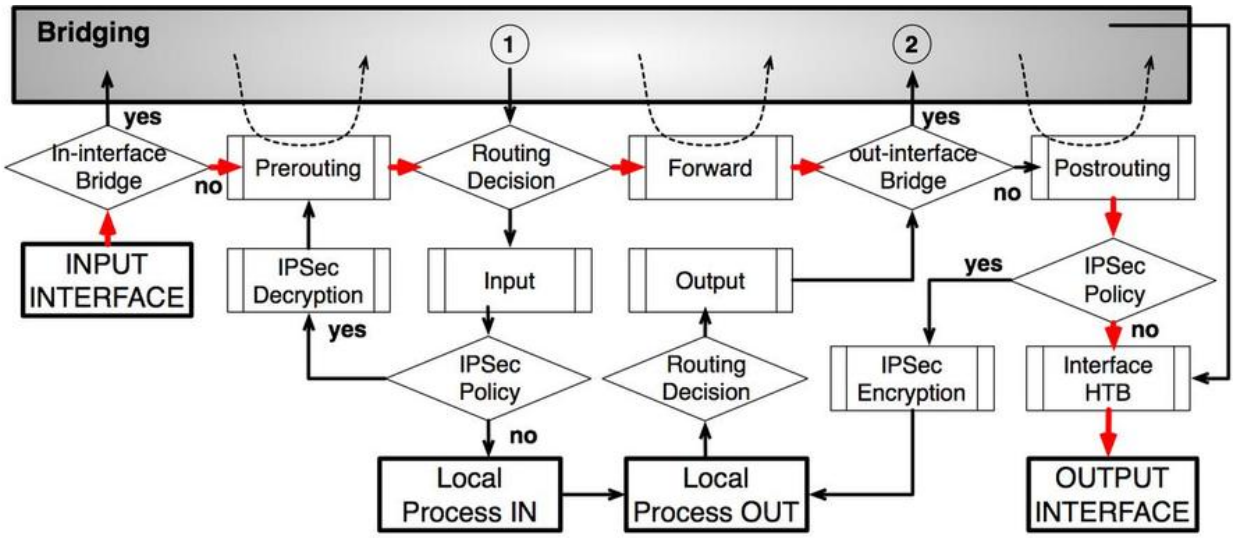
- undo all that was done by hotspot-in for the packets that is going back to client.

Examples

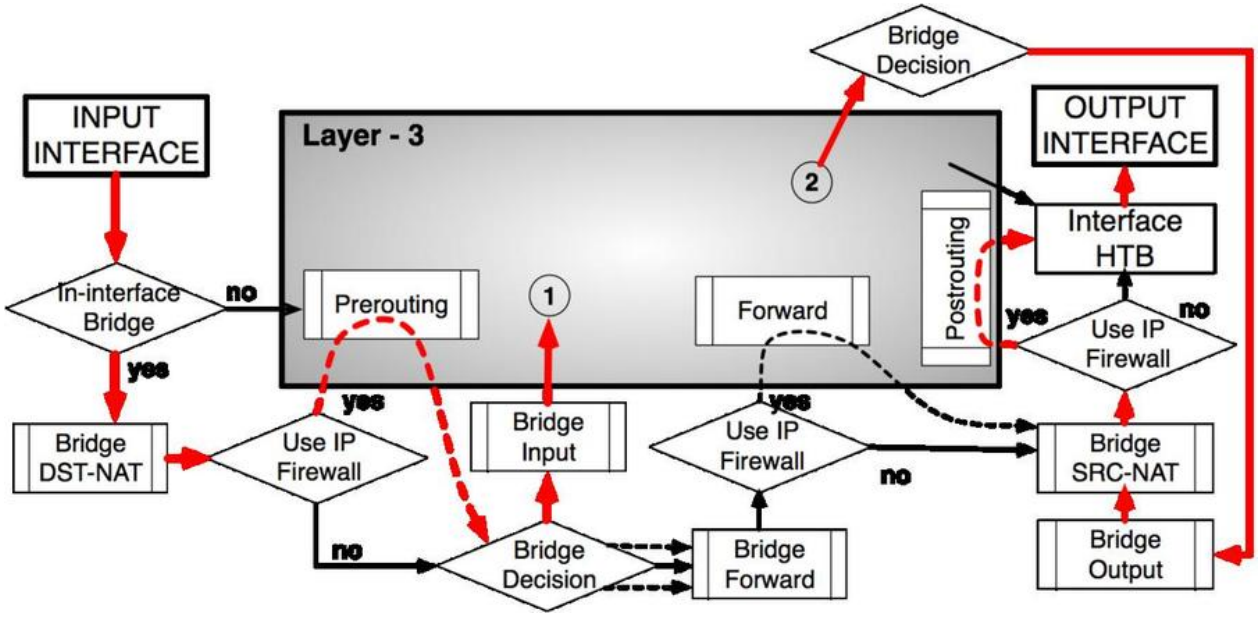
Bridging with use-ip-firewall=yes



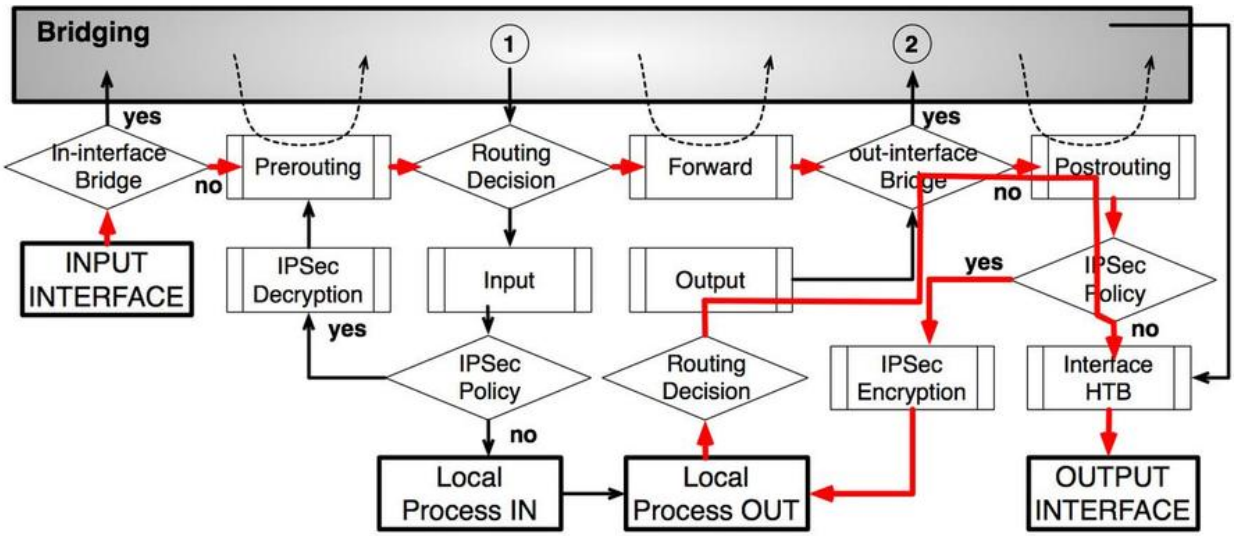
Routing - from Ethernet to Ethernet interface



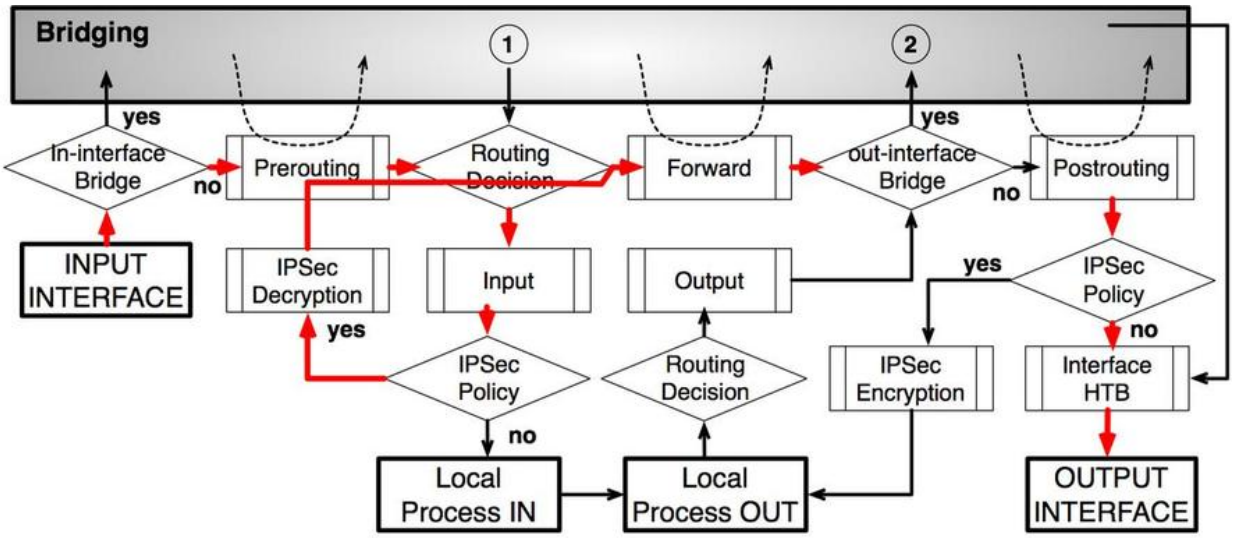
Routing from one Bridge interface to different Bridge interface



IPsec encryption



IPsec decryption



References

[1] http://wiki.mikrotik.com/images/1/1b/Traffic_Flow_Diagram_RouterOS_3.x.pdf

Manual:Performance Testing with Traffic Generator

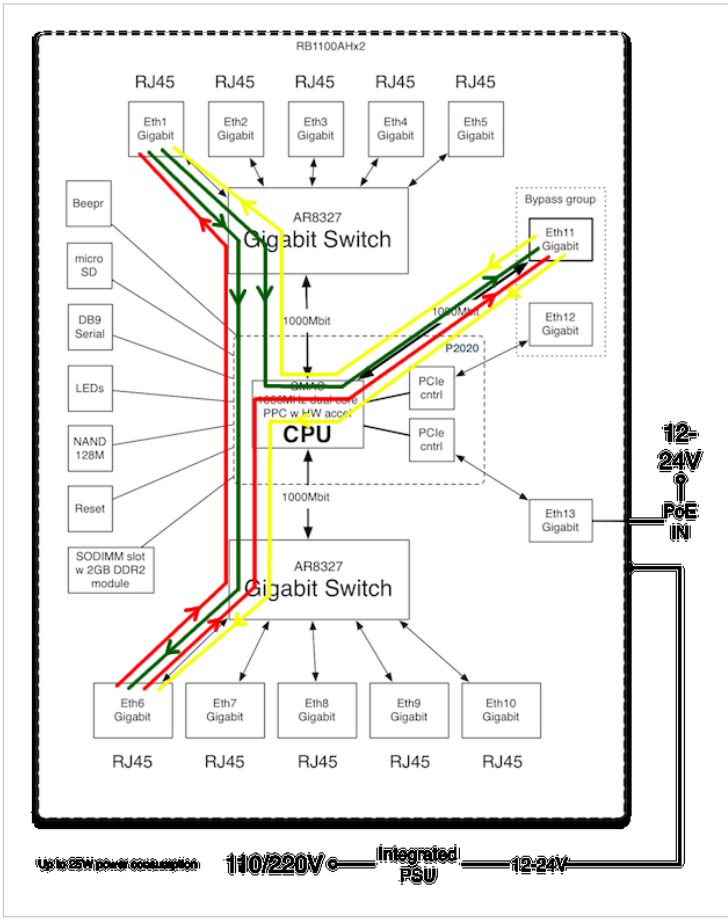
Summary

RouterOS Version 6 introduces a new tool - "traffic generator", which allows to perform performance testing without expensive testing hardware. Traffic is generated from one more router in the network.

This article shows necessary configuration and hardware to replicate the tests published in routerboard.com [1].

RB1100AHx2 Test setup

First step is to choose which ports we will be using for testing.



If we look at the diagram how ports are connected to CPU, fastest combinations are:

- port from switch1 to port form switch chip2,
- ether11 to switch chip,
- ether12/13 to switch chip or to ether11.

To get the maximum out of RB1100AHx2 we will be running 6 streams in total:

- from ether1 to ether6
- from ether1 to ether11
- from ether6 to ether1
- from ether6 to ether11
- from ether11 to ether6

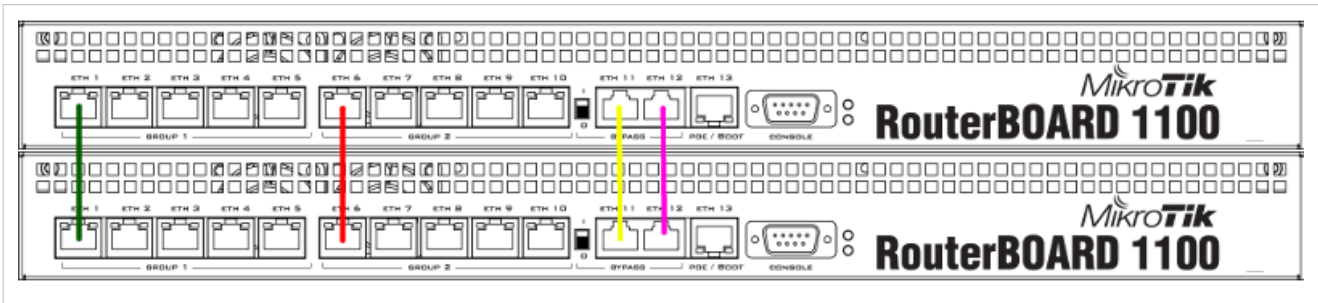
- from ether11 to ether1

In our test environment one RB1100AHx2 will be device under test (DUT) and other RB1100AHx2 will be a Traffic generator device.

Connecting the routers

Note: RouterOS v6 should be used on both test routers.

Connect cables like this: ether1 to ether1, ether6 to ether6, ether11 to ether11



Note: Ether12 will be added where test reaches wire speed.

Now proceed with software configuration. Either it will be routing (layer3) testing or bridging (layer2) testing.

Routing Performance Testing

DUT Config

```
/ip address
add address=1.1.1.254/24 interface=ether1 network=1.1.1.0
add address=2.2.2.254/24 interface=ether6 network=2.2.2.0
add address=3.3.3.254/24 interface=ether11 network=3.3.3.0
```

Traffic Generator Config

```
/ip address
add address=1.1.1.1/24 interface=ether1 network=1.1.1.0
add address=2.2.2.2/24 interface=ether6 network=2.2.2.0
add address=3.3.3.3/24 interface=ether11 network=3.3.3.0

/tool traffic-generator packet-template
add name=r12 header-stack=mac,ip,udp ip-gateway=1.1.1.254 ip-dst=2.2.2.2
add name=r13 header-stack=mac,ip,udp ip-gateway=1.1.1.254 ip-dst=3.3.3.3
add name=r21 header-stack=mac,ip,udp ip-gateway=2.2.2.254 ip-dst=1.1.1.1
add name=r23 header-stack=mac,ip,udp ip-gateway=2.2.2.254 ip-dst=3.3.3.3
add name=r32 header-stack=mac,ip,udp ip-gateway=3.3.3.254 ip-dst=2.2.2.2
add name=r31 header-stack=mac,ip,udp ip-gateway=3.3.3.254 ip-dst=1.1.1.1
```

Note: To force MAC address re-discovery (on device/configuration change, just apply emply "set" command to necessary packet-templates)

Running Tests

```
/tool traffic-generator
quick tx-template=r12,r13,r21,r23,r31,r32 packet-size=60 mbps=300
```

Note: We are specifying 60 byte packet in traffic generator to get a 64 byte packet on ethernet.

```
[admin@TrafficGen] > /tool traffic-gen quick tx-template=r12,r13,r21,r23,r31,r32 packet-size=60
mbps=120
```

```
24     0         185 422  91.9Mbps       185 190  88.8Mbps                  232   3.0Mbps 16us
24     1         213 397 105.8Mbps       212 747 102.1Mbps                  650   3.7Mbps 10.6us
24     2         186 245  92.3Mbps       186 185  89.3Mbps                   60   3.0Mbps 16.4us
24     3         213 685 105.9Mbps       212 961 102.2Mbps                  724   3.7Mbps 10.8us
24     4         249 142 119.5Mbps       180 400  86.5Mbps               68 742  32.9Mbps 13.2us
24     5         249 141 119.5Mbps       193 158  92.7Mbps               55 983  26.8Mbps 11.1us
24     TOT     1 297 032 635.3Mbps     1 170 641 561.9Mbps              126 391  73.4Mbps 10.6us
```

You can also check in the **DUT** if forwarding is actually happening:

```
[admin@DUT] > /interface monitor-traffic aggregate,ether1,ether6,ether11
                      name:               ether1    ether6   ether11
    rx-packets-per-second:  1 235 620   481 094   487 045   267 469
      rx-drops-per-second:           0         0         0         0
     rx-errors-per-second:           0         0         0         0
       rx-bits-per-second:  593.0Mbps 230.9Mbps 233.7Mbps 128.3Mbps
    tx-packets-per-second:  1 233 862   360 750   360 402   512 692
      tx-drops-per-second:           0         0         0         0
     tx-errors-per-second:           0         0         0         0
       tx-bits-per-second:  603.9Mbps 178.9Mbps 178.7Mbps 246.0Mbps
```

After running the test you can see that total throughput of 64byte packets is **1'170'641**pps which is a lot faster than shown in routerboard.com [1] results.

This is because by default fast-path mode is enabled.

Lets enable connection tracking on DUT:

```
/ip firewall connection tracking set enabled=yes
```

And run the test again. As you can see now it is close to advertised pps rate.

```
46     0         249 793 123.8Mbps       127 410  61.1Mbps                122 383  62.7Mbps 3.22ms
46     1         249 791 123.8Mbps        87 232  41.8Mbps                162 559  82.0Mbps 5.2ms
46     2         249 792 123.8Mbps       127 424  61.1Mbps                122 368  62.7Mbps 3.15ms
46     3         249 792 123.8Mbps        87 219  41.8Mbps                162 573  82.0Mbps 5.18ms
46     4         249 792 119.9Mbps        40 492  19.4Mbps                209 300 100.4Mbps 5.54ms
46     5         249 791 119.8Mbps        46 736  22.4Mbps                203 055  97.4Mbps 5.41ms
46     TOT     1 498 751 735.3Mbps       516 513 247.9Mbps                982 238 487.4Mbps 3.15ms
```

We can now add more firewall rules, queues and any other configuration and see how much router can actually handle.

Lets add some firewall rules

We will take the customer protection rules from the manual

Start by adding default rules that should present on any firewall:

```
/ip firewall filter
add chain=forward protocol=tcp connection-state=invalid \
      action=drop comment="drop invalid connections"
add chain=forward connection-state=established action=accept \
      comment="allow already established connections"
add chain=forward connection-state=related action=accept \
      comment="allow related connections"
```

We get approximately **18%** less packets

```
53     TOT     1 492 520 732.3Mbps       435 546 209.0Mbps              1 056 974 523.2Mbps 3.08ms
```

Now add more rules from the manual to see how count of firewall rules affects the performance of the board

```
/ip firewall filter
add chain=forward protocol=icmp action=jump jump-target=icmp

add chain=icmp protocol=icmp icmp-options=0:0 action=accept \
       comment="echo reply"
add chain=icmp protocol=icmp icmp-options=3:0 action=accept \
       comment="net unreachable"
add chain=icmp protocol=icmp icmp-options=3:1 action=accept \
       comment="host unreachable"
add chain=icmp protocol=icmp icmp-options=3:4 action=accept \
       comment="host unreachable fragmentation required"
add chain=icmp protocol=icmp icmp-options=4:0 action=accept \
       comment="allow source quench"
add chain=icmp protocol=icmp icmp-options=8:0 action=accept \
       comment="allow echo request"
add chain=icmp protocol=icmp icmp-options=11:0 action=accept \
       comment="allow time exceed"
add chain=icmp protocol=icmp icmp-options=12:0 action=accept \
       comment="allow parameter bad"
add chain=icmp action=drop comment="deny all other types"
```

```
33     TOT     1 500 908 736.4Mbps       424 197 203.6Mbps              1 076 711 532.8Mbps 4.07ms
```

There are almost no performance changes. You can add any amount of rules and see that there is only a small influence on performance of the router.

Perform the same test with different packet sizes:

```
/tool traffic-generator
quick tx-template=r12,r13,r21,r23,r31,r32 packet-size=508 mbps=500
/tool traffic-generator
quick tx-template=r12,r13,r21,r23,r31,r32 packet-size=1514 mbps=500
```

If we run the test with 1518 packet size then max throughput will be only **2.9Gbps** This is because wire speed of all interfaces are reached.

We will need to add one more port to our test and add streams.

Connect **ether12** to **ether12** and proceed with configuration

On DUT:

```
/ip address
add address=4.4.4.254/24 interface=ether12
```

On TrafficGen

```
/ip address
add address=4.4.4.4/24 interface=ether12

/tool traffic-generator packet-template
add header-stack=mac,ip,udp ip-dst=4.4.4.4/32 ip-gateway=1.1.1.254 ip-src=1.1.1.1/32 name=\
    r14
add header-stack=mac,ip,udp ip-dst=4.4.4.4/32 ip-gateway=2.2.2.254 ip-src=2.2.2.2/32 name=\
    r24
add header-stack=mac,ip,udp ip-dst=4.4.4.4/32 ip-gateway=3.3.3.254 ip-src=3.3.3.3/32 name=\
    r34
add header-stack=mac,ip,udp ip-dst=1.1.1.1/32 ip-gateway=4.4.4.254 ip-src=4.4.4.4/32 name=\
    r41
add header-stack=mac,ip,udp ip-dst=2.2.2.2/32 ip-gateway=4.4.4.254 ip-src=4.4.4.4/32 name=\
    r42
add header-stack=mac,ip,udp ip-dst=3.3.3.3/32 ip-gateway=4.4.4.254 ip-src=4.4.4.4/32 name=\
    r43
```

And now run the test:

```
/tool traffic-generator quick tx-template=r12,r13,r14,r21,r23,r24,r31,r32,r34,r41,r42,r43 \
 packet-size=1514 mbps=350
```

```
30     6          23 472 284.2Mbps        23 328 282.5Mbps                   144 1744.1... 3.22ms
30     7          28 890 349.9Mbps        28 741 348.1Mbps                   149 1804.6... 1.74ms
30     8          28 889 349.9Mbps        26 870 325.4Mbps                 2 019  24.4Mbps 984us
30     9          23 455 284.0Mbps        23 083 279.5Mbps                   372   4.5Mbps 866us
30     10         28 876 349.7Mbps        28 709 347.7Mbps                   167   2.0Mbps 922us
30     11         28 875 349.7Mbps        27 277 330.3Mbps                 1 598  19.3Mbps 3.33ms
30     TOT       323 389   3.9Gbps       311 743   3.7Gbps                11 646 143.6Mbps 341us
```

As you can see we get **3.7Gbps**.

And with all firewalls enabled from previous tests we get 2.8Gbps which is approximately **30%** slower:

```
18     TOT       275 405   3.3Gbps       238 143   2.8Gbps                37 262 453.9Mbps 1.57ms
```

Note: mind that speed in quick mode is specified per stream, so if you have two streams per port, you need to send 1/2 of traffic per stream

Bridging Performance Testing

DUT Config

```
/interface bridge add
/interface bridge port
add bridge=bridge1 interface=ether1
add bridge=bridge1 interface=ether6
add bridge=bridge1 interface=ether11
```

Traffic Generator Config

```
/ip address
add address=1.1.1.1/24 interface=ether1 network=1.1.1.0
add address=2.2.2.2/24 interface=ether6 network=2.2.2.0
add address=3.3.3.3/24 interface=ether11 network=3.3.3.0

/tool traffic-generator packet-template
add header-stack=mac,ip,udp ip-src=1.1.1.1/32 ip-dst=2.2.2.2/32 name=b12
add header-stack=mac,ip,udp ip-src=1.1.1.1/32 ip-dst=3.3.3.3/32 name=b13
add header-stack=mac,ip,udp ip-src=2.2.2.2/32 ip-dst=1.1.1.1/32 name=b21
add header-stack=mac,ip,udp ip-src=2.2.2.2/32 ip-dst=3.3.3.3/32 name=b23
add header-stack=mac,ip,udp ip-src=3.3.3.3/32 ip-dst=1.1.1.1/32 name=b31
add header-stack=mac,ip,udp ip-src=3.3.3.3/32 ip-dst=2.2.2.2/32 name=b32
```

Running Tests

```
/tool traffic-generator
quick tx-template=b12,b13,b21,b23,b31,b32 packet-size=60 mbps=200
/tool traffic-generator
quick tx-template=b12,b13,b21,b23,b31,b32 packet-size=508 mbps=500
/tool traffic-generator
quick tx-template=b12,b13,b21,b23,b31,b32 packet-size=1514 mbps=500
```

With small packets we get approximately 1.4 mil packets per second

```
187    0         195 659  97.0Mbps       195 640  93.9Mbps                    19   3.1Mbps 22us
187    1         236 906 117.5Mbps       221 901 106.5Mbps                15 005  10.9Mbps 18.7us
187    2         202 678 100.5Mbps       202 678  97.2Mbps                     0   3.2Mbps 18.7us
187    3         238 750 118.4Mbps       231 348 111.0Mbps                 7 402   7.3Mbps 12.1us
187    4         263 906 126.6Mbps       256 146 122.9Mbps                 7 760   3.7Mbps 23.9us
187    5         263 906 126.6Mbps       256 030 122.8Mbps                 7 876   3.7Mbps 14.3us
187    TOT     1 401 805 686.8Mbps     1 363 743 654.5Mbps                38 062  32.2Mbps 12.1us
```

With 1518 byte packets we will get wire speed maximum

```
11     TOT       243 587   2.9Gbps       241 695   2.9Gbps                  1 892  25.5Mbps 1.04ms
```

So we will need to use ether12 and add few more streams just like in routing test.

See More

- Traffic Generator Manual
- Fast Path

References

[1] http://routerboard.com/RB1100AHx2#tests

Manual:System/Packages

Summary

RouterOS supports a lot of different features and since every installation requires specific set of features supprted it is possible to add or remove certain groups of features using package system. As result user is able to control what features are available and size of installation. Packages are provided only by MikroTik and no 3rd parties are allowed to make them.

Acquiring packages

Packages can be downloaded from MikroTik download [1] page or mirrors listed on that page. Either of provided download methods can be used.

RouterOS packages

for each architecture

Package	Features
advanced-tools (*mipsle, mipsbe, ppc, x86*)	advanced ping tools. netwatch, ip-scan, sms tool, wake-on-LAN
calea (*mipsle, mipsbe, ppc, x86*)	data gathering tool for specific use due to "Communications Assistance for Law Enforcement Act" in USA
dhcp (*mipsle, mipsbe, ppc, x86*)	Dynamic Host Control Protocol client and server
gps (*mipsle, mipsbe, ppc, x86*)	Global Positioning System devices support
hotspot (*mipsle, mipsbe, ppc, x86*)	HotSpot user management
ipv6 (*mipsle, mipsbe, ppc, x86*)	IPv6 addressing support
mpls (*mipsle, mipsbe, ppc, x86*)	Multi Protocol Labels Switching support
multicast (*mipsle, mipsbe, ppc, x86*)	Protocol Independent Multicast - Sparse Mode; Internet Group Managing Protocol - Proxy

ntp (*mipsle, mipsbe, ppc, x86*)	Network protocol client and service
ppp (*mipsle, mipsbe, ppc, x86*)	MlPPP client, PPP, PPTP, L2TP, PPPoE, ISDN PPP clients and servers
routerboard (*mipsle, mipsbe, ppc, x86*)	accessing and managing RouterBOOT. RouterBOARD specific imformation.
routing (*mipsle, mipsbe, ppc, x86*)	dynamic routing protocols like RIP, BGP, OSPF and routing utilities like BFD, filters for routes.
security (*mipsle, mipsbe, ppc, x86*)	IPSEC, SSH, Secure WinBox
system (*mipsle, mipsbe, ppc, x86*)	basic router features like *static routing, ip addresses, sNTP, telnet, API, queues, firewall, web proxy, DNS cache, TFTP, IP pool, SNMP, packet sniffer, e-mail send tool, graphing, bandwidth-test, torch, EoIP, IPIP, bridging, VLAN, VRRP etc.). Also, for RouterBOARD platform - MetaROUTER	Virtualization*
ups (*mipsle, mipsbe, ppc, x86*)	APC ups
user-manager (*mipsle, mipsbe, ppc, x86*)	MikroTik User Manager
wireless (*mipsle, mipsbe, ppc, x86*)	wireless interface support
arlan (*x86*)	legacy Aironet Arlan support
isdn (*x86*)	ISDN support
lcd (*x86*)	LCD panel support
radiolan (*x86*)	RadioLan cards support
synchronous (*x86*)	FarSync support
xen (*discontinued x86*)	XEN Virtualization
kvm (*x86*)	KVM Virtualization
routeros-mipsle (*mipsle*)	combined package for mipsle (RB100, RB500) (includes *system, hotspot, wireless, ppp, security, mpls, advanced-tools, dhcp, routerboard, ipv6, routing*)
routeros-mipsbe (*mipsbe*)	combined package for mipsbe (RB400) (includes *system, hotspot, wireless, ppp, security, mpls, advanced-tools, dhcp, routerboard, ipv6, routing*)
routeros-powerpc (*ppc*)	combined package for powerpc (RB300, RB600, RB1000) (includes *system, hotspot, wireless, ppp, security, mpls, advanced-tools, dhcp, routerboard, ipv6, routing*)
routeros-x86 (*x86*)	combined package for x86 (Intel/AMD PC, RB230) (includes *system, hotspot, wireless, ppp, security, mpls, advanced-tools, dhcp, routerboard, ipv6, routing*)
mpls-test (*mipsle, mipsbe, ppc, x86*)	Multi Protocol Labels Switching support improvements
routing-test (*mipsle, mipsbe, ppc, x86*)	routing protocols (RIP, OSPF, BGP) improvements

Working with packages

Menu: */system package*

Commands executed in this menu will take place only on restart of the router. Until then, user can freely schedule or revert set actions.

Command	Desciption
disable	schedule package to be disabled after next reboot. All features provided by package will not be accessible
downgrade	will prompt for reboot. During reboot process will try to downgrade RouterOS to oldest version possible by checking packages that are uploaded to the router.
print	outputs information about packages, like: version, package state, planned state changes etc.
enable	schedule package to be enabled after next reboot
uninstall	schedule package to be removed from router. That will take place during reboot.
unschedule	remove scheduled task for package.

Examples

Upgrade process is described here.

List available packages

```
 /system package print
Flags: X - disabled
 #   NAME                    VERSION                    SCHEDULED
 0 X ipv6                    3.13
 1   system                  3.13
 2 X mpls                    3.13
 3 X hotspot                 3.13
 4   routing                 3.13
 5   wireless                3.13
 6 X dhcp                    3.13
 7   routerboard             3.13
 8   routeros-mipsle         3.13
 9   security                3.13
 10 X ppp                     3.13
 11   advanced-tools          3.13
```

Uninstall package

Schedules package for uninstallation and reboots router.

```
 /system package uninstall ppp; /system reboot;
Reboot, yes? [y/N]:
```

Disable package

```
 /system package disable hotspot; /system reboot;
Reboot, yes? [y/N]:
```

Downgrade

```
 /system package downgrade; /system reboot;
Reboot, yes? [y/N]:
```

Cancel uninstall or disable action

```
 /system package unschedule ipv6
```

Manual:Tools/Profiler

Applies to RouterOS: v5beta7 +

Summary

Command: `/tool profile`
Standards:

Profiler tool shows CPU usage for each process running in RouterOS. It helps to identify which process is using most of the CPU resources.

```
[admin@dzeltenais_burkaans] > /tool profile
NAME              USAGE
sstp              9%
ppp               0.5%
ethernet          0%
queue-mgmt        0%
console           0.5%
dns               0%
winbox            0%
logging           0%
management        1.5%
ospf              0%
idle              87.5%
profiling         0.5%
queuing           0%
routing           0%
bridging          0%
unclassified      0.5%
```

CPU usage on multi-core systems

On multi-core systems tool allows to specify per core CPU usage. For example, to view CPU usage on second core use following command:

```
[admin@x86-test] > /tool profile cpu=2
NAME              CPU         USAGE
ethernet          1           0%
kvm               1           2.5%
management        1           0.5%
idle              1           96.5%
profiling         1           0%
unclassified      1           0.5%
```

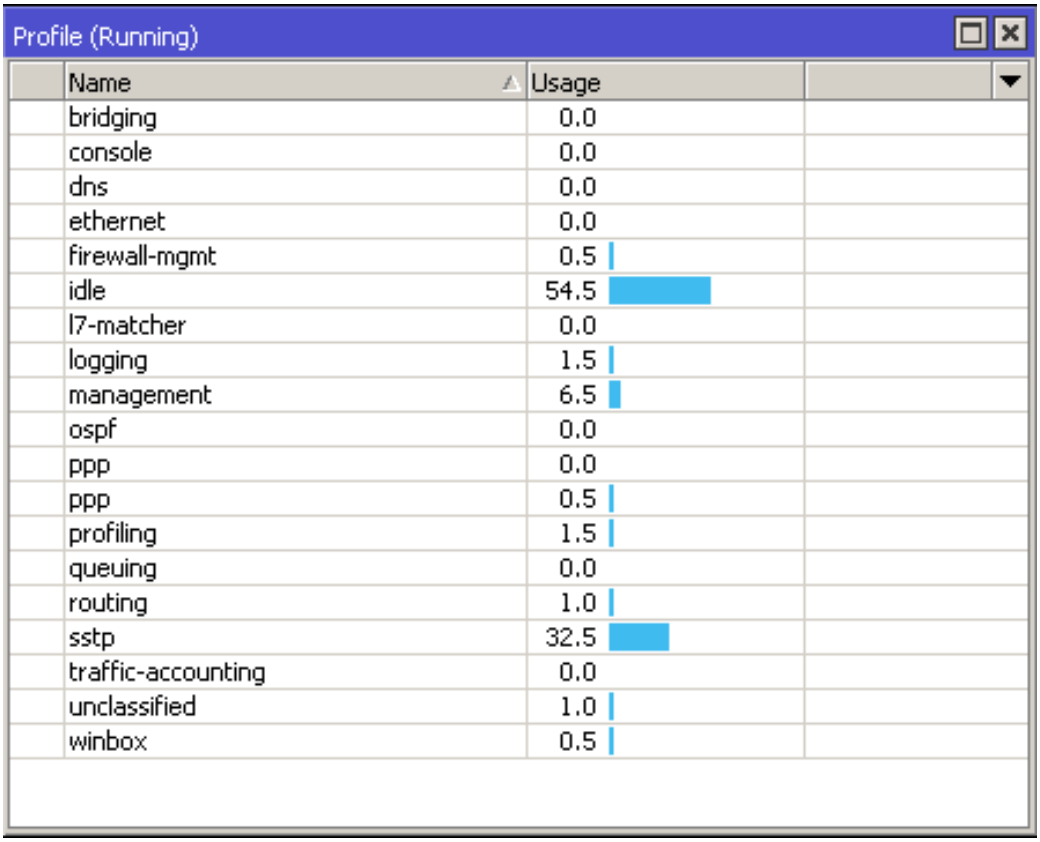
"cpu" parameter allows to specify integer number which represents a core or two of predefined values **all** and **total**

- total - this value sets to show sum of all core usages;
- all - value sets to show cpu usages separately for every available core

Example with both values on two core system:

```
[admin@x86-test] > /tool profile cpu=all
NAME              CPU         USAGE
ethernet          1           0%
kvm               0           0%
kvm               1           4.5%
management        0           0%
management        1           0.5%
idle              0           100%
idle              1           93%
profiling         0           0%
profiling         1           2%

[admin@x86-test] > /tool profile cpu=total
NAME              CPU         USAGE
ethernet          all         0%
console           all         0%
kvm               all         2.7%
management        all         0%
idle              all         97.2%
profiling         all         0%
bridging          all         0%
```

Tool is also available in winbox:



Classifiers

Profile classifies processes in several classifiers. Most of them are self explanatory and does not require detailed explanation.

- **idle** - shows unused CPU. Typically idle=100%-(sum of all process cpu usages).
- **ppp**
- **pppoe**
- **ppp-compression**
- **ppp-mppe**
- **ethernet** - cpu used by ethernets when sending/receiving packets
- **bridging**
- **encrypting** - cpu used by packet encryption
- **ipsec** - IP security
- **queuing** - packet queuing
- **firewall** - packet processing in Ip firewall
- **l7-matcher** - cpu used by Layer7 matcher.
- **p2p-matcher** - Peer-to-peer traffic matcher in ip firewall
- **gre** - Gre tunnels
- **eoip** - EoIP tunnels
- **m3p** - MikroTik Packet Packer Protocol
- **radius**
- **ip-pool**
- **routing**
- **sniffing**
- **traffic-accounting**
- **traffic-flow**

- **console**
- **telnet**
- **ssh**
- **ftp**
- **tfpt**
- **www**
- **dns**
- **snmp**
- **socks**
- **web-proxy**
- **winbox**
- **metarouter-fs**
- **metarouter-net**
- **kvm**
- **profiling** - cpu used by Profiler tool itself
- **btest** - bandwidth test tool
- **logging**
- **flash** - cpu usage when writing to NAND
- **disk** - cpu usage when wiring to Disk
- **serial**
- **usb**
- **firewall-mgmt**
- **queue-mgmt**
- **e-mail**
- **fetcher**
- **backup**
- **graphing**
- **health**
- **isdn**
- **dhcp**
- **hotspot**
- **radv** - IPv6 route advertisement
- **ntp** - NTP server/client
- **ldp**
- **mpls**
- **pim** - Multicast routing protocol
- **igmp-proxy**
- **bgp**
- **ospf**
- **rip**
- **mme**
- **synchronous** - cpu usage by synchronous cards
- **gps**
- **user-manager**
- **wireless**
- **dude**
- **supout.rif** - cpu used by supout.rif file creator.

- **management** - RouterOS management processes that do not fall into any other classifier. For example, when routes added to kernel, internal messaging exchange between RouterOS applications, etc.
- **unclassified** - any other processes that were not classified.

[Top | Back to Content]

Manual:IP/Packing

Overview

IP Packing provides packet packaging service on network links. It allows simple packet aggregation into larger packets and compression of contents of packets.

Requirements

Packet packing is part of system package and has to have discovery protocol enabled on interface.

Configuration

Menu: *[/ip packing](#)*

It required to have configuration in two places, both routers should be set up symetrically:

- *ip packing* - to enable packet aggregation and/or compression on interface
- */ip neighbor discovery*- to enable discovery protocol on the interface

Packing configuration

Property	Desciption
aggregated-size (*20 .. 16384 default:**1500***)	size of aggregated packet that packing will try to achieve before sending packet over network
disabled (*yes|no*)	state of packing rule, if value is *yes* it will be ignored and will not be part of active configuration
interface (*interface name*)	packing will try to aggregate and/or compress packets from this interface
packing (*simple|compress-all|compress-headers|none*)	action it should perform when packet is leaving interface packing rule is configured on. • *simple* - do just aggregation of packets • *compress-all - do aggregation and attempt to compress headers and payload of packet* • *compress-headers - do aggregation and attempt to compress headers and leaving payload of packet as is* • *none* - send packets as is
unpacking (*simple|compress-all|compress-headers|none*)	action it should perform when packet is received on interface packing rule is configured on. • *simple* - unpack received packets from aggregated packet received from interface • *compress-all - unpack aggregated packet and uncompress headers and payload of packet* • *compress-headers - unpack aggregated packet and decompress headers of packet* • *none* - do nothing with received packet

Warning: Router should be seen as neighbour of router over interface you want to enable packing on. If in neighbour list there are no entry indicating packing, packing is not working!

Note: Packing may increase latency on the link it is configured on!

Example

Router-A and Router-B are connected with cable with interface ether1 on Router-A and ether3 on Router-B. This example will aggregate packets coming from Router-A, but will leave packets from Router-B intact On Router-A:

- make sure discovery is enabled

```
/ip neighbor discovery set ether1 discover=yes
```

- add packing rule for the interface

```
/ip packing add interface=ether1 aggregated-size=1500 packing=simple unpacking=none
```

On Router-B:

- make sure discovery is enabled

```
/ip neighbor discovery set ether3 discover=yes
```

- add packing rule for the interface

```
/ip packing add interface=ether3 aggregated-size=1500 packing=none unpacking=simple
```

Manual:Password reset

RouterOS password can only be reset by reinstalling the router, or using the reset button (or jumper hole) in case the hardware is RouterBOARD.

For X86 devices, only complete reinstall will clear the password, along with other configuration. For RouterBOARD devices, several methods exist, depending on our model.

Button reset

Most RouterBOARD devices are fitted with a reset button.

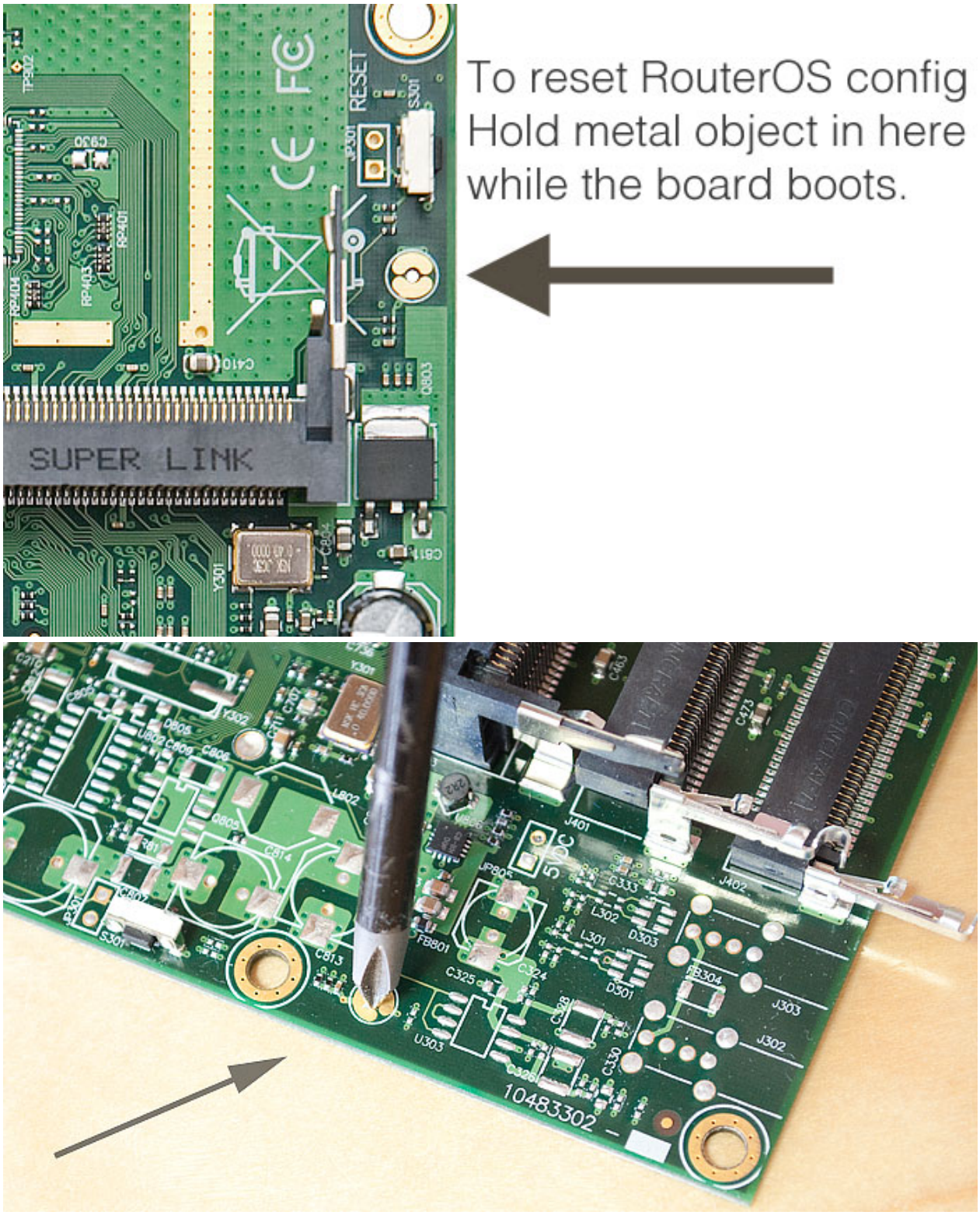
Using: unplug the device power, hold the button, apply power and wait until the USER LED starts flashing. Now release the button to clear configuration.

Note: If you wait until LED stops flashing, and only then release the button - this will instead launch Netinstall mode, to reinstall RouterOS.

Jumper hole reset

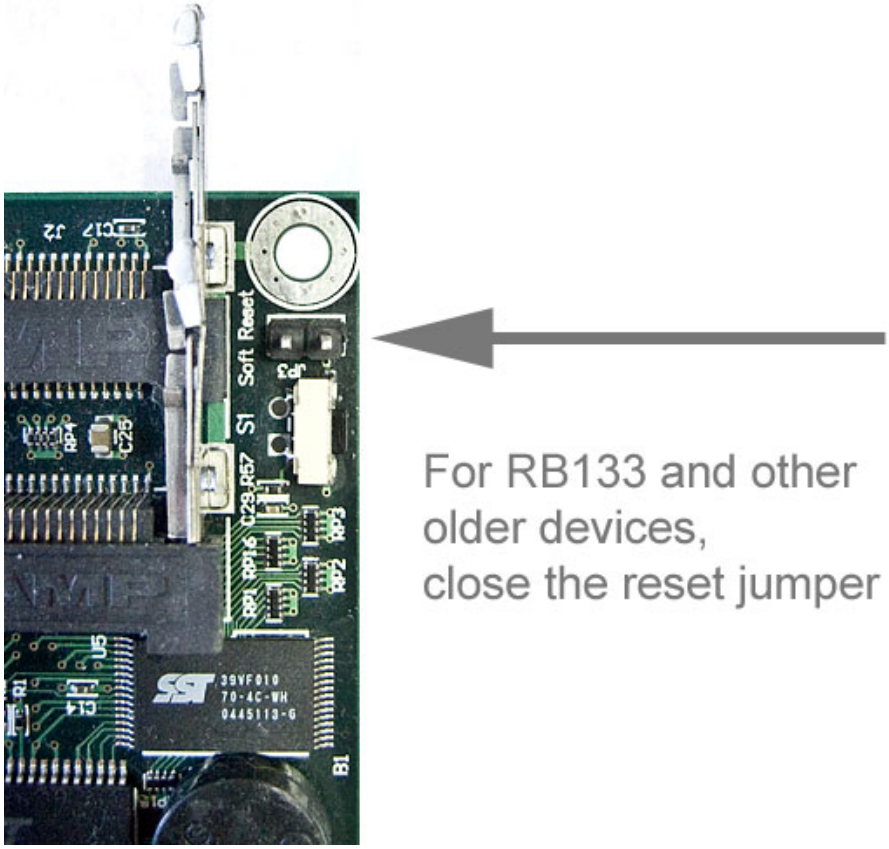
All RouterBOARD current models are also fitted with a reset jumper hole. Some devices might need opening of the enclosure, RB750/RB951/RB751 have the jumper hole under one of the rubber feet of the enclosure.

Using: Close the jumper with a metal screwdriver, and boot the board until the configuration is cleared.



Jumper reset for older models

The **below** image shows the location of the Reset Jumper on older RouterBOARDs like RB133C:



Note: Don't forget to remove the jumper after configuration has been reset, or it will be reset every time you reboot.

Manual:PCC

Applies to RouterOS: v3, v4

Introduction

PCC matcher will allow you to divide traffic into equal streams with ability to keep packets with specific set of options in one particular stream (you can specify this set of options from src-address, src-port, dst-address, dst-port)

Theory

PCC takes selected fields from IP header, and with the help of a hashing algorithm converts selected fields into 32-bit value. This value then is divided by a specified *Denominator* and the remainder then is compared to a specified *Remainder*, if equal then packet will be captured. You can choose from src-address, dst-address, src-port, dst-port from the header to use in this operation.

```
per-connection-classifier=
PerConnectionClassifier ::= [!]ValuesToHash:Denominator/Remainder
  Remainder ::= 0..4294967295    (integer number)
  Denominator ::= 1..4294967295    (integer number)
  ValuesToHash ::= both-addresses|both-ports|dst-address-and-port|
  src-address|src-port|both-addresses-and-ports|dst-address|dst-port|src-address-and-port
```

Example

This configuration will divide all connections into 3 groups based on source address and port

```
/ip firewall mangle add chain=prerouting action=mark-connection \
 new-connection-mark=1st_conn per-connection-classifier=src-address-and-port:3/0
/ip firewall mangle add chain=prerouting action=mark-connection \
  new-connection-mark=2nd_conn per-connection-classifier=src-address-and-port:3/1
/ip firewall mangle add chain=prerouting action=mark-connection \
  new-connection-mark=3rd_conn per-connection-classifier=src-address-and-port:3/2
```

Notes

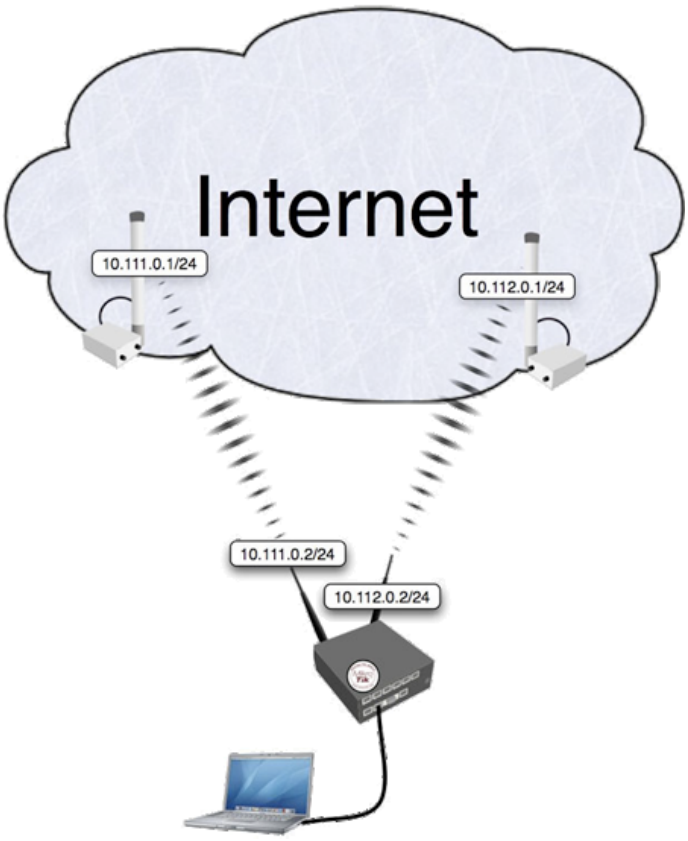
PCC is available in RouterOS since v3.24. This option was introduced to address configuration issues with load balancing over multiple gateways with masquerade

Previous configurations:

- ECMP load balancing with masquerade
- NTH load balancing with masquerade
- NTH load balancing with masquerade (another approach)

Application Example - Load Balancing

Consider the following network layout:



Quick Start for Impatient

Configuration export from the gateway router:

```
/ ip address
add address=192.168.0.1/24 network=192.168.0.0 broadcast=192.168.0.255 interface=LAN
add address=10.111.0.2/24 network=10.111.0.0 broadcast=10.111.0.255 interface=ISP1
add address=10.112.0.2/24 network=10.112.0.0 broadcast=10.112.0.255 interface=ISP2

/ ip firewall mangle
add chain=prerouting dst-address=10.111.0.0/24  action=accept in-interface=LAN
add chain=prerouting dst-address=10.112.0.0/24  action=accept in-interface=LAN
add chain=prerouting in-interface=ISP1 connection-mark=no-mark action=mark-connection \
    new-connection-mark=ISP1_conn
add chain=prerouting in-interface=ISP2 connection-mark=no-mark action=mark-connection \
    new-connection-mark=ISP2_conn
add chain=prerouting  in-interface=LAN connection-mark=no-mark dst-address-type=!local \
    per-connection-classifier=both-addresses:2/0 action=mark-connection new-connection-mark=ISP1_conn
add chain=prerouting  in-interface=LAN connection-mark=no-mark dst-address-type=!local \
    per-connection-classifier=both-addresses:2/1 action=mark-connection new-connection-mark=ISP2_conn
add chain=prerouting connection-mark=ISP1_conn in-interface=LAN action=mark-routing \
    new-routing-mark=to_ISP1
add chain=prerouting connection-mark=ISP2_conn in-interface=LAN action=mark-routing \
    new-routing-mark=to_ISP2
```

```
add chain=output connection-mark=ISP1_conn action=mark-routing new-routing-mark=to_ISP1
add chain=output connection-mark=ISP2_conn action=mark-routing new-routing-mark=to_ISP2

/ ip route
add dst-address=0.0.0.0/0 gateway=10.111.0.1 routing-mark=to_ISP1 check-gateway=ping
add dst-address=0.0.0.0/0 gateway=10.112.0.1 routing-mark=to_ISP2 check-gateway=ping
add dst-address=0.0.0.0/0 gateway=10.111.0.1 distance=1 check-gateway=ping
add dst-address=0.0.0.0/0 gateway=10.112.0.1 distance=2 check-gateway=ping

/ ip firewall nat
add chain=srcnat out-interface=ISP1 action=masquerade
add chain=srcnat out-interface=ISP2 action=masquerade
```

Explanation

Let's assume this configuration:

IP Addresses

```
/ ip address
add address=192.168.0.1/24 network=192.168.0.0 broadcast=192.168.0.255 interface=LAN
add address=10.111.0.2/24 network=10.111.0.0 broadcast=10.111.0.255 interface=ISP1
add address=10.112.0.2/24 network=10.112.0.0 broadcast=10.112.0.255 interface=ISP2
```

The router has two upstream (ISP) interfaces with the addresses of 10.111.0.2/24 and 10.112.0.2/24. The LAN interface has IP address of 192.168.0.1/24.

Policy routing

```
/ ip firewall mangle
add chain=prerouting dst-address=10.111.0.0/24  action=accept in-interface=LAN
add chain=prerouting dst-address=10.112.0.0/24  action=accept in-interface=LAN
```

With policy routing it is possible to force all traffic to the specific gateway, even if traffic is destined to the host (other that gateway) from the connected networks. This way routing loop will be generated and communications with those hosts will be impossible. To avoid this situation we need to allow usage of default routing table for traffic to connected networks.

```
add chain=prerouting in-interface=ISP1 connection-mark=no-mark action=mark-connection \
    new-connection-mark=ISP1_conn
add chain=prerouting in-interface=ISP2 connection-mark=no-mark action=mark-connection \
    new-connection-mark=ISP2_conn
```

First it is necessary to manage connection initiated from outside - replies must leave via same interface (from same Public IP) request came. We will mark all new incoming connections, to remember what was the interface.

```
add chain=prerouting  in-interface=LAN connection-mark=no-mark dst-address-type=!local \
    per-connection-classifier=both-addresses:2/0 action=mark-connection new-connection-mark=ISP1_conn
add chain=prerouting  in-interface=LAN connection-mark=no-mark dst-address-type=!local \
    per-connection-classifier=both-addresses:2/1 action=mark-connection new-connection-mark=ISP2_conn
```

Action mark-routing can be used only in mangle chain **output** and **prerouting**, but mangle chain **prerouting** is capturing all traffic that is going to the router itself. To avoid this we will use *dst-address-type=!local*. And with the

help of the new PCC we will divide traffic into two groups based on source and destination addressees.

```
add chain=prerouting connection-mark=ISP1_conn in-interface=LAN action=mark-routing \
    new-routing-mark=to_ISP1
add chain=prerouting connection-mark=ISP2_conn in-interface=LAN action=mark-routing \
    new-routing-mark=to_ISP2
add chain=output connection-mark=ISP1_conn action=mark-routing new-routing-mark=to_ISP1
add chain=output connection-mark=ISP2_conn action=mark-routing new-routing-mark=to_ISP2
```

Then we need to mark all packets from those connections with a proper mark. As policy routing is required only for traffic going to the Internet, do not forget to specify in-interface option.

```
/ ip route
add dst-address=0.0.0.0/0 gateway=10.111.0.1 routing-mark=to_ISP1 check-gateway=ping
add dst-address=0.0.0.0/0 gateway=10.112.0.1 routing-mark=to_ISP2 check-gateway=ping
```

Create a route for each routing-mark

```
add dst-address=0.0.0.0/0 gateway=10.111.0.1 distance=1 check-gateway=ping
add dst-address=0.0.0.0/0 gateway=10.112.0.1 distance=2 check-gateway=ping
```

To enable failover, it is necessary to have routes that will jump in as soon as others will become inactive on gateway failure. (and that will happen only if check-gateway option is active)

NAT

```
/ ip firewall nat
add chain=srcnat out-interface=ISP1 action=masquerade
add chain=srcnat out-interface=ISP2 action=masquerade
```

As routing decision is already made we just need rules that will fix src-addresses for all outgoing packets. If this packet will leave via wlan1 it will be NATed to 10.112.0.2, if via wlan2 then NATed to 10.111.0.2

Manual:PoE-Out

Summary

this page describes how poe-out feature can be used on *RB750UP* [1] and *OmniTik UPA-5HnD* [2]. These routers provided power only over the spare pairs.

This feature will become available on new device featuring SwOS interface that will also provide power only over spare pairs. It will use new PoE-out controller firmware.

Warning: When updated to newer PoE-Out controller firmware version it is not possible to downgrade it

Note: RouterOS is scapable to work with 1.x and 2.x firmware releases

Configuration

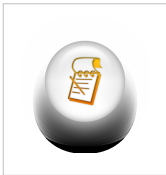
RouterOS

Global settings

Global PoE settings (affects all ports) can be configured in */interface ethernet poe settings* menu.

Property	Desciption
version	version of PoE controller firmware. It is independent of RouterOS and bootloader version.
ether1-poe-in-long-cable (*default: no*)	setting to yes will disable short detection on all poe-out ports to enable use of longer ethernet cables. This is potentially dangerous settings and should be used with caution.

Using command ***/interface ethernet poe settings upgrade*** it is possible to update poe-out controller firmware. This firmware is not available separately from RouterOS, so newest version of RouterOS should be used.



Note: New PoE-Out controler firmware is available since RouterOS 5.20 release.

Port settings

PoE Out can be configured in `/interface ethernet` menu or in */interface ethernet poe* menu. Each port can be controlled independently.

Power on 100MBps interface is provided over spare pairs (blue and brown) where blue pair carries positive and brown - negative.

Note: Since RouterOS 5.20 it is possible to upgrade PoE-Out controller firmware to newer version. Doing so will enable certain new features. Use of older firmware with new RouterOS releases is possible

Old firmware supported configuration features:

Configuration options are:

- ***auto-on*** - board will attempt to detect if power can be applied on the port. For power-on to happen there should be resistance on spare pairs in range from 3kΩ to 26.5kΩ. It is the default setting;

- ***forced-on*** - detection range is removed. As a result power over Ethernet will be always on;
- ***off*** - all detection and power is turned off for this port.

Starting PoE-Out controller firmware 2.0 these configuration options are available:

All of the configuration can be done via */interface ethernet poe* menu

Property	Desciption
poe-out (*default: auto-on*)	• ***auto-on*** - board will attempt to detect if power can be applied on the port. For power-on to happen there should be resistance on spare pairs in range from 3kΩ to 26.5kΩ. It is the default setting; • ***forced-on*** - detection range is removed. As a result power over Ethernet will be always on; • ***off*** - all detection and power is turned off for this port.
poe-priority (*0 .. 99, default: 10*)	set port priority regarding other PoE-Out ports. Highest priority is 0, lowest priority is 99. If there are 2 or more ports with similar priority then port with smallest port number will have higher priority. For example, if ether2 and ether3 are of same priority and over-current is detected then PoE-Out on ether3 will be turned off. Every 6 seconds ports will be checked for possibility to provide PoE-Out if it was turned off due to port priority.

Command	Desciption
print	prints all the properties in menu
upgrade	flash PoE-Out controller with firmware included into RouterOS.
export	export setting of the current menu.

SwOS

PoE Out feature configuration will be available from SwOS Link page where one of PoE Out profiles can be for each PoE Out enabled port individually:

- ***auto*** - detection is done regarding resistance on the spare pairs to check if port has PoE capability. For port to be turned on measured value should be within range from 3kΩ to 26.5kΩ;
- ***on*** - PoE Out is enabled on the port regardless of the resistance on the port. Use this with caution as that can damage connected equipment;
- ***off*** - PoE Out is turned off.

Monitoring

New PoE Out controller firmware enables monitoring of the PoE Out state

RouterOS

It is possible to monitor poe-out current usage in RouterOS via */interface ethernet poe* menu. Monitoring currently will show power usage in watts, current load in amperes and supplied voltage.

SwOS

PoE Out controller with new firmware will enable certain monitoring features such as PoE Out state, current power usage and voltage supplied to the board:

- ***PoE Out state*** - will enable to see PoE Out state remotely just by logging into the board
- ***Current power usage*** - measured in Amperes will show current usage on the port
- ***Supplied voltage*** - as this value directly impacts how much power can be given to the connected devices it is important to know value that is available to the SwOS device itself.

How it works

auto-on mode

If *auto-on* is selected then port operates in this strict order:

- with low voltage it checks for resistance on the connected port. If it is detected that resistance is in range (3kΩ to 26.5kΩ) power is turned on;
- when power is given it is continuously checked if overload limit is not reached or short circuit is detected
- when cable is unplugged port returns in detection state and will remain off until suitable port that could accept PoE is detected

forced-on mode

If *forced-on* is selected then port operates in this strict order:

- it is checked if resistance is in range is 0Ω to ∞Ω, so that even if cable is unplugged power will be given on the port
- when power is given it is continuously checked if overload limit is not reached or short circuit is detected
- when cable is unplugged port remains with power enabled on the port

off mode

If *off* mode is used, poe-out on the port will be turned off at all times and port on the router will behave as simple Ethernet port.

Safety

PoE out feature has the following safety features:

Port eligibility detection for poe out

auto-on mode is considered safe, it will check if resistance on the port is within range and only then enable PoE out on the interface. The range is 3kΩ to 26.5kΩ

Overload protection

when port is on (PoE out is enabled) port is checked for overload, If that is detected to avoid hardware damage of powered device or powering device PoE out is turned of on this port.

With new firmware after overload is detected starts 120 second countdown after what PoE Out feature will be turned on again to see if environment has changed and device connected can be supplied with power again. That is important for configurations that are not connected to mains (solar installations, equipment running on batteries due to mains failure) so that when voltage drops - overload will be detected and connected devices turned off. After a while when voltage level returns to usual operating value - connected equipment can be powered on again.

Note: PoE-Out controller firmware version 2.0 allows 1A load on port with 2.2A of total limit on all ports

Short circuit detection

when port is on (PoE out is enabled) port is checked for short circuit. If it is detected to ensure that there is no additional damage on powered device and no damage on powering device PoE out is turned off on all ports

PoE Out polarity

By default PoE out polarity on MikroTik manufactured devices on 100MBps ports is standard

Standard

PoE out on RouterBOARD devices is standard - over blue pair - positive and over brown pair - negative voltage

Reversed

Some devices use different polarity to power their devices that is reverse of the standard: over blue pair - negative and over brown pair - positive voltage

Any

Device that supports either of 2 possible polarities - standard or reversed

Wiring

If you want to power device that supports only reverse polarity using 'RB750UP [1] *and* OmniTik UPA [2] *you have to make special cable that will swap required wires, thus enabling you to power these devices.*

On one end you have to have standard RJ45 T568B wiring (orange striped, orange, green striped, blue, blue striped, green, brown striped, brown) and on other and has to have blue and brown pair swapped resulting in (orange striped, orange, green striped, brown, brown striped, green, blue striped, blue)

RB750UP [1] and *OmniTik UPA-5HnD* [2] Powered devices

Table contain information on what devices can be powered using *RB750UP* [1] and *OmniTik UPA-5HnD* [2] as power source for PoE. Table contains devices that are explicitly tested. Routers from same series or with modifications will also work.

Polarity: standard - blue pair positive, brown pair negative; reversed - blue pair negative , brown pair positive, any - either standard or reversed.

Device	forced-on	auto-on	polarity
RouterBOARD RB433GL [3]	yes	yes	any
RouterBOARD RB450G [4]	yes	yes	any
OmniTik UPA-5HnD [2]	yes	yes	standard
OmniTik U-5HnD [5]	yes	yes	standard
RouterBOARD RB711G-5HnD [6]	yes	yes	any
RouterBOARD RB411U [7]	yes	yes	standard
RouterBOARD RB800 [8]	no	no	any
RouterBOARD RB450 [9]	yes	yes	standard
RouterBOARD RB433AH [10]	yes	yes	standard
RouterBOARD RB750 [11]	yes	yes	standard
RouterBOARD RB750UP [1]	yes	yes	standard
RouterBOARD SXT[12]	yes	yes	standard

RouterBOARD RB435G [13]	yes	no	any
RouterBOARD RB493AH [14]	yes	yes	standard
RouterBOARD RB493 [15]	yes	yes	standard
RouterBOARD RB532	yes	yes	standard
RouterBOARD RB153	yes	yes	standard
RouterBOARD RB112	yes	yes	standard
Motorolla Canopy	yes	yes	reversed
Ubiquity LOCO900	yes	yes	standard
Ubiquity Bullet M2	yes	yes	standard
Ubiquity Bullet M5	yes	yes	standard
Ubiquity NanoStation 2N	yes	yes	standard
Ubiquity NanoStation M365	yes	yes	standard
Ubiquity UniFi AP	yes	yes	standard
Ubiquity UniFi AP long range	yes	yes	standard

Upgrade PoE Out firmware

PoE-Out controller firmware is not provided separately from RouterOS. For newer PoE-Out firmware it is required to install newer RouterOS release.

Note: Firmware upgrade is possible since RouterOS 5.20

Upgrade procedure is as follows:

- upgrade router with newest RouterOS release.
- check current firmware using */interface ethernet poe settings print* command
- to upgrade issue */interface ethernet poe settings upgrade* command

- when prompted with this message follow instructions as required

```
 Do you really want to upgrade PoE firmware? [y/n]
```

- you have to reboot the router for changes to take place

```
Please reboot to finish PoE firmware upgrade.
```

- reboot the router

PoE Out examples

PoE-Out firmware starting from 2.0 allows to set port priority. Priority of *0* is the highest priority there is, so if your installation is going over power budged port with the lowest priority is going to be turned off first.

setting up priority

here is example on how priorities look

setting up priorities:

```
/interface ethernet poe set ether2 poe-priority=10
/interface ethernet poe set ether3 poe-priority=13
/interface ethernet poe set ether4 poe-priority=11
/interface ethernet poe set ether5 poe-priority=14
```

what will happen when power budget will go over set total limit - first, if over-load is detected ether5 will be turned off, then recheck is done and if still total limit over-load is detected next port in priority will be turned off, in this example, ether3 will have its poe-out turned off. Both of these ports will be reched every 6 seconds is it possible to turn poe-out on for these ports. Power up will happen in reverse order as the power was cut.

same priority

if all, or some ports will have same poe-priority than port with lower port number will have higher priority

```
/interface ethernet poe set ether2 poe-priority=10
/interface ethernet poe set ether3 poe-priority=10
/interface ethernet poe set ether4 poe-priority=10
/interface ethernet poe set ether5 poe-priority=10
```

in this example, if over-load total limit is reached ether5 will be turned off first, then ether4 then ether3 as all of these ports have same poe priority.

monitoring poe-out

all the monitoring of poe-out can be done via */interface ethernet poe monitor <interface>* menu

```
[admin@MikroTik] > interface ethernet poe monitor [find]
              name: ether2 ether3 ether4 ether5
   poe-out-voltage: 23.2V  23.2V  23.2V
   poe-out-current: 224mA  116mA  64mA
     poe-out-power: 5.1W   2.6W   1.4W
```

here RB750UP is powering up 3 devices on ports ether2, ether3 and ether4

Changes between 1.x and 2.0 PoE-Out controller firmware

feature	version 1.x	version 2.0
max limit on port	500mA	1A
total limit	2A	2.2A
poe-priority	not available	per port priority adjustable
monitoring	not available	information about each port
PoE in long cable mode	not available	available
Check PoE controller FW version	newer RouterOS	available

References

[1] http://routerboard.com/RB750UP
[2] http://routerboard.com/RBOMNITIKUPA5HnD
[3] http://routerboard.com/RB433GL
[4] http://routerboard.com/RB450G
[5] http://routerboard.com/OMN5HnD
[6] http://routerboard.com/RB711G-5HnD
[7] http://routerboard.com/RB411U
[8] http://routerboard.com/RB800
[9] http://routerboard.com/RB450
[10] http://routerboard.com/RB433AH
[11] http://routerboard.com/RB750
[12] http://routerboard.com/RBSXTG-5HnD
[13] http://routerboard.com/RB435G
[14] http://routerboard.com/RB493AH
[15] http://routerboard.com/RB493

Manual:IP/Pools

Applies to RouterOS: 2.9, v3, v4 +

IP pools are used to define range of IP addresses that is used for DHCP server and Point-to-Point servers

Specifications

- Packages required: system
- License required: Level1
- Submenu level: /ip pool
- Standards and Technologies: none
- Hardware usage: Not significant

Description

IP pools simply group IP addresses for further usage. It is a single configuration point for all features that assign IP addresses to clients.

Note: Whenever possible, the same ip address is given out to each client (OWNER/INFO pair).

Setup

Sub-menu: `/ip pool`

Property Description

- **name** (name) - the name of the pool
- **next-pool** (name) - when address is acquired from pool that has no free addresses, and next-pool property is set to another pool, then next IP address will be acquired from next-pool
- **ranges** (IP address) - IP address list of non-overlapping IP address ranges in form of: from1-to1,from2-to2,...,fromN-toN. For example, 10.0.0.1-10.0.0.27,10.0.0.32-10.0.0.47

Example

To define a pool named ip-pool with the 10.0.0.1-10.0.0.125 address range excluding gateway's address 10.0.0.1 and server's address 10.0.0.100, and the other pool dhcp-pool, with the 10.0.0.200-10.0.0.250 address range:

```
[admin@MikroTik] ip pool> add name=ip-pool ranges=10.0.0.2-10.0.0.99,10.0.0.101
10.0.0.126
[admin@MikroTik] ip pool> add name=dhcp-pool ranges=10.0.0.200-10.0.0.250
[admin@MikroTik] ip pool> print
  # NAME                                      RANGES
  0 ip-pool                                   10.0.0.2-10.0.0.99
                                              10.0.0.101-10.0.0.126
  1 dhcp-pool                                 10.0.0.200-10.0.0.250

[admin@MikroTik] ip pool>
```

Used Addresses from Pool

- Submenu level: **/ip pool used**

Description

Here you can see all used IP addresses from IP pools.

Property Description

- **address** (read-only: IP address) - IP address that is assigned to client form the pool
- **info** (read-only: name) - name of the interface to which the client is connected to
- **owner** (read-only: MAC address) - MAC address of the client
- **pool** (read-only: name) - name of the IP pool

Example

See used addresses from pool:

```
[admin@MikroTik] ip pool used> print
POOL  ADDRESS         OWNER                     INFO
local 192.168.0.100   00:0C:42:03:1F:60         test
local 192.168.0.99    00:0C:42:03:21:0F         test
```

[Top | Back to Content]

Manual:IPv6/Pool

Applies to RouterOS: v5.7+

Summary

Sub-menu: `/ipv6 pool`
Standards:
Package : `IPv6`

IPv6 pools are used to define range of IPv6 addresses that is used for DHCPv6 server and Point-to-Point servers

IPv6 pools simply group IPv6 addresses for further usage. It is a single configuration point for all features that assign IPv6 addresses to clients.

Pool Configuration

Property	Description
name (*string*; Default:)	Descriptive name of the pool.
prefix (*IPv6/0..128*; Default:)	Ipv6 address prefix
prefix-length (*integer [1..128]*; Default:)	Option represents the prefix size that will be give out to the client.

Read-only properties

Property	Description
dynamic (*yes* | *no*)	Whether pool is dynamic.
id (*integer*)	
expire-time (*time*)	Expire time is set to dynamic pools added by DHCPv6 client.

Example

Define a pool named "test" with prefix "2001::/64":

```
[admin@test-host] /ipv6 pool> add
name: test
prefix: 2001::/60
prefix-length: 62
[admin@test-host] /ipv6 pool> print
 # NAME                  PREFIX                                          PREFIX-LENGTH
 0 test                  2001::/60                                            62bits
[admin@test-host] /ipv6 pool>
```

Used Addresses from Pool

Sub-menu: `/ipv6 pool used`

In this menu you can see all used IPv6 addresses from the pools.

All properties are read-only.

Property	Description
info (*string*)	Shows DUID related information received from client (value in hex).Can contain also raw timestamp in hex.
owner (*string*)	What reserved the prefix ("DHCP", etc.)
pool (*string*)	Name of the pool.
prefix (*IPv6/0..128*)	IPv6 prefix that is assigned to client form the pool.

[Top | Back to Content]

Manual:Port

Applies to RouterOS: v5+

Summary

There are many ways how to use ports on the routers. Most obvious one is to use serial port for initial RouterOS configuration after installation(by default serial0 is used by serial-terminal).

Serial and USB ports can also be used to:

- connect 3G modems;
- connect to another device through a serial cable
- access device connected to serial cable remotely.

General

Sub-menu: `/port`

Menu lists all available serial, usb, ... ports on the router and allows to configure port parameters, like baud-rate, flow-control, etc.

Below you can see default port configuration on RB493.

```
[admin@RB493G] /port> print
Flags: I - inactive
 #    NAME                            CHANNELS USED-BY                        BAUD-RATE
 0    serial0                                1 serial-terminal                 115200
```

Note: List of the ports are maintained automatically by the RouterOS.

Properties

Property	Description
baud-rate (*integer* | *auto*; Default: **auto**)	Baud rate (speed) used by the port. If set to **auto**, then RouterOS tries to detect baud rate automatically.
data-bits (*7* | *8*; Default:)	The number of data bits in each character. • **7** - true ASCII • **8** - any data (matches the size of a byte)
dtr (*on* | *off*; Default:)	Whether to enable RS-232 DTR signal circuit used by flow control.
flow-control (*hardware* | *none* | *xon-xoff*; Default:)	method of flow control to pause and resume the transmission of data.
name (*string*; Default:)	Name of the port.
parity (*even* | *none* | *odd*; Default:)	Error detection method. If enabled, extra bit is sent to detect the communication errors. In most cases parity is set to **none** and errors are handled by the communication protocol.
rts (*on* | *off*; Default:)	Whether to enable RS-232 RTS signal circuit used by flow control.
stop-bits (*1* | *2*; Default:)	Stop bits sent after each character. Electronic devices usually uses 1 stop bit.

Read-only properties

Property	Description
channels (*integer*)	Number of channels supported by the port.
inactive (*yes* | *no*)	
line-state ()	
used-by (*string*)	Shows what is using current port. For example, by default **Serial0** is used by serial-console.

Firmware

Sub-menu: `/port firmware`

This submenu allows to specify **directory** where drivers for 3g modems can be uploaded and used.

Remote Access

Sub-menu: `/port remote-access`

If you want to access serial device that can only talk to COM ports and is located somewhere else behind router, then you can use remote-access.

As defined in RFC 2217 RouterOS can transfer data from/to a serial device over TCP connection.

Enabling remote access on RouterOS is very easy:

```
/port remote-access add port=serial0 protocol=rfc2217 tcp-port=9999
```

Note: By default serial0 is used by serial-terminal. Without releasing the port, it cannot be used by remote-access or other services

Properties

Property	Description
allowed-addresses (*IP address range*; Default: **0.0.0.0/0**)	Range of IP addresses allowed to access port remotely.
channel (*integer [0..4294967295]*; Default: **0**)	Port channel that will be used. If port has only one channel then channel number should always be **0**.
disabled (*yes* | *no*; Default: **no**)	
local-address (*IP address*; Default:)	IP address used as source address.
log-file (*string*; Default: "")	Name of the file, where communication will be logged. By default logging is disabled.
port (*string*; Default:)	Name of the port from Port list.
protocol (*raw* | *rfc2217*; Default: **rfc2217**)	RFC 2217 defines a protocol to transfer data from/to a serial device over TCP. If set to **raw**, then data is sent to serial as is.
tcp-port (*integer [1..65535]*; Default: **0**)	TCP port on which to listen for incoming connections.

Read-only properties

Property	Description
active (*yes* | *no*)	Whether remote access is active and ready to accept connection.
busy (*yes* | *no*)	Whether port is currently busy.
inactive (*yes* | *no*)	
logging-active (*yes* | *no*)	Whether logging to file is currently running
remote-address (*IP address*)	IP address of remote location that is currently connected.

See More

- Special Login
- Serial Console
- Serial Port Usage

[Top | Back to Content]

Manual:IPv6 PD over PPP

Applies to RouterOS: v5.9

Summary

This example demonstrates how to set up PPPoE server and client to use IPv6 Prefix Delegation.

Starting from v5.9 IPv6 Prefixes can be delegated over ppp interfaces. When client connects, ppp will automatically add dynamic DHCPv6-PD server. This allows to run dhcpv6 client on ppp interfaces.

Configuration

Server

There is a new parameter in ppp profile called `dhcpv6-pd-pool` used to enable PPP-PD. PPP will use specified IPv6 pool to create dynamic DHCP server.

So first step is to add IPv6 pool:

```
/ipv6 pool
add name=myPool prefix=2001:db8:7501:ff00::/60 prefix-length=62
```

Now we can configure ppp profile and add pppoe server

```
/ppp profile set default dhcpv6-pd-pool=myPool

/interface pppoe-server server
add service-name=test interface=ether1
```

Client

On client side we need to set up pppoe-client interface and run dhcp client on it.

```
/interface pppoe-client
add name=client-test interface=ether1 user=a1 service-name=test

/ipv6 dhcp-client
add interface=client-test pool-name=ppp-test pool-refix-length=64
```

Testing status

On server side check if dynamic DHCP server is added and prefix is bound to specific client:

```
[admin@RB1100] /ipv6 dhcp-server> print
Flags: D - dynamic, X - disabled, I - invalid
 #    NAME              INTERFACE            ADDRESS-POOL             LEASE-TIME
 0 D  <pppoe-a1>        <pppoe-a1>           myPool                   3d

[admin@RB1100] /ipv6 dhcp-server binding> print
Flags: X - disabled, D - dynamic
 #   ADDRESS                                       DU       IAID SER.. STATUS
 1 D 2001:db8:7501:ff04::/62                        247 <pp.. bound
```

On client side check if dhcp client is bound and pool is added

```
[admin@x86-test] /ipv6 dhcp-client> print
Flags: D - dynamic, X - disabled, I - invalid
 #    INTERFACE           STATUS        PREFIX                          EXPIRES-AFTER
0    client-test          bound         2001:db8:7501:ff04::/62         2d23h18m17s

[admin@x86-test] /ipv6 pool> print
Flags: D - dynamic
 #   NAME                      PREFIX                                  PREFIX-LENGTH
 0 D ppp-test                  2001:db8:7501:ff04::/62                            64
```

[Top | Back to Content]

Manual:PPP AAA

Applies to RouterOS: 2.9, v3, v4, v5

Summary

Sub-menu: `/ppp`

The MikroTik RouterOS provides scalable Authentication, Athorization and Accounting (AAA) functionality.

Local authentication is performed using the User Database and the Profile Database. The actual configuration for the given user is composed using respective user record from the User Database, associated item from the Profile Database and the item in the Profile database which is set as default for a given service the user is authenticating to. Default profile settings from the Profile database have lowest priority while the user access record settings from the User Database have highest priority with the only exception being particular IP addresses take precedence over IP pools in the local-address and remote-address settings, which described later on.

Support for RADIUS authentication gives the ISP or network administrator the ability to manage PPP user access and accounting from one server throughout a large network. The MikroTik RouterOS has a RADIUS client which can authenticate for PPP, PPPoE, PPTP, L2TP and ISDN connections. The attributes received from RADIUS server override the ones set in the default profile, but if some parameters are not received they are taken from the respective default profile.

User Profiles

Sub-menu: `/ppp profile`

PPP profiles are used to define default values for user access records stored under `/ppp secret` submenu. Settings in `/ppp secret` User Database override corresponding `/ppp profile` settings except that single IP addresses always take precedence over IP pools when specified as local-address or remote-address parameters.

Properties

Property	Description
address-list (*string*; Default:)	Address list name to which ppp assigned address will be added.
bridge (*string*; Default:)	Name of the bridge interface to which ppp interface will be added as slave port.
change-tcp-mss (*yes* | *no* | *default*; Default: **default**)	Modifies connection MSS settings: • `yes` - adjust connection MSS value • `no` - do not atjust connection MSS value • `default` - derive this value from the interface default profile; same as no if this is the interface default profile
comment (*string*; Default:)	
dhcpv6-pd-pool (*string*; Default:)	Name of the IPv6 pool which will be used by dynamically created DHCPv6-PD server when client connects. `Read more >>`
dns-server (*IP*; Default:)	IP address of the DNS server that is supplied to ppp clients
idle-timeout (*time*; Default:)	Specifies the amount of time after which the link will be terminated if there are no activity present. Timeout is not set by default

incoming-filter (*string*; Default:)
Firewall chain name for incoming packets. Specified chain gets control for each packet coming from the client. The ppp chain should be manually added and rules with action=jump jump-target=ppp should be added to other relevant chains in order for this feature to work. For more information look at the examples section

local-address (*IP address* | *pool*; Default:)
Tunnel address or name of the pool from which address is assigned to ppp interface locally.

name (*string*; Default:)
PPP profile name

only-one (*yes* | *no* | *default*; Default: **default**)
Defines whether a user is allowed to have more than one connection at a time

- `yes` - a user is not allowed to have more than one connection at a time
- `no` - the user is allowed to have more than one connection at a time
- `default` - derive this value from the interface default profile; same as no if this is the interface default profile

outgoing-filter (*string*; Default:)
Firewall chain name for outgoing packets. Specified chain gets control for each packet going to the client. The ppp chain should be manually added and rules with action=jump jump-target=ppp should be added to other relevant chains in order for this feature to work. For more information look at the Examples section.

rate-limit (*string*; Default:)
Rate limitation in form of **rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time] [priority] [rx-rate-min[/tx-rate-min]]]]** from the point of view of the router (so "rx" is client upload, and "tx" is client download). All rates are measured in bits per second, unless followed by optional 'k' suffix (kilobits per second) or 'M' suffix (megabits per second). If tx-rate is not specified, rx-rate serves as tx-rate too. The same applies for tx-burst-rate, tx-burst-threshold and tx-burst-time. If both rx-burst-threshold and tx-burst-threshold are not specified (but burst-rate is specified), rx-rate and tx-rate are used as burst thresholds. If both rx-burst-time and tx-burst-time are not specified, 1s is used as default. Priority takes values 1..8, where 1 implies the highest priority, but 8 - the lowest. If rx-rate-min and tx-rate-min are not specified rx-rate and tx-rate values are used. The rx-rate-min and tx-rate-min values can not exceed rx-rate and tx-rate values.

remote-address (*IP*; Default:)
Tunnel address or name of the pool from which address is assigned to remote ppp interface.

remote-ipv6-prefix-pool (*string* | *none*; Default: **none**)
Assign prefix from IPv6 pool to the client and install corresponding IPv6 route.

session-timeout (*time*; Default:)
Maximum time the connection can stay up. By default no time limit is set.

use-compression (*yes* | *no* | *default*; Default: **default**)
Specifies whether to use data compression or not.

- `yes` - enable data compression
- `no` - disable data compression
- `default` - derive this value from the interface default profile; same as no if this is the interface default profile

use-encryption (*yes* | *no* | *default* | *require*; Default: **default**)
Specifies whether to use data encryption or not.

- `yes` - enable data encryption
- `no` - disable data encryption
- `default` - derive this value from the interface default profile; same as no if this is the interface default profile
- `require` - explicitly requires encryption

use-ipv6 (*yes* | *no* | *default* | *require*; Default: **default**)
Specifies whether to allow IPv6. By default is enabled if IPv6 package is installed.

- `yes` - enable IPv6 support
- `no` - disable IPv6 support
- `default` - derive this value from the interface default profile; same as **no** if this is the interface default profile
- `require` - explicitly requires IPv6 support

use-mpls (*yes* | *no* | *default* | *require*; Default: **default**)
Specifies whether to allow MPLS over PPP.
- `yes` - enable MPLS support
- `no` - disable MPLS support
- `default` - derive this value from the interface default profile; same as **no** if this is the interface default profile
- `require` - explicitly requires MPLS support

use-vj-compression (*yes* | *no* | *default*; Default: **default**)
Specifies whether to use Van Jacobson header compression algorithm.
- `yes` - enable Van Jacobson header compression
- `no` - disable Van Jacobson header compression
- `default` - derive this value from the interface default profile; same as no if this is the interface default profile

wins-server (*IP address*; Default:)
IP address of the WINS server to supply to Windows clients

Notes

There are two default profiles that cannot be removed:

```
[admin@rb13] ppp profile> print
Flags: * - default
 0 * name="default" use-compression=no use-vj-compression=no use-encryption=no only-one=no
     change-tcp-mss=yes
 1 * name="default-encryption" use-compression=default use-vj-compression=default use-encryption=yes
     only-one=default change-tcp-mss=default
[admin@rb13] ppp profile>
```

Use Van Jacobson compression only if you have to because it may slow down the communications on bad or congested channels.

incoming-filter and outgoing-filter arguments add dynamic jump rules to chain ppp, where the jump-target argument will be equal to incoming-filter or outgoing-filter argument in /ppp profile. Therefore, chain ppp should be manually added before changing these arguments.

`only-one` parameter is ignored if RADIUS authentication is used.

If there are more that 10 simultaneous PPP connections planned, it is recommended to turn the `change-mss` property off, and use one general MSS changing rule in mangle table instead, to reduce CPU utilization.

User Database

Sub-menu: `/ppp secret`

PPP User Database stores PPP user access records with PPP user profile assigned to each user.

Properties

Property	Description
caller-id (*string*; Default:)	For PPTP and L2TP it is the IP address a client must connect from. For PPPoE it is the MAC address (written in CAPITAL letters) a client must connect from. For ISDN it is the caller's number (that may or may not be provided by the operator) the client may dial-in from
comment (*string*; Default:)	Short description of the user.
disabled (*yes* | *no*; Default: **no**)	Whether secret will be used.
limit-bytes-in (*integer*; Default: **0**)	Maximal amount of bytes for a session that client can upload.
limit-bytes-out (*integer*; Default: **0**)	Maximal amount of bytes for a session that client can download.
local-address (*IP address*; Default:)	IP address that will be set locally on ppp interface.
name (*string*; Default:)	Name used for authentication
password (*string*; Default:)	Password used for authentication
profile (*string*; Default: **default**)	Which user profile to use.
remote-address (*IP*; Default:)	IP address that will be assigned to remote ppp interface.
remote-ipv6-prefix (*IPv6 prefix*; Default:)	IPv6 prefix assigned to ppp client. Prefix is added to ND prefix list enabling stateless address auto-configuration on ppp interface.Available starting from v5.0.
routes (*string*; Default:)	Routes that appear on the server when the client is connected. The route format is: dst-address gateway metric (for example, 10.1.0.0/ 24 10.0.0.1 1). Several routes may be specified separated with commas. This parameter will be ignored for OpenVPN.
service (*any* | *async* | *isdn* | *l2tp* | *pppoe* | *pptp* | *ovpn* | *sstp*; Default: **any**)	Specifies the services that particular user will be able to use.

Active Users

Sub-menu: `/ppp active`

This submenu allows to monitor active (connected) users.

`/ppp active print` command will show all currently connected users.

`/ppp active print stats` command will show received/sent bytes and packets

Properties

Property	Description
address (*IP address*)	IP address the client got from the server
bytes (*integer*)	Amount of bytes transfered through tis connection. First figure represents amount of transmitted traffic from the router's point of view, while the second one shows amount of received traffic.
caller-id (*string*)	For PPTP and L2TP it is the IP address the client connected from. For PPPoE it is the MAC address the client connected from.
encoding (*string*)	Shows encryption and encoding (separated with '/' if asymmetric) being used in this connection
limit-bytes-in (*integer*)	Maximal amount of bytes the user is allowed to send to the router.
limit-bytes-out (*integer*)	Maximal amount of bytes the user is allowed to send to the client.
name (*string*)	User name supplied at authentication stage
packets (*integer/integer*)	Amount of packets transfered through tis connection. First figure represents amount of transmitted traffic from the router's point of view, while the second one shows amount of received traffic
service (*async* | *isdn* | *l2tp* | *pppoe* | *pptp* | *ovpn* | *sstp*)	Type of service the user is using.

session-id (*string*)	Shows unique client identifier.
uptime (*time*)	User's uptime

Remote AAA

Sub-menu: `/ppp aaa`

Settings in this submenu allows to set RADIUS accounting and authentication. Note that RADIUS user database is consulted only if the required username is not found in local user database.

Properties

Property	Description
accounting (*yes* | *no*; Default: **yes**)	Enable RADIUS accounting
interim-update (*time*; Default: **0s**)	Interim-Update time interval
use-radius (*yes* | *no*; Default: **no**)	Enable user authentication via RADIUS. If entry in local secret database is not found, then client will be authenticated via RADIUS.

Examples

Add new profile

To add the profile ex that assigns the router itself the 10.0.0.1 address, and the addresses from the ex pool to the clients, filtering traffic coming from clients through mypppclients chain:

```
[admin@rb13] ppp profile> add name=ex local-address=10.0.0.1 remote-address=ex incoming-filter=mypppclients
[admin@rb13] ppp profile> print
Flags: * - default
 0 * name="default" use-compression=no use-vj-compression=no use-encryption=no only-one=no
     change-tcp-mss=yes
 1   name="ex" local-address=10.0.0.1 remote-address=ex use-compression=default
     use-vj-compression=default use-encryption=default only-one=default change-tcp-mss=default
     incoming-filter=mypppclients
 2 * name="default-encryption" use-compression=default use-vj-compression=default use-encryption=yes
     only-one=default change-tcp-mss=default
[admin@rb13] ppp profile>
```

Add new user

To add the user ex with password lkjrht and profile ex available for PPTP service only, enter the following command:

```
[admin@rb13] ppp secret> add name=ex password=lkjrht service=pptp profile=ex
[admin@rb13] ppp secret> print
Flags: X - disabled
 #   NAME                SERVICE CALLER-ID         PASSWORD          PROFILE            REMOTE-ADDRESS
 0   ex                  pptp                      lkjrht            ex                 0.0.0.0
[admin@rb13] ppp secret>
```

Manual:Routing/Prefix list

Applies to RouterOS: 2.9, v3, v4 +

Sub-menu: `/routing prefix-list`

Filtering by prefix list involves matching the prefixes of routes with those listed in the prefix list. When there is a match, the rule is used. The prefix lists can be used to filter out RIP routes, and are used if specified under `/routing rip interface`.

Property	Description
action (*accept* | *discard*; Default: **accept**)	action to perform on route matching the rule
chain (*string*; Default: **""**)	chain name to place this rule in. If a chain with the specified name does not exist it will be automatically created
invert-math (*yes* | *no*; Default: **no**)	invert this match, i.e. apply the rule to routes that would fail to match it and vice versa
prefix (*IP prefix*; Default: **0.0.0.0/0**)	network prefix to match. If prefix-length is not set, only exact match is done. For example, 0.0.0.0/0 then matches only the default route and nothing else

prefix-length (*integer*; Default: **0-32**)	network prefix mask length to match. If prefix-length is set, for a route to match the prefix and prefix-length of a rule, the following should hold:
	• the network prefix of the route falls within the range of the prefix of the rule, (i.e.
	  • the network mask of the route is greater of equal than the network mask of the prefix;
	  • the network address of the route masked out by the network mask of the prefix is equal to the network address of the prefix;)
	• the length of the network mask of the route falls within the range of the prefix-length

set-metric (*integer*; Default:)	Set metric

Proxylizer

Note: MikroTik has discontinued the Proxylizer project, it will no longer receive updates, and technical support will not be available

Introduction

- What is Proxylizer
- Features
- Architecture
- Requirements

Getting Started

- Download
- Install
- First report

Concepts

- Collected Log Data
- Reports
- Time calculation
- Created Reports
- Report Generation
- Report History
- Inactive Reports
- Report Editing
- IP Users
- Database Statistics

Web Page

- Setup
- Login
- Status
- Reports
- IP users
- Config

Proxylizer/Getting Started

Download

Scripts for install method 1

You can download proxylizer archive here [1]

VMware image download for install method 2

There are 2 ways to download this image file (318 MB) :

- direct download [2]
- torrent network [3]

Install

All the examples assume that Proxylizer server IP address is 10.1.1.2 and syslog-ng uses port 514 that is its default

The installation includes steps for setting up the following:

- Mikrotik router:
  - Web-proxy log export to remote host
- Proxylizer server:
  - Method 1
    - Required packages
    - Web page scripts
    - Permissions for directories
    - Syslog deamon
    - MySQL user for proxylizer database
    - Scheduled scripts for forwarding records and report generation
    - Database and web page access configuration
    - Mail sending configuration
  - Method 2

Mikrotik router

Web-proxy log export to remote host (Proxylizer server)

To forward logs from Mikrotik Router to Proxylizer server, open RouterOS console and type in the following commands (assuming that Proxylizer Server IP Address is 10.1.1.2):

```
/system logging action add name=sendToProxylizer target=remote remote=10.1.1.2:514
/system logging add topics=web-proxy,!debug action=sendToProxylizer
```

Note that logs are sent to port number 514, it must be equal with the port on which Syslog daemon on Proxylizer server is listening.

And then just set up web proxy:

[admin@Proxylizer pruebas] > ip [admin@Proxylizer pruebas] /ip> proxy [admin@Proxylizer pruebas] /ip proxy> print

```
                enabled: yes
            src-address: 0.0.0.0
```

```
                  port: 8080
           parent-proxy: 0.0.0.0
      parent-proxy-port: 0
    cache-administrator: "webmaster"
         max-cache-size: none
          cache-on-disk: no
 max-client-connections: 600
 max-server-connections: 600
         max-fresh-time: 3d
  serialize-connections: no
      always-from-cache: no
         cache-hit-dscp: 4
            cache-drive: system
```

and redirect the traffic to the web proxy:

[admin@Proxylizer pruebas] /ip firewall nat> print Flags: X - disabled, I - invalid, D - dynamic

```
0   chain=dstnat action=redirect to-ports=8080 protocol=tcp dst-port=80
```

Note: remember protect the proxy

Proxylizer server

Install method 1

All the examples assume that web page root directory is "/var/www/proxylizer", web server user is "www-data", Proxylizer server system user is "proxylizer" and .pipe file destination/name is "/home/proxylizer/mysql.pipe".

Required packages

- Syslog-ng [4] daemon
- Web server with PHP [5] and PHP-Pear [6]
  - Apache2 [7] (recomended), PHP5 [5], PHP5-cli [8] and PHP-Pear [6] : DB, Mail, Mail_Mime and Net_SMTP packages
- MySQL [9] database server

For Ubuntu issue this command to install all required packages:

```
sudo apt-get install syslog-ng libapache2-mod-php5 php5-cli php-pear
php-db php-mail php-mail-mime php-net-smtp php5-mysql mysql-server
mysql-client
```

WARNING : If you have Ubuntu syslog-ng can conflict with ubuntu-minimal package! You can remove this package.

Web page scripts

Download proxylizer archive. Create directory and extract it in web page root directory:

```
sudo tar -xvzf  proxylizer.tar.gz -C /var/www/
```

Permissions for directories

Change ownership of web page root directory for web server user:

```
 chown proxylizer:www-data /var/www/proxylizer -R
```

Set write permissions to web page root directory for web server user:

```
 chmod g+w /var/www/proxylizer -R
```

Set permissions to execute 3 shell script files for web server user group:

```
cd /var/www/proxylizer
chmod ug+x checkwebproxy.sh mail_send.php webproxylogtomysql.php
```

Syslog daemon

Change syslog-ng config to receive logs from Mikrotik router and put them into mysql.pipe file. Open /etc/syslog-ng/syslog-ng.conf and add these lines next to "#destinations" :

```
destination d_mysql {
pipe("/home/proxylizer/mysql.pipe"
template("$HOST $YEAR-$MONTH-$DAY $HOUR:$MIN:$SEC $MSG\n") template-escape(yes));
};
log { source(net); destination(d_mysql); };
```

And this line next to "#sources":

```
source net { udp(); };
```

Create pipe file:

```
mkfifo /home/proxylizer/mysql.pipe
```

Set destination of .pipe file in /var/www/proxylizer/webproxylogtomysql.php. At the beginning of the file you must change variable value in the line:

```
$MYSQL_PIPE = "/home/proxylizer/mysql.pipe";
```

restart syslog:

```
/etc/init.d/syslog-ng restart
```

Note: The last version of Syslog-NG don't project the string "web-proxy,account" to the pipe, so the $rawlog[] depending variables from the php script are bad setted. If you use syslog-ng 3.1.x or more please use this line to avoid problem:

```
destination d_mysql {
pipe("/home/proxylizer/mysql.pipe"
template("$HOST $YEAR-$MONTH-$DAY $HOUR:$MIN:$SEC web-proxy,account $MSG\n") template-escape(yes));
};
log { source(net); destination(d_mysql); };
```

MySQL user for proxylizer database

Default user name for mysql database is root with no password. But we recomend to change it for security reasons. To create new database `proxylizerdb` and user `proxylizer` with password `passwd` connect to mysql server, using command:

```
mysql –u root
```

and issue the following commands in mysql frontend:

```
CREATE DATABASE proxylizerdb;
GRANT ALL PRIVILEGES ON proxylizerdb.* TO proxylizer@localhost IDENTIFIED BY "password" WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

If you use mysql user other than root without password, connect to mysql server, using

```
mysql –u usrname –p
```

and you will be asked to enter the mysql user's password.

Scheduled scripts for forwarding records from syslog to MySQL and report generation

Create directory for script logs and set permesions:

```
sudo mkdir /var/log/proxylizer
sudo chown proxylizer:proxylizer /var/log/proxylizer
sudo chmod u+w /var/log/proxylizer
```

If you want to write logs in different directory you must edit bash script "checkwebproxy.sh" and change "/var/log/proxylizer" to preferred directory.

Put two scripts in cron sheduler. First create crontab file for web server system user:

```
nano /home/proxylizer/proxylizercrontab
```

and copy these lines:

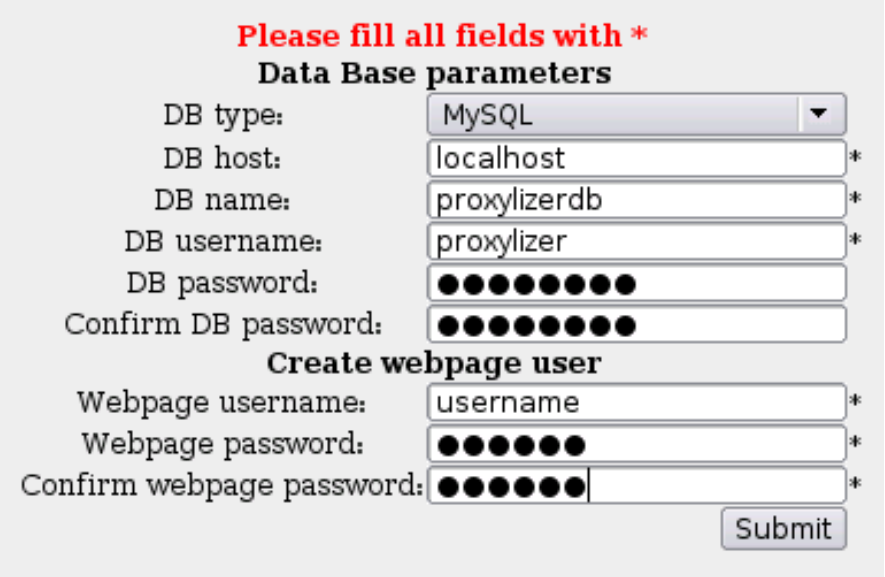
```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
* *     * * *           /var/www/proxylizer/mail_send.php >> /var/log/proxylizer/mail_send_log.log
* *     * * *           /var/www/proxylizer/checkwebproxy.sh >> /var/log/proxylizer/checkwebproxy.log &
```

Set scheduler tasks from this file:

```
crontab /home/proxylizer/proxylizercrontab
```

Database and web page access configuration

When all previous settings is set. Open web browser and point it to proxylizer server. First page must be like this :



- DB type - for now Proxylizer supports only MySQL, in future PostgreSQL, Interbase and other data bases will be added;
- DB host - by default "localhost", i.e,. database is located on the Proxylizer server;
- DB name - by default "proxylizer", must be equal with the one set here;
- DB username and password - as you have set here;
- Webpage username and password - as you prefer;

Setup page is shown always when the config file `config_constants.php` is not found in the Proxylizer root directory. On successful setup the configuration is written to this file. Configuration file contains database access and web page access parameters, no report or IP user configuration is included.

Mail sending configuration

To start receive reports to email, go to IP users page and add user with email address, then to Config page and configure Mail server access (any SMTP account needed).

Install method 2

It is posible to download already installed linux(debian) and proxylizer VMware virtual machine image and use proxylizer on any platform supported by VMware.

- Download VMware player [10].
- Download archived VMware proxylizer image
- Network settings:
  - if not in DHCP network open /etc/network/interfaces and change address, netmask, gateway etc.
- Passwords and usernames :
  - root password "rootroot";
  - username - "proxylizer", password - "rootroot";
  - mysql: root password - "proxylizer"; proxylizer data base username - "proxylizer", password - "password";
  - webpage: username- "proxylizer", password - "rootroot";

First report

First read documentation of web interface here. If you wan't to just check users web usage - create once report for date interval you are interested in and after a few moments report will be ready. If you wan't to see all users visited domains - create domain report, but remember it is only possible to get report for date interval which is already passed. For example if you want data for today report will be generated only tomorrow.

References

[1] http://www.mikrotik.com/download/proxylizer/proxylizer_0.1.1b.tar.gz
[2] http://www.mikrotik.com/download/proxylizer_vmware_image.zip
[3] http://www.mikrotik.com/download/proxylizer_01b_vmware.torrent
[4] http://en.wikipedia.org/wiki/Syslog-ng
[5] http://www.php.net
[6] http://pear.php.net/
[7] http://httpd.apache.org/
[8] http://lv.php.net/features.commandline
[9] http://www.mysql.com
[10] http://www.vmware.com/download/player/

Proxylizer/Introduction

What is Proxylizer

Mikrotik Proxylizer is a convenient system with a web interface, for **web-proxy log storage** in database and **report generation** from stored logs.

Mikrotik web-proxy [1] is able to send log entries to a remote location using syslog [2] protocol. Remote host must process the log entries to get required statistics. Mikrotik Proxylizer is designed to accomplish this task in a convenient way.

With Mikrotik Proxylizer, received web-proxy logs are stored in a database for further processing. Using web interface system administrators define reports which are sent to a certain email address, what data must be collected, and when the report has to be generated. Periodic reports are available (daily, weekly and monthly).

Mikrotik Proxylizer can be used to:

- Collect statistics about company staff member visited sites;
- Detect spyware which sends information to remote web sites.

Features

Mikrotik Proxylizer features:

- Web-proxy logs stored in SQL database (MySQL [3] supported at the time);
- Log filtering in reports based on host IP address and requested domain;
- Reports include:
  - Overall user report - list of IP addresses with time spent on browsing web;
  - Specific user report - list of domains with time spent for specified IP address;
  - Domain report - list of the most popular domains with IP count and spent time;
- Reports contain day of the week and time of the day restrictions;
- Scheduled report generation: daily, weekly, monthly;
- Reports are generated automatically without any user intervention;
- Reports sent in emails as attached PDF [4] (portable document format, platform independent);

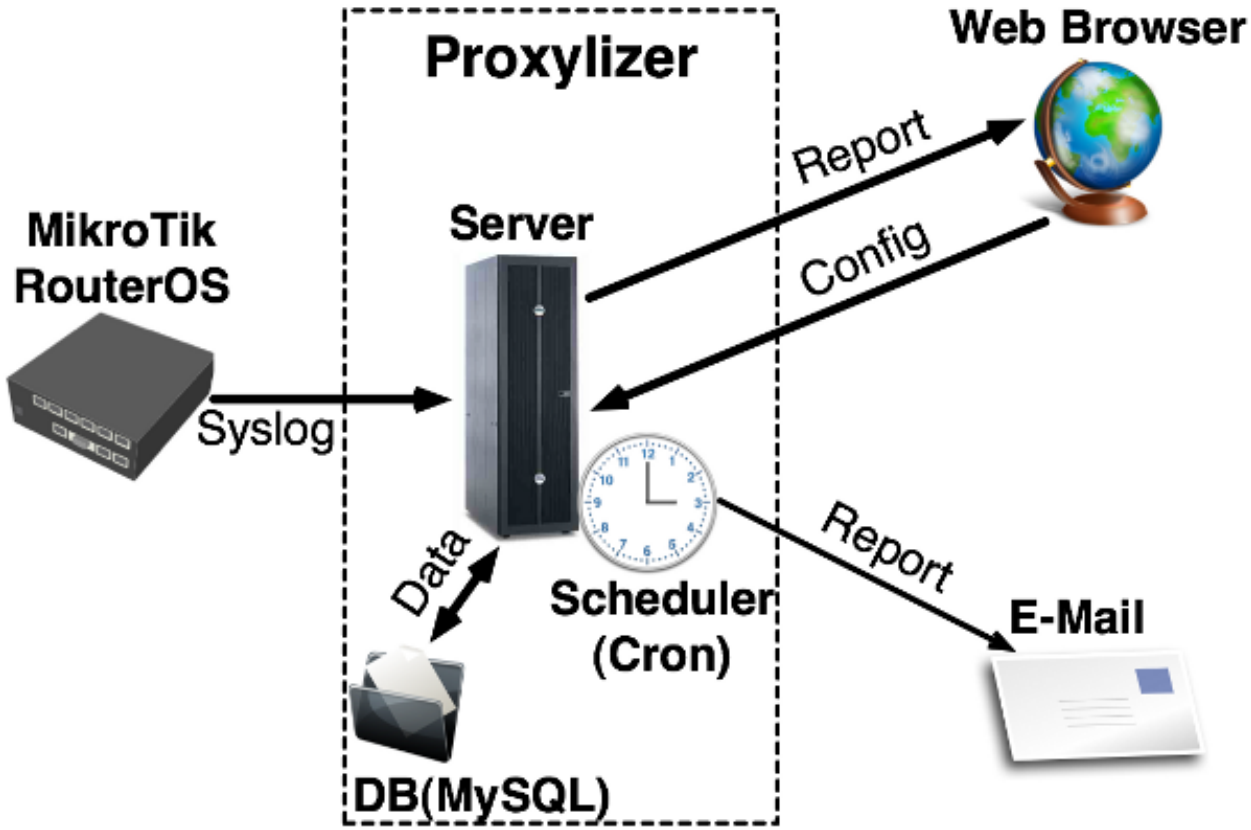
- All generated report history accessible in the web interface;
- Reports are generated in background process;
- Multiple reports can be generated in parallel to utilize multi-core processors efficiently;

Architecture

Mikrotik Proxylizer consists of multiple parts:

- SQL database - storage of all log entries. MySQL [9] supported at the time, other SQL [5] database support planned in future;
- Log reader - script responsible for log entry transfer from syslog [2] to SQL [5] database. Syslog-ng [4] is used to listen on incoming syslog [2] entries and write them to pipe file [6]. PHP [5] script is used to read the pipe and insert entries in database afterwards;
- Report generator - every minute a script is started which takes the first report which must be generated and collects the required data from database. The result is sent to email and stored in database for later access in the web;
- Web interface - configuration, statistics and history user interface for the system administrator.

The Proxylizer is interconnected with other system components: syslog entry source, SQL database, scheduler, SMTP [7] server, web browser. Proxylizer contains scripts for database table structures and periodic tasks, therefore database and scheduler are treated as part of the Proxylizer. However both SQL database server and scheduler service are third party applications: MySQL [9] and Cron [8] in the current version. The collaboration is shown in the following diagram:



Requirements

Hardware requirements

Recommended requirements :

- CPU: 1 GHz
- RAM: at least the size of database,

We have tested it on a server with the following hardware:

- CPU: Intel(R) Pentium(R) Dual Core 2.80GHz
- RAM:1 GB

This server accepted insert of approximately 500 records per second. 1GB of disk space was used by approximately 4.5 million records.

Operating system and software requirements

Currently only the Linux distributions are supported. Proxylizer is tested on Ubuntu [9] platform but there should be no problem running it on other distros, including Debian [10], SUSE [11], and others. Technically it is possible to install it on a MacOS machine, however it is not tested yet.

On the server side Proxylizer requires the following third-party software:

- Web server, for example, Apache [7];
- Database server. Currently only MySQL [9] is supported;
- PHP 5.x [5] with PHP-Pear [6];
- Syslog deamon, for example Syslog-ng [12];
- Scheduler, for example Cron [8];
- SMTP server, for example Postfix [13].

On the client side Proxylizer requires only a modern Web-browser. The following browsers are supported:

- Mozilla Firefox [14], version 2.x or 3.x (This browser is recommended);
- Opera [15] 9.x or 10.x;
- Google Chrome [16] 1.x
- Safari [17] 3.x
- Internet Explorer [18], version 6.x or 7.x;

Network infrastructure requirements

Mikrotik Proxylizer utilizes the following communications (shown as arrows in the collaboration picture):

- Mikrotik router to Proxylizer server;
- Mikrotik router to Proxylizer server;
- Between Proxylizer server and user browser;
- Proxylizer server to SMTP server.

Mikrotik router to Proxylizer server

Mikrotik Router needs access to Proxylizer server to send syslog entries to it. It is sufficient to allow only communication through one port. Syslog uses port 514 by default.

Between Proxylizer server and database server

Both web scripts and data collector/report scripts need access to MySQL database. Usually MySQL database and web scripts are located on the same host machine without any additional infrastructure requirements. But it is also possible to use database server on remote host. In that case two requirements must be met:

1. Proxylizer server needs access to the remote hosts database port, 3306 by default.
2. MySQL must be configured to allow remote connections. This can be done in MySQL config files. On Ubuntu distribution they are located in `/etc/mysql`.

Between Proxylizer server and user browser

Proxylizer has Web-interface for configuration and report history review. To deliver the interface to users browser, HTTP [19] traffic has to be sent over TCP connection. Ie., TCP connection between the user machine and Proxylizer server's HTTP port (80 by default or 443 for secure HTTPS [20] connection) must be enabled. If the Proxylizer server is behind a firewall router, the HTTP traffic must be forwarded from the firewall to Proxylizer server, using DST-NAT [21]. For example, if the firewall is Mikrotik router and Proxylizer has IP address 10.1.1.2, the NAT is enabled by typing the following command in the firewall router's console:

```
/ip firewall nat add action=dst-nat protocol=tcp dst-port=80 to-addresses=10.1.1.2 to-ports=80
```

Proxylizer server to SMTP server

Proxylizer can send automatic email reports with statistics. To send email, it requires SMTP [7] server. If the SMTP server and Proxylizer are running on the same host machine no additional infrastructure is required. If the SMTP server is located on a remote host, access to it for Proxylizer must be enabled. The default SMTP port is 25.

References

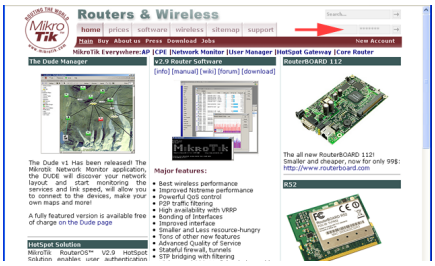
[1] http://www.mikrotik.com/testdocs/ros/3.0/pnp/proxy.php
[2] http://en.wikipedia.org/wiki/Syslog
[3] http://www.mysql.com/
[4] http://en.wikipedia.org/wiki/Pdf
[5] http://en.wikipedia.org/wiki/Sql
[6] http://en.wikipedia.org/wiki/Pipe_(Unix)
[7] http://en.wikipedia.org/wiki/Smtp
[8] http://en.wikipedia.org/wiki/Cron
[9] http://www.ubuntu.com
[10] http://www.debian.com
[11] http://www.novell.com/linux/
[12] http://www.balabit.com/network-security/syslog-ng/
[13] http://www.postfix.org/
[14] http://www.mozilla.com/en-US/firefox/
[15] http://www.opera.com/
[16] http://www.google.com/chrome
[17] http://www.apple.com/safari/
[18] http://www.microsoft.com/windows/products/winfamily/ie/default.mspx
[19] http://en.wikipedia.org/wiki/Http
[20] http://en.wikipedia.org/wiki/Https
[21] http://en.wikipedia.org/wiki/Network_Address_Translation

Manual:Purchasing a License for RouterOS

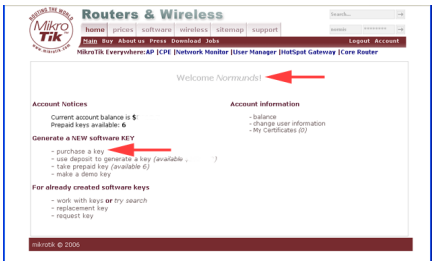
First you have to make an account on the Account Server, this can be done on the mikrotik.com main page, and is a free and easy process.

Important! Before purchasing a key, you have to install RouterOS. It will generate a SoftID that will be required during the purchase. Before entering the SoftID in the purchase form, make sure it has not changed on your router. After installation, you have 24 hours to enter a key. If you are close to running out of time - shut down the router. The timer will stop.

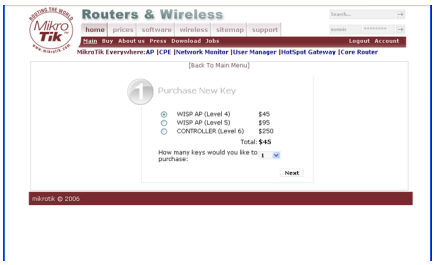
After you have an account, start by logging in, here is an example process:



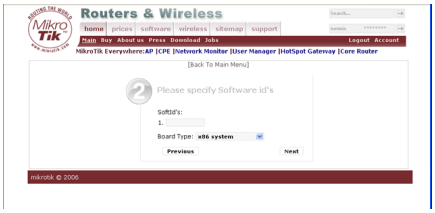
Log into your account



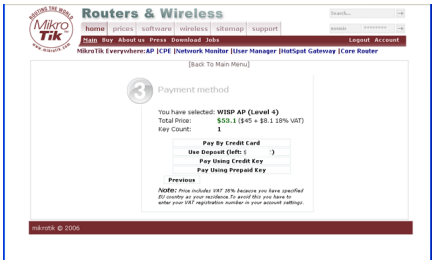
Click on **Purchase a Key**



Select your License Level and the number of licenses you need



Enter your SoftIDs and select the system kind, remember that SoftID will be given to you after installation of RouterOS. The system kind is a choice between RouterBOARD and X86. Basically if you have a RouterBOARD(TM) device, select RouterBOARD, if you have some other kind of device - select X86. NOTE!: Older RouterBOARD 230 model is an X86 device too.



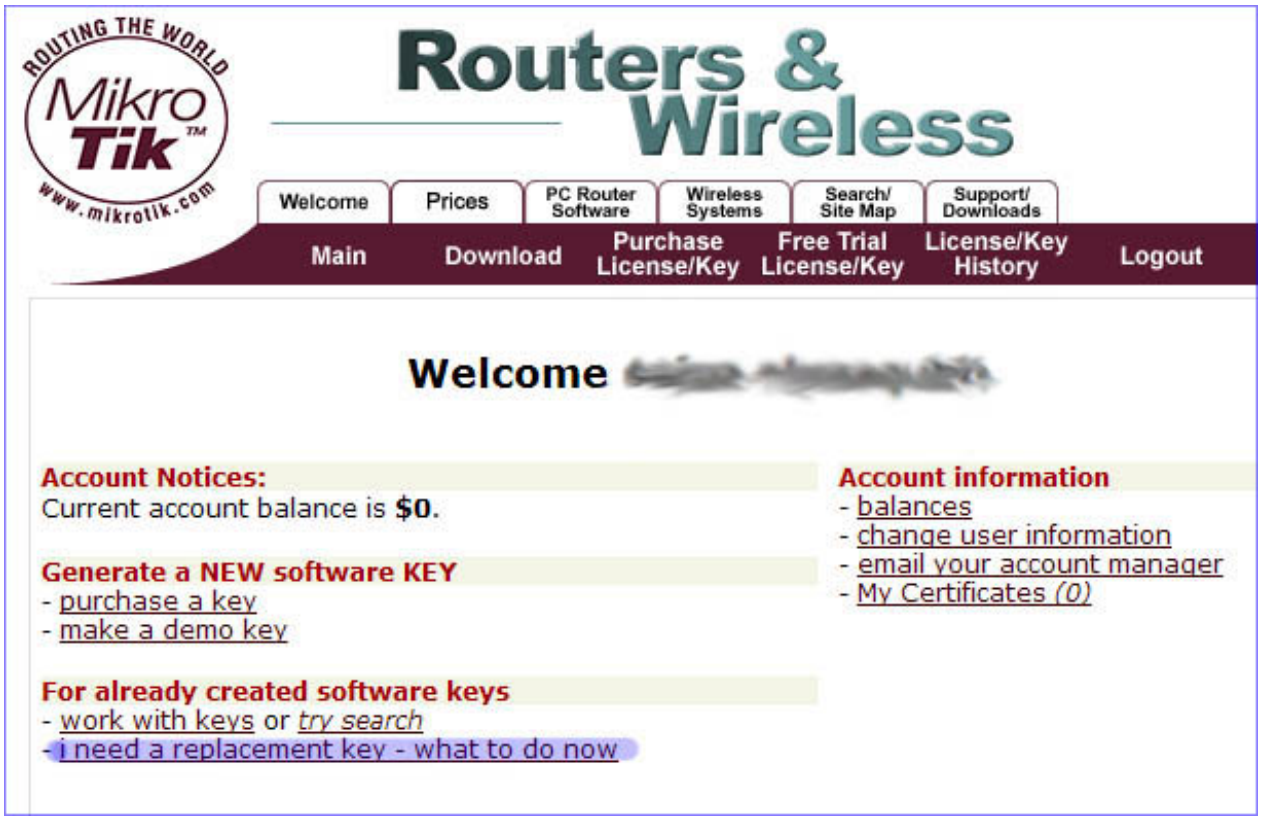
Click on Pay By Credit Card and You will be presented the bank payment page

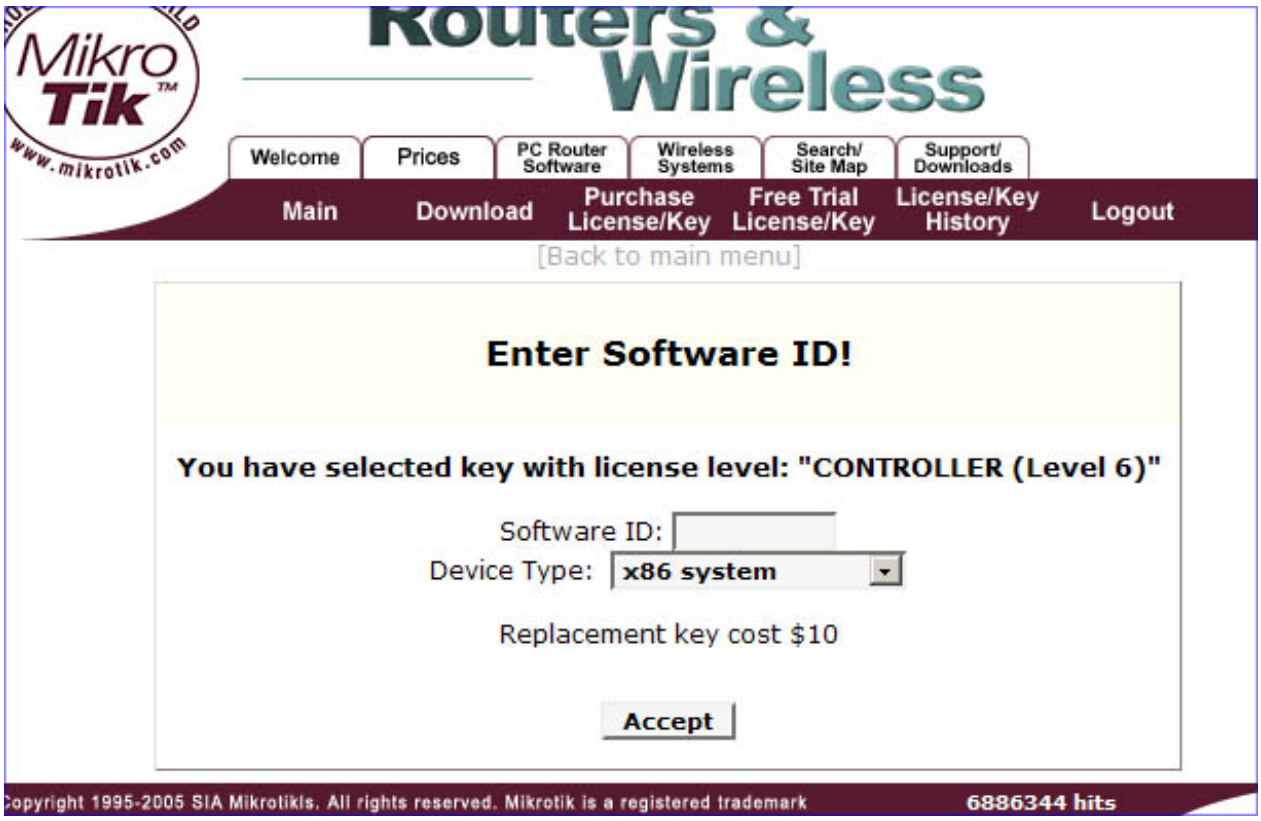
In the Bank page you will be asked for your **Credit Card Number**, CVC/CVV code, expiry date of the card and the name on the card. The CVC/CVV card can be found on the back of the card and is a three digit code. After you enter all the details and submit the information, your credit card will be charged. Do not close the browser or push any buttons until the process is complete. Then you will receive your new key in your email, and it will also appear in the "**work with keys**" section of your account.

Instructions how to apply license on your router are here.

Manual:Replacement Key

If you have been given the so-called "Replacement Key", follow these instructions to take it from your account:





Manual:Queue

Applies to RouterOS: 2.9, v3, v4

List of reference sub-pages **Case studies** **List of examples**

<splist showparent=yes />

Summary

Queues are used to limit and prioritize traffic:

- limit data rate for certain IP addresses, subnets, protocols, ports, and other parameters
- limit peer-to-peer traffic
- prioritize some packet flows over others
- configure traffic bursts for faster web browsing
- apply different limits based on time
- share available traffic among users equally, or depending on the load of the channel

Queue implementation in MikroTik RouterOS is based on Hierarchical Token Bucket (HTB). HTB allows to create hierarchical queue structure and determine relations between queues.

In RouterOS, these hierarchical structures can be attached at 4 different places:

- *global-in*: represents all the input interfaces in general (INGRESS queue). Queues attached to global-in apply to traffic that is received by the router before the packet filtering

- *global-out*: represents all the output interfaces in general (EGRESS queue).
- *global-total*: represents all input and output interfaces together (in other words it is aggregation of global-in and global-out). Used in case when customers have single limit for both, upload and download.
- <*interface name*>: - represents one particular outgoing interface. Only traffic that is designated to go out via this interface will pass this HTB queue.

There are two different ways how to configure queues in RouterOS:

- **/queue simple** menu - designed to ease configuration of simple, everyday queuing tasks (such as single client upload/download limitation, p2p traffic limitation, etc.).
- **/queue tree** menu - for implementing advanced queuing tasks (such as global prioritization policy, user group limitations). Requires marked packet flows from **/ip firewall mangle** facility.

Rate limitation principles

Rate limiting is used to control the rate of traffic flow sent or received on a network interface. Traffic which rate that is less than or equal to the specified rate is sent, whereas traffic that exceeds the rate is dropped or delayed.

Rate limiting can be performed in two ways:

1. discard all packets that exceed rate limit – ***rate limiting (dropper or shaper)*** *(100% rate limiter when queue-size=0)*
2. delay packets that exceed specific rate limit in queue and transmit its when it is possible – ***rate equalizing (scheduler)*** "(100% rate equalizing when queue-size=unlimited)

Next figure explains difference between *rate limiting* and rate *equalizing*:

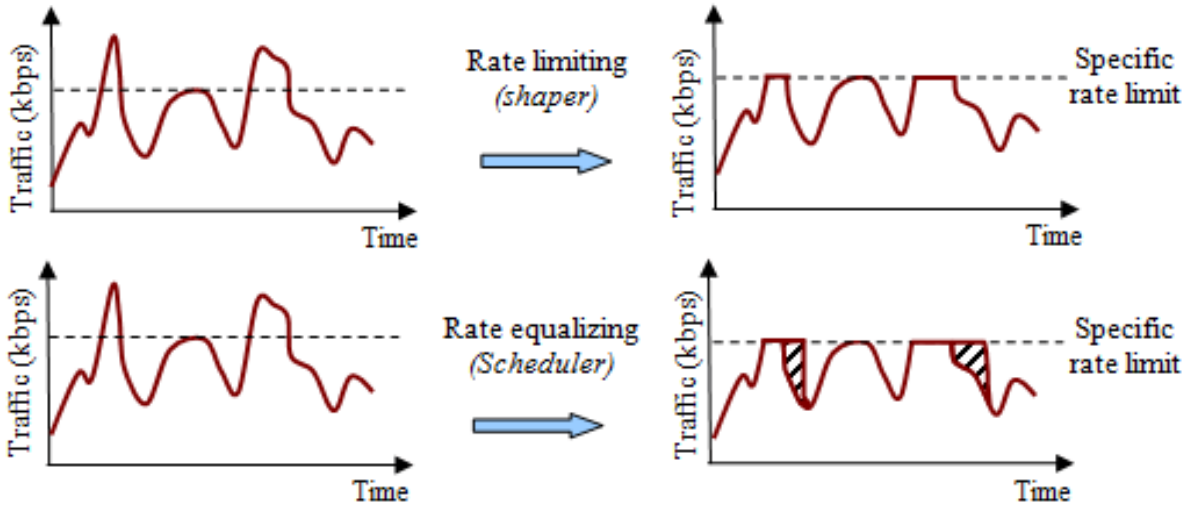


Figure 8.1. Principles of rate limiting and equalizing

As you can see in first case all traffic exceeds specific rate and is dropped. In other case traffic exceeds specific rate and is delayed in queue and transmitted later when it is possible, but note that packet can be delayed only until queue is not full. If there is not more space in queue buffer, packets are dropped.

For each queue we can define two rate limits:

- **CIR** (Committed Information Rate) – (**limit-at** in RouterOS) worst case scenario, flow will get this amount of traffic rate regardless of other traffic flows. At any given time, the bandwidth should not fall below this committed rate.
- **MIR** (Maximum Information Rate) – (**max-limit** in RouterOS) best case scenario, maximum available data rate for flow, if there is free any part of bandwidth.

Simple Queues

Sub-menu: `/queue simple`

The simplest way to limit data rate for specific IP addresses and/or subnets, is to use simple queues.

You can also use simple queues to build advanced QoS applications. They have useful integrated features:

- Peer-to-peer traffic queuing
- Applying queue rules on chosen time intervals
- Priorities
- Using multiple packet marks from */ip firewall mangle*
- Shaping (scheduling) of bidirectional traffic (one limit for the total of upload + download)

One configuration item in */queue simle*' can create from 0 to 3 separate queues - one queue in *global-in*, one queue in *global-out* and one queue in *global-total*. If all properties of a queue have default values (no set limits, queue type is *default*), and queue has no children, then it is not actually created. This way, for exanple, creation of *global-total* queues can be avoided if only upload/download limitation is used.

Simple queues have strict order - each packet must go through every queue until it will meet conditions. (In case of 1000 queues, packet for last queue will need to proceed through 999 queues before it will reach the destination)

Configuration Example

Assume we have network topology like Figure 8.6 and we want to limited download and upload for private network (upload - 256kbps, and download – 512kbps).

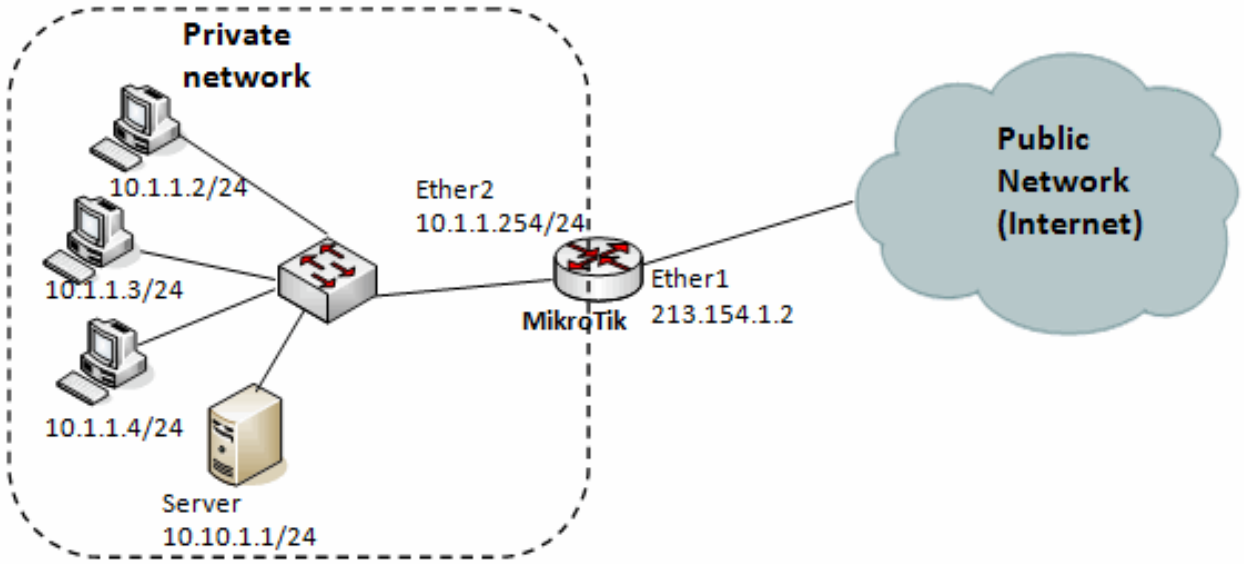


Figure 8.6. Network example

Add a simple queue rule, which will limit the download traffic to 512kbps and upload to 256kbps for the network **10.1.1.0/24**, served by the interface **Ether2**:

```
[admin@MikroTik] /queue simple> add name=private target-addresses=10.1.1.0/24 max-limit=256K/512K \
interface=ether2
```

In this case statement works right also if we indicate only one of parameters: *"target-addresses="* or *"interface="*, because both of these define where and for which traffic this queue will be implemented.

Check your configuration:

```
[admin@Augsha] /queue simple> print
```

```
Flags: X - disabled, I - invalid, D - dynamic
 0    name="private" target-addresses=10.1.1.0/24 dst-address=0.0.0.0/0
      interface=ether2 parent=none direction=both priority=8
      queue=default-small/default-small limit-at=0/0 max-limit=256k/512k
      burst-limit=0/0 burst-threshold=0/0 burst-time=0s/0s
      total-queue=default-small
```

The **max-limit** parameter cuts down the maximum available bandwidth. The value **max-limit=256k/512k** means that clients from private network will get maximum of 512kbps for download and 256kbps for upload. The **target-addresses** allows to define the source IP addresses to which the queue rule will be applied.

Probably, you want to exclude the server from being limited, if so, add a queue for it without any limitation (max-limit=0/0 which means no limitation). Move this rule to the beginning of the list, because items in /queue simple are executed in order one by one if router finds rule that satisfy certain packet next rules aren't compared:

```
[admin@MikroTik] /queue simple> add name=server target-addresses=10.1.1.1/32 max-limit=0/0 \
interface=ether2
```

Flow Identifiers

- **target-addresses** (multiple choice: IP address/netmask) : list of IP address ranges that will be limited by this queue.
- **interface** (Name of the interface, or *all*) : identifies interface the target is connected to. Useful when it is not possible to specify targets addresses.

Each of these two properties can be used to determine which direction is target upload and which is download.

Be careful to configure both of these options for the same queue - in case they will point to opposite directions queue will not work.

If neither value of **target-addresses** nor of **interface** is specified, the queue will not be able to make difference between upload and download, and will limit all traffic twice.

Other properties

- **name** (Text) : Unique queue identifier that can be used as **parent** option value for other queues
- **direction** (One of *both*, *upload*, *download*, *none*; default: *both*) : allow to enable one-directional limitation for simple queues (disable other direction)
  - *both* - limit both download and upload traffic
  - *upload* - limit only traffic to the target
  - *download* - limit only traffic from the target
- **time** (*TIME-TIME,sun,mon,tue,wed,thu,fri,sat* - *TIME* is local time, all day names are optional; default: not set) : allow to specify time when particular queue will be active. Router must have correct time settings.
- **dst-address** (IP address/netmask) : allows to select only specific stream (from target address to this destination address) for limitation explain what is target and what is dst and what is upload and what not
- **p2p** (one of *all-p2p*, *bit-torrent*, *blubster*, *direct-connect*, *edonkey*, *fasttrack*, *gnutella*, *soulseek*, *winmx*; default: not set) : allow to select unencrypted packets of particular p2p for limitation
- **packet-marks** (Comma separated list of packet mark names) : allows to use marked packets from **/ip firewall mangle**. Take look at the RouterOS packet flow diagram. It is necessary to mark packets before the simple queues (before *global-in* HTB queue) or else target's download limitation will not work. The only mangle chain before *global-in* is *prerouting*.

HTB Properties

- **parent** (Name of parent simple queue, or *none*) : assigns this queue as a child queue for selected target {{{...}}}. Target queue can be HTB queue or any other previously created simple queue. In order for traffic to reach child queues, parent queues must capture all necessary traffic.
- **priority** (1..8) : Prioritize one child queue over other child queue. Does not work on parent queues (if queue has at least one child). One is the highest, eight is the lowest priority. Child queue with higher priority will have chance to reach its **limit-at** before child with lower priority and after that child queue with higher priority will have chance to reach its **max-limit** before child with lower priority. Priority have nothing to do with bursts.
- **queue** (*SOMETHING/SOMETHING*) : Choose the type of the upload/download queue. Queue types can be created in **/queue type**.
- **limit-at** (*NUMBER/NUMBER*) : normal upload/download data rate that is guaranteed to a target
- **max-limit** (*NUMBER/NUMBER*) : maximal upload/download data rate that is allowed for a target to reach to reach what
- **burst-limit** (*NUMBER/NUMBER*) : maximal upload/download data rate which can be reached while the burst is active
- **burst-time** (*TIME/TIME*) : period of time, in seconds, over which the average upload/download data rate is calculated. (This is NOT the time of actual burst)
- **burst-threshold** (*NUMBER/NUMBER*) : when average data rate is below this value - burst is allowed, as soon as average data rate reach this value - burst is denied. (basically this is burst on/off switch). For optimal burst behavior this value should above **limit-at** value and below **max-limit** value

And corresponding options for *global-total* HTB queue:

- **total-queue** (*SOMETHING/SOMETHING*): corresponds to **queue**
- **total-limit-at** (*NUMBER/NUMBER*): corresponds to **limit-at**
- **total-max-limit** (*NUMBER/NUMBER*): corresponds to **max-limit**
- **total-burst-limit** (*NUMBER/NUMBER*): corresponds to **burst-limit**
- **total-burst-time** (*TIME/TIME*): corresponds to **burst-time**
- **total-burst-threshold** (*NUMBER/NUMBER*): corresponds to **burst-threshold**

Good practice suggests that:

> Sum of children's **limit-at** values must be less or equal to **max-limit** of the parent.
>
> Every child's **max-limit** must be less than **max-limit** of the parent. This way you will leave some traffic for the other child queues, and they will be able to get traffic without fighting for it with other child queues.

Statistics

- **rate** (read-only/read-only) : average queue passing data rate in bytes per second
- **packet-rate** (read-only/read-only) : average queue passing data rate in packets per second
- **bytes** (read-only/read-only) : number of bytes processed by this queue
- **packets** (read-only/read-only) : number of packets processed by this queue
- **queued-bytes** (read-only/read-only) : number of bytes waiting in the queue
- **queued-packets** (read-only/read-only) : number of packets waiting in the queue
- **dropped** (read-only/read-only) : number of dropped packets
- **borrows** (read-only/read-only) : packets that passed queue over its "limit-at" value (and was unused and taken away from other queues)
- **lends** (read-only/read-only) : packets that passed queue below its "limit-at" value OR if queue is a parent - sum of all child borrowed packets
- **pcq-queues** (read-only/read-only) : number of PCQ substreams, if queue type is PCQ

And corresponding options for *global-total* HTB queue:

- **total-rate** (read-only): corresponds to *rate*
- **total-packet-rate** (read-only): corresponds to *packet-rate*
- **total-bytes** (read-only): corresponds to *bytes*
- **total-packets** (read-only): corresponds to *packets*
- **total-queued-bytes** (read-only): corresponds to *queued-bytes*
- **total-queued-packets** (read-only): corresponds to *queued-packets*
- **total-dropped** (read-only): corresponds to *dropped*
- **total-lends** (read-only): corresponds to *lends*
- **total-borrows** (read-only): corresponds to *borrows*
- **total-pcq-queues** (read-only): corresponds to *pcq-queues*

Queue Tree

Sub-menu: `/queue tree`

Queue tree creates only one directional queue in one of the HTBs. It is also the only way how to add queue on the separate interface. This way it is possible to ease mangle configuration - you don't need separate marks for download and upload - only upload will get to Public interface and only download will get to Private interface.

Also it is possible to have double queuing (example:prioritization of traffic in global-in or global-out, limitation per client on the outgoing interface) If you have simple queues and queue tree in the same HTB - simple queues will get traffic first.

Queue tree is not ordered - all traffic pass it together.

Read more about HTB and see configuration examples.

Flow Identifiers

- **name** (Text) : Unique queue identifier that can be used as **parent** option value for other queues
- **packet-marks** (Comma separated list of) : allows to use marked packets from **/ip firewall mangle**. Take look at this packet flow diagram. You need to make sure that packets are marked before the simple queues (before global-in HTB queue)

HTB Properties

- **parent** (Name of , or *none*) : assigns this queue as a child queue for selected target. Target queue can be HTB queue or any other previously created queue
- **priority** (1..8) : Prioritize one child queue over other child queue. Does not work on parent queues (if queue has at least one child). One is the highest, eight is the lowest priority. Child queue with higher priority will have chance to reach its **nax-limit** before child with lower priority. Priority have nothing to do with bursts.
- **queue** (*SOMETHING*) : Choose the type of the queue. Queue types can be created here
- **limit-at** (*NUMBER*) : normal data rate that is guaranteed to a target
- **max-limit** (*NUMBER*) : maximal data rate that is allowed for a target to reach
- **burst-limit** (*NUMBER*) : maximal data rate which can be reached while the burst is active
- **burst-time** (*TIME*) : period of time, in seconds, over which the average data rate is calculated. (This is NOT the time of actual burst)
- **burst-threshold** (*NUMBER*) : when average data rate is below this value - burst is allowed, as soon as average data rate reach this value - burst is denied. (basically this is burst on/off switch). For optimal burst behavior this value should above **limit-at** value and below **max-limit** value

Statistics

Command: `/queue tree print stats`

- **rate** (read-only) : average queue passing data rate in bytes per second
- **packet-rate** (read-only) : average queue passing data rate in packets per second
- **bytes** (read-only) : number of bytes processed by this queue
- **packets** (read-only) : number of packets processed by this queue
- **queued-bytes** (read-only) : number of bytes waiting in the queue
- **queued-packets** (read-only) : number of packets waiting in the queue
- **dropped** (read-only) : number of dropped packets
- **borrows** (read-only) : packets that passed queue over its "limit-at" value (and was unused and taken away from other queues)
- **lends** (read-only) : packets that passed queue below its "limit-at" value OR if queue is a parent - sum of all child borrowed packets
- **pcq-queues** (read-only) : number of PCQ substreams, if queue type is PCQ

Queue Types

Sub-menu: `/queue type`

This sub-menu lists by default created queue types and allows to add new user specific ones.

By default RouterOS creates following pre-defined queue types:

```
[admin@MikroTik] /queue type> print
 0 name="default" kind=pfifo pfifo-limit=50

 1 name="ethernet-default" kind=pfifo pfifo-limit=50

 2 name="wireless-default" kind=sfq sfq-perturb=5 sfq-allot=1514

 3 name="synchronous-default" kind=red red-limit=60 red-min-threshold=10 red-max-threshold=50 red-burst=20
   red-avg-packet=1000

 4 name="hotspot-default" kind=sfq sfq-perturb=5 sfq-allot=1514

 5 name="only-hardware-queue" kind=none

 6 name="multi-queue-ethernet-default" kind=mq-pfifo mq-pfifo-limit=50

 7 name="default-small" kind=pfifo pfifo-limit=10
```

Note: Starting from v5.8 there is new kind **none** and new default queue **only-hardware-queue**. All RouterBOARDS will have this new queue type set as default interface queue

only-hardware-queue leaves interface with only hw transmit descriptor ring buffer which acts as a queue in itself. Usually at least 100 packets can be queued for transmit in transmit descriptor ring buffer. Transmit descriptor ring buffer size and the amount of packets that can be queued in it varies for different types of ethernet MACs.

Having no software queue is especially beneficial on SMP systems because it removes the requirement to synchronize access to it from different cpus/cores which is expensive.

multi-queue-ethernet-default can be beneficial on SMP systems with ethernet interfaces that have support for multiple transmit queues and have a linux driver support for multiple transmit queues. By having one software queue for each hardware queue there might be less time spent for synchronizing access to them.

Note: having possibility to set only-hardware-queue requires support in ethernet driver so it is available only for some ethernet interfaces mostly found on RBs.

Note: improvement from only-hardware-queue and multi-queue-ethernet-default is present only when there is no "/queue tree" entry with paticular interface as a parent.

Kinds

Queue kinds or Queuing (scheduling) algorithms describe which packet will be transmitted next in line. RouterOS supports several queuing algorithms:

- BFIFO, PFIFO, MQ PFIFO
- RED
- SFQ
- PCQ

PFIFO, BFIFO and MQ PFIFO

These queuing disciplines are based on the FIFO algorithm (First-In First-Out). The difference between PFIFO and BFIFO is that one is measured in packets and the other one in bytes.

Every packet that cannot be enqueued (if the queue is full), is dropped. Large queue sizes can increase latency, but utilize channel better.

These queues uses **pfifo-limit** and **bfifo-limit** parameters.

mq-pfifo is pfifo with support for multiple transmit queues. This queue is beneficial on SMP systems with ethernet interfaces that have support for multiple transmit queues and have a linux driver support for multiple transmit queues.

mq-pfifo uses `mq-pfifo-limit` parameter.

RED

Random Early Drop is a queuing mechanism which tries to avoid network congestion by controlling the average queue size. The average queue size is compared to two thresholds: a minimum (min_{th}) and maximum (max_{th}) threshold. If average queue size (avg_q) is less than the minimum threshold, no packets are dropped. When average queue size is greater than the maximum threshold, all incoming packets are dropped. But if the average queue size is between the minimum and maximum thresholds packets are randomly dropped with probability P_d where probability is exact a function of the average queue size: $P_d = P_{max}(avg_q - min_{th})/ (max_{th} - min_{th})$. If average queue grows, the probability for dropping incoming packets grows too. P_{max} - ratio, which can adjust the packet discarding probability abruptness, (the simplest case P_{max} can be equal to one. The diagram in Figure 8.2. shows the packet drop probability in RED algorithm.

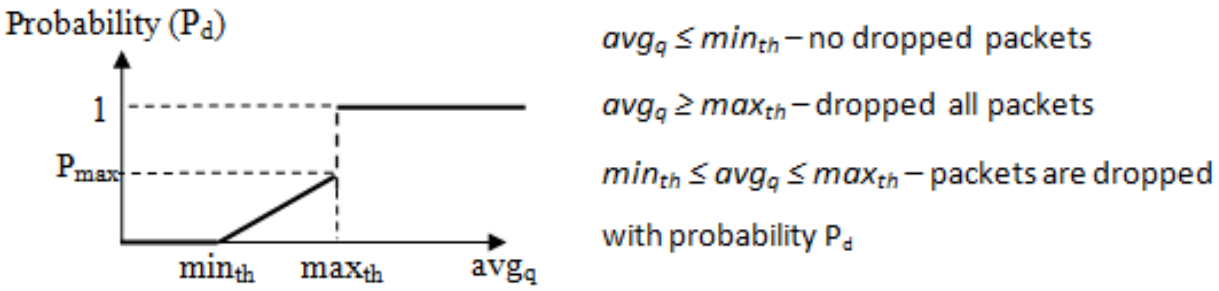


Figure 8.2. RED operation

SFQ

Stochastic Fairness Queuing (SFQ) is ensured by hashing and round-robin algorithms. A traffic flow may be uniquely identified by a 4 options(src-address, dst-address, src-port and dst-port), so these parameters are used by SFQ hashing algorithm to classify packets into one of 1024 possible sub-streams. Then round-robin algorithm will start to distribute available bandwidth to all sub-streams, on each round giving **sfq-allot** bytes of traffic. The whole SFQ queue can contain 128 packets and there are 1024 sub-streams available.

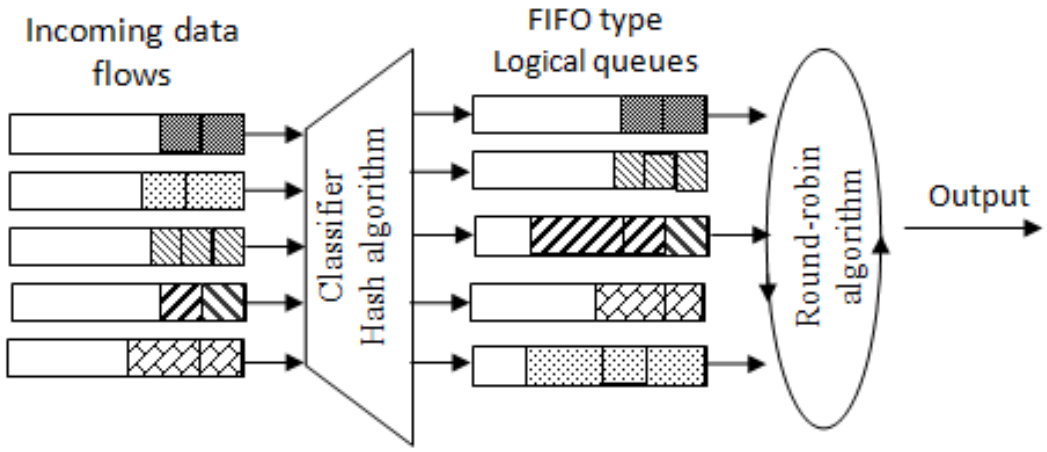


Figure 8.3. SFQ operation

SFQ is called "Stochastic" because it does not really allocate a queue for each flow, it has an algorithm which divides traffic over a limited number of queues (1024) using a hashing algorithm.

PCQ

Per Connection Queuing (PCQ) is a similar to SFQ, but it has additional features.

It is possible to choose flow identifiers (from dst-address | dst-port | src-address | src-port). For example if you classify flows by src-address on local interface (interface with your clients), each PCQ sub-stream will be one particular client's upload.

It is possible to assign speed limitation to sub-streams with **pcq-rate** option. If **pcq-rate**=0 sub-streams will divide available traffic equally.

More information and examples of PCQ are available here.

Properties

Properties that start with particular queue kind name, is applied only to particular kind. For example all properties starting with pcq applies only to queue **kind=pcq**.

Property	Description
bfifo-limit (*integer [1000..4294967295]*; Default: **15000**)	Maximum number of bytes that the BFIFO queue can hold. Applies if **kind** is bfifo.
kind (*bfifo* | *mq-pfifo* | *none* | *pcq* | *pfifo* | *red* | *sfq*; Default:)	Kind of particular queue type. `Read more >>`
mq-pfifo-limit (*integer [1..4294967295]*; Default: **50**)	Multi-queue PFIFO limit.
name (*string*; Default:)	Descriptive name of queue type
pcq-burst-rate (*integer [0..4294967295]*; Default: **0**)	Maximal upload/download data rate which can be reached while the burst for substream is allowed
pcq-burst-threshold (*integer [0..4294967295]*; Default: **0**)	This is value of burst on/off switch
pcq-burst-time (*time*; Default: **10s**)	Period of time, in seconds, over which the average data rate is calculated. (This is NOT the time of actual burst)
pcq-classifier (*list of src-address|dst-address|src-port|dst-port*; Default: **""**)	Selection of sub-stream identifiers
pcq-dst-address-mask (*integer [0..32]* | *IPNetmask*; Default: **32**)	size of IPv4 network that will be used as dst-address sub-stream identifier
pcq-dst-address6-mask (*integer [0..128]*; Default: **128**)	size of IPV6 network that will be used as dst-address sub-stream identifier
pcq-limit (*integer [1..4294967295]*; Default: **50**)	Queue size of single sub-stream (in KB)
pcq-rate (*integer [0..4294967295]*; Default: **0**)	Maximal available data rate of each sub-steam
pcq-src-address-mask (*integer [0..32]* | *IPNetmask*; Default: **32**)	size of IPv4 network that will be used as src-address sub-stream identifier
pcq-src-address6-mask (*integer [0..128]*; Default: **128**)	size of IPV6 network that will be used as src-address sub-stream identifier
pcq-total-limit (*integer [1..4294967295]*; Default: **2000**)	Queue size of single sub-stream (in KB)
pfifo-limit (*integer [1..4294967295]*; Default: **50**)	Maximum number of packets that the PFIFO queue can hold. Applies if **kind** is pfifo.
red-avg-packet (*integer [1..65535]*; Default: **1000**)	Used by RED for average queue size calculations (for packet to byte translation)
red-burst (*integer [0..4294967295]*; Default: **20**)	Number of packets allowed for bursts of packets when there are no packets in the queue
red-limit (*integer [0..4294967295]*; Default: **60**)	RED queue limit in packets
red-max-threshold (*integer [0..4294967295]*; Default: **50**)	The average queue size at which packet marking probability is the highest.
red-min-threshold (*integer [0..4294967295]*; Default: **10**)	Average queue size in bytes.
sfq-allot (*integer [0..32767]*; Default: **1514**)	Amount of data in bytes that can be sent in one round-robin round
sfq-perturb (*integer [0..4294967295]*; Default: **5**)	How often hash function must be refreshed

Interface Queue

Sub-menu: `/queue interface`

Before sending data over an interface, it is processed by the queue. This sub menu list all available interfaces in RouterOS and allows to change queue type for particular interface.

Note: You cannot add new interfaces to this menu. List is generated automatically.

Properties

Property	Description
interface (*string*)	Interface name to which queue is applied. Read-only parameter.
queue (*string*; Default:)	Queue type assigned to particular interface.

[Top | Back to Content]

Manual:Queues - Burst

Applies to RouterOS: v2.9 and newer

Theory

Burst is a feature that allows to satisfy queue requirement for additional bandwidth even if required rate is bigger that **MIR** (**max-limit**) for a limited period of time.

Burst can occur only if **average-rate** of the queue for the last **burst-time** seconds is smaller that **burst-threshold**. Burst will stop if **average-rate** of the queue for the last **burst-time** seconds is bigger or equal to **burst-threshold**

Burst mechanism is simple - if burst is allowed **max-limit** value is replaced by **burst-limit** value. When burst is disallowed **max-limit** value remains unchanged.

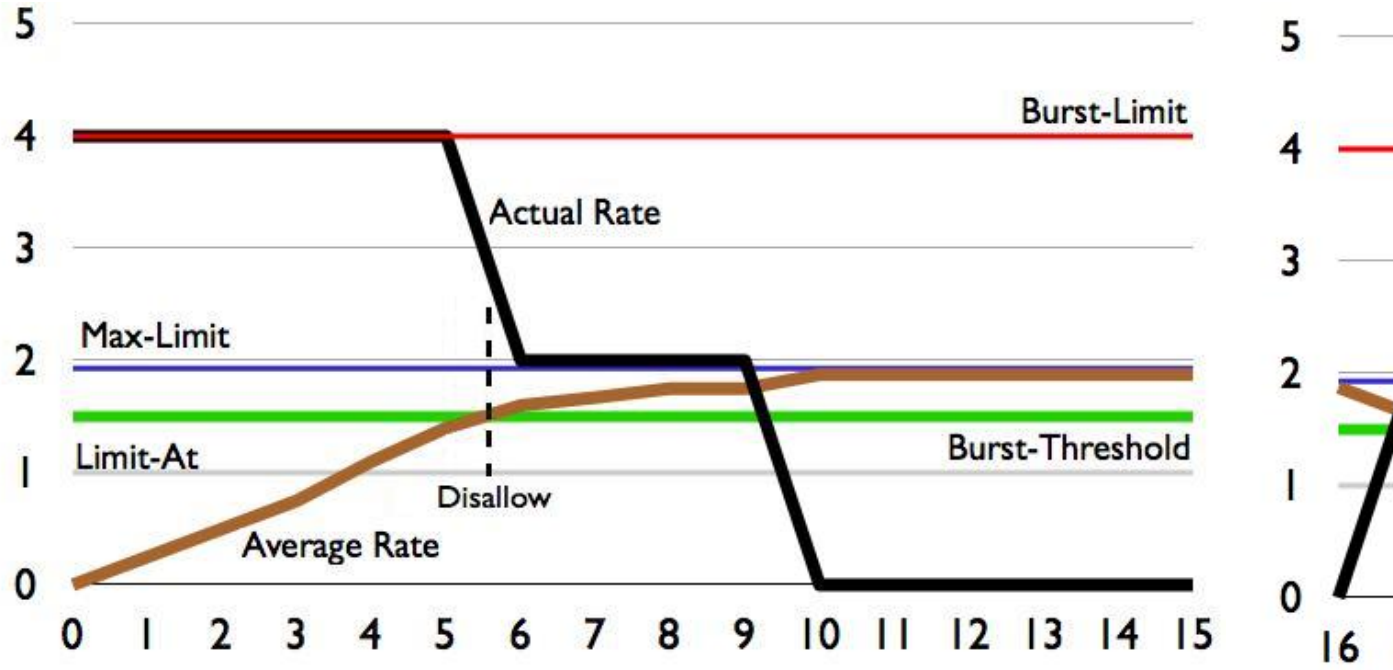
1. **burst-limit** (NUMBER) : maximal upload/download data rate which can be reached while the burst is allowed
2. **burst-time** (TIME) : period of time, in seconds, over which the average data rate is calculated. (This is NOT the time of actual burst)
3. **burst-threshold** (NUMBER) : this is value of burst on/off switch
4. **average-rate** (read-only) : Every 1/16 part of the **burst-time**, the router calculates the average data rate of each class over the last **burst-time** seconds
5. **actual-rate** (read-only) : actual traffic transfer rate of the queue

Example

Values: **limit-at=1M** , **max-limit=2M** , **burst-threshold=1500k** , **burst-limit=4M**

Client will try to download two 4MB (32Mb) blocks of data, first download will start at zero seconds, second download will start at 17th second. Traffic was unused for last minute.

Burst-time=16s



As we can see as soon as client requested bandwidth it was able to get 4Mpbs burst for 6 seconds. This is longest possible burst with given values *(longest-burst-time = burst-threshold * burst-time / burst-limit)*. As soon as burst runs out rest of the data will be downloaded with 2Mbps. This way block of data was downloaded in 9 seconds - without burst it would take 16 seconds. Burst have 7 seconds to recharge before next download will start.

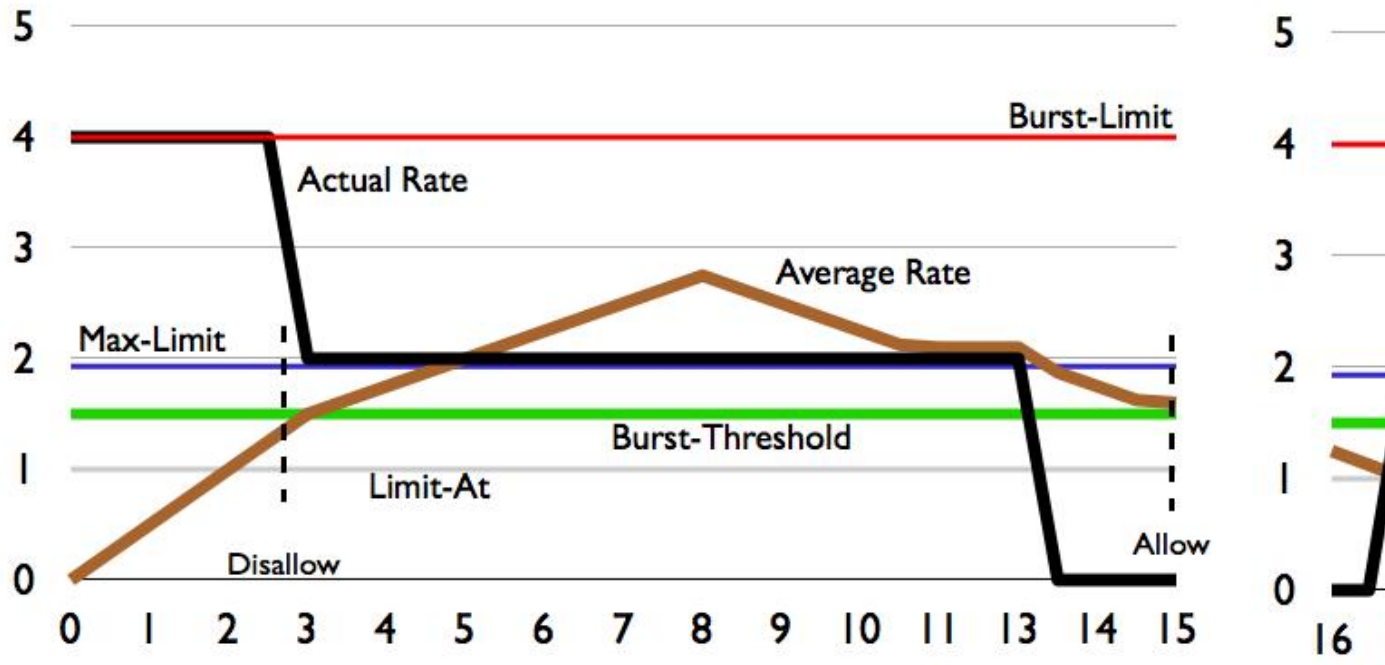
Note that burst is still disallowed when download started and it kicks in only afterwards - in the middle of download. So with this example we proved that burst may happen in the middle of download. Burst was ~4 seconds long and second block of was downloaded 4 seconds faster then without burst.

Average rate is calculated every 1/16 of burst time, so in this case 1s

Time	average-rate	burst	actual-rate
0	(0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0)/16=0Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps
1	(0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+4)/16=250Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps
2	(0+0+0+0+0+0+0+0+0+0+0+0+0+0+4+4)/16=500Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps
3	(0+0+0+0+0+0+0+0+0+0+0+0+0+4+4+4)/16=750Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps
4	(0+0+0+0+0+0+0+0+0+0+0+0+4+4+4+4)/16=1000Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps
5	(0+0+0+0+0+0+0+0+0+0+0+4+4+4+4+4)/16=1250Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps
6	(0+0+0+0+0+0+0+0+0+0+4+4+4+4+4+4)/16=1500Kbps	average-rate = burst-threshold → Burst **not** allowed	**2Mbps**
7	(0+0+0+0+0+0+0+0+0+4+4+4+4+4+4+2)/16=1625Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps
8	(0+0+0+0+0+0+0+0+4+4+4+4+4+4+2+2)/16=1750Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps
9	(0+0+0+0+0+0+0+4+4+4+4+4+4+2+2+2)/16=1750Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps
10	(0+0+0+0+0+0+4+4+4+4+4+4+2+2+2+2)/16=1875Kbps	average-rate > burst-threshold → Burst not allowed	**0Mbps**

11	(0+0+0+0+0+4+4+4+4+4+4+2+2+2+2+0)/16=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps
12	(0+0+0+0+4+4+4+4+4+4+4+2+2+2+2+0+0)/16=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps
13	(0+0+0+4+4+4+4+4+4+4+2+2+2+2+0+0+0)/16=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps
14	(0+0+4+4+4+4+4+4+4+2+2+2+2+0+0+0+0)/16=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps
15	(0+4+4+4+4+4+4+4+2+2+2+2+0+0+0+0+0)/16=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps
16	(4+4+4+4+4+4+4+2+2+2+2+0+0+0+0+0+0)/16=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps
17	(4+4+4+4+4+4+2+2+2+2+0+0+0+0+0+0+0)/16=1625Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps
18	(4+4+4+4+2+2+2+2+0+0+0+0+0+0+0+2)/16=1500Kbps	average-rate = burst-threshold → Burst not allowed	2Mbps
19	(4+4+4+2+2+2+2+0+0+0+0+0+0+2+2)/16=1375Kbps	average-rate < burst-threshold → Burst **is** allowed	4Mbps
20	(4+4+2+2+2+2+0+0+0+0+0+0+2+2+4)/16=1375Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps
21	(4+2+2+2+2+0+0+0+0+0+0+2+2+4+4)/16=1375Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps
22	(2+2+2+2+0+0+0+0+0+0+2+2+4+4+4)/16=1375Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps
23	(2+2+2+0+0+0+0+0+0+2+2+4+4+4+4)/16=1500Kbps	average-rate = burst-threshold → Burst **not** allowed	2Mbps
24	(2+2+0+0+0+0+0+0+2+2+4+4+4+4+2)/16=1500Kbps	average-rate = burst-threshold → Burst not allowed	2Mbps
25	(2+0+0+0+0+0+0+2+2+4+4+4+4+2+2)/16=1500Kbps	average-rate = burst-threshold → Burst not allowed	2Mbps
26	(0+0+0+0+0+0+2+2+4+4+4+4+2+2+2)/16=1500Kbps	average-rate = burst-threshold → Burst not allowed	2Mbps
27	(0+0+0+0+0+2+2+4+4+4+4+2+2+2+2)/16=1625Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps
28	(0+0+0+0+2+2+4+4+4+4+2+2+2+2+2)/16=1750Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps
29	(0+0+0+2+2+4+4+4+4+2+2+2+2+2+2)/16=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps
30	(0+0+2+2+4+4+4+4+2+2+2+2+2+2+0)/16=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps
31	(0+0+2+2+4+4+4+4+2+2+2+2+2+2+0+0)/16=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps

Burst-time=8s



If we decrease burst-time to 8 seconds - we are able to see that in this case bursts are only at the beginning of downloads

Average rate is calculated every 1/16th of burst time, so in this case every 0.5 seconds.

Time	average-rate	burst	actual-rate
0.0	(0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0)/8=0Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
0.5	(0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+2)/8=250Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
1.0	(0+0+0+0+0+0+0+0+0+0+0+0+0+0+2+2)/8=500Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
1.5	(0+0+0+0+0+0+0+0+0+0+0+0+0+2+2+2)/8=750Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
2.0	(0+0+0+0+0+0+0+0+0+0+0+0+2+2+2+2)/8=1000Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
2.5	(0+0+0+0+0+0+0+0+0+0+0+2+2+2+2+2)/8=1250Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
3.0	(0+0+0+0+0+0+0+0+0+0+2+2+2+2+2+2)/8=1500Kbps	average-rate = burst-threshold → Burst **not** allowed	**2Mbps** (1Mb per 0,5sek)
3.5	(0+0+0+0+0+0+0+0+0+2+2+2+2+2+2+1)/8=1625Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
4.0	(0+0+0+0+0+0+0+0+2+2+2+2+2+2+1+1)/8=1750Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
4.5	(0+0+0+0+0+0+0+2+2+2+2+2+2+1+1+1)/8=1875Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
5.0	(0+0+0+0+0+0+2+2+2+2+2+2+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
5.5	(0+0+0+0+0+2+2+2+2+2+2+1+1+1+1+1)/8=2125Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
6.0	(0+0+0+0+2+2+2+2+2+2+1+1+1+1+1+1)/8=2250Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
6.5	(0+0+0+2+2+2+2+2+2+1+1+1+1+1+1+1)/8=2375Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
7.0	(0+0+2+2+2+2+2+2+1+1+1+1+1+1+1+1)/8=2500Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
7.5	(0+2+2+2+2+2+2+1+1+1+1+1+1+1+1+1)/8=2625Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
8.0	(2+2+2+2+2+2+1+1+1+1+1+1+1+1+1+1)/8=2750Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
8.5	(2+2+2+2+2+1+1+1+1+1+1+1+1+1+1+1)/8=2625Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
9.0	(2+2+2+2+1+1+1+1+1+1+1+1+1+1+1+1)/8=2500Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
9.5	(2+2+2+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2375Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
10.0	(2+2+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2250Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
10.5	(2+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2125Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
11.0	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
11.5	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
12.0	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
12.5	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
13.0	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	**0Mbps** (0Mb per 0,5sek)
13.5	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+0)/8=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps (0Mb per 0,5sek)
14.0	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+0+0)/8=1750Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps (0Mb per 0,5sek)
14.5	(1+1+1+1+1+1+1+1+1+1+1+1+1+0+0+0)/8=1625Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps (0Mb per 0,5sek)
15.0	(1+1+1+1+1+1+1+1+1+1+1+1+0+0+0+0)/8=1500Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps (0Mb per 0,5sek)
15.5	(1+1+1+1+1+1+1+1+1+1+1+0+0+0+0+0)/8=1375Kbps	average-rate < burst-threshold → Burst **is** allowed	0Mbps (0Mb per 0,5sek)
16.0	(1+1+1+1+1+1+1+1+1+1+0+0+0+0+0+0)/8=1250Kbps	average-rate < burst-threshold → Burst is allowed	0Mbps (0Mb per 0,5sek)
16.5	(1+1+1+1+1+1+1+1+1+0+0+0+0+0+0+0)/8=1125Kbps	average-rate < burst-threshold → Burst is allowed	0Mbps (0Mb per 0,5sek)
17.0	(1+1+1+1+1+1+1+1+0+0+0+0+0+0+0+0)/8=1000Kbps	average-rate < burst-threshold → Burst is allowed	**2Mbps** (1Mb per 0,5sek)

17.5	(1+1+1+1+1+1+1+0+0+0+0+0+0+0+0+1)/8=1000Kbps	average-rate < burst-threshold → Burst is allowed	**4Mbps** (2Mb per 0,5sek)
18.0	(1+1+1+1+1+1+0+0+0+0+0+0+0+0+1+2)/8=1125Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
18.5	(1+1+1+1+1+0+0+0+0+0+0+0+0+1+2+2)/8=1250Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
19.0	(1+1+1+1+0+0+0+0+0+0+0+0+1+2+2+2)/8=1375Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
19.5	(1+1+1+0+0+0+0+0+0+0+0+1+2+2+2+2)/8=1500Kbps	average-rate = burst-threshold → Burst **not** allowed	2Mbps (1Mb per 0,5sek)
20.0	(1+1+0+0+0+0+0+0+0+0+1+2+2+2+2+1)/8=1500Kbps	average-rate = burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
20.5	(1+0+0+0+0+0+0+0+0+1+2+2+2+2+1+1)/8=1500Kbps	average-rate = burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
21.0	(0+0+0+0+0+0+0+0+1+2+2+2+2+1+1+1)/8=1500Kbps	average-rate = burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
21.5	(0+0+0+0+0+0+0+1+2+2+2+2+1+1+1+1)/8=1625Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
22.0	(0+0+0+0+0+0+1+2+2+2+2+1+1+1+1+1)/8=1750Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
22.5	(0+0+0+0+0+1+2+2+2+2+1+1+1+1+1+1)/8=1875Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
23.0	(0+0+0+0+1+2+2+2+2+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
23.5	(0+0+0+1+2+2+2+2+1+1+1+1+1+1+1+1)/8=2125Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
24.0	(0+0+1+2+2+2+2+1+1+1+1+1+1+1+1+1)/8=2250Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
24.5	(0+1+2+2+2+2+1+1+1+1+1+1+1+1+1+1)/8=2375Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
25.0	(1+2+2+2+2+1+1+1+1+1+1+1+1+1+1+1)/8=2500Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
25.5	(2+2+2+2+1+1+1+1+1+1+1+1+1+1+1+1)/8=2500Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
26.0	(2+2+2+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2375Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
26.5	(2+2+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2250Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
27.0	(2+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2125Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
27.5	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
28.0	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
28.5	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
29.0	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
29.5	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
30.0	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
30.5	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps (0Mb per 0,5sek)
31.0	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+0)/8=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps (0Mb per 0,5sek)

Manual:Queues - PCQ

Applies to RouterOS: 2.9, v3, v4

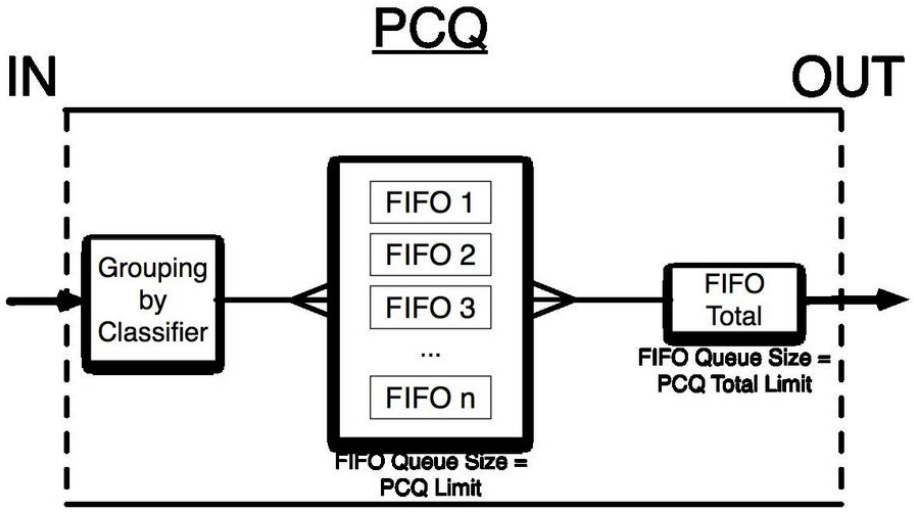
Usage

PCQ was introduced to optimize massive QoS systems, where most of the queues are exactly the same for different sub-streams. For example a sub-stream can be download or upload for one particular client (IP) or connection to server.

PCQ algorithm is very simple - at first it uses selected classifiers to distinguish one sub-stream from another, then applies individual FIFO queue size and limitation on every sub-stream, then groups all sub-streams together and applies global FIFO queue size and limitation.

PCQ parameters:

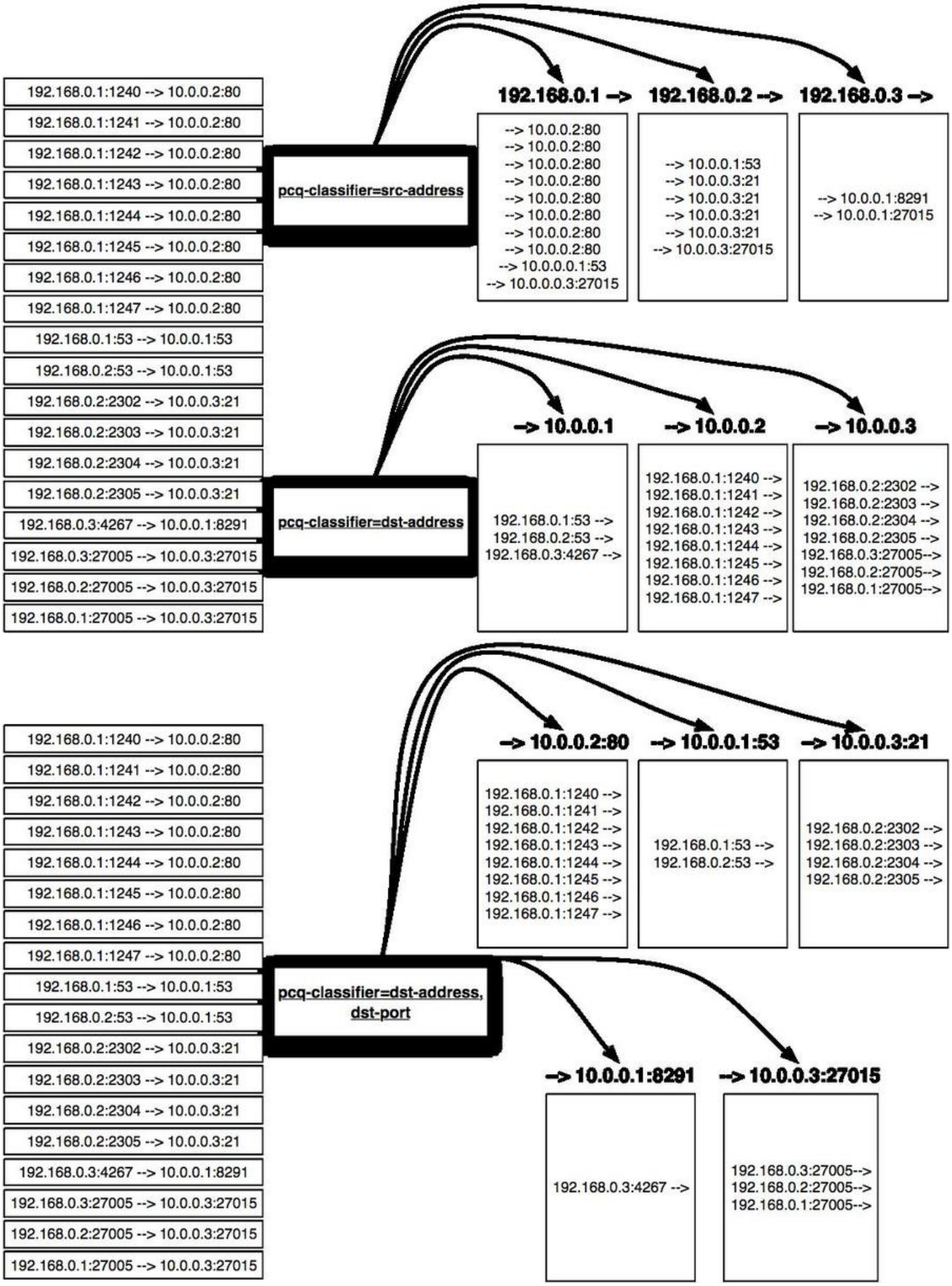
- **pcq-classifier** (dst-address | dst-port | src-address | src-port; default: "") : selection of sub-stream identifiers
- **pcq-rate** (number) : maximal available data rate of each sub-steam
- **pcq-limit** (number) : queue size of single sub-stream (in KB)
- **pcq-total-limit** (number) : queue size of global FIFO queue (in KB)



So instead of having 100 queues with 1000kbps limitation for download we can have one PCQ queue with 100 sub-streams

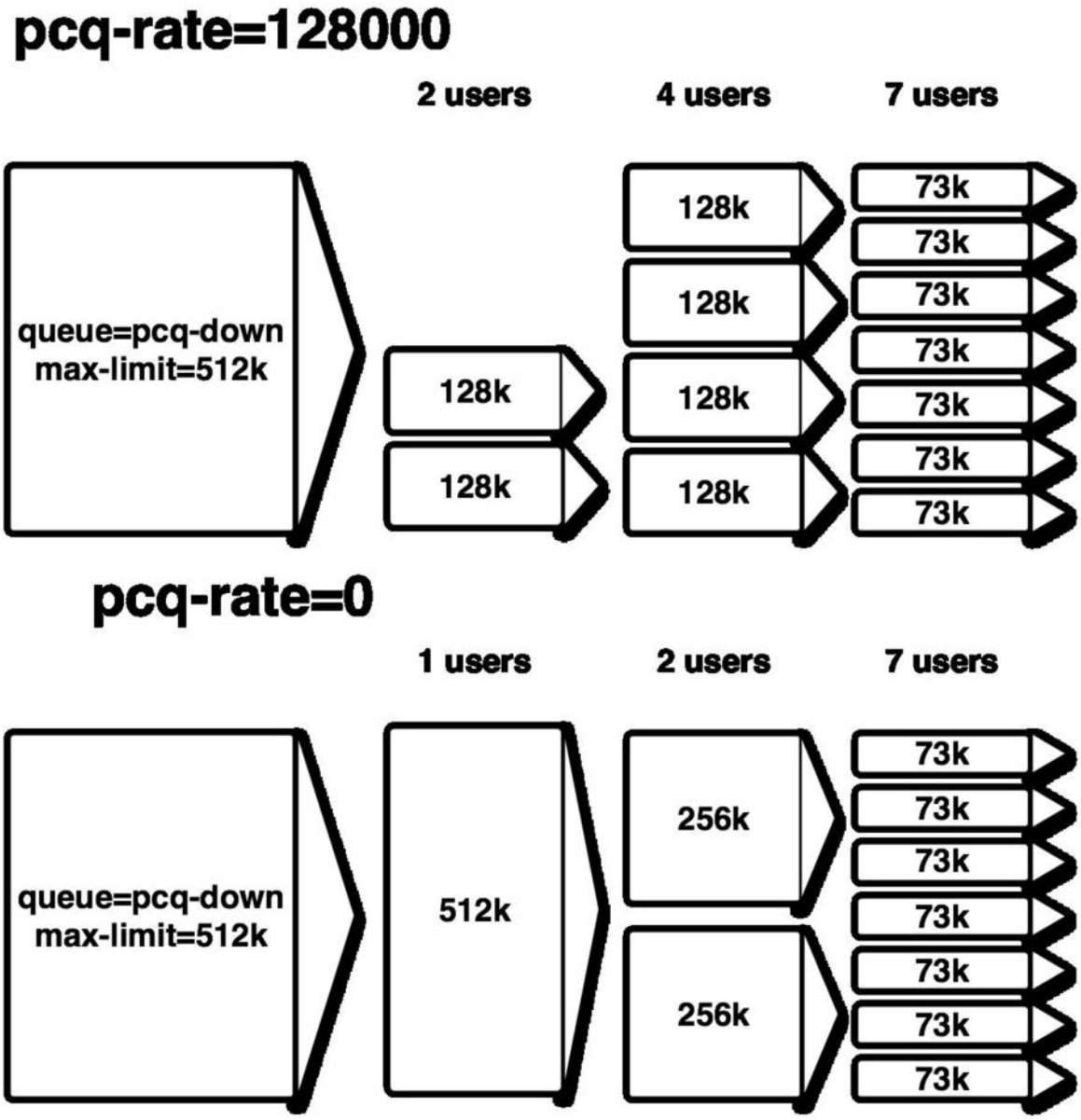
Classification Examples

To better understand classification we will take a list of 18 packet streams from specific address and port, to a specific address and port. Then we will choose a classifier and divide all 18 packet streams into PCQ sub-streams



PCQ Rate Examples

Here it is possible to see what happens if PCQ-rate is, or isn't specified. I must noted that if both limits (pcq-rate and max-limit) are unspecified, queue behavior can be imprecise. So it is strongly suggested to have at least one of these options set.



New PCQ implementation (v5.0RC5)

PCQ was rewritten in v5.0RC4 to optimize it high throughput both in Mbps and pps. This implementation properly utilize all new Linux Kernel features, this makes PCQ faster and less resource demanding.

Now as soon as new stream activates it will get 1/4th of rate with highest priority. If rate is "0" sub-stream will not have this feature (as 1/4th of "0" is "0")

This is necessary to know for one good reason: Lets assume that sub-stream's rate is 10Mbps, so in the moment when new sub-stream will request traffic it will get first 2500k of traffic without limitation. This may result in higher that expected results in such programs as Speedtest.net. To avoid that make sure that Speedtest.net is not the first program that utilize bandwidth that you run on PC.

Also starting from v5.0RC5 PCQ have new features

PCQ Burst for sub-streams. PCQ will have burst implementation identical to Simple Queues and Queue Tree

PCQ parameters:

- **pcq-burst-rate** (number) : maximal upload/download data rate which can be reached while the burst for substream is allowed
- **pcq-burst-threshold** (number) : this is value of burst on/off switch
- **pcq-burst-time** (time) : period of time, in seconds, over which the average data rate is calculated. (This is NOT the time of actual burst)

For detailed burst explanation refer to:

- Burst

PCQ also allows to use different size IPv4 and IPv6 networks as sub-stream identifiers . Before it was locked to single IP address. This is done mainly for IPv6 as customers from ISP point of view will be represented by /64 network, but devices in customers network will be /128. PCQ can be used for both of these scenarios and more.

PCQ parameters:

- **pcq-dst-address-mask** (number) : size of IPv4 network that will be used as dst-address sub-stream identifier
- **pcq-src-address-mask** (number) : size of IPv4 network that will be used as src-address sub-stream identifier
- **pcq-dst-address6-mask** (number) : size of IPV6 network that will be used as dst-address sub-stream identifier
- **pcq-src-address6-mask** (number) : size of IPV6 network that will be used as src-address sub-stream identifier

See Also

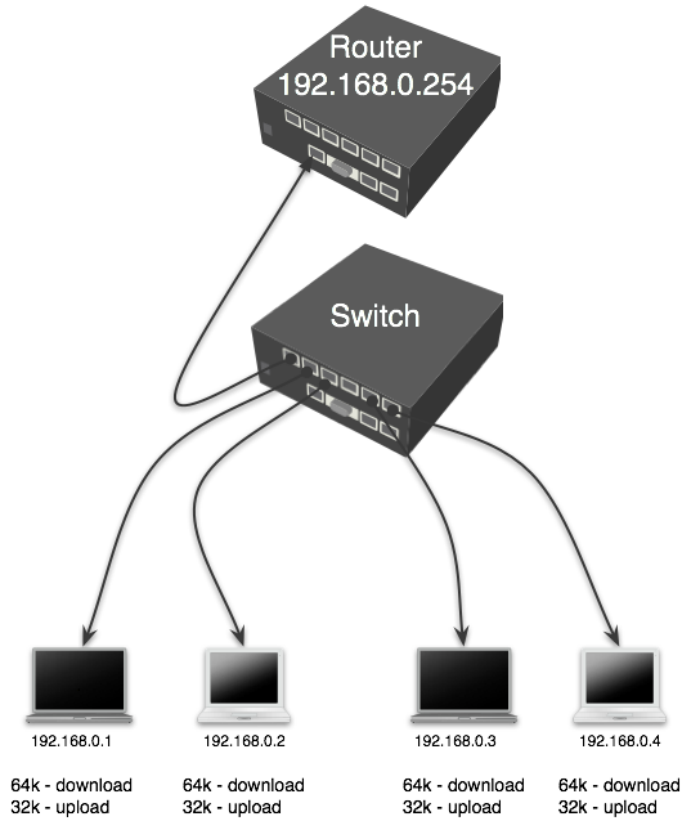
- PCQ Examples

Manual:Queues - PCQ Examples

Per Connection Queue (PCQ) is a queuing discipline that can be used to dynamically equalize or shape traffic for multiple users, using little administration. It is possible to divide PCQ scenarios into three major groups: equal bandwidth for a number of users, certain bandwidth equal distribution between users, unknown bandwidth equal distribution between users.

Equal Bandwidth for a Number of Users

Use PCQ type queue when you need to equalize the bandwidth [and set max limit] for a number of users. We will set the 64kbps download and 32kbps upload limits.



There are two ways how to make this: using mangle and queue trees, or, using simple queues.

1. Mark all packets with packet-marks upload/download: (lets constider that ether1-LAN is public interface to the Internet and ether2-LAN is local interface where clients are connected

```
/ip firewall mangle add chain=prerouting action=mark-packet \
   in-interface=ether1-LAN new-packet-mark=client_upload
/ip firewall mangle add chain=prerouting action=mark-packet \
   in-interface=ether2-WAN new-packet-mark=client_download
```

2. Setup two PCQ queue types - one for download and one for upload. *dst-address* is classifier for user's download traffic, *src-address* for upload traffic:

```
/queue type add name="PCQ_download" kind=pcq pcq-rate=64000 pcq-classifier=dst-address
/queue type add name="PCQ_upload" kind=pcq pcq-rate=32000 pcq-classifier=src-address
```

3. Finally, two queue rules are required, one for download and one for upload:

```
/queue tree add parent=global-in queue=PCQ_download packet-mark=client_download
/queue tree add parent=global-out queue=PCQ_upload packet-mark=client_upload
```

If you don't like using mangle and queue trees, you can skip step 1, do step 2, and step 3 would be to create one simple queue as shown here:

```
/queue simple add target-addresses=192.168.0.0/24 queue=PCQ_upload/PCQ_download \
   packet-marks=client_download,client_upload
```

Note: More information about certain and unknown Distribution between routers can be found in PCQ manual.

See Also

- PCQ

Manual:Queue Size

Applies to RouterOS: 2.9, v3, v4

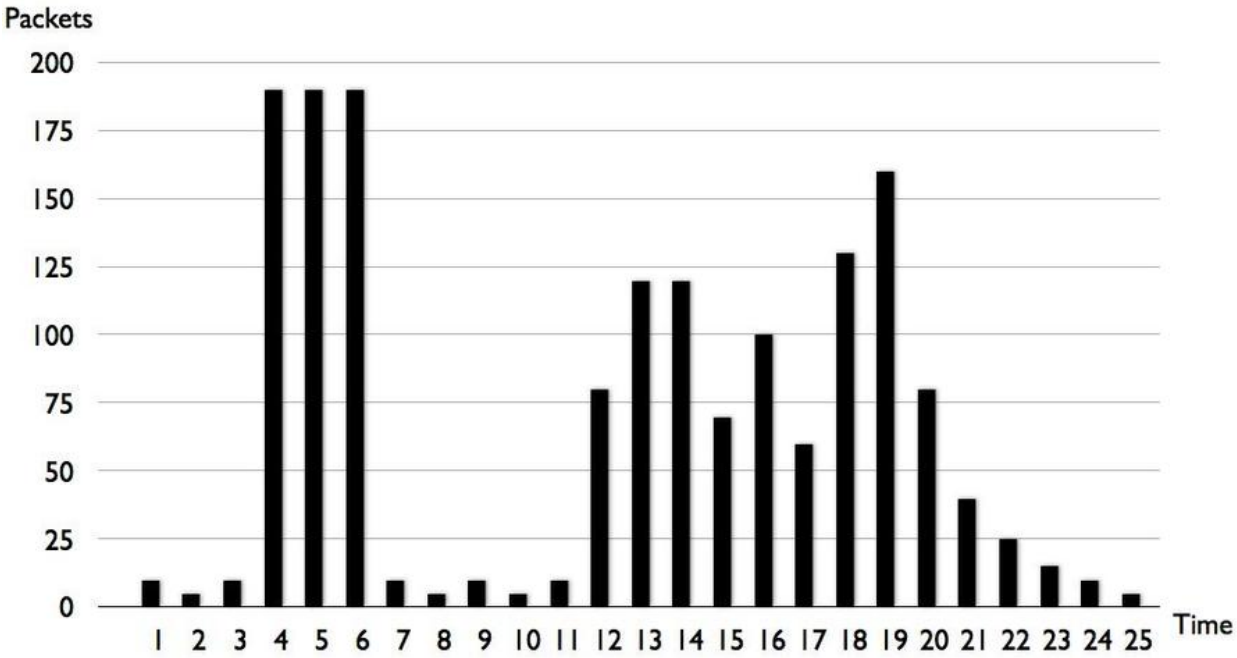
Queue Size Example

This example was created to highlight queue size impact on traffic that was queued by specific queue.

In Mikrotik RouterOS queue size can be specified in the "/queue type" menu. Each queue type have a different option for specifying queue size (pfifo-limit, bfifo-limit, pcq-limit, pcq-total-limit, red-limit), but all principles are the same - queue size is main option that decide should the package be dropped or scheduled for later time.

In real time environment this process is happening continuously without any stops, steps or other interruptions, but in order to show it as an example we will divide it into steps, where it is possible to know exactly how many packets will be received/transited in every step.

We will not go into specific details of TCP and dropped packet retransmission - consider these packets as simple UDP stream.

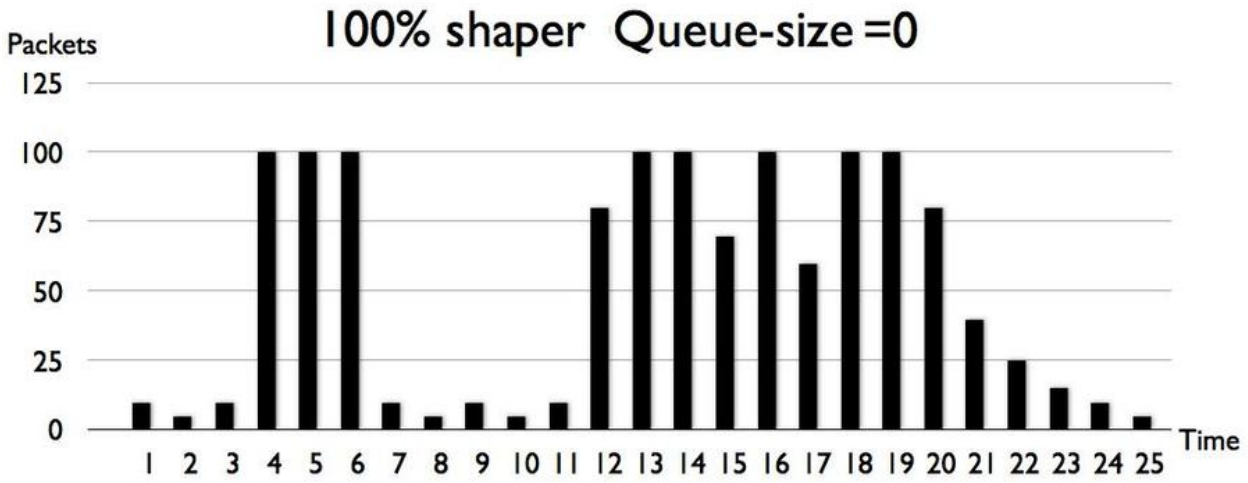


As you can see in the picture above there are **25 steps** and there are total of **1610 incoming packets** over this time frame.

100% Shaper

Queue is 100% shaper when every packet that is over allowed limits will be dropped immediately. This way all packages that are not dropped will be sent out without any delay.

Lets apply **max-limit=100 packets per step** limitation to our example:



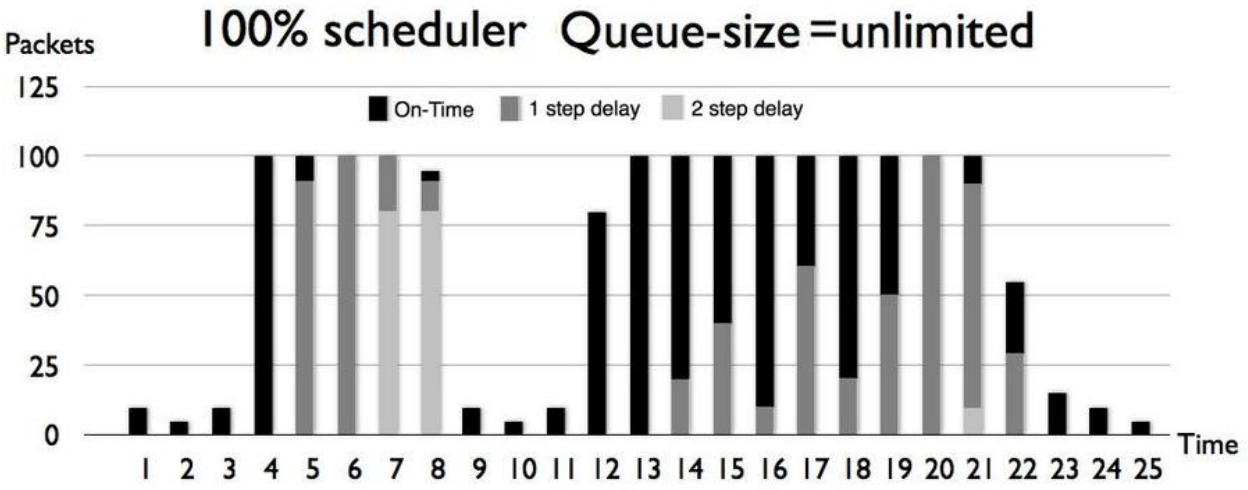
With this type of limitation only 1250 out of 1610 packets were able to pass the queue (**22,4% packet drop**), but all packets arrive without delay.

100% Scheduler

Queue is 100% Scheduler when there is no packet drops at all, all packets are queued and will be sent out at the first possible moment.

In each step queue must send out queued packets from previous steps first and only then sent out packets from this step, this way it is possible to keep right sequence of packets.

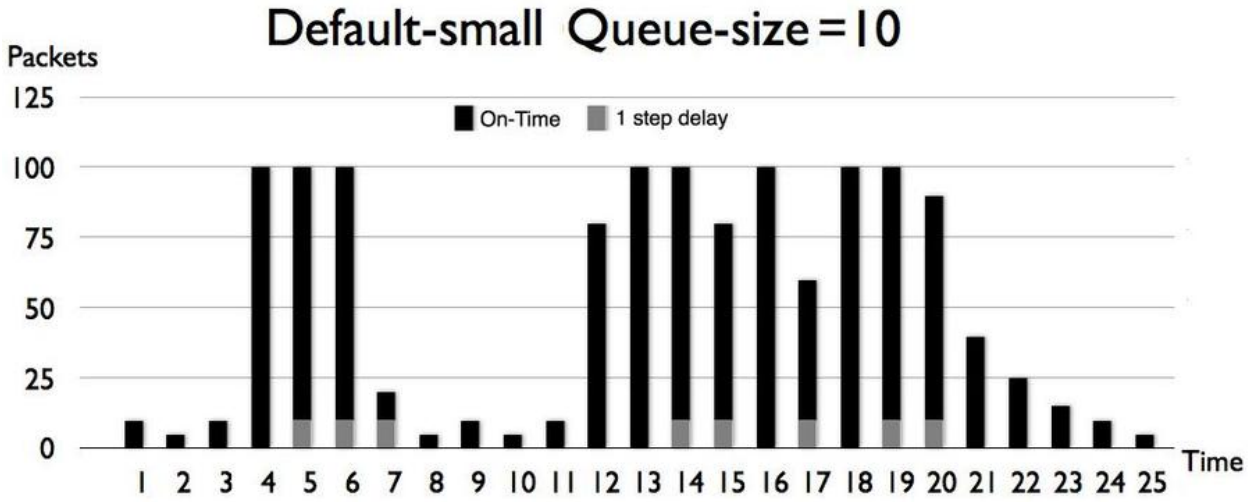
We will again use same limit (**100 packets per step**)



There was no packet loss, but 630 **(39,1%) packets had 1 step delay**, and other 170 **(10,6%) packets had 2 step delay**. (delay = latency)

Default-small queue type

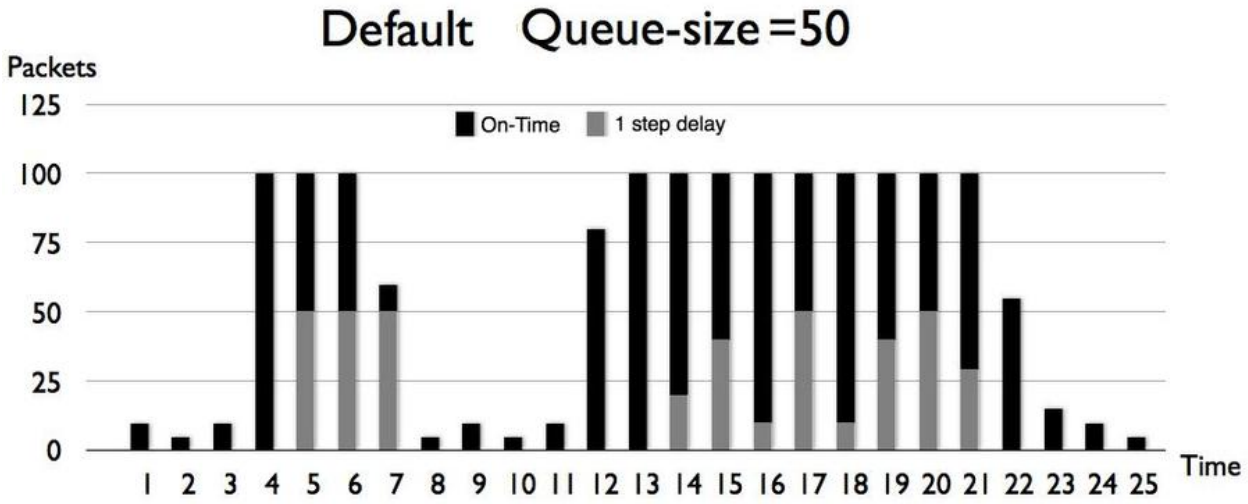
It is also possible to choose the middle way, when queue use both of these queuing aspects (shaping and scheduling) By default most of the queues in RouterOS have queue size of 10.



There were 320 **(19,9%) packets dropped** and 80 **(5,0%) packets had 1 step delay**.

Default queue type

Other popular queue size in RouterOS is 50



There were 190 **(11,8%) packets dropped** and 400 **(24,8%) packets had 1 step delay**.

Manual:IP/Route

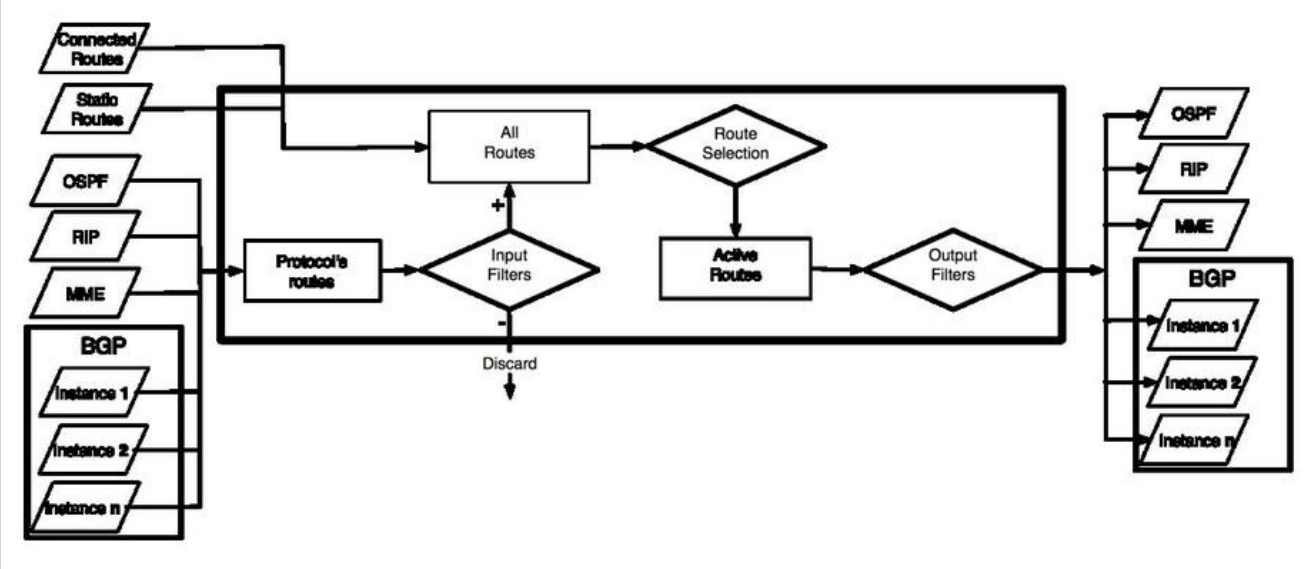
Applies to RouterOS: v3, v4, v5+

Overview

Router keeps routing information in several separate spaces:

- FIB (Forwarding Information Base), that is used to make packet forwarding decisions. It contains a copy of the necessary routing information.
- Each routing protocol (except BGP) has it's own internal tables. This is where per-protocol routing decisions are made. BGP does not have internal routing tables and stores complete routing information from all peers in the RIB.
- RIB contains routes grouped in separate routing tables based on their value of **routing-mark**. All routes without **routing-mark** are kept in the *main* routing table. These tables are used for best route selection. The *main* table is also used for nexthop lookup.

Routing Information Base



RIB (Routing Information Base) contains complete routing information, including static routes and policy routing rules configured by the user, routing information learned from routing protocols, information about connected networks. RIB is used to filter routing information, calculate best route for each destination prefix, build and update Forwarding Information Base and to distribute routes between different routing protocols.

By default forwarding decision is based only on the value of destination address. Each route has **dst-address** property, that specifies all destination addresses this route can be used for. If there are several routes that apply to a particular IP address, the most specific one (with largest netmask) is used. This operation (finding the most specific route that matches given address) is called *routing table lookup*.

If routing table contains several routes with the same **dst-address**, only one of them can be used to forward packets. This route is installed into FIB and marked as *active*.

When forwarding decision uses additional information, such as a source address of the packet, it is called *policy routing*. Policy routing is implemented as a list of policy routing rules, that select different routing table based on destination address, source address, source interface, and routing mark (can be changed by firewall mangle rules) of the packet.

All routes by default are kept in the *main* routing table. Routes can be assigned to specific routing table by setting their **routing-mark** property to the name of another routing table. Routing tables are referenced by their name, and are created automatically when they are referenced in the configuration.

Each routing table can have only one active route for each value of **dst-address** IP prefix.

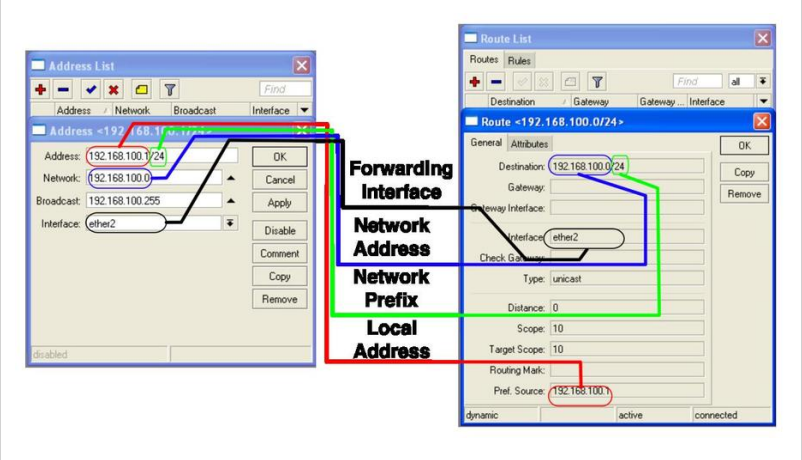
There are different groups of routes, based on their origin and properties.

Default route

Route with **dst-address** *0.0.0.0/0* applies to every destination address. Such route is called the *default route*. If routing table contains an active default route, then routing table lookup in this table will never fail.

Connected routes

Connected routes are created automatically for each IP network that has at least one enabled interface attached to it (as specifie in the `/ip address` configuration). RIB tracks status of connected routes, but does not modify them. For each connected route there is one *ip address* item such that:



- address part of **dst-address** of connected route is equal to **network** of *ip address* item.
- netmask part of **dst-address** of connected route is equal to netmask part of **address** of *ip address* item.
- **pref-src** of connected route is equal to address part of **address** of *ip address* item.
- **interface** of connected route is equal to **actual-interface** of *ip address* item (same as **interface**, except for bridge interface ports).

Multipath (ECMP) routes

Because results of the forwarding decision are cached, packets with the same **source address, destination address, source interface, routing mark** and **ToS** are sent to the same gateway. This means that one connection will use only one link in each direction, so ECMP routes can be used to implement per-connection load balancing. See interface bonding if you need to achieve per-packet load balancing.

To implement some setups, such as load balancing, it might be necessary to use more than one path to given destination. However, it is not possible to have more than one active route to destination in a single routing table.

ECMP (Equal cost multi-path) routes have multiple **gateway** nexthop values. All reachable nexthops are copied to FIB and used in forwarding packets.

OSPF protocol can create ECMP routes. Such routes can also be created manually.

Routes with interface as a gateway

Value of **gateway** can be specified as an interface name instead of the nexthop IP address. Such route has following special properties:

- Unlike connected routes, routes with interface nexthops are not used for nexthop lookup.
- It is possible to assign several interfaces as a value of **gateway**, and create ECMP route. It is not possible to have connected route with multiple **gateway** values.

Route selection

Each routing table can have one active route for each destination prefix. This route is installed into FIB. Active route is selected from all candidate routes with the same **dst-address** and **routing-mark**, that meet the criteria for becoming an active route. There can be multiple such routes from different routing protocols and from static configuration. Candidate route with the lowest **distance** becomes an active route. If there is more than one candidate route with the same **distance**, selection of active route is arbitrary (except for BGP routes).

BGP has the most complicated selection process (described in separate article). Notice that this protocol-internal selection is done only after BGP routes are installed in the main routing table; this means there can be one candidate route from each BGP peer. Also note that BGP routes from different BGP instances are compared by their distance, just like other routes.

Criteria for selecting candidate routes

To participate in route selection process, route has to meet following criteria:

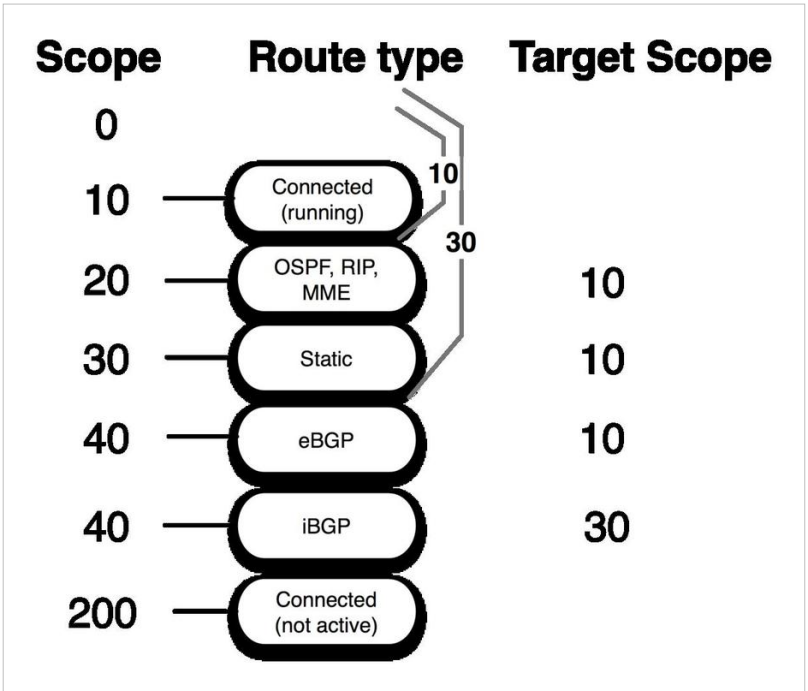
- route is not disabled.
- **distance** is not *255*. Routes that are rejected by route filter have **distance** value of *255*.
- **pref-src** is either not set or is a valid local address of the router.
- **routing-mark** is either not set or is referred by firewall or policy routing rules.
- If type of route is *unicast* and it is not a connected route, it must have at least one reachable nexthop.

Nexthop lookup

Nexthop lookup is a part of the route selection process.

Routes that are installed in the FIB need to have interface associated with each gateway address. Gateway address (nexthop) has to be directly reachable via this interface. Interface that should be used to send out packets to each gateway address is found by doing nexthop lookup.

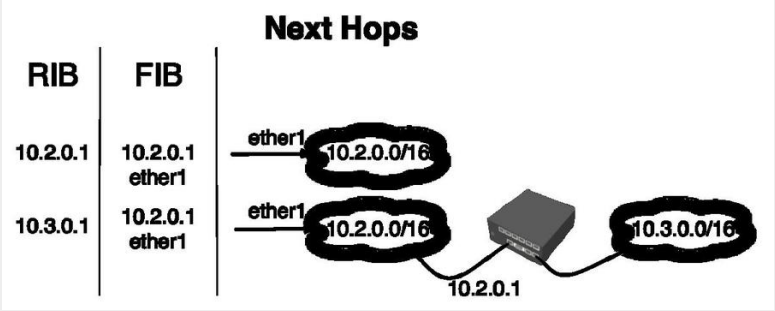
Some routes (e.g. iBGP) may have gateway address that is several hops away from this router. To install such routes in the FIB, it is necessary to find the address of the directly reachable gateway (an immediate nexthop), that



should be used to reach the gateway address of this route. Immediate nextop addresses are also found by doing nexthop lookup.

Nexthop lookup is done only in the *main* routing table, even for routes with different value of **routing-mark**. It is necessary to restrict set of routes that can be used to look up immediate nexthops. Nexthop values of RIP or OSPF routes, for example, are supposed to be directly reachable and should be looked up only using connected routes. This is achieved using **scope** and **target-scope** properties.

- Routes with interface name as the value of gateway are not used for nexthop lookup. If route has both interface nexthops and active IP address nexthops, then interface nexthops are ignored.
- Routes with **scope** greater than the maximum accepted value are not used for nexthop lookup. Each route specifies maximum accepted **scope** value for it's nexthops in the **target-scope** property. Default value of this property allows nexthop lookup only through connected routes, with the exception of iBGP routes that have larger default value and can lookup nexthop also through IGP and static routes.



Recursive nexthop lookup example

- nexthop 10.2.0.1 is resolved through a connected route, it's status is *reachable*.
- nexthop 10.3.0.1 is resolved recursively through a 10.3.0.0/16 route, it's status is *recursive*, and it uses 10.2.0.1 as the immediate nexthop value that is installed in the FIB.

Interface and immediate nexthop are selected based on the result of nexthop lookup:

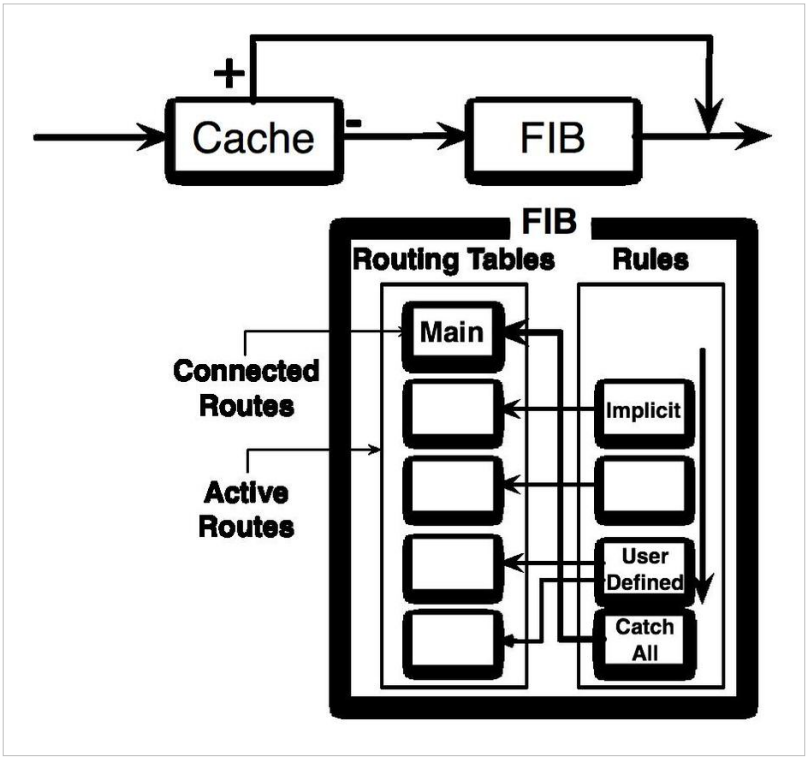
- If most specific active route that nexthop lookup finds is connected route, then interface of this connected route is used as the nexthop interface, and this gateway is marked as *reachable*. Since gateway is directly reachable through this interface (that's exactly what connected route means), the gateway address is used as the immediate nexthop address.
- If most specific active route that nexthop lookup finds has nexthop that is already resolved, immediate nexthop address and interface is copied from that nexthop and this gateway is marked as *recursive*.
- If most specific active route that nexthop lookup finds is ECMP route, then it uses first gateway of that route that is not *unreachable*.
- If nexthop lookup does not find any route, then this gateway is marked as *unreachable*.

Forwarding Information Base

FIB (Forwarding Information Base) contains copy of information that is necessary for packet forwarding:

- all active routes
- policy routing rules

By default (when no **routing-mark** values are used) all active routes are in the *main* table, and there is only one hidden implicit rule ("catch all" rule) that uses the *main* table for all destination lookups.



Routing table lookup

FIB uses following information from packet to determine it's destination:

- source address
- destination address
- source interface
- routing mark
- ToS (not used by RouterOS in policy routing rules, but it is a part of routing cache lookup key)

Possible routing decisions are:

- receive packet locally
- discard packet (either silently or by sending ICMP message to the sender of the packet)
- send packet to specific IP address on specific interface

Results of routing decision are remembered in the routing cache. This is done to improve forwarding performance. When another packet with the same **source address, destination address, source interface, routing mark** and **ToS** is routed, cached results are used. This also allows to implement per-connection load balancing using ECMP routes, because values used to lookup entry in the routing cache are the same for all packets that belong to the same connection and go in the same direction.

If there is no routing cache entry for this packet, it is created by running routing decision:

- check that packet has to be locally delivered (destination address is address of the router)
- process implicit policy routing rules
- process policy routing rules added by user
- process implicit catch-all rule that looks up destination in the *main* routing table
- return result is "network unreachable"

Result of routing decision can be:

- IP address of nexthop + interface
- point-to-point interface
- local delivery
- discard
- ICMP prohibited
- ICMP host unreachable
- ICMP network unreachable

Rules that do not match current packet are ignored. If rule has action *drop* or *unreachable*, then it is returned as a result of the routing decision process. If action is *lookup* then destination address of the packet is looked up in routing table that is specified in the rule. If lookup fails (there is no route that matches destination address of packet), then FIB proceeds to the next rule. Otherwise:

- if type of the route is *blackhole*, *prohibit* or *unreachable*, then return this action as the routing decision result;
- if this is a connected route, or route with an interface as the **gateway** value, then return this interface and the destination address of the packet as the routing decision result;
- if this route has IP address as the value of **gateway**, then return this address and associated interface as the routing decision result;
- if this route has multiple values of nexthop, then pick one of them in round robin fashion.

Result of this routing decision is stored in new routing cache entry.

Properties

Route flags

Property(Flag)	Description
disabled (*X*)	Configuration item is disabled. It does not have any effect on other routes and is not used by forwarding or routing protocols in any way.
active (*A*)	Route is used for packet forwarding. See route selection.
dynamic (*D*)	Configuration item created by software, not by management interface. It is not exported, and cannot be directly modified.
connect (*C*)	connected route.
static (*S*)	static route.
rip (*r*)	RIP route.
bgp (*b*)	BGP route.
ospf (*o*)	OSPF route.
mme (*m*)	MME route.
blackhole (*B*)	Silently discard packet forwarded by this route.
unreachable (*U*)	Discard packet forwarded by this route. Notify sender with ICMP host unreachable (type 3 code 1) message.
prohibit (*P*)	Discard packet forwarded by this route. Notify sender with ICMP communication administratively prohibited (type 3 code 13) message.

General properties

Property	Description
check-gateway (*arp* | *ping*; Default: **""**)	Periodically (every 10 seconds) check gateway by sending either ICMP echo request (*ping*) or ARP request (*arp*). If no response from gateway is received for 10 seconds, request times out. After two timeouts gateway is considered unreachable. After receiving reply from gateway it is considered reachable and timeout counter is reset.
comment (*string*; Default: **""**)	Description of particular route
distance (*integer[1..255]*; Default: **"1"**)	Value used in route selection. Routes with smaller distance value are given preference. If value of this property is not set, then the default depends on route protocol: • connected routes: 0 • static routes: 1 • eBGP: 20 • OSPF: 110 • RIP: 120 • MME: 130 • iBGP: 200
dst-address (*IP prefix*; Default: **0.0.0.0/0**)	IP prefix of route, specifies destination addresses that this route can be used for. Netmask part of this property specifies how many of the most significant bits in packet destination address must match this value. If there are several active routes that match destination address of packet, then the most specific one (with largest netmask value) is used.
gateway (*IP IP%interface* | *IP@table[, IP* | *string, [..*; Default: **""**)	Array of IP addresses or interface names. Specifies which host or interface packets should be sent to. Connected routes and routes with *blackhole*, *unreachable* or *prohibit* type do not have this property. Usually value of this property is a single IP address of a gateway that can be directly reached through one of router's interfaces (but see nexthop lookup). ECMP routes have more than one gateway value. Value can be repeated several times.
pref-src (*IP*; Default: **""**)	Which of the local IP addresses to use for locally originated packets that are sent via this route. Value of this property has no effect on forwarded packets. If value of this property is set to IP address that is not local address of this router then the route will be inactive. If **pref-src** value is not set, then for locally originated packets that are sent using this route router will choose one of local addresses attached to the output interface that match destination prefix of the route (an example).
route-tag (*integer*; Default: **""**)	Value of route tag attribute for RIP or OSPF. For RIP only values 0..4294967295 are valid.
routing-mark (*string*; Default: **""**)	Name of routing table that contains this route. Not set by default which is the same as *main*. Packets that are marked by firewall with this value of **routing-mark** will be routed using routes from this table, unless overridden by policy routing rules. Not more than 250 routing marks are possible per router.
scope (*integer[0..255]*; Default: **"30"**)	Used in nexthop resolution. Route can resolve nexthop only through routes that have **scope** less than or equal to the **target-scope** of this route. Default value depends on route protocol: • connected routes: 10 (if interface is running) • OSPF, RIP, MME routes: 20 • static routes: 30 • BGP routes: 40 • connected routes: 200 (if interface is not running)
target-scope (*integer[0..255]*; Default: **"10"**)	Used in nexthop resolution. This is the maximum value of **scope** for a route through which a nexthop of this route can be resolved. See nexthop lookup. For iBGP value is set to 30 by default.
type (*unicast* | *blackhole* | *prohibit* | *unreachabl*; Default: **unicast**)	Routes that do not specify nexthop for packets, but instead perform some other action on packets have type different from the usual *unicast*. *blackhole* route silently discards packets, while *unreachable* and *prohibit* routes send ICMP Destination Unreachable message (code 1 and 13 respectively) to the source address of the packet.
vrf-interface (*string*; Default: **"10"**)	VRF interface name

Other Read-only properties

Property	Description
gateway-status (*array*)	Array of gateways, gateway states and which interface is used for forwarding. Syntax "IP state interface", for example "10.5.101.1 reachable bypass-bridge". State can be *unreachable*, *reachable* or *recursive*. See nexthop lookup for details.
ospf-metric (*integer*)	Used OSPF metric for particular route
ospf-type (*string*)	

BGP Route Properties

These properties contain information that is used by BGP routing protocol. However, values of these properties can be set for any type of route, including static and connected. It can be done either manually (for static routes) or using route filters.

Property	Description
bgp-as-path (*string*; Default: "")	Value of BGP AS_PATH attribute. Comma separated list of AS numbers with confederation AS numbers enclosed in *()* and AS_SETs enclosed in *{}*. Used to check for AS loops and in BGP route selection algorithm: routes with shorter AS_PATH are preferred (but read how AS_PATH length is calculated).
bgp-atomic-aggregate (*yes* | *no*; Default:)	Value of BGP ATOMIC_AGGREGATE attribute.
bgp-communities (*array of (integer:integer* | *internet* | *no-advertise* | *no-export* |*local-as*; Default:)	Value of BGP communities list. This attribute can be used to group or filter routes. Named values have special meanings: • *internet* - advertise this route to the Internet community (i.e. all routers) • *no-advertise* - do not advertise this route to any peers • *no-export* - do not advertise this route to EBGP peers • *local-as* - same as *no-export*, except that route is also advertised to EBGP peers inside local confederation
bgp-local-pref (*integer*; Default:)	Value of BGP LOCAL_PREF attribute. Used in BGP route selection algorithm: routes with greater LOCAL_PREF value are preferred. If value is not set then it is interpreted as 100.
bgp-med (*integer*; Default:)	Value of BGP MULTI_EXIT_DISC BGP attribute. Used in BGP route selection algorithm: routes with lower MULTI_EXIT_DISC value are preferred.. If value is not set then it is interpreted as 0.
bgp-origin (*igp* | *egp* | *incomplete*; Default:)	Value of BGP ORIGIN attribute. Used in BGP route selection algorithm: *igp* routes are preferred over *egp* and *egp* over *incomplete*.
bgp-prepend (*integer [0..16]*; Default:)	How many times to prepend router's own AS number to AS_PATH attribute when announcing route via BGP. Affects only routes sent to eBGP peers (for iBGP value 0 is always used).

Read-only

Property	Description
bgp-ext-communities (*string*)	Value of BGP extended communities attribute
bgp-weight (*integer*)	Additional value used by BGP best path selection algorithm. Routes with higher weight are preferred. It can be set by incoming routing filters and is useful only for BGP routes. If value is not set then it is interpreted as 0.
received-from (*string*)	Name of the BGP peer from which route is received.

User:Prince500

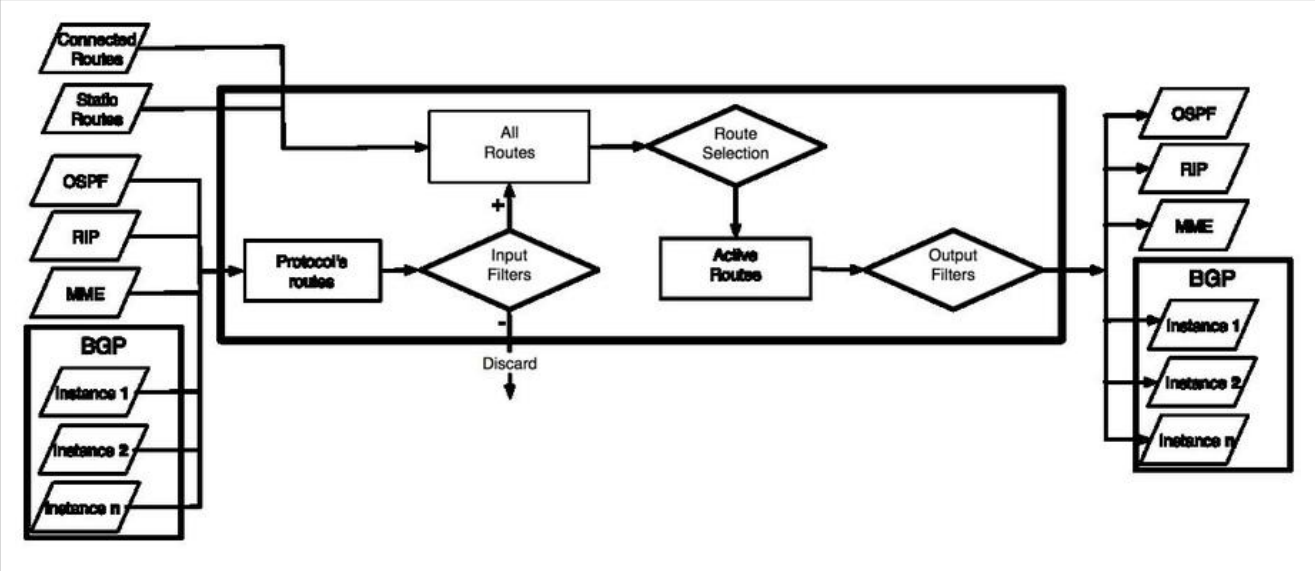
Applies to RouterOS: v3, v4, v5+

Обзор

Маршрутизатор хранит информацию о маршрутах в следующих раздельных областях:

- FIB (Forwarding Information Base), который используется для решения о пересылке пакета. Он содержит копию необходимой информации о маршрутах.
- Каждый протокол маршрутизации (кроме BGP) имеет собственные внутренние таблицы. На их основе принимаются решения о маршрутизации. BGP не имеет внутренних таблиц маршрутизации и хранит полную информацию о маршрутах всех соединений в RIB.
- RIB хранит маршруты, сгруппированные в раздельных таблицах, основываясь на значении параметра **routing-mark**. Все маршруты без параметра **routing-mark** помещаются в таблицу маршрутизации *main*. Эти таблицы используются для выбора лучшего маршрута. Таблица *main* также используется для поиска следующего узла (nexthop lookup).

Routing Information Base



RIB (Routing Information Base) содержит полную информацию о маршрутах, включая статические маршруты и политики правил маршрутизации, сконфигурированные пользователем, информацию о маршрутах, полученную от протоколов маршрутизации, информацию о присоединенных сетях. RIB используется для фильтрации информации о маршрутах, подсчета лучшего маршрута для каждой точки назначения, создания и редактирования Forwarding Information Base и для распространения информации о маршрутах между различными протоколами маршрутизации.

By default forwarding decision is based only on the value of destination address. Each route has **dst-address** property, that specifies all destination addresses this route can be used for. If there are several routes that apply to a particular IP address, the most specific one (with largest netmask) is used. This operation (finding the most specific route that matches given address) is called *routing table lookup*.

If routing table contains several routes with the same **dst-address**, only one of them can be used to forward packets. This route is installed into FIB and marked as *active*.

When forwarding decision uses additional information, such as a source address of the packet, it is called *policy routing*. Policy routing is implemented as a list of policy routing rules, that select different routing table based on destination address, source address, source interface, and routing mark (can be changed by firewall mangle rules) of the packet.

All routes by default are kept in the *main* routing table. Routes can be assigned to specific routing table by setting their **routing-mark** property to the name of another routing table. Routing tables are referenced by their name, and are created automatically when they are referenced in the configuration.

Each routing table can have only one active route for each value of **dst-address** IP prefix.

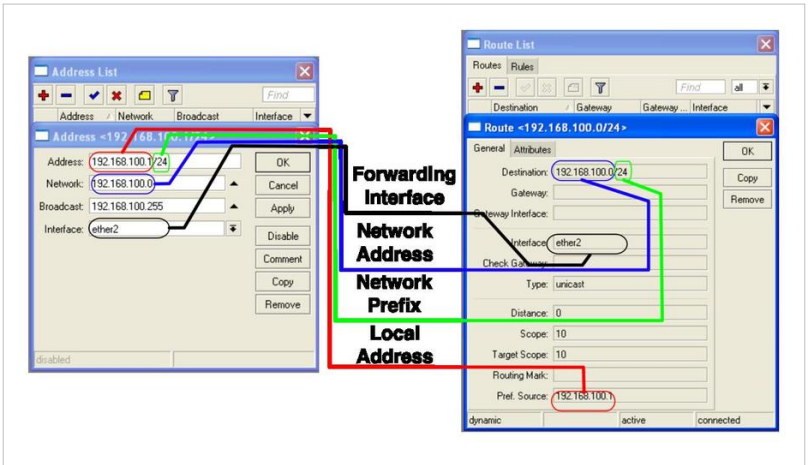
There are different groups of routes, based on their origin and properties.

Default route

Route with **dst-address** *0.0.0.0/0* applies to every destination address. Such route is called the *default route*. If routing table contains an active default route, then routing table lookup in this table will never fail.

Connected routes

Connected routes are created automatically for each IP network that has at least one enabled interface attached to it (as specifie in the `/ip address` configuration). RIB tracks status of connected routes, but does not modify them. For each connected route there is one *ip address* item such that:



- address part of **dst-address** of connected route is equal to **network** of *ip address* item.
- netmask part of **dst-address** of connected route is equal to netmask part of **address** of *ip address* item.
- **pref-src** of connected route is equal to address part of **address** of *ip address* item.
- **interface** of connected route is equal to **actual-interface** of *ip address* item (same as **interface**, except for bridge interface ports).

Multipath (ECMP) routes

Because results of the forwarding decision are cached, packets with the same **source address, destination address, source interface, routing mark** and **ToS** are sent to the same gateway. This means that one connection will use only one link in each direction, so ECMP routes can be used to implement per-connection load balancing. See interface bonding if you need to achieve per-packet load balancing.

To implement some setups, such as load balancing, it might be necessary to use more than one path to given destination. However, it is not possible to have more than one active route to destination in a single routing table.

ECMP (Equal cost multi-path) routes have multiple **gateway** nexthop values. All reachable nexthops are copied to FIB and used in forwarding packets.

OSPF protocol can create ECMP routes. Such routes can also be created manually.

Routes with interface as a gateway

Value of **gateway** can be specified as an interface name instead of the nexthop IP address. Such route has following special properties:

- Unlike connected routes, routes with interface nexthops are not used for nexthop lookup.
- It is possible to assign several interfaces as a value of **gateway**, and create ECMP route. It is not possible to have connected route with multiple **gateway** values.

Route selection

Each routing table can have one active route for each destination prefix. This route is installed into FIB. Active route is selected from all candidate routes with the same **dst-address** and **routing-mark**, that meet the criteria for becoming an active route. There can be multiple such routes from different routing protocols and from static configuration. Candidate route with the lowest **distance** becomes an active route. If there is more than one candidate route with the same **distance**, selection of active route is arbitrary (except for BGP routes).

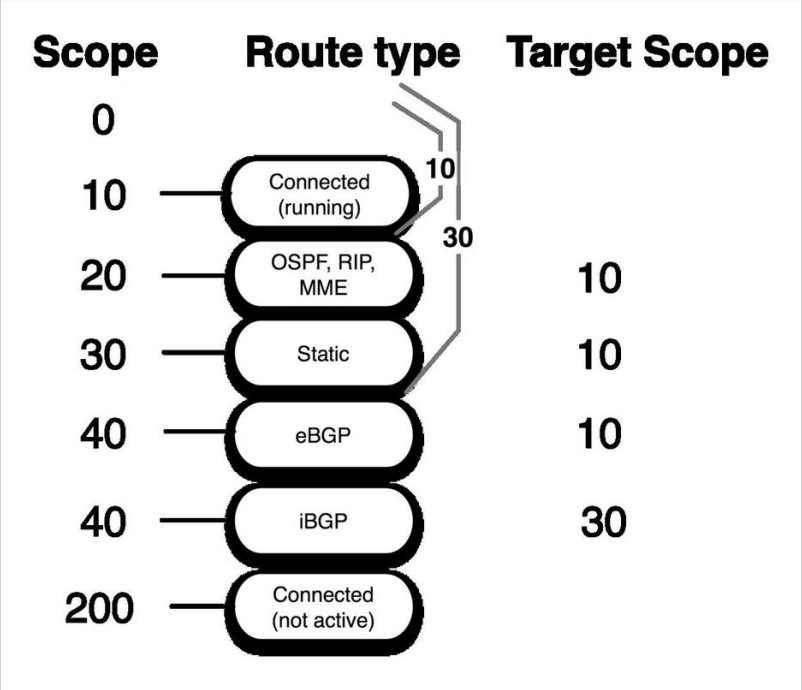
BGP has the most complicated selection process (described in separate article). Notice that this protocol-internal selection is done only after BGP routes are installed in the main routing table; this means there can be one candidate route from each BGP peer. Also note that BGP routes from different BGP instances are compared by their distance, just like other routes.

Criteria for selecting candidate routes

To participate in route selection process, route has to meet following criteria:

- route is not disabled.
- **distance** is not *255*. Routes that are rejected by route filter have **distance** value of *255*.
- **pref-src** is either not set or is a valid local address of the router.
- **routing-mark** is either not set or is referred by firewall or policy routing rules.
- If type of route is *unicast* and it is not a connected route, it must have at least one reachable nexthop.

Nexthop lookup



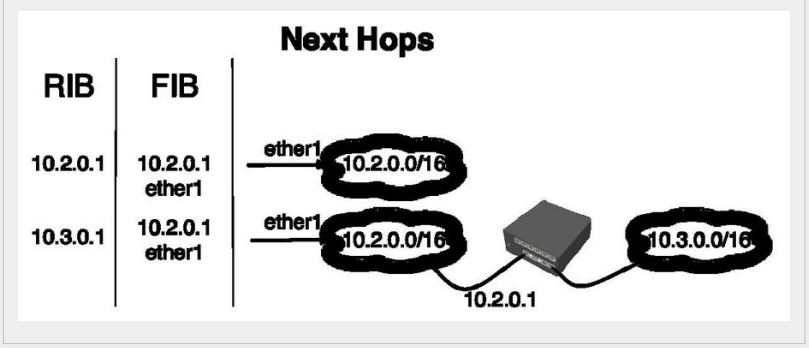
Nexthop lookup is a part of the route selection process.

Routes that are installed in the FIB need to have interface associated with each gateway address. Gateway address (nexthop) has to be directly reachable via this interface. Interface that should be used to send out packets to each gateway address is found by doing nexthop lookup.

Some routes (e.g. iBGP) may have gateway address that is several hops away from this router. To install such routes in the FIB, it is necessary to find the address of the directly reachable gateway (an immediate nexthop), that should be used to reach the gateway address of this route. Immediate nextop addresses are also found by doing nexthop lookup.

Nexthop lookup is done only in the *main* routing table, even for routes with different value of **routing-mark**. It is necessary to restrict set of routes that can be used to look up immediate nexthops. Nexthop values of RIP or OSPF routes, for example, are supposed to be directly reachable and should be looked up only using connected routes. This is achieved using **scope** and **target-scope** properties.

- Routes with interface name as the value of gateway are not used for nexthop lookup. If route has both interface nexthops and active IP address nexthops, then interface nexthops are ignored.
- Routes with **scope** greater than the maximum accepted value are not used for nexthop lookup. Each route specifies maximum accepted **scope** value for it's nexthops in the **target-scope** property. Default value of this property allows nexthop lookup only through connected routes, with the exception of iBGP routes that have larger default value and can lookup nexthop also through IGP and static routes.



Recursive nexthop lookup example

- nexthop 10.2.0.1 is resolved through a connected route, it's status is *reachable*.
- nexthop 10.3.0.1 is resolved recursively through a 10.3.0.0/16 route, it's status is *recursive*, and it uses 10.2.0.1 as the immediate nexthop value that is installed in the FIB.

Interface and immediate nexthop are selected based on the result of nexthop lookup:

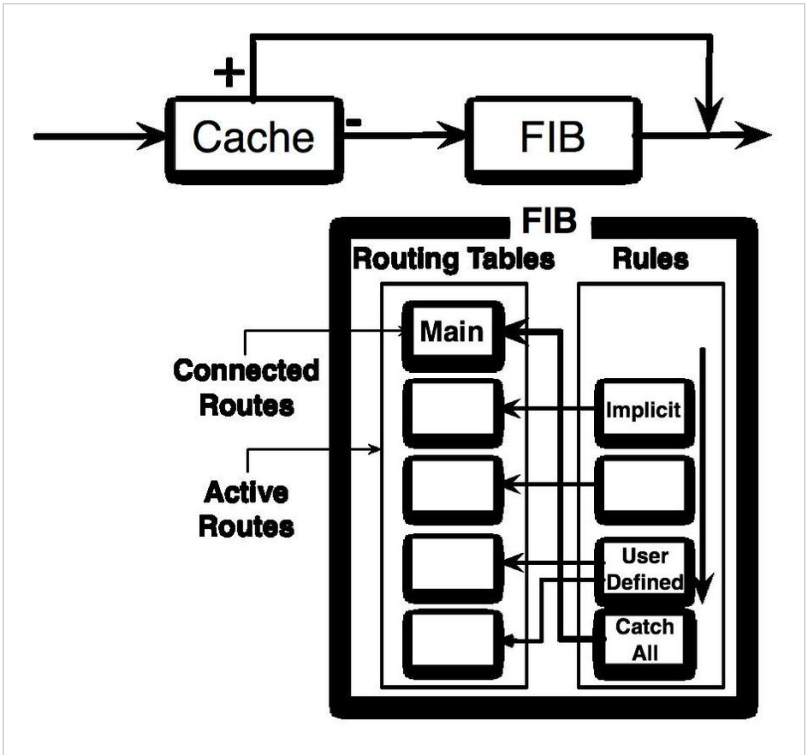
- If most specific active route that nexthop lookup finds is connected route, then interface of this connected route is used as the nexthop interface, and this gateway is marked as *reachable*. Since gateway is directly reachable through this interface (that's exactly what connected route means), the gateway address is used as the immediate nexthop address.
- If most specific active route that nexthop lookup finds has nexthop that is already resolved, immediate nexthop address and interface is copied from that nexthop and this gateway is marked as *recursive*.
- If most specific active route that nexthop lookup finds is ECMP route, then it uses first gateway of that route that is not *unreachable*.
- If nexthop lookup does not find any route, then this gateway is marked as *unreachable*.

Forwarding Information Base

FIB (Forwarding Information Base) contains copy of information that is necessary for packet forwarding:

- all active routes
- policy routing rules

By default (when no **routing-mark** values are used) all active routes are in the *main* table, and there is only one hidden implicit rule ("catch all" rule) that uses the *main* table for all destination lookups.



Routing table lookup

FIB uses following information from packet to determine it's destination:

- source address
- destination address
- source interface
- routing mark
- ToS (not used by RouterOS in policy routing rules, but it is a part of routing cache lookup key)

Possible routing decisions are:

- receive packet locally
- discard packet (either silently or by sending ICMP message to the sender of the packet)
- send packet to specific IP address on specific interface

Results of routing decision are remembered in the routing cache. This is done to improve forwarding performance. When another packet with the same **source address, destination address, source interface, routing mark** and **ToS** is routed, cached results are used. This also allows to implement per-connection load balancing using ECMP routes, because values used to lookup entry in the routing cache are the same for all packets that belong to the same connection and go in the same direction.

If there is no routing cache entry for this packet, it is created by running routing decision:

- check that packet has to be locally delivered (destination address is address of the router)
- process implicit policy routing rules
- process policy routing rules added by user

- process implicit catch-all rule that looks up destination in the *main* routing table
- return result is "network unreachable"

Result of routing decision can be:

- IP address of nexthop + interface
- point-to-point interface
- local delivery
- discard
- ICMP prohibited
- ICMP host unreachable
- ICMP network unreachable

Rules that do not match current packet are ignored. If rule has action *drop* or *unreachable*, then it is returned as a result of the routing decision process. If action is *lookup* then destination address of the packet is looked up in routing table that is specified in the rule. If lookup fails (there is no route that matches destination address of packet), then FIB proceeds to the next rule. Otherwise:

- if type of the route is *blackhole*, *prohibit* or *unreachable*, then return this action as the routing decision result;
- if this is a connected route, or route with an interface as the **gateway** value, then return this interface and the destination address of the packet as the routing decision result;
- if this route has IP address as the value of **gateway**, then return this address and associated interface as the routing decision result;
- if this route has multiple values of nexthop, then pick one of them in round robin fashion.

Result of this routing decision is stored in new routing cache entry.

Properties

Route flags

Property(Flag)	Description
`disabled` (*X*)	Configuration item is disabled. It does not have any effect on other routes and is not used by forwarding or routing protocols in any way.
`active` (*A*)	Route is used for packet forwarding. See route selection.
`dynamic` (*D*)	Configuration item created by software, not by management interface. It is not exported, and cannot be directly modified.
`connect` (*C*)	connected route.
`static` (*S*)	static route.
`rip` (*r*)	RIP route.
`bgp` (*b*)	BGP route.
`ospf` (*o*)	OSPF route.
`mme` (*m*)	MME route.
`blackhole` (*B*)	Silently discard packet forwarded by this route.
`unreachable` (*U*)	Discard packet forwarded by this route. Notify sender with ICMP host unreachable (type 3 code 1) message.
`prohibit` (*P*)	Discard packet forwarded by this route. Notify sender with ICMP communication administratively prohibited (type 3 code 13) message.

General properties

Property	Description
check-gateway (*arp* | *ping*; Default: **""**)	Periodically (every 10 seconds) check gateway by sending either ICMP echo request (*ping*) or ARP request (*arp*). If no response from gateway is received for 10 seconds, request times out. After two timeouts gateway is considered unreachable. After receiving reply from gateway it is considered reachable and timeout counter is reset.
comment (*string*; Default: **""**)	Description of particular route
distance (*integer[1..255]*; Default: **"1"**)	Value used in route selection. Routes with smaller distance value are given preference. If value of this property is not set, then the default depends on route protocol: • connected routes: 0 • static routes: 1 • eBGP: 20 • OSPF: 110 • RIP: 120 • MME: 130 • iBGP: 200
dst-address (*IP prefix*; Default: **0.0.0.0/0**)	IP prefix of route, specifies destination addresses that this route can be used for. Netmask part of this property specifies how many of the most significant bits in packet destination address must match this value. If there are several active routes that match destination address of packet, then the most specific one (with largest netmask value) is used.
gateway (*IP IP%interface* | *IP@table[, IP* | *string, [..*; Default: **""**)	Array of IP addresses or interface names. Specifies which host or interface packets should be sent to. Connected routes and routes with *blackhole*, *unreachable* or *prohibit* type do not have this property. Usually value of this property is a single IP address of a gateway that can be directly reached through one of router's interfaces (but see nexthop lookup). ECMP routes have more than one gateway value. Value can be repeated several times.
pref-src (*IP*; Default: **""**)	Which of the local IP addresses to use for locally originated packets that are sent via this route. Value of this property has no effect on forwarded packets. If value of this property is set to IP address that is not local address of this router then the route will be inactive. If **pref-src** value is not set, then for locally originated packets that are sent using this route router will choose one of local addresses attached to the output interface that match destination prefix of the route (an example).
route-tag (*integer*; Default: **""**)	Value of route tag attribute for RIP or OSPF. For RIP only values 0..4294967295 are valid.
routing-mark (*string*; Default: **""**)	Name of routing table that contains this route. Not set by default which is the same as *main*. Packets that are marked by firewall with this value of **routing-mark** will be routed using routes from this table, unless overridden by policy routing rules. Not more than 250 routing marks are possible per router.
scope (*integer[0..255]*; Default: **"30"**)	Used in nexthop resolution. Route can resolve nexthop only through routes that have **scope** less than or equal to the **target-scope** of this route. Default value depends on route protocol: • connected routes: 10 (if interface is running) • OSPF, RIP, MME routes: 20 • static routes: 30 • BGP routes: 40 • connected routes: 200 (if interface is not running)
target-scope (*integer[0..255]*; Default: **"10"**)	Used in nexthop resolution. This is the maximum value of **scope** for a route through which a nexthop of this route can be resolved. See nexthop lookup. For iBGP value is set to 30 by default.
type (*unicast* | *blackhole* | *prohibit* | *unreachabl*; Default: **unicast**)	Routes that do not specify nexthop for packets, but instead perform some other action on packets have type different from the usual *unicast*. *blackhole* route silently discards packets, while *unreachable* and *prohibit* routes send ICMP Destination Unreachable message (code 1 and 13 respectively) to the source address of the packet.
vrf-interface (*string*; Default: **"10"**)	VRF interface name

Other Read-only properties

Property	Description
gateway-status (*array*)	Array of gateways, gateway states and which interface is used for forwarding. Syntax "IP state interface", for example "10.5.101.1 reachable bypass-bridge". State can be *unreachable*, *reachable* or *recursive*. See nexthop lookup for details.
ospf-metric (*integer*)	Used OSPF metric for particular route
ospf-type (*string*)	

BGP Route Properties

These properties contain information that is used by BGP routing protocol. However, values of these properties can be set for any type of route, including static and connected. It can be done either manually (for static routes) or using route filters.

Property	Description
bgp-as-path (*string*; Default: "")	Value of BGP AS_PATH attribute. Comma separated list of AS numbers with confederation AS numbers enclosed in *()* and AS_SETs enclosed in *{}*. Used to check for AS loops and in BGP route selection algorithm: routes with shorter AS_PATH are preferred (but read how AS_PATH length is calculated).
bgp-atomic-aggregate (*yes* | *no*; Default:)	Value of BGP ATOMIC_AGGREGATE attribute.
bgp-communities (*array of (integer:integer* | *internet* | *no-advertise* | *no-export* |*local-as*; Default:)	Value of BGP communities list. This attribute can be used to group or filter routes. Named values have special meanings: • *internet* - advertise this route to the Internet community (i.e. all routers) • *no-advertise* - do not advertise this route to any peers • *no-export* - do not advertise this route to EBGP peers • *local-as* - same as *no-export*, except that route is also advertised to EBGP peers inside local confederation
bgp-local-pref (*integer*; Default:)	Value of BGP LOCAL_PREF attribute. Used in BGP route selection algorithm: routes with greater LOCAL_PREF value are preferred. If value is not set then it is interpreted as 100.
bgp-med (*integer*; Default:)	Value of BGP MULTI_EXIT_DISC BGP attribute. Used in BGP route selection algorithm: routes with lower MULTI_EXIT_DISC value are preferred.. If value is not set then it is interpreted as 0.
bgp-origin (*igp* | *egp* | *incomplete*; Default:)	Value of BGP ORIGIN attribute. Used in BGP route selection algorithm: *igp* routes are preferred over *egp* and *egp* over *incomplete*.
bgp-prepend (*integer [0..16]*; Default:)	How many times to prepend router's own AS number to AS_PATH attribute when announcing route via BGP. Affects only routes sent to eBGP peers (for iBGP value 0 is always used).

Read-only

Property	Description
bgp-ext-communities (*string*)	Value of BGP extended communities attribute
bgp-weight (*integer*)	Additional value used by BGP best path selection algorithm. Routes with higher weight are preferred. It can be set by incoming routing filters and is useful only for BGP routes. If value is not set then it is interpreted as 0.
received-from (*string*)	Name of the BGP peer from which route is received.

Manual:RADIUS Client

Applies to RouterOS: 2.9, v3, v4, v5

Summary

Sub-menu: `/radius`

Standards: `RADIUS RFC 2865`

RADIUS, short for Remote Authentication Dial-In User Service, is a remote server that provides authentication and accounting facilities to various network apliances. RADIUS authentication and accounting gives the ISP or network administrator ability to manage PPP user access and accounting from one server throughout a large network. The MikroTik RouterOS has a RADIUS client which can authenticate for HotSpot, PPP, PPPoE, PPTP, L2TP and ISDN connections. The attributes received from RADIUS server override the ones set in the default profile, but if some parameters are not received they are taken from the respective default profile.

The RADIUS server database is consulted only if no matching user acces record is found in router's local database.

Traffic is accounted locally with MikroTik Traffic Flow and Cisco IP pairs and snapshot image can be gathered using Syslog utilities. If RADIUS accounting is enabled, accounting information is also sent to the RADIUS server default for that service.

Radius Client

This sub-menu allows to add/remove radius clients.

Note: The order of added items in this list is significant.

Properties

Property	Description
accounting-backup (*yes* | *no*; Default: **no**)	Whether configuration is for backup RADIUS server
accounting-port (*integer [1..65535]*; Default: **1813**)	RADIUS server port used for accounting
address (*IPv4/IPv6 address*; Default: **0.0.0.0**)	IPv4 or IPv6 address of RADIUS server.
authentication-port (*integer [1..65535]*; Default: **1812**)	RADIUS server port used for authentication.
called-id (*string*; Default:)	Value depends on Point-to-Point protocol: PPPoE - service name, PPTP - server's IP address, L2TP - server's IP address.
comment (*string*; Default:)	
disabled (*yes* | *no*; Default: **no**)	
domain (*string*; Default:)	Microsoft Windows domain of client passed to RADIUS servers that require domain validation.
realm (*string*; Default:)	Explicitly stated realm (user domain), so the users do not have to provide proper ISP domain name in user name.
secret (*string*; Default:)	Shared secret used to access the RADIUS server.
service (*ppp|login|hotspot|wireless|dhcp*; Default:)	Router services that will use this RADIUS server: • `hotspot` - HotSpot authentication service • `login` - router's local user authentication • `ppp` - Point-to-Point clients authentication • `wireless` - wireless client authentication (client's MAC address is sent as User-Name) • `dhcp` - DHCP protocol client authentication (client's MAC address is sent as User-Name)
src-address (*ipv4/ipv6 address*; Default: **0.0.0.0**)	Source IP/IPv6 address of the packets sent to RADIUS server
timeout (*time*; Default: **100ms**)	Timeout after which the request should be resend

Note: Microsoft Windows clients send their usernames in form domain\username

Note: When RADIUS server is authenticating user with CHAP, MS-CHAPv1, MS-CHAPv2, it is not using shared secret, secret is used only in authentication reply, and router is verifying it. So if you have wrong shared secret, RADIUS server will accept request, but router won't accept reply. You can see that with /radius monitor command, "bad-replies" number should increase whenever somebody tries to connect.

Example

To set a RADIUS server for HotSpot and PPP services that has 10.0.0.3 IP address and ex shared secret, you need to do the following:

```
[admin@MikroTik] radius> add service=hotspot,ppp address=10.0.0.3 secret=ex
[admin@MikroTik] radius> print
Flags: X - disabled
  #    SERVICE         CALLED-ID     DOMAIN        ADDRESS         SECRET
  0    ppp,hotspot                                 10.0.0.3        ex
[admin@MikroTik] radius>
AAA for the respective services should be enabled too:
[admin@MikroTik] radius> /ppp aaa set use-radius=yes
[admin@MikroTik] radius> /ip hotspot profile set default use-radius=yes
To view some statistics for a client:
[admin@MikroTik] radius> monitor 0
             pending: 0
            requests: 10
             accepts: 4
             rejects: 1
             resends: 15
            timeouts: 5
         bad-replies: 0
    last-request-rtt: 0s
[admin@MikroTik] radius>
```

Connection Terminating from RADIUS

Sub-menu: `/radius incoming`

This facility supports unsolicited messages sent from RADIUS server. Unsolicited messages extend RADIUS protocol commands, that allow to terminate a session which has already been connected from RADIUS server. For this purpose DM (Disconnect-Messages) are used. Disconnect messages cause a user session to be terminated immediately.

Note: RouterOS doesn't support POD (Packet of Disconnect) the other RADIUS access request packet that performs a similar function as Disconnect Messages

Properties

Property	Description
accept (*yes* | *no*; Default: **no**)	Whether to accept the unsolicited messages
port (*integer*; Default: **1700**)	The port number to listen for the requests on

Supported RADIUS Attributes

Here you can download the RADIUS reference dictionary, which incorporates all the needed RADIUS attributes. This dictionary is the minimal dictionary, which is enough to support all features of MikroTik RouterOS. It is designed for FreeRADIUS [1], but may also be used with many other UNIX RADIUS servers (eg. XTRadius [2]).

Note: it may conflict with the default configuration files of RADIUS server, which have references to the Attributes, absent in this dictionary. Please correct the configuration files, not the dictionary, as no other Attributes are supported by MikroTik RouterOS.

There is also the RADIUS MikroTik specific dictionary that can be included in an existing dictionary to support MikroTik vendor-specific Attributes.

Definitions

- **PPPs** - PPP, PPTP, PPPoE and ISDN
- **default configuration** - settings in default profile (for PPPs) or HotSpot server settings (for HotSpot)

Access-Request

- **Service-Type** - always is "Framed" (only for PPPs)
- **Framed-Protocol** - always is "PPP" (only for PPPs)
- **NAS-Identifier** - router identity
- **NAS-IP-Address** - IP address of the router itself
- **NAS-Port** - unique session ID
- **Acct-Session-Id** - unique session ID
- **NAS-Port-Type** - async PPP - "Async"; PPTP and L2TP - "Virtual"; PPPoE - "Ethernet"; ISDN - "ISDN Sync"; HotSpot - "Ethernet | Cable | Wireless-802.11" (according to the value of nas-port-type parameter in /ip hotspot p
- **Calling-Station-Id** - PPPoE and HotSpot- client MAC address in capital letters; PPTP and L2TP - client public IP address; ISDN - client MSN
- **Called-Station-Id** - PPPoE - service name; PPTP and L2TP - server IP address; ISDN - interface MSN; HotSpot - name of the HotSpot server
- **NAS-Port-Id** - async PPP - serial port name; PPPoE - ethernet interface name on which server is running; HotSpot - name of the physical HotSpot interface (if bridged, the bridge port name is showed here); not present for ISDN, PPTP and L2TP
- **Framed-IP-Address** - IP address of HotSpot client after Universal Client translation
- **Mikrotik-Host-IP** - IP address of HotSpot client before Universal Client translation (the original IP address of the client)
- **User-Name** - client login name
- **MS-CHAP-Domain** - User domain, if present
- **Mikrotik-Realm** - If it is set in /radius menu, it is included in every RADIUS request as Mikrotik-Realm attribute. If it is not set, the same value is sent as in MS-CHAP-Domain attribute (if MS-CHAP-Domain is missing, Realm is not included neither)

- **WISPr-Location-ID** - text string specified in radius-location-id property of the HotSpot server
- **WISPr-Location-Name** - text string specified in radius-location-name property of the HotSpot server
- **WISPr-Logoff-URL** - full link to the login page (for example, http://10.48.0.1/lv/logout)

Depending on authentication methods (NOTE: HotSpot uses CHAP by default and may use also PAP if unencrypted passwords are enabled, it can not use MSCHAP):

- **User-Password** - encrypted password (used with PAP authentication)
- **CHAP-Password, CHAP-Challenge** - encrypted password and challenge (used with CHAP authentication)
- **MS-CHAP-Response, MS-CHAP-Challenge** - encrypted password and challenge (used with MS-CHAPv1 authentication)
- **MS-CHAP2-Response, MS-CHAP-Challenge** - encrypted password and challenge (used with MS-CHAPv2 authentication)

Access-Accept

- **Framed-IP-Address** - IP address given to client. If address belongs to 127.0.0.0/8 or 224.0.0.0/3 networks, IP pool is used from the default profile to allocate client IP address. If Framed-IP-Address is specified, Framed-Pool is ignored
- **Framed-IP-Netmask** - client netmask. PPPs - if specified, a route will be created to the network Framed-IP-Address belongs to via the Framed-IP-Address gateway; HotSpot - ignored by HotSpot
- **Framed-Pool** - IP pool name (on the router) from which to get IP address for the client. If Framed-IP-Address is specified, this attribute is ignored
- **Framed-IPv6-Prefix** - Ipv6 prefix assigned for the client. Added in v5.8
- **Mikrotik-Delegated-IPv6-Pool** - IPv6 pool used for Prefix Delegation. Added in v5.9

NOTE: if Framed-IP-Address or Framed-Pool is specified it overrides remote-address in default configuration

- **Idle-Timeout** - overrides idle-timeout in the default configuration
- **Session-Timeout** - overrides session-timeout in the default configuration
- **Port-Limit** - maximal mumber of simultaneous connections using the same username (overrides te shared-users property of the HotSpot user profile)
- **Class** - cookie, will be included in Accounting-Request unchanged
- **Framed-Route** - routes to add on the server. Format is specified in RFC 2865 (Ch. 5.22), can be specified as many times as needed
- **Filter-Id** - firewall filter chain name. It is used to make a dynamic firewall rule. Firewall chain name can have suffix .in or .out, that will install rule only for incoming or outgoing traffic. Multiple Filter-id can be provided, but only last ones for incoming and outgoing is used. For PPPs - filter rules in ppp chain that will jump to the specified chain, if a packet has come to/from the client (that means that you should first create a ppp chain and make jump rules that would put actual traffic to this chain). The same applies for HotSpot, but the rules will be created in hotspot chain
- **Mikrotik-Mark-Id** - firewall mangle chain name (HotSpot only). The MikroTik RADIUS client upon receiving this attribute creates a dynamic firewall mangle rule with action=jump chain=hotspot and jump-target equal to the atribute value. Mangle chain name can have suffixes .in or .out, that will install rule only for incoming or outgoing traffic. Multiple Mark-id attributes can be provided, but only last ones for incoming and outgoing is used.
- **Acct-Interim-Interval** - interim-update for RADIUS client. PPP - if 0 uses the one specified in RADIUS client; HotSpot - only respected if radius-interim-update=received in HotSpot server profile
- **MS-MPPE-Encryption-Policy** - require-encryption property (PPPs only)
- **MS-MPPE-Encryption-Types** - use-encryption property, non-zero value means to use encryption (PPPs only)

- **Ascend-Data-Rate** - tx/rx data rate limitation if multiple attributes are provided, first limits tx data rate, second - rx data rate. If used together with Ascend-Xmit-Rate, specifies rx rate. 0 if unlimited. Ignored if Rate-Limit attribute is present
- **Ascend-Xmit-Rate** - tx data rate limitation. It may be used to specify tx limit only instead of sending two sequental Ascend-Data-Rate attributes (in that case Ascend-Data-Rate will specify the receive rate). 0 if unlimited. Ignored if Rate-Limit attribute is present
- **MS-CHAP2-Success** - auth. response if MS-CHAPv2 was used (for PPPs only)
- **MS-MPPE-Send-Key, MS-MPPE-Recv-Key** - encryption keys for encrypted PPPs provided by RADIUS server only is MS-CHAPv2 was used as authentication (for PPPs only)
- **Ascend-Client-Gateway** - client gateway for DHCP-pool HotSpot login method (HotSpot only)
- **Mikrotik-Recv-Limit** - total receive limit in bytes for the client
- **Mikrotik-Recv-Limit-Gigawords** - 4G (2^32) bytes of total receive limit (bits 32..63, when bits 0..31 are delivered in Mikrotik-Recv-Limit)
- **Mikrotik-Xmit-Limit** - total transmit limit in bytes for the client
- **Mikrotik-Xmit-Limit-Gigawords** - 4G (2^32) bytes of total transmit limit (bits 32..63, when bits 0..31 are delivered in Mikrotik-Recv-Limit)
- **Mikrotik-Wireless-Forward** - not forward the client's frames back to the wireless infrastructure if this attribute is set to "0" (Wireless only)
- **Mikrotik-Wireless-Skip-Dot1x** - disable 802.1x authentication for the particulat wireless client if set to non-zero value (Wireless only)
- **Mikrotik-Wireless-Enc-Algo** - WEP encryption algorithm: 0 - no encryption, 1 - 40-bit WEP, 2 - 104-bit WEP (Wireless only)
- **Mikrotik-Wireless-Enc-Key** - WEP encruption key for the client (Wireless only)
- **Mikrotik-Rate-Limit** - Datarate limitation for clients. Format is: rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time] [priority] [rx-rate-min[/tx-rate-min]]]] from the point of view of the router (so "rx" is client upload, and "tx" is client download). All rates should be numbers with optional 'k' (1,000s) or 'M' (1,000,000s). If tx-rate is not specified, rx-rate is as tx-rate too. Same goes for tx-burst-rate and tx-burst-threshold and tx-burst-time. If both rx-burst-threshold and tx-burst-threshold are not specified (but burst-rate is specified), rx-rate and tx-rate is used as burst thresholds. If both rx-burst-time and tx-burst-time are not specified, 1s is used as default. Priority takes values 1..8, where 1 implies the highest priority, but 8 - the lowest. If rx-rate-min and tx-rate-min are not specified rx-rate and tx-rate values are used. The rx-rate-min and tx-rate-min values can not exceed rx-rate and tx-rate values.
- **Mikrotik-Group** - Router local user group name (defines in /user group) for local users. HotSpot default profile for HotSpot users.
- **Mikrotik-Advertise-URL** - URL of the page with advertisements that should be displayed to clients. If this attribute is specified, advertisements are enabled automatically, including transparent proxy, even if they were explicitly disabled in the corresponding user profile. Multiple attribute instances may be send by RADIUS server to specify additional URLs which are choosen in round robin fashion.
- **Mikrotik-Advertise-Interval** - Time interval between two adjacent advertisements. Multiple attribute instances may be send by RADIUS server to specify additional intervals. All interval values are threated as a list and are taken one-by-one for each successful advertisement. If end of list is reached, the last value is continued to be used.
- **WISPr-Redirection-URL** - URL, which the clients will be redirected to after successfull login
- **WISPr-Bandwidth-Min-Up** - minimal datarate (CIR) provided for the client upload
- **WISPr-Bandwidth-Min-Down** - minimal datarate (CIR) provided for the client download
- **WISPr-Bandwidth-Max-Up** - maxmal datarate (MIR) provided for the client upload

- **WISPr-Bandwidth-Max-Down** - maxmal datarate (MIR) provided for the client download
- **WISPr-Session-Terminate-Time** - time, when the user should be disconnected; in "YYYY-MM-DDThh:mm:ssTZD" form, where Y - year; M - month; D - day; T - separator symbol (must be written between date and time); h - hour (in 24 hour format); m - minute; s - second; TZD - time zone in one of these forms: "+hh:mm", "+hhmm", "-hh:mm", "-hhmm"

Note: the received attributes override the default ones (set in the default profile), but if an attribute is not received from RADIUS server, the default one is to be used.

Rate-Limit takes precedence over all other ways to specify data rate for the client. Ascend data rate attributes are considered second; and WISPr attributes takes the last precedence.

Here are some Rate-Limit examples:

- **128k** - rx-rate=128000, tx-rate=128000 (no bursts)
- **64k/128M** - rx-rate=64000, tx-rate=128000000
- **64k 256k** - rx/tx-rate=64000, rx/tx-burst-rate=256000, rx/tx-burst-threshold=64000, rx/tx-burst-time=1s
- **64k/64k 256k/256k 128k/128k 10/10** - rx/tx-rate=64000, rx/tx-burst-rate=256000, rx/tx-burst-threshold=128000, rx/tx-burst-time=10s

Accounting-Request

The accounting request carries the same attributes as Access Request, plus these ones:

- **Acct-Status-Type** - Start, Stop, or Interim-Update
- **Acct-Authentic** - either authenticated by the RADIUS or Local authority (PPPs only)
- **Class** - RADIUS server cookie, as received in Access-Accept
- **Acct-Delay-Time** - how long does the router try to send this Accounting-Request packet

Stop and Interim-Update Accounting-Request

Additionally to the accounting start request, the following messages will contain the following attributes:

- **Acct-Session-Time** - connection uptime in seconds
- **Acct-Input-Octets** - bytes received from the client
- **Acct-Input-Gigawords** - 4G (2^32) bytes received from the client (bits 32..63, when bits 0..31 are delivered in Acct-Input-Octets)
- **Acct-Input-Packets** - nubmer of packets received from the client
- **Acct-Output-Octets** - bytes sent to the client
- **Acct-Output-Gigawords** - 4G (2^32) bytes sent to the client (bits 32..63, when bits 0..31 are delivered in Acct-Output-Octets)
- **Acct-Output-Packets** - number of packets sent to the client

Stop Accounting-Request

These packets will, additionally to the Interim Update packets, have:

- **`Acct-Terminate-Cause`** - session termination cause (see RFC 2866 ch. 5.10)

Change of Authorization

RADIUS disconnect and Change of Authorization (according to RFC3576) are supported as well. These attributes may be changed by a CoA request from the RADIUS server:

- Mikrotik-Group
- Mikrotik-Recv-Limit
- Mikrotik-Xmit-Limit
- Mikrotik-Rate-Limit
- Ascend-Data-Rate (only if Mikrotik-Rate-Limit is not present)
- Ascend-XMit-Rate (only if Mikrotik-Rate-Limit is not present)
- Mikrotik-Mark-Id
- Filter-Id
- Mikrotik-Advertise-Url
- Mikrotik-Advertise-Interval
- Session-Timeout
- Idle-Timeout
- Port-Limit

Note that it is not possible to change IP address, pool or routes that way - for such changes a user must be disconnected first.

MikroTik Specific RADIUS Attribute Numeric Values

Click here to get plain text attribute list of MikroTik specific attributes (FreeRadius comaptible) .

Name	VendorID	Value	RFC
MIKROTIK_RECV_LIMIT	14988	1	
MIKROTIK_XMIT_LIMIT	14988	2	
MIKROTIK_GROUP	14988	3	
MIKROTIK_WIRELESS_FORWARD	14988	4	
MIKROTIK_WIRELESS_SKIPDOT1X	14988	5	
MIKROTIK_WIRELESS_ENCALGO	14988	6	
MIKROTIK_WIRELESS_ENCKEY	14988	7	
MIKROTIK_RATE_LIMIT	14988	8	
MIKROTIK_REALM	14988	9	
MIKROTIK_HOST_IP	14988	10	
MIKROTIK_MARK_ID	14988	11	
MIKROTIK_ADVERTISE_URL	14988	12	
MIKROTIK_ADVERTISE_INTERVAL	14988	13	
MIKROTIK_RECV_LIMIT_GIGAWORDS	14988	14	
MIKROTIK_XMIT_LIMIT_GIGAWORDS	14988	15	
MIKROTIK_WIRELESS_PSK	14988	16	

MIKROTIK_TOTAL_LIMIT	14988	17
MIKROTIK_TOTAL_LIMIT_GIGAWORDS	14988	18
MIKROTIK_ADDRESS_LIST	14988	19
MIKROTIK_WIRELESS_MPKEY	14988	20
MIKROTIK_WIRELESS_COMMENT	14988	21
MIKROTIK_DELEGATED_IPV6_POOL	14988	22

All Supported Attribute Numeric Values

Note: FreeRadius already has these attributes predefined. If you are using other radius server then use table below to create dictionary file

Name	VendorID	Value	RFC
Acct-Authentic		45	RFC 2866
Acct-Delay-Time		41	RFC 2866
Acct-Input-Gigawords		52	RFC 2869
Acct-Input-Octets		42	RFC 2866
Acct-Input-Packets		47	RFC 2866
Acct-Interim-Interval		85	RFC 2869
Acct-Output-Gigawords		53	RFC 2869
Acct-Output-Octets		43	RFC 2866
Acct-Output-Packets		48	RFC 2866
Acct-Session-Id		44	RFC 2866
Acct-Session-Time		46	RFC 2866
Acct-Status-Type		40	RFC 2866
Acct-Terminate-Cause		49	RFC 2866
Ascend-Client-Gateway	529	132	
Ascend-Data-Rate	529	197	
Ascend-Xmit-Rate	529	255	
Called-Station-Id		30	RFC 2865
Calling-Station-Id		31	RFC 2865
CHAP-Challenge		60	RFC 2866
CHAP-Password		3	RFC 2865
Class		25	RFC 2865
Filter-Id		11	RFC 2865
Framed-IP-Address		8	RFC 2865
Framed-IP-Netmask		9	RFC 2865
Framed-IPv6-Prefix		97	RFC 3162

Framed-Pool		88	RFC 2869
Framed-Protocol		7	RFC 2865
Framed-Route		22	RFC 2865
Idle-Timeout		28	RFC 2865
MS-CHAP-Challenge	311	11	RFC 2548
MS-CHAP-Domain	311	10	RFC 2548
MS-CHAP-Response	311	1	RFC 2548
MS-CHAP2-Response	311	25	RFC 2548
MS-CHAP2-Success	311	26	RFC 2548
MS-MPPE-Encryption-Policy	311	7	RFC 2548
MS-MPPE-Encryption-Types	311	8	RFC 2548
MS-MPPE-Recv-Key	311	17	RFC 2548
MS-MPPE-Send-Key	311	16	RFC 2548
NAS-Identifier		32	RFC 2865
NAS-Port		5	RFC 2865
NAS-IP-Address		4	RFC 2865
NAS-Port-Id		87	RFC 2869
NAS-Port-Type		61	RFC 2865
Port-Limit		62	RFC 2865
Redback-Agent-Remote-Id	2352	96	
Redback-Agent-Circuit-Id	2352	97	
Service-Type		6	RFC 2865
Session-Timeout		27	RFC 2865
User-Name		1	RFC 2865
User-Password		2	RFC 2865
WISPr-Bandwidth-Max-Down	14122	8	wi-fi.org
WISPr-Bandwidth-Max-Up	14122	7	wi-fi.org
WISPr-Bandwidth-Min-Down	14122	6	wi-fi.org
WISPr-Bandwidth-Min-Up	14122	5	wi-fi.org
WISPr-Location-Id	14122	1	wi-fi.org
WISPr-Location-Name	14122	2	wi-fi.org
WISPr-Logoff-URL	14122	3	wi-fi.org
WISPr-Redirection-URL	14122	4	wi-fi.org
WISPr-Session-Terminate-Time	14122	9	wi-fi.org

Troubleshooting

My radius server accepts authentication request from the client with "Auth: Login OK:...", but the user cannot log on. The bad replies counter is incrementing under radius monitor.

> This situation can occur, if the radius client and server have high delay link between them. Try to increase the radius client's timeout to 600ms or more instead of the default 300ms! Also, double check, if the secrets match on client and server!

[Top | Back to Content **]**

References

[1] http://freeradius.org
[2] http://xtradius.sourceforge.net/

Manual:Routing

List of reference sub-pages **Routing protocol case studies** **List of examples**

<splist showparent=yes />

Manual:Routing/RIP

Applies to RouterOS: 2.9, v3, v4 +

Summary

MikroTik RouterOS implements RIP Version 1 (RFC 1058) and Version 2 (RFC 2453). RIP enables routers in an autonomous system to exchange routing information. It always uses the best path (the path with the fewest number of hops (i.e. routers)) available.

General

Sub-menu: `/routing rip`

Property	Description
distribute-default (*always* | *if-installed* | *never*; Default: **never**)	specifies how to distribute default route.
redistribute-static (*yes* | *no*; Default: **no**)	if enabled, redistribute static routes to neighbor routers
redistribute-connected (*yes* | *no*; Default: **no**)	if enabled, redistribute connected routes to neighbor routers
redistribute-ospf (*yes* | *no*; Default: **no**)	if enabled, redistribute OSPF routes to neighbor routers
redistribute-bgp (*yes* | *no*; Default: **no**)	if enabled, redistribute BGP routes to neighbor routers
metric-default (*integer*; Default: **1**)	specifies metric for default route
metric-static (*integer*; Default: **1**)	specifies metric for static routes
metric-connected (*integer*; Default: **1**)	specifies metric for connected routes
metric-ospf (*integer*; Default: **1**)	specifies metric (the number of hops) for the routes learned via OSPF protocol
metric-bgp (*integer*; Default: **1**)	specifies metric (the number of hops) for the routes learned via BGP protocol
update-timer (*time*; Default: **30s**)	specifies frequency of RIP updates
timeout-timer (*time*; Default: **3m**)	specifies time interval after which the route is considered invalid
garbage-timer (*time*; Default: **2m**)	specifies time interval after which the invalid route will be dropped from neighbor router table

Note: The maximum metric of RIP route is 15. Metric higher than 15 is considered 'infinity' and routes with such metric are considered unreachable. Thus RIP cannot be used on networks with more than 15 hops between any two routers, and using redistribute metrics larger that 1 further reduces this maximum hop count.

Interface

Sub-menu: `/routing rip interface`

Property	Description
interface (*string* | *all*; Default: **all**)	interface on which RIP runs. If set to 'all' settings will be applied to all interfaces
send (*v1* | *v1-2* | *v2*; Default: **v2**)	specifies RIP protocol update versions to distribute
receive (*v1* | *v1-2* | *v2*; Default: **v1-2**)	specifies RIP protocol update versions the router will be able to receive
passive (*yes* | *no*; Default: **no**)	if enabled, do not send routing packets via this interface, only receive
authentication (*none* | *simple* | *md5*; Default: **none**)	specifies authentication method to use on RIP messages
authentication-key (*string*; Default: "")	specifies authentication key
key-chain (*string*; Default: "")	chain name for MD5 authentication passwords
in-prefix-list (*string*; Default: "")	name of the filtering prefix list for received routes
out-prefix-list (*string*; Default: "")	name of the filtering prefix list for advertised routes

Keys

Sub-menu: `/routing rip keys`

MD5 authentication key chains.

Property	Description
chain (*string*; Default: "")	chain name to place this key in. If a chain with the specified name does not exist it will be automatically created
key (*string*; Default: "")	authentication key. Maximal length 16 characters
key-id (*integer:0..255*; Default:)	key identifier. This number is included in MD5 authenticated RIP messages, and determines witch key to use to check authentication for a specific message.
from-date (*date*; Default: **tomorrow system date**)	key is valid from this date
from-time (*time*; Default: **00:00:00**)	key is valid until this time in the specified date

Network

Sub-menu: `/routing rip network`

To start the RIP protocol, you have to define the networks on which RIP will run.

Property	Description
network (*IP prefix*; Default:)	the network prefix. RIP will be enabled on all interfaces that has at least one address falling within this range. Note that the network prefix of the address is used for this check (i.e. not the local address). For PtP interfaces this means the address of the remote endpoint.

Neighbor

Sub-menu: `/routing rip neighbor`

This submenu is used to define a neighboring routers to exchange routing information with. Normally there is no need to add the neighbors, if multicasting is working properly within the network. If there are problems with exchanging routing information, neighbor routers can be added to the list. It will force the router to exchange the routing information with the neighbor using regular unicast packets.

Property	Description
address (*IP address*; Default: **0.0.0.0**)	IP address of neighboring router

Route

Sub-menu: `/routing rip route`

Read only properties:

Property	Description
dst-address (*IP prefix*)	destination network
gateway (*IP address*)	last gateway on the route to destination
metric (*integer*)	distance vector length to the destination network
from (*IP address*)	specifies the IP address of the router from which the route was received
timeout (*time*)	for valid RIP routes (metric < 16): time until the route will expire. For routes with metric 16: time until advertising of the route will be stopped

[Top | Back to Content]

Manual:RouterOS FAQ

See also: Mikrotik_RouterOS_Preguntas_Frecuentes_(español/spanish)

What is MikroTik RouterOS™?

What does MikroTik RouterOS™ do?

MikroTik RouterOS™ is a router operating system and software which turns a regular Intel PC or MikroTik RouterBOARD™ hardware into a dedicated router.

What features does RouterOS™ have?

RouterOS feature list

Can I test the MikroTik RouterOS™ functionality before I buy the license?

Yes, you can download the installation from MikroTik's webpage and install your own MikroTik router. The router has full functionality without the need for a license key for 24h total running time. That's enough time to test the router for 3 days at 8h a day, if you shut down the router at the end of each 8h day.

Where can I get the License Key?

Create an account on MikroTik's webpage (the top right-hand corner of www.mikrotik.com). You can use a credit card to pay for the key.

Can I use MikroTik router to hook up to a service provider via a T1, T3, or other high speed connection?

Yes, you can install various NICs supported by MikroTik RouterOS™ and get your edge router, backbone router, firewall, bandwidth manager, VPN server, wireless access point, HotSpot and much more in one box. Please check the Specification Sheet [1] and Manual [2] for supported interfaces!

How fast will it be?

An Intel PC is faster than almost any proprietary router, and there is plenty of processing power even in a 100MHz CPU.

How does this software compare to using a Cisco router?

You can do almost everything that a proprietary router does at a fraction of the cost of such a router and have flexibility in upgrading, ease of management and maintenance.

What OS do I need to install the MikroTik RouterOS™?

No Operating System is needed. The MikroTik RouterOS™ is standalone Operating System. The OS is Linux kernel based and very stable. Your hard drive will be wiped completely by the installation process. No additional disk support, just one PRIMARY MASTER HDD or FlashDisk, except for WEB proxy cache.

How secure is the router once it is setup?

Access to the router is protected by username and password. Additional users can be added to the router, specific rights can be set for user groups. Remote access to the router can be restricted by user, IP address. Firewall filtering is the easiest way to protect your router and network.

Installation

How can I install RouterOS?

RouterOS can be installed with CD Install or Netinstall.

How large HDD can I use for the MikroTik RouterOS™?

MikroTik RouterOS™ supports disks larger than 8GB (usually up to 120GB). But make sure the BIOS of the router's motherboard is able to support these large disks.

Can I run MikroTik RouterOS™ from any hard drive in my system?

Yes

Is there support for multiple hard drives in MikroTik RouterOS™?

A secondary drive is supported for web cache. This support has been added in 2.8, older versions don't support multiple hard drives.

Why the CD installation stops at some point and does not go "all the way through"?

The CD installation is not working properly on some motherboards. Try to reboot the computer and start the installation again. If it does not help, try using different hardware.

Logging on and Passwords

What is the username and password when logging on to the router for the first time?

Username is 'admin', and there is no password (hit the 'Enter' key). You can change the password using the '/password' command.

How can I recover a lost password?

If you have forgotten the password, there is no recovery for it. You have to reinstall the router.

After power failure the MikroTik router is not starting up again

If you haven't shut the router down, the file system has not been unmounted properly. When starting up, the RouterOS™ will perform a file system check. Depending on the HDD size, it may take several minutes to complete. Do not interrupt the file system check! It would make your installation unusable.

How can I access the router if the LAN interface has been disabled?

You can access the router either locally (using monitor and keyboard) or through the serial console.

Licensing Issues

How many MikroTik RouterOS™ installations does one license cover?

The license is per RouterOS installation. Each installed router needs a separate license.

Does the license expire?

The license never expires. The router runs for ever. Your only limitation is to which versions you can upgrade. For example if it says "Upgradable to v4.x", it means you can use all v4 releases, but not v5 This doesn't mean you can't stay on v4.x as long as you want.

How can I reinstall the MikroTik RouterOS™ software without losing my software license?

You have to use CD, Floppies or Netinstall procedure and install the MikroTik RouterOS™ on the HDD with the previous MikroTik RouterOS™ installation still intact. The license is kept with the HDD. Do not use format or partitioning utilities, they will delete your key! Use the same (initial) BIOS settings for your HDD!

Can I use my MikroTik RouterOS™ software license on a different hardware?

Yes, you can use different hardware (motherboard, NICs), but you should use the same HDD. The license is kept with the HDD unless format or fdisk utilities are used. It is not required to reinstall the system when moving to different hardware. When paying for the license, please be aware, that it cannot be used on another harddrive than the one it was installed upon.

License transfer to another hard drive costs 10$. Contact support to arrange this.

What to do, if my hard drive with MikroTik RouterOS™ crashes, and I have to install another one?

If you have paid for the license, you have to write to support[at]mikrotik.com and describe the situation. We may request you to send the broken hard drive to us as proof prior to issuing a replacement key.

What happens if my hardware breaks again, and I lose my replacement key?

The same process is used as above, but this time, we need physical proof that there is in fact been another incident.

If you have a free demo license, no replacement key can be issued. Please obtain another demo license, or purchase the base license.

More information available here All_about_licenses

How can I enter a new Software Key?

Entering the key from Console/FTP:

- import the attached file with the command '/system license import' (you should upload this file to the router's FTP server)

Entering the key with Console/Telnet:

- use copy/paste to enter the key into a Telnet window (no matter which submenu). Be sure to copy the whole key, including the lines "--BEGIN MIKROTIK SOFTWARE KEY--" and "--END MIKROTIK SOFTWARE KEY--"

Entering the key from Winbox:

- use 'system -> license' menu in Winbox to Paste or Import the key

I have mis-typed the software ID when I purchased the Software Key. How can I fix this?

In the Account Server choose `work with keys`, then select your mis-typed key, and then choose `fix key`.

About entering keys, see more on this page

Entering a RouterOS License key

All other information about License Keys can be found here

All_about_licenses

Upgrading

How can I install additional feature packages?

You have to use the same version package files (extension .npk) as the system package. Use the **/system package print** command to see the list of installed packages. Check the free space on router's HDD using the **/system resource print** command before uploading the package files. Make sure you have at least 2MB free disk space on the router after you have uploaded the package files!

Upload the package files using the ftp BINARY mode to the router and issue **/system reboot** command to shut down the router and reboot. The packages are installed (upgraded) while the router is going for shutdown. You can monitor the installation process on the monitor screen connected to the router. After reboot, the installed packages are listed in the **/system package print** list.

How can I upgrade?

To upgrade the software, you will need to download the latest package files (*.npk) from our website (the 'system' package plus the ones that you need). Then, connect to the router via FTP and upload the new packages to it by using Binary transfer mode.

Then reboot the router by issuing **/system reboot** command. More information here: Upgrading_RouterOS

I installed additional feature package, but the relevant interface does not show up under the **/interface print** list.

You have to obtain (purchase) the required license level or install the NPK package for this interface (for example package 'wireless').

If I do upgrade RouterOS, will I lose my configuration?

No, configuration is kept intact for upgrades within one version family. When upgrading version families (for example, V2.5 to V2.6) you may lose the configuration of some features that have major changes. For example when upgrading from V2.4, you should upgrade to the last version of 2.4 first.

How much free disk space do I need when upgrading to higher version?

You need space for the system package and the additional packages you have to upgrade. After uploading the newer version packages to the router you should have at least 2MB free disk space left. If not, do not try to make the upgrade! Uninstall the unnecessary packages first, and then upgrade the remaining ones.

Downgrading

How can I downgrade the MikroTik RouterOS™ installation to an older version?

You can downgrade by reinstalling the RouterOS™ from any media. The software license will be kept with the HDD as long as the disk is not repartitioned/reformatted. The configuration of the router will be lost (it is possible to save the old configuration, but this option has unpredictable results when downgrading and it is not recommended to use it).

Another way is to use the **/system package downgrade** command. This works only if you downgrade to 2.7.20 and not lower. Upload the older packages to the router via FTP and then use the **/system package downgrade** command.

TCP/IP Related Questions

I have two NIC cards in the MikroTik router and they are working properly. I can ping both networks from the router but can't ping from one network through the router to the other network and to the Internet. I have no firewall setup.

This is a typical problem, where you do not have routing set up at your main Internet gateway. Since you have introduced a new network, you need to 'tell' about it your main gateway (your ISP). A route should be added for your new network. Alternatively, you can 'hide' your new network by means of masquerading to get access to the Internet. Please take time to study the Basic Setup Guide, where the problem is described and the solution is given.

There is an example how to masquerade your private LAN:

```
[admin@MikroTik] ip firewall nat> add chain=srcnat action=masquerade out-interface=Public
[admin@MikroTik] ip firewall nat> print
Flags: X - disabled, I - invalid, D - dynamic
```

```
 0   chain=srcnat out-interface=Public action=masquerade
```

How can I change the TCP port number for telnet or http services, if I do not want to use the ports 23 and 80, respectively?

You can change the allocated ports under /ip service.

When I use the IP address/mask in the form 10.1.1.17/24 for my filtering or queuing rules, they do not work.

The rules 'do not work', since they do not match the packets due to the incorrectly specified address/mask. The correct form would be:

```
10.1.1.0/24 for the IP addresses in the range 10.1.1.0-10.1.1.255, or,
   10.1.1.17/32 for just one IP address 10.1.1.17.
```

I need to set up DHCP client, but there is no menu '/ip dhcp-client'.

The DHCP feature is not included in the system software package. You need to install the dhcp package. Upload it to the router and reboot!

Can I statically bind IP's to MAC addresses via DHCP?

Yes, you can add static leases to the DHCP server leases list. However, DHCP is insecure by default, and it is better to use PPPoE for user authentication and handing out IP addresses. There you can request the user to log on from a specified MAC address as well.

How can I masquerade two different subnets using two different external IP addresses for them?

Use **/ip firewall nat** rule with **chain=srcnat action=nat**, specify the **to-src-address** argument value. It should be one of the router's external addresses. If you use **action=masquerade**, the **to-src-address** is not taken into account, since it is substituted by the external address of the router automatically.

I cannot surf some sites when I use PPPoE.

Use **/ip firewall mangle** to change MSS (maximum segment size) 40 bytes less than your connection MTU. For example, if you have encrypted PPPoE link with MTU=1492, set the mangle rule as follows:

```
/ ip firewall mangle
 add chain=forward protocol=tcp tcp-flags=syn action=change-mss tcp-mss=!0-1448 new-mss=1448
```

Bandwidth Management Related Questions

How can I controll bandwidth(bandwidth shaping)in Bridge mode?

In bridge settings enable **use-ip-firewall**.

Can I use MikroTik as a bridge and a traffic shaper in one machine?

Yes. You can use all the extensive queue management features. Set the queue to the interface where the traffic is actually leaving the router, when passing through the router. It is not the bridge interface! The queue on the bridge interface is involved only for the traffic generated from the router.

Can I limit bandwidth based on MAC addresses?

For download:

```
1. connection-mark all packets from the MAC of each client with different marks
   for each client using action=passthrough:
   /ip firewall mangle add chain=prerouting src-mac-address=11:11:11:11:11:11 \
   action=mark-connection new-connection-mark=host11 passthrough=yes
```

```
   2. Remark these packets with flow-mark (again different flow-marks for each connection-marks):
   /ip firewall mangle add chain=prerouting connection-mark=host11 new-packet-mark=host11
```

```
   3. We can use these flow-marks in queue trees now.
```

While this solution should function, it is fundamentally flawed as the first packet of each connection destined to these clients will not be taken into account.

For upload:

```
[admin@AP] ip firewall mangle> add chain=prerouting src-mac-address=11:11:11:11:11:11 \
   action=mark-packet new-packet-mark=upload
```

Wireless Questions

Can I bridge wlan interface operating in the *station* mode?

No, you cannot.

See more >>

BGP Questions

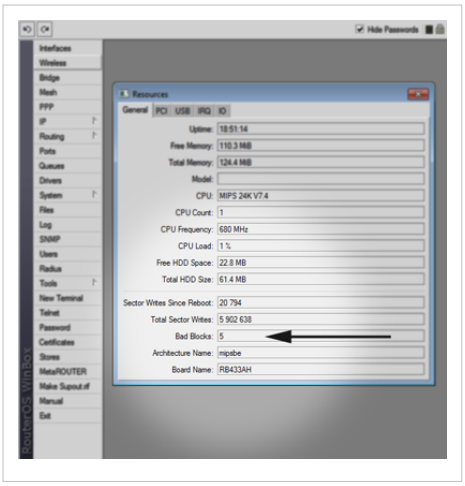
See BGP FAQ and HowTo

[Top | Back to Content]

References

[1] http://www.mikrotik.com/docs/ros/2.9/guide/specs
[2] http://www.mikrotik.com/docs/ros/2.9/

Manual:RouterBOARD bad blocks



Every once in a while, one can notice a number of **bad blocks** appearing in the RouterBOARD resource page. A bad block indicates a problem to write in one part of the NAND storage device, but it doesn't affect the performance of your router, and it doesn't give any indication of quality. According to the manufacturer of NAND chips, up to 5 blocks can be bad when NAND is manufactured, and up to 80 bad blocks could develop during operation, but it will **not** disturb the operation of your router, because complex workaround mechanisms are in place, that will copy the data to another block and attempt to fix the bad block.

- **Note**! *If more than a 100 bad blocks (more than 5% in new versions) should develop in the NAND chip of a RouterBOARD device, you should try to reinstall RouterOS with Netinstall, and it will reduce the number of bad blocks*

Since RouterOS v5.18, NAND is refreshed every few minutes which increases sector writes by ~32 per refresh. This is to ensure evenly distributed NAND wear, to increase protection against data loss as compared to previous versions. As the manufacturer of the NAND chip guarantees 100'000 write cycles to the NAND sectors, 32 sector writes per refresh would mean NAND life is about 100'000 weeks.

If you are planning to use the RouterBOARD as an active caching or logging server, an external drive or a replaceable memory card could be a better alternative than keeping them on the NAND chip. Note that several RouterBOARD models support USB external drives and different types of memory cards.

Important! As you can see in the screenshot above, RouterOS shows you writes per NAND TOTAL, not writes per sector. This is different than the given 100'000 write guarantee per sector.

Manual:RouterOS features

RouterOS features

RouterOS is MikroTik's stand-alone operating system based on linux v2.6 kernel. The following list shows features found in the latest RouterOS release:

Hardware Support

- i386 compatible architecture
- SMP – multi-core and multi-CPU compatible
- Minimum 32MB of RAM (maximum supported 2GB)
- IDE, SATA, USB and flash storage medium with minimum of 64MB space
- Network cards supported by linux v2.6 kernel (PCI, PCI-X)
- Partial hardware compatibility list (user maintained)
- Switch chip configuration support

Installation

- M:Netinstall: Full network based installation from PXE or EtherBoot enabled network card
- Netinstall: Installation to a secondary drive mounted in Windows
- CD based installation

Configuration

- MAC based access for initial configuration
- WinBox – standalone Windows GUI configuration tool
- Webfig - advanced web based configuration interface
- Basic web interface configuration tool
- Powerful command-line configuration interface with integrated scripting capabilities, accessible via local terminal, serial console, telnet and ssh
- API - the way to create your own configuration and monitoring applications.

Backup/Restore

- Binary configuration backup saving and loading
- Configuration export and import in human readable text format

Firewall

- Statefull filtering
- Source and destination NAT
- NAT helpers (h323, pptp, quake3, sip, ftp, irc, tftp)
- Internal connection, routing and packet marks
- Filtering by IP address and address range, port and port range, IP protocol, DSCP and many more
- Address lists
- Custom Layer7 matcher

- IPv6 support
- PCC - per connection classifier, used in load balancing configurations

Routing

- Static routing
- Virtual Routing and Forwarding (VRF)
- Policy based routing
- Interface routing
- ECMP routing
- IPv4 dynamic routing protocols: RIP v1/v2, OSPFv2, BGP v4
- IPv6 dynamic routing protocols: RIPng, OSPFv3, BGP
- Bidirectional Forwarding Detection (BFD)

MPLS

- Static Label bindings for IPv4
- Label Distribution protocol for IPv4
- RSVP Traffic Engineering tunnels
- VPLS MP-BGP based autodiscovery and signaling
- MP-BGP based MPLS IP VPN
- complete list of MPLS features

VPN

- Ipsec – tunnel and transport mode, certificate or PSK, AH and ESP security protocols. Hardware encryption support on RouterBOARD 1000 [1].
- Point to point tunneling (OpenVPN, PPTP, PPPoE, L2TP, SSTP)
- Advanced PPP features (MLPPP, BCP)
- Simple tunnels (IPIP, EoIP) IPv4 andIPv6 support
- 6to4 tunnel support (IPv6 over IPv4 network)
- VLAN – IEEE802.1q Virtual LAN support, Q-in-Q support
- MPLS based VPNs

Wireless

- IEEE802.11a/b/g wireless client and access point
- Full IEEE802.11n support
- Nstreme and Nstreme2 proprietary protocols
- NV2 protocol
- Wireless Distribution System (WDS)
- Virtual AP
- WEP, WPA, WPA2
- Access control list
- Wireless client roaming
- WMM
- HWMP+ Wireless MESH protocol
- MME wireless routing protocol

DHCP

- Per interface DHCP server
- DHCP client and relay
- Static and dynamic DHCP leases
- RADIUS support
- Custom DHCP options
- DHCPv6 Prefix Delegation (DHCPv6-PD)
- DHCPv6 Client

Hotspot

- Plug-n-Play access to the Network
- Authentication of local Network Clients
- Users Accounting
- RADIUS support for Authentication and Accounting

QoS

- Hierarchical Token Bucket (HTB) QoS system with CIR, MIR, burst and priority support
- Simple and fast solution for basic QoS implementation - Simple queues
- Dynamic client rate equalization (PCQ)

Proxy

- HTTP caching proxy server
- Transparent HTTP proxy
- SOCKS protocol support
- DNS static entries
- Support for caching on a separate drive
- Parent proxy support
- Access control list
- Caching list

Tools

- Ping, traceroute
- Bandwidth test, ping flood
- Packet sniffer, torch
- Telnet, ssh
- E-mail and SMS send tools
- Automated script execution tools
- CALEA
- File Fetch tool
- Advanced traffic generator

Other features

- Bridging – spanning tree protocol (STP, RSTP), bridge firewall and MAC natting.
- Dynamic DNS update tool
- NTP client/server and synchronization with GPS system
- VRRP v2 and v3 support
- SNMP
- M3P - MikroTik Packet packer protocol for wireless links and ethernet
- MNDP - MikroTik neighbor discovery protocol, supports CDP (Cisco discovery protocol)
- RADIUS authentication and accounting
- TFTP server
- Synchronous interface support (Farsync cards only) (Removed in v5.x)
- Asynchronous – serial PPP dial-in/dial-out, dial on demand
- ISDN – dial-in/dial-out, 128K bundle support, Cisco HDLC, x75i, x75ui, x75bui line protocols, dial on demand

[Top | Back to Content]

Manual:Route Selection Algorithm in RouterOS

The information in this article may be deprecated, and is described better elsewhere in the Wiki.

Who is who in route selection

There are more than one routing table in a running router.

- There is FIB, Forwarding Information Base, that makes packet forwarding decisions. In RouterOS case, FIB is inside Linux kernel.
- There is main routing table. This is where route selection decisions are made. For each destination there can be at least one route from each routing protocol, and one from static configuration, but only the route with the lowest distance will be installed in FIB. If there is more than one route with the same distance, selection is done in random (except for BGP routes). For non-BGP routes this can happen only if you are playing with filters. Scope attribute also determines which routes can be active.
- Each routing protocol (except BGP) has it's own internal tables. This is where per-protocol routing decisions are made. For example, in OSPF there are different types of routes: intra-area, inter-area, external-type-1, external-type-2. If all four type of routes to a single destination are present in OSPF internal LSA tables, only one of them -- intra-area route -- will make it to the common rotuing table.
- BGP has the most complicated selection process (described in BGP Best Path Selection Algorithm). Notice that this protocol-internal selection is done only after a BGP route is installed in main routing table; that means there can be one route for each BGP peer. Also note that BGP routes from different instances are compared by their distance.

Route distance

Cisco documentation describes "administrative distance" as "This is the measure of trustworthiness of the source of the route. If a router learns about a destination from more than one routing protocol, administrative distance is compared and the preference is given to the routes with lower administrative distance. In other words, it is the believability of the source of the route."

Default distances in RouterOS are:

protocol	distance
connected	0
static	1
eBGP	20
OSPF	110
RIP	120
MME	130
iBGP	200

Load balancing and dynamic routing protocols

With multiple nexthops

Normally, there can be only one active route per destination. If you want to do load balancing, multiple nexthops per route is the solution.

Note: This will only provide per-connection, not per-packet load balancing!

Example: suppose that you have a router that has 10.0.0.0/24 and 10.0.1.0/24 upstream networks.

```
[admin@II] > routing filter add chain=bgp-in prefix=0.0.0.0/0 set-in-nexthop=10.0.0.1,10.0.1.1
# use this filter for a bgp peer
[admin@II] > routing bgp peer set peer1 in-filter=bgp-in
# observe that it's really working
[admin@II] > ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE INTERFACE
 0 ADb  0.0.0.0/0                          r 10.0.0.1        1        ether1
                                           r 10.0.1.1                 ether2
```

What if you want to divide the traffic in uneven parts? Unlike in Linux (for example), you cannot set the weight for nexhtops. But you can simulate different weights, using specific nexthop more than once.

e.g.

```
[admin@II] > routing filter add chain=bgp-in prefix=0.0.0.0/0 set-in-nexthop=10.0.0.1,10.0.0.1,10.0.1.1
```

Now about 2/3 of traffic are routed to 10.0.0.1 and remaining 1/3 - to 10.0.1.1

Note: Another way to achieve load balancing is to use multiple recursive next-hop resolution, as described here.

With policy routing

Another solution is to use policy routes. This way you can have more than one active route to a particular destination. Suppose you already have a static default route, and want to have another one from bgp.

Static route

```
[admin@II] > /ip route p
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE INTERFACE
 0 A S  0.0.0.0/0                          r 10.0.1.1        1        ether2
```

Add policy route rule and tell the filter to set it:

```
[admin@II] > /ip route rule add table=aaa routing-mark=aaa
[admin@II] > /routing filter add chain=bgp-in prefix=0.0.0.0/0 set-routing-mark=aaa
```

Now, when default route is received from BGP:

```
[admin@II] > /ip route print detail
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 0 ADb  dst-address=0.0.0.0/0 gateway=10.0.0.1 interface=ether1 gateway-state=reachable
        distance=200 scope=255 target-scope=30 routing-mark=aaa bgp-local-pref=100
        bgp-origin=incomplete received-from=10.0.0.133
 1 A S  dst-address=0.0.0.0/0 gateway=10.0.1.1 interface=ether2 gateway-state=reachable
        distance=1 scope=255 target-scope=10
```

Now mark the traffic to route appropiately.

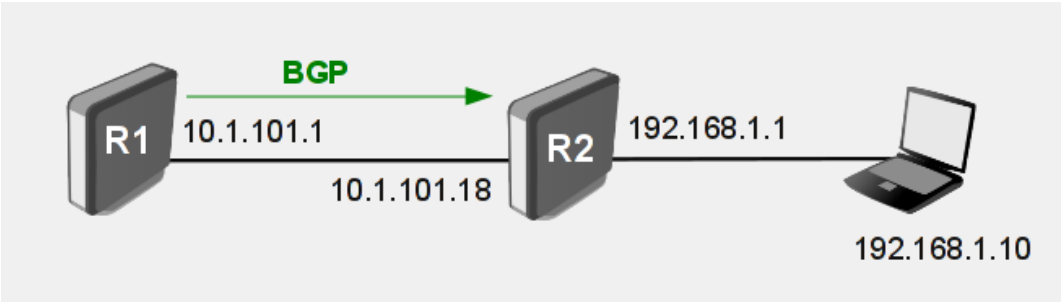
Manual:Routing Table Matcher

Sometimes ISP's are giving different local and overseas bandwidth. To set up QoS you had to make static address list of local IP addresses, keep track of Ip ranges used in your country and update address list accordingly. Here you can find article describing mentioned approach.

With introduction of **routing-table** matcher it is possible to **match packet which destination address is resolved in specific routing table**. So we just need BGP peering with ISP and ask them to send all routes local to your country, add them to routing table and set up mangle rules accordingly.

Note: It is not possible to match source address against routing table.

Consider following setup:



R1 is ISP router sending BGP routes R2 is client's main gateway and clients local network is 192.168.1.0/24

After setting up bgp peering (which is not covered in this article) we get following BGP routes

```
[admin@MikroTik] /ip route> print where bgp
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 ..
 1 ADb  10.10.1.0/24                       10.1.101.1         20
 2 ADb  10.10.10.4/32                      10.1.101.1         20
```

Next step is to add all received BGP rotues to another routing table, to do that we set up routing filters

```
#at first we have to specify input filter chain
/routing bgp peer set 0 in-filter=bbgp

#now we set up filter itself
/routing filter
add action=passthrough chain=bbgp set-routing-mark=local
```

As you can see now routes are added to **"local"** routing table

```
[admin@MikroTik] /ip route> print detail where routing-mark="local"
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit

...
```

```
 1 ADb  dst-address=10.10.1.0/24 gateway=10.1.101.1
        gateway-status=10.1.101.1 reachable ether1 distance=20 scope=255
        target-scope=255 routing-mark=local
        bgp-as-path="3001,3001,3010,3002,3000" bgp-origin=incomplete
        received-from=ISP

 2 ADb  dst-address=10.10.10.4/32 gateway=10.1.101.1
        gateway-status=10.1.101.1 reachable ether1 distance=20 scope=255
        target-scope=255 routing-mark=local
        bgp-as-path="3001,3001,3010,3002,3000" bgp-origin=incomplete
        bgp-communities=3000:120,3000:200 received-from=ISP
```

Following mangle rule will match all packets that destination is resolved in **"local"** routing table.

```
/ip firewall mangle
add action=log chain=forward  routing-table=local
```

Now when we try to send packets from the client for example to address 10.10.10.4, mangle rule will not match anything. This is because by default every destination is resolved in **"main"** routing table.

To fix this we have to explicitly specify to resolve all packets coming from client in **"local"** routing table.

```
/ip route rule
add action=lookup src-address=192.168.1.0/24 table=local
```

To verify if packets are actually matched:

```
[admin@MikroTik] /ip firewall mangle> print stats
Flags: X - disabled, I - invalid, D - dynamic
 #   CHAIN              ACTION                  BYTES           PACKETS
 0   forward            log                     28736           449
```

Also check log messages

```
[admin@MikroTik] /log> print
...
11:06:31 firewall,info forward: in:bridge1 out:ether1, src-mac 00:0c:42:21:f1:ec
, proto ICMP (type 8, code 0), 192.168.1.10->10.10.10.4, len 44
11:06:32 firewall,info forward: in:bridge1 out:ether1, src-mac 00:0c:42:21:f1:ec
, proto ICMP (type 8, code 0), 192.168.1.10->10.10.10.4, len 44
...
```

As you can see from the logs only packets coming from the client are matched. The reason for this is because **routing-table** matcher is matching only packet which destination address is resolved in **local** routing table. In our example 192.168.1.10 as destination is resolved in "main" routing table.

From what was said above, this approach is useful **only** for upload traffic marking and shaping.

Manual:Routing/MME

Applies to RouterOS: v3, v4+

Summary

Sub-menu: `/routing mme`
Packages required: `routing`

MME (Mesh Made Easy) is a MikroTik routing protocol suited for IP level routing in wireless mesh networks. It is based on ideas from B.A.T.M.A.N. (Better Approach To Mobile Ad-hoc Networking) routing protocol.

This is MME configuration reference only; for description of the protocol and configuration examples see Manual:MME wireless routing protocol.

General Setup

Property	Description
origination-interval (*time*; Default: **5s**)	Interval between originator messages. Obviously, this value should be less than **timeout** value.
timeout (*time*; Default: **1m**)	Node timeout. If no messages at all are received from an originator node during this interval, that node is purged from protocol tables, and so are all routes it has announced.
bidirectional-timeout (*integer*; Default: **2**)	How many originator messages from a node can be lost in sequence, while still considering it a bidirectional neighbor. We are assuming that every node originates messages with the same rate as this router (i.e. the value from **origination-interval**).
ttl (*integer*; Default: **50**)	How many times to forward originator messages.
gateway-class (*none* | *56-KBit* | *64-KBit* | *128-KBit* | *256-KBit* | *512-KBit* | *1-MBit* | *2-MBit* | *3-MBit* | *5-MBit* | *6-MBit* | *>6-MBit* | *integer*; Default: **none**)	Announce internet gateway capability in the originator messages sent by this node.
gateway-selection (*no-gateway* | *best-adjusted* | *best-statistic*; Default: **no-gateway**)	This node is a MME gateway protocol client. • **no-gateway** - don't install default route via MME. • **best-adjusted** - select best gateway node based on received message statistics and announced gateway class; • **best-statistic** - select best gateway node based only on received message statistics;
gateway-keepalive (*time*; Default: **1m**)	The time interval between successive gateway keepalive messages. For gateway client, this specifies how often to send out keepalive messages. For gateway server, as client hold time is used 3 * **gateway-keepalive** seconds. If the server does not receive keepalive messages from a client during this time interval, the client is considered dead. All state information associated with it are deleted, including the dynamic IPIP tunnel.
preferred-gateway (*IP*; Default: **0.0.0.0**)	Always prefer this node as internet gateway to any others, if it is present in originator tables.

Note: The node running MME with gateway-class option is supposed to have a link to Internet and a default route to that.

The symbolic values of `gateway-class` are compatible with B.A.T.M.A.N. This table describes the mapping from integers to symbolic values:

- 0 no gateway
- 1 modem
- 2 ISDN
- 3 Double ISDN
- 4 256 KBit
- 5 UMTS/ 0.5 MBit
- 6 1 MBit
- 7 2 MBit
- 8 3 MBit
- 9 5 MBit
- 10 6 MBit
- 11 >6 MBit

Entering integer value > 11 means even better gateway class.

Interfaces

Sub-menu: `/routing mme interface`

List of interfaces on which to run the MME protocol.

Property	Description
interface (*string*; Default: **all**)	Interface on which MME will run • **all** - is used for the interfaces not having any specific settings
passive (*yes* | *no* ; Default: **no**)	If true, do not send originator messages via this interface, only receive.
primary (*yes* | *no* ; Default: **no**)	Include routing information (i.e. network announcements) in self-originated packets send via this interface. (For forwared packets the information is always included.) Only one interface can be primary. If no interfaces are configured as primary, one is selected automatically in a random fashion.

Command `/routing mme interface print status` allows to view status of interfaces.

Property	Description
messages-tx (*integer*)	Originator messages transmitted via this interface. For **all** interface: cumulative statistics
messages-rx (*integer*)	Originator messages received via this interface. For **all** interface: cumulative statistics.

Networks

Sub-menu: `/routing mme network`

MME Networks is a list of networks to be advertised.

Property	Description
network (*IP prefix*; Default:)	Network to advertise

Note: The usage of MME networks is similar to BGP networks, and different from IGP (i.e. RIP and OSPF) networks. They determine which networks to announce via MME, not on which networks to run the protocol.

Originators

Sub-menu: `/routing mme originators`

This submenu contains information about active neighbor nodes.

Property	Description
originator (*IP*)	IP address of the node.
gateway (*IP*)	The nexthop for this node.
gateway-class (*none* | *56-KBit* | *64-KBit* | *128-KBit* | *256-KBit* | *512-KBit* | *1-MBit* | *2-MBit* | *3-MBit* | *5-MBit* | *6-MBit* | *>6-MBit* | *integer*)	If **none**, then this node is not a gateway server. Otherwise this node is a gateway server with specified gateway bandwidth.
last-packet-before (*time*)	Seconds elapsed since last received packet.

[Top | Back to Content]

Manual:Interface/SSTP

Applies to RouterOS: v5

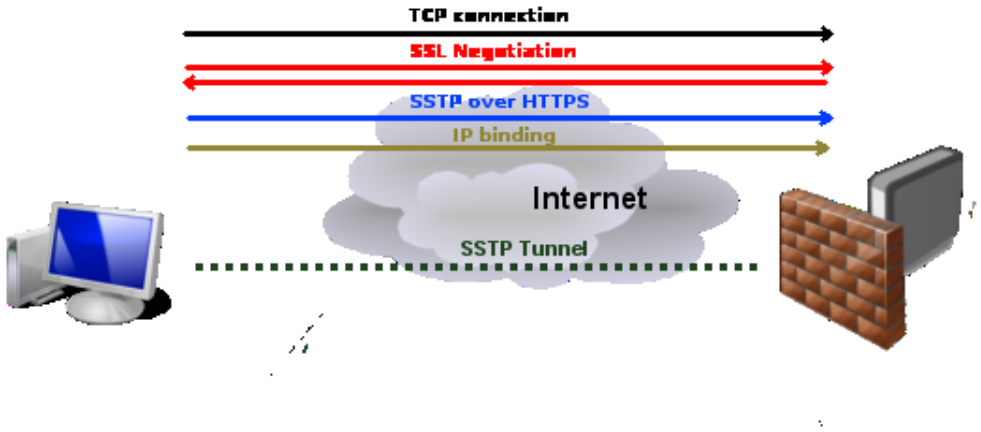
Summary

Standards: `SSTP specification` [1]

Package: `ppp`

Secure Socket Tunneling Protocol (SSTP) is the way to transport PPP tunnel over SSL 3.0 channel. The use of SSL over TCP port 443 allows SSTP to pass through virtually all firewalls and proxy servers.

SSTP connection mechanism



- TCP connection is established from client to server (by default on port 443);
- SSL validates server certificate. If certificate is valid connection is established otherwise connection is torn down.
- The client sends SSTP control packets within the HTTPS session which establishes the SSTP state machine on both sides.
- PPP negotiation over SSTP. Client authenticates to the server and binds IP addresses to SSTP interface
- SSTP tunnel is now established and packet encapsulation can begin.

If both client and server are Mikrotik routers, then it is possible to establish SSTP tunnel without certificates and with any available authentication type. Otherwise to establish secure tunnels **mschap** authentication and client/server certificates from the same chain should be used. `Read more>>`

Note: Starting from v5.0beta2 SSTP does not require certificates to operate. This feature will work only between two MikroTik routers, as it is not according to standards.

Currently, SSTP clients exist only in Windows Vista, Windows 7 and RouterOS.

Note: While connecting to SSTP server, Windows does CRL (certificate revocation list) checking on server certificate which can introduce **significant delay** to complete connection or even prevent user from accessing sstp server at all if Windows is unable to access CRL distribution point! Custom generated CA which does not include CRLs can be used to minimize connection delays and certificate costs (signed certificates with known CA usually are not for free), but this custom CA must be imported into each Windows client individually. It is possible to disable CRL check in Windows registry, but it is supported only by Windows Server 2008 http://support.microsoft.com/kb/947054

Certificates

To set up secure SSTP tunnel, certificates are required. On the server authentication is done only by username and password, but on the client - server is authenticated using server certificate. It is also used by client to cryptographicly bind SSL and PPP authentication, meaning - the clients sends a special value over SSTP connection to server, this value is derived from the key data that is generated during PPP authentication and server certificate, this allows server to check if both channels are secure.

If sstp clients are Windows PCs then only way to set up secure SSTP tunnel when using self-signed certificate is by importing "server" certificate on SSTP server and on windows PC add CA certificate in trusted root [2].

Note: If your server certificate is issued by CA which is known by Windows, then Windows client will work witout any additional certificates.

Warning: RSA Key length must be at least 472 bits if certificate is used by SSTP. Shorter keys are considered as security threats.

Similar configuration on RouterOS client would be, importing CA certificate and enabling **verify-server-certificate** option. In this scenario Man-in-the-Middle attacks are not possible.

Between two Mikrotik routers it is also possible to set up insecure tunnel by not using certificates at all. In this case data going through SSTP tunnel is using anonymous DH and Man-in-the-Middle attacks are easily accomplished. **This scenario is not compatible with Windows clients**.

It is also possible to make secure SSTP tunnel by adding additional authorization with client certificate. Configuration requirements are:

- certificates on both server and client
- verification options enabled on server and client

This scenario is also not possible with Windows clients, because there is no way to set up client certificate on Windows.

Certificate error messages

When ssl handshake fails, you will see one of the following certificate errors:

- **certificate is not yet valid** - *notBefore* date is after the current time.
- **certificate has expired** - *notAfter* date is before the current time.
- **invalid certificate purpose** - the supplied certificate cannot be used for the specified purpose.
- **self signed certificate in chain** - the certificate chain could be built up using the untrusted certificates but the root could not be found locally.
- **unable to get issuer certificate locally** - CA certificate is not imported locally.
- **server's IP address does not match certificate** - server address verification is enabled, but address provided in certificate does not match servers address.

Hostname verification

Starting from v5.6 when server ceritificate verification is enabled on sstp client, additionally IP addresses found in certificate's **subjectAltName** and then **issuer CN** will be compared to the real address. DNS names are ignored. v5.7 adds new parameter `verify-server-address-from-certificate` to disable/enable hostname verification.

SSTP Client

Sub-menu: `/interface sstp-client`

Properties

Property	Description
add-default-route (*yes* | *no*; Default: **no**)	Whether to add SSTP remote address as a default route.
authentication (*mschap2* | *mschap1* | *chap* | *pap*; Default: **mschap2, mschap1, chap, pap**)	Allowed authentication methods.
certificate (*string* | *none*; Default: **none**)	
comment (*string*; Default:)	Descriptive name of an item
connect-to (*IP:Port*; Default: **0.0.0.0:443**)	Remote address and port of SSTP server.
dial-on-demand (*yes* | *no*; Default: **no**)	
disabled (*yes* | *no*; Default: **yes**)	Whether interface is disabled or not. By default it is disabled.
keepalive-timeout (*integer* | *disabled*; Default: **60**)	
max-mru (*integer*; Default: **1500**)	Maximum Receive Unit. Max packet size that SSTP interface will be able to receive without packet fragmentation.
max-mtu (*integer*; Default: **1500**)	Maximum Transmission Unit. Max packet size that SSTP interface will be able to send without packet fragmentation.
mrru (*disabled* | *integer*; Default: **disabled**)	Maximum packet size that can be received on the link. If a packet is bigger than tunnel MTU, it will be split into multiple packets, allowing full size IP or Ethernet packets to be sent over the tunnel. `Read more >>`
name (*string*; Default:)	Descriptive name of the interface.
password (*string*; Default: "")	Password used for authentication.
profile (*name*; Default: **default-encryption**)	Used PPP profile.
proxy (*IP:Port*; Default: **0.0.0.0:443**)	Address and port of HTTP proxy server.
user (*string*; Default:)	User name used for authentication.
verify-server-certificate (*yes* | *no*; Default: **no**)	If set to yes, then client checks whether certificate belongs to the same certificate chain as server's certificate. To make it work CA certificate must be imported.
verify-server-address-from-certificate (*yes* | *no*; Default: **yes**)	If set to yes, server's IP address will be compared to one set in certificate. `Read More >>`

Quick example

This example demonstrates how to set up SSTP client with username "sstp-test", password "123" and server 10.1.101.1

```
[admin@MikroTik]  /interface sstp-client>add user=sstp-test password=123 \
\... connect-to=10.1.101.1 disabled=no
[admin@MikroTik] /interface sstp-client> print
Flags: X - disabled, R - running
 0  R name="sstp-out1" max-mtu=1500 max-mru=1500 mrru=disabled connect-to=10.1.101.1:443
      user="sstp-test" password="123" proxy=0.0.0.0:443 profile=default
      certificate=none keepalive-timeout=60 add-default-route=no dial-on-demand=no
      authentication=pap,chap,mschap1,mschap2
```

SSTP Server

Sub-menu: `/interface sstp-server`

This sub-menu shows interfaces for each connected SSTP clients.

An interface is created for each tunnel established to the given server. There are two types of interfaces in SSTP server's configuration

- Static interfaces are added administratively if there is a need to reference the particular interface name (in firewall rules or elsewhere) created for the particular user.
- Dynamic interfaces are added to this list automatically whenever a user is connected and its username does not match any existing static entry (or in case the entry is active already, as there can not be two separate tunnel interfaces referenced by the same name).

Dynamic interfaces appear when a user connects and disappear once the user disconnects, so it is impossible to reference the tunnel created for that use in router configuration (for example, in firewall), so if you need a persistent rules for that user, create a static entry for him/her. Otherwise it is safe to use dynamic configuration.

Note: in both cases PPP users must be configured properly - static entries do not replace PPP configuration.

Server configuration

Sub-menu: `/interface sstp-server server`

Properties:

Property	Description
`authentication` (*pap* | *chap* | *mschap1* | *mschap2*; Default: **pap,chap,mschap1,mschap2**)	Authentication methods that server will accept.
`certificate` (*name* | *none*; Default: **none**)	Name of the certificate that SSTP server will use.
`default-profile` (*name*; Default: **default**)	
`enabled` (*yes* | *no*; Default: **no**)	Defines whether SSTP server is enabled or not.
`keepalive-timeout` (*integer* | *disabled*; Default: **60**)	Defines the time period (in seconds) after which the router is starting to send keepalive packets every second. If no traffic and no keepalive responses has came for that period of time (i.e. 2 * keepalive-timeout), not responding client is proclaimed disconnected
`max-mru` (*integer*; Default: **1500**)	Maximum Receive Unit. Max packet size that SSTP interface will be able to receive without packet fragmentation.
`max-mtu` (*integer*; Default: **1500**)	Maximum Transmission Unit. Max packet size that SSTP interface will be able to send without packet fragmentation.
`mrru` (*disabled* | *integer*; Default: **disabled**)	Maximum packet size that can be received on the link. If a packet is bigger than tunnel MTU, it will be split into multiple packets, allowing full size IP or Ethernet packets to be sent over the tunnel. `Read more >>`
`verify-client-certificate` (*yes* | *no*; Default: **no**)	If set to yes, then server checks whether client's certificate belongs to the same certificate chain.

```
[admin@MikroTik] /interface sstp-server server> set certificate=server
[admin@MikroTik] /interface sstp-server server> set enabled=yes
[admin@MikroTik] /interface sstp-server server> print
                    enabled: yes
                       port: 443
                    max-mtu: 1500
                    max-mru: 1500
```

```
                        mrru: disabled
           keepalive-timeout: 60
             default-profile: default
                 certificate: server
  verify-client-certificate: no
              authentication: pap,chap,mschap1,mschap2
[admin@MikroTik] /interface sstp-server server>
```

Warning: It is very important that date on the router is in the range of certificate's date of expiration . To overcome any certificate verification problems, enable **NTP** date synchronization on both server and client.

Monitoring

Monitor command can be used to monitor status of the tunnel on both client and server.

```
[admin@dzeltenais_burkaans] /interface sstp-server> monitor 0
     status: "connected"
     uptime: 17m47s
  idle-time: 17m47s
       user: "sstp-test"
  caller-id: "10.1.101.18:43886"
        mtu: 1500
```

Read-only properties

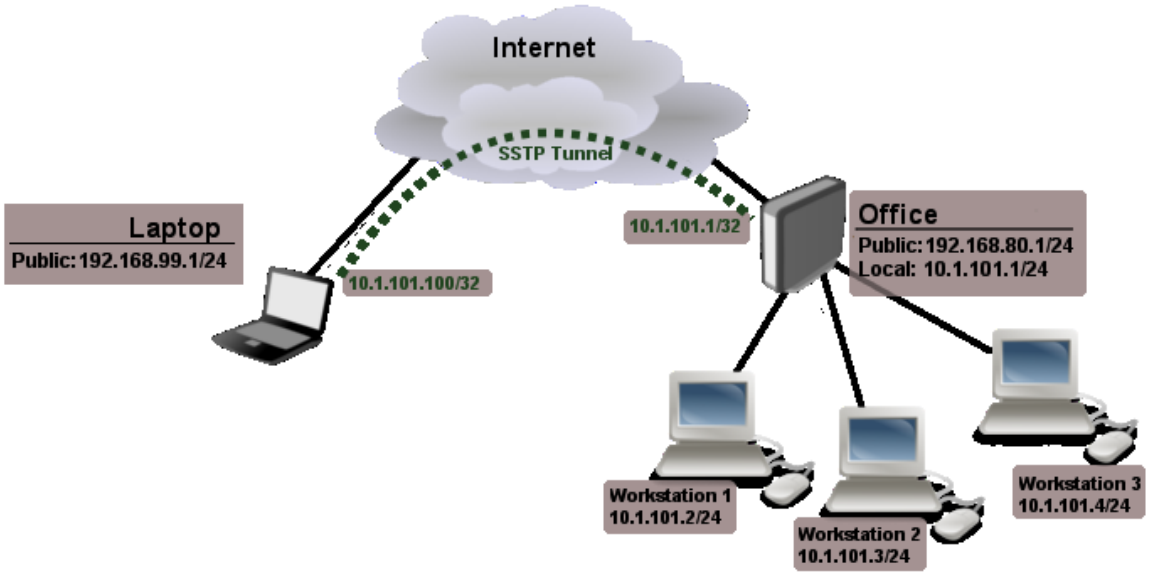
Property	Description
status ()	Current SSTP status. Value other than "connected" indicates that there are some problems estabising tunnel.
uptime (*time*)	Elapsed time since tunnel was established.
idle-time (*time*)	Elapsed time since last activity on the tunnel.
user (*string*)	Username used to establish the tunnel.
mtu (*integer*)	Negotiated and used MTU
caller-id (*IP:ID*)	

Application Examples

Connecting Remote Client

The following example shows how to connect a computer to a remote office network over **secure SSTP encrypted** tunnel giving that computer an IP address from the same network as the remote office has (without need of bridging over EoIP tunnels)

Consider following setup:



Office router is connected to internet through **ether1**. Workstations are connected to **ether2**. Laptop is connected to the internet and can reach Office router's public IP (in our example it is 192.168.80.1).

Before you begin to configure SSTP you need to create a server certificate and import it to the router (instructions here).

Now it is time to create a user

```
[admin@RemoteOffice] /ppp secret> add name=Laptop service=sstp password=123
local-address=10.1.101.1 remote-address=10.1.101.100
[admin@RemoteOffice] /ppp secret> print detail
Flags: X - disabled
  0   name="Laptop" service=sstp caller-id="" password="123" profile=default
      local-address=10.1.101.1 remote-address=10.1.101.100 routes==""

[admin@RemoteOffice] /ppp secret>
```

Notice that SSTP local address is the same as routers address on local interface and remote address is form the same range as local network (10.1.101.0/24).

Next step is to enable sstp server and sstp client on the laptop.

```
[admin@RemoteOffice] /interface sstp-server server> set certificate=server
[admin@RemoteOffice] /interface sstp-server server> set enabled=yes
[admin@RemoteOffice] /interface sstp-server server> set authentication=mschap2
[admin@RemoteOffice] /interface sstp-server server> print
                     enabled: yes
```

```
                        port: 443
                     max-mtu: 1500
                     max-mru: 1500
                        mrru: disabled
           keepalive-timeout: 60
             default-profile: default
                 certificate: server
  verify-client-certificate: no
              authentication: mschap2

[admin@RemoteOffice] /interface sstp-server server>
```

Notice that authentication is set to **mschap**. These are the only authentication options that are valid to establish secure tunnel.

SSTP client from the laptop should connect to routers public IP which in our example is 192.168.80.1.

Please, consult the respective manual on how to set up a SSTP client with the software You are using. If you set up SSTP client on Windows and self-signed certificates are used, then CA certificate should be added to trusted root [2].

Note: Currently SSTP is supported on Windows 2008, Windows Vista and Vista SP1. Other OS will not be able to connect to SSTP server

To verify if sstp client is connected

```
[admin@RemoteOffice] /interface sstp-server> print
Flags: X - disabled, D - dynamic, R - running
 #      NAME       USER          MTU         CLIENT-ADDRESS    UPTIME    ENCODING
 0  DR <sstp-... Laptop      1500       10.1.101.18:43886 1h47s

[admin@RemoteOffice] /interface sstp-server>monitor 0
     status: "connected"
     uptime: 1h45s
  idle-time: 1h45s
       user: "Laptop"
  caller-id: "192.168.99.1:43886"
        mtu: 1500
```

At this point (when SSTP client is successfully connected) if you will try to ping any workstation form the laptop, ping will time out, because Laptop is unable to get ARPs from workstations. Solution is to set up `proxy-arp` on local interface

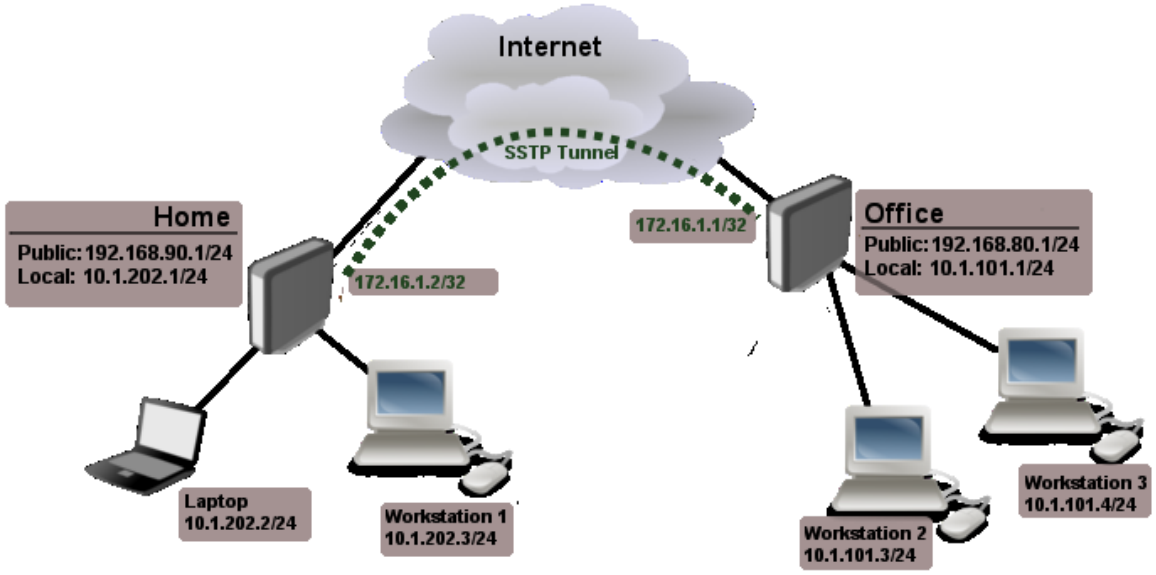
```
[admin@RemoteOffice] /interface ethernet> set ether2 arp=proxy-arp
[admin@RemoteOffice] /interface ethernet> print
Flags: X - disabled, R - running
  #    NAME                 MTU   MAC-ADDRESS         ARP
  0  R ether1               1500  00:30:4F:0B:7B:C1 enabled
  1  R ether2               1500  00:30:4F:06:62:12 proxy-arp
[admin@RemoteOffice] interface ethernet>
```

After `proxy-arp` is enabled client can successfully reach all workstations in local network behind the router.

Site-to-Site SSTP

The following is an example of connecting two Intranets using SSTP tunnel over the Internet.

Consider following setup



Office and Home routers are connected to internet through **ether1**, workstations and laptops are connected to **ether2**. In this example both local networks are routed through sstp client, thus they are not in the same broadcast domain. To overcome this problem as any other ppp tunnel SSTP also supports BCP which allows to bridge SSTP tunnel with local interface.

First step is to create a user

```
[admin@RemoteOffice] /ppp secret> add name=Home service=sstp password=123
local-address=172.16.1.1 remote-address=172.16.1.2 routes="10.1.202.0/24 172.16.1.2 1"
[admin@RemoteOffice] ppp secret> print detail
Flags: X - disabled
  0   name="Home" service=sstp caller-id="" password="123" profile=default
      local-address=172.16.1.1 remote-address=172.16.1.2 routes=="10.1.101.0/24 172.16.1.1 1"

[admin@RemoteOffice] /ppp secret>
```

Notice that we set up SSTP to add route whenever client connects. If this option is not set, then you will need static routing configuration on the server to route traffic between sites through SSTP tunnel.

Now we need to upload and import CA and server/client certificates. Assume that files are already uploaded use following commands:

```
admin@RemoteOffice] /certificate> import file-name=ca.crt
passphrase:
admin@RemoteOffice] /certificate> import file-name=server.crt
passphrase: ****
admin@RemoteOffice] /certificate> import file-name=server.key
passphrase: ****
```

Set up proper names:

```
admin@RemoteOffice] /certificate>set 0 name=CA
admin@RemoteOffice] /certificate>set 1 name=server
admin@RemoteOffice] /certificate> print
Flags: K - decrypted-private-key, Q - private-key, R - rsa, D - dsa
 0  D name="CA" subject=C=LV,ST=RI,L=Riga,O=MT,CN=MT CA,emailAddress=xx@mt.lv
      issuer=C=LV,ST=RI,L=Riga,O=MT,CN=MT CA,emailAddress=xx@mt.lv
      serial-number="DF626FA846090BCC" email=xx@mt.lv invalid-before=jun/25/2008 07:23:50
      invalid-after=jun/23/2018 07:23:50 ca=yes

 1 KR name="server" subject=C=LV,ST=RI,L=Riga,O=MT,CN=server,emailAddress=xx@mt.lv
      issuer=C=LV,ST=RI,L=Riga,O=MT,CN=MT CA,emailAddress=xx@mt.lv serial-number="01"
      email=xx@mt.lv invalid-before=jun/25/2008 07:24:33 invalid-after=jun/23/2018 07:24:33
      ca=yes
```

Do the same on client side, but instead of server's certificate import client's certificate.

Next step is to enable SSTP server on the office router and configure SSTP client on the Home router.

```
[admin@RemoteOffice] /interface sstp-server server> set certificate=server
[admin@RemoteOffice] /interface sstp-server server> set enabled=yes
[admin@RemoteOffice] /interface sstp-server server> set verify-client-certificate=yes
[admin@RemoteOffice] /interface sstp-server server> print
                     enabled: yes
                        port: 443
                     max-mtu: 1500
                     max-mru: 1500
                        mrru: disabled
           keepalive-timeout: 60
             default-profile: default
                 certificate: server
  verify-client-certificate: yes
              authentication: pap,chap,mschap1,mschap2
```

```
[admin@Home] /interface sstp-client> add user=Home password=123 connect-to=192.168.80.1 disabled=no
  certificate=client verify-server-certificate=yes
[admin@Home] /interface sstp-client> print
Flags: X - disabled, R - running
 0  R name="sstp-out1" max-mtu=1500 max-mru=1500 mrru=disabled connect-to=192.168.80.1:443
       user="Home" password="123" proxy=0.0.0.0:443 profile=default certificate=client
       keepalive-timeout=60 add-default-route=no dial-on-demand=no
       authentication=pap,chap,mschap1,mschap2 verify-server-certificate=yes

[admin@Home] /interface sstp-client>
```

Now we need to add static route on Home router to reach local network behind Office router

```
[admin@Home] /ip route> add dst-address=10.1.101.0/24 gateway=172.16.1.1
```

After tunnel is established you should be able to ping remote network.

Troubleshooting

After Windows 7 upgrade SSTP is unable to connect (windows error 631) ?

> MS Patch KB2585542 changes cypher to RC4 which was not supported on RouterOS. Starting from RouterOS v5.13 RC4 is the preferred cipher and AES will be used only if peer does not advertise RC4.

I get following error when trying to connect Windows 7 client. Error 0x80070320 The oplock that was associated with this handle is now associated with a different handle.

> Disable **verify-client-certificate** option on the server.

Read More

- Creating Certificates
- BCP (Bridge Control Protocol)
- http://technet.microsoft.com/en-us/library/cc731352(WS.10).aspx
- Free trusted Class1 certificates [3] online

[Top | Back to Content]

References

[1] http://msdn.microsoft.com/en-us/library/cc247338(PROT.10).aspx
[2] http://technet.microsoft.com/en-us/library/dd458982.aspx
[3] http://www.startssl.com/

Manual:IP/Services

Applies to RouterOS: v3, v4

Summary

Sub-menu: `/ip service`

This document lists protocols and ports used by various MikroTik RouterOS services. It helps you to determine why your MikroTik router listens to certain ports, and what you need to block/allow in case you want to prevent or grant access to the certain services. Please see the relevant sections of the Manual for more explanations.

Properties

Note that it is not possible to add new services, only existing service modifications are allowed.

Property	Description
address (*IP address/netmask | IPv6/0..128*; Default:)	List of IP/IPv6 prefixes from which the service is accessible.
certificate (*name*; Default: **none**)	The name of the certificate used by particular service. Applicable only for services that depends on certificates (*www-ssl*)
name (*name*; Default: **none**)	Service name
port (*integer: 1..65535*; Default:)	The port particular service listens on

Example

For example allow telnet only from specific IPv6 address range

```
[admin@dzeltenais_burkaans] /ip service> set api address=10.5.101.0/24,2001:db8:fade::/64
[admin@dzeltenais_burkaans] /ip service> print
Flags: X - disabled, I - invalid
 #   NAME     PORT  ADDRESS                                       CERTIFICATE
 0   telnet   23
 1   ftp      21
 2   www      80
 3   ssh      22
 4 X www-ssl  443                                                 none
 5   api      8728  10.5.101.0/24
                    2001:db8:fade::/64
 6   winbox   8291
```

Service Ports

Sub-menu: `/ip firewall service-port`

Hosts behind a NAT-enabled router do not have true end-to-end connectivity. Therefore some Internet protocols might not work in scenarios with NAT.
To overcome these limitations RouterOS includes a number of NAT helpers, that enable NAT traversal for various protocols.

Helper	Description
FTP	FTP service helper
h323	H323 service helper
irc	
PPTP	PPTP tunneling helper.
SIP	
tftp	

Protocols and ports

Table below shows the list of protocols and ports used by RouterOS.

Proto/Port	Description
20/tcp	FTP data connection
21/tcp	FTP control connection
22/tcp	Secure Shell (SSH) remote Login protocol
23/tcp	Telnet protocol
53/tcp **53/udp**	DNS
67/udp	Bootstrap protocol or DHCP Server
68/udp	Bootstrap protocol or DHCP Client
80/tcp	World Wide Web HTTP
123/udp	Network Time Protocol (NTP)
161/udp	Simple Network Management Protocol (SNMP)
179/tcp	Border Gateway Protocol (BGP)
443/tcp	Secure Socket Layer (SSL) encrypted HTTP
500/udp	Internet Key Exchange (IKE) protocol
520/udp **521/udp**	RIP routing protocol
646/tcp	LDP transport session
646/udp	LDP hello protocol
1080/tcp	SOCKS proxy protocol
1701/udp	Layer 2 Tunnel Protocol (L2TP)
1723/tcp	Point-To-Point Tunneling Protocol (PPTP)
1900/udp **2828/tcp**	Universal Plug and Play (uPnP)
2000/tcp	Bandwidth test server
5678/udp	Mikrotik Neighbor Discovery Protocol
8080/tcp	HTTP Web Proxy
8291/tcp	Winbox
8728/tcp	API
20561/udp	MAC winbox
/1	ICMP
/4	IPIP encapsulation
/41	IPv6 (encapsulation)
/47	General Routing Encapsulation (GRE) - used for PPTP and EoIP tunnels
/50	Encapsulating Security Payload for IPv4 (ESP)
/51	Authentication Header for IPv4 (AH)
/89	OSPF routing protocol
/103	Multicast | IGMP

/112 VRRP

[Top | Back to Content]

Manual:IP/Settings

Applies to RouterOS: v6+

Summary

Sub-menu: `/ip settings`

IP Settings allows to configure several IP related kernel parameters.

Properties

Property	Description
accept-redirects (*yes* | *no*; Default: **no**)	Whether to accept ICMP redirect messages. Typically should be enabled on host and disabled on routers.
accept-source-route (*yes* | *no*; Default: **no**)	Whether to accept packets with SRR option. Typically should be enabled on router.
ip-forwarding (*yes* | *no*; Default: **yes**)	Emable/disable packet forwarding between interfaces. Resets all configuration parameters to defaults according to RFC1812 for routers.
rp_filter (*loose* | *no* | *strict*; Default: **no**)	Disables enables source validation. • no - No source validation. • strict - Strict mode as defined in RFC3704 Strict Reverse Path. Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded. • loose - Loose mode as defined in RFC3704 Loose Reverse Path. Each incoming packet's source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail. Current recommended practice in RFC3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.
secure-redirects (*yes* | *no*; Default: **yes**)	Accept ICMP redirect messages only for gateways, listed in default gateway list.
send-redirects (*yes* | *no*; Default: **yes**)	Whether to send ICMP redirects. Recommended to be enabled on routers.
tcp_syncookies (*yes* | *no*; Default: **no**)	Send out syncookies when the syn backlog queue of a socket overflows. This is to prevent against the common 'SYN flood attack'. syncookies seriously violate TCP protocol, do not allow o use TCP extensions, can result in serious degradation of some services (f.e. SMTP relaying), visible not by you, but your clients and relays, contacting you.

[Top | Back to Content]

Manual:Scripting

Applies to RouterOS: v3, v4

Scripting language manual

This manual provides introduction to RouterOS built-in powerful scripting language.
Scripting host provides a way to automate some router maintenance tasks by means of executing user-defined scripts bounded to some event occurrence.
Scripts can be stored in Script repository or can be written directly to console. The events used to trigger script execution include, but are not limited to the System Scheduler, the Traffic Monitoring Tool, and the Netwatch Tool generated events.

Line structure

RouterOS script is divided into number of command lines. Command lines are executed one by one until the end of script or until runtime error occur.

Command line

RouterOS console uses following command syntax:
`[prefix] [path] command [uparam] [param=[value]] .. [param=[value]]`

- [prefix] - ":" or "/" character which indicates if command is ICE or path. May or may not be required.
- [path] - relative path to the desired menu level. May or may not be required.
- command - one of the commands available at the specified menu level.
- [uparam] - unnamed parameter, must be specified if command requires it.
- [params] - sequence of named parameters followed by respective values

The end of command line is represented by the token *";"* or *NEWLINE*. Sometimes *";"* or *NEWLINE* is not required to end the command line.
Single command inside `(), [] or {}` does not require any end of command character. End of command is determined by content of whole script

```
:if ( true ) do={ :put "lala" }
```

Each command line inside another command line starts and ends with square brackets "[ ]" (command concatenation).

```
:put [/ip route get [find gateway=1.1.1.1]];
```

Notice that code above contains three command lines:

- :put
- /ip route get
- find gateway=1.1.1.1

Command line can be constructed from more than one physical line by following line joining rules.

Physical Line

A physical line is a sequence of characters terminated by an end-of-line (EOL) sequence. Any of the standard platform line termination sequences can be used:

- **unix** – ASCII LF;
- **windows** – ASCII CR LF;
- **mac** – ASCII CR;

Standard C conventions for new line characters can be used (the \n character).

Comments

A comment starts with a hash character (#) and ends at the end of the physical line. Whitespace or any other symbols are not allowed before hash symbol. Comments are ignored by syntax. If (#) character appear inside string it is not considered a comment.

```
# this is a comment
 # bad comment
:global a; # bad comment

:global myStr "lala # this is not a comment"
```

Line joining

Two or more physical lines may be joined into logical lines using backslash character (\). A line ending in a backslash cannot carry a comment. A backslash does not continue a comment. A backslash does not continue a token except for string literals. A backslash is illegal elsewhere on a line outside a string literal.

```
:if ($a = true \
      and $b=false) do={ :put “$a $b”; }

:if ($a = true \      # bad comment
      and $b=false) do={ :put “$a $b”; }

# comment \
    continued – invalid  (syntax error)
```

Whitespace between tokens

Whitespace can be used to separate tokens. Whitespace is necessary between two tokens only if their concatenation could be interpreted as a different token. Example:

```
{
   :local a true; :local b false;
# whitespace is not required
   :put (a&&b);
# whitespace is required
   :put (a and b);
}
```

Whitespace are not allowed

- between '<parameter>='
- between 'from=' 'to=' 'step=' 'in=' 'do=' 'else='

Example:

```
#incorrect:
:for i from = 1 to = 2 do = { :put $i }
#correct syntax:
:for i from=1 to=2 do={ :put $i }
:for i from= 1 to= 2 do={ :put $i }

#incorrect
/ip route add gateway = 3.3.3.3
#correct
/ip route add gateway=3.3.3.3
```

Scopes

Variables can be used only in certain regions of the script. These regions are called scopes. Scope determines visibility of the variable. There are two types of scopes - `global` and `local`. A variable declared within a block is accessible only within that block and blocks enclosed by it, and only after the point of declaration.

Global scope

Global scope or root scope is default scope of the script. It is created automatically and can not be turned off.

Local scope

User can define its own groups to block access to certain variables, these scopes are called local scopes. Each local scope is enclosed in curly braces ("{ }").

```
{
   :local a 3;
   {
      :local b 4;
      :put ($a+$b);
   }
#line below will generate error
   :put ($a+$b);
}
```

In code above variable `b` has local scope and will not be accessible after closed curly brace.

Note: Each line written in terminal is treated as local scope

So for example, defined local variable will not be visible in next command line and will generate syntax error

```
[admin@MikroTik] > :local myVar a;
[admin@MikroTik] > :put $myVar
syntax error (line 1 column 7)
```

Warning: Do not define global variables inside local scopes.

Note that even variable can be defined as global, it will be available only from its scope unless it is not already defined.

```
{
   :local a 3;
   {
       :global b 4;
   }
   :put ($a+$b);
}
```

Code above will generate an error.

Keywords

The following words are keywords and cannot be used as variable and function names:

```
and       or        not       in
```

Delimiters

The following tokens serve as delimiters in the grammar:

```
()  []  {}  :  ;  $  /
```

Data types

RouterOS scripting language has following data types:

Type	Description
number	- 64bit signed integer, possible hexadecimal input;
boolean	- values can bee `true` or `false`;
string	- character sequence;
IP	- IP address;
internal ID	- hexadecimal value prefixed by '*' sign. Each menu item has assigned unique number - internal ID;
time	- date and time value;
array	- sequence of values organized in an array;
nil	- default variable type if no value is assigned;

Constant Escape Sequences

Following escape sequences can be used to define certain special character within string:

\"	Insert double quote
****	Insert backslash
\n	Insert newline
\r	Insert carriage return
\t	Insert horizontal tab
\$	Output $ character. Otherwise $ is used to link variable.
\?	Output ? character. Otherwise ? is used to print "help" in console.
_	- space
\a	- BEL (0x07)
\b	- backspace (0x08)
\f	- form feed (0xFF)
\v	Insert vertical tab
\xx	Print character from hex value. Hex number should use capital letters.

```
:put "\48\45\4C\4C\4F\r\nThis\r\nis\r\na\r\ntest";
```

which will show on display

```
HELLO
This
is
a
test
```

Operators

Arithmetic Operators

Usual arithmetic operators are supported in RouterOS scripting language

Opearator	Description	Example
"+"	binary addition	`:put (3+4);`
"-"	binary subtraction	`:put (1-6);`
"*"	binary multiplication	`:put (4*5);`
"/"	binary division	`:put (10 / 2); :put ((10)/2)`
"-"	unary negation	`{ :local a 1; :put (-a); }`

Note: for division to work you have to use braces or spaces around dividend so it is not mistaken as IP address

Relational Operators

Opearator	Description	Example
"<"	less	`:put (3<4);`
">"	greater	`:put (3>4);`
"="	equal	`:put (2=2);`
"<="	less or equal	
">="	greater or equal	
"!="	not equal	

Logical Operators

Opearator	Description	Example
"!", "not"	logical NOT	`:put (!true);`
"&&", "and"	logical AND	`:put (true&&true)`
"||", "or"	logical OR	`:put (true||false);`
"in"		`:put (1.1.1.1/32 in 1.0.0.0/8);`

Bitwise Operators

Bitwise operators are working on number and ip address data types.

Opearator	Description	Example
"~"	bit inversion	`:put (~0.0.0.0)`
"|"	bitwise OR. Performs logical OR operation on each pair of corresponding bits. In each pair the result is "1" if one of bits or both bits are "1", otherwise the result is "0".	
"^"	bitwise XOR. The same as OR, but the result in each position is "1" if two bits are not equal, and "0" if bits are equal.	
"&"	bitwise AND. In each pair the result is "1" if first and second bit is "1". Otherwise the result is "0".	
"<<"	left shift by given amount of bits	
">>"	right shift by given amount of bits	

Concatenation Operators

Opearator	Description	Example
“.”	concatenates two strings	`:put (“concatenate” . “ “ . “string”);`
“,”	concatenates two arrays or adds element to array	`:put ({1;2;3} , 5 );`

It is possible to add variable values to strings without concatenation operator:

```
:global myVar "world";

:put ("Hello " . $myVar);
# next line does the same as above
:put "Hello $myVar";
```

By using $[] and $() in string it is possible to add expressions inside strings:

```
:local a 5;
:local b 6;
:put " 5x6 = $($a * $b)";

:put " We have $[ :len [/ip route find] ] routes";
```

Other Operators

Opearator	Description	Example
“`[]`”	command substitution. Can contain only single command line	`:put [ :len "my test string"; ];`
“`()`”	sub expression or grouping operator	`:put ( "value is " . (4+5));`
“`$`”	substitution operator	`:global a 5; :put $a;`
“`~`”	binary operator that matches value against POSIX extended regular expression	Print all routes which gateway ends with 202 `/ip route print where gateway~"^[0-9\\.]*202"`

Variables

Scripting language has two types of variables:

- `global` - accessible from all scripts created by current user, defined by global keyword;
- `local` - accessible only within the current scope, defined by local keyword.

Every variable, except for built in RouterOS variables, must be declared before usage by local or global keywords. Undefined variables will be marked as undefined and will result in compilation error. Example:

```
# following code will result in compilation error, because myVar is used without declaration
:set myVar "my value";
:put $myVar
```

Correct code:

```
:local myVar;
:set myVar "my value";
:put $myVar;
```

Valid characters in variable names are letters and digits. If variable name contains any other character, then variable name should be put in double quotes. Example:

```
#valid variable name
:local myVar;
#invalid variable name
:local my-var;
#valid because double quoted
:global "my-var";
```

If variable is initially defined without value then variable data type is set to *nil*, otherwise data type is determined automatically by scripting engine. Sometimes conversion from one data type to another is required. It can be achieved using data conversion commands. Example:

```
#convert string to array
:local myStr "1,2,3,4,5";
:put [:typeof $myStr];
:local myArr [:toarray $myStr];
:put [:typeof $myArr]
```

Variable names are case sensitive.

```
:local myVar "hello"
# following line will generate error, because variable myVAr is not defined
:put $myVAr
# correct code
:put $myVar
```

Commands

Every global command should start with *":"* token, otherwise it will be treated as variable.

Command	Syntax	Description	Example
/		go to root menu	
..		go back by one menu level	
?		list all available menu commands and brief descriptions	
global	`:global <var> [<value>]`	define global variable	`:global myVar "something"; :put $myVar;`
local	`:local <var> [<value>]`	define local variable	`{ :local myLocalVar "I am local"; :put $myVar; }`
beep	`:beep <freq> <length>`	beep built in speaker	
delay	`:delay <time>`	do nothing for a given period of time	
put	`:put <expression>`	put supplied argument to console	
len	`:len <expression>`	return string length or array element count	`:put [:len "length=8"];`
typeof	`:typeof <var>`	return data type of variable	`:put [:typeof 4];`
pick	`:pick <var> <start>[<end>]`	return range of elements or substring. If end position is not specified, will return only one element from an array.	`:put [:pick "abcde" 1 3]`
log	`:log <topic> <message>`	write message to system log. Available topics are "debug, error, info and warning"	`:log info "Hello from script";`

time	:time <expression>	return interval of time needed to execute command	:put [:time {:for i from=1 to=10 do={ :delay 100ms }}];
set	:set <var> [<value>]	assign value to declared variable.	:global a; :set a true;
find	:find <arg> <arg> <start>	return position of substring or array element	:put [:find "abc" "a" -1];
environment	:environment print <start>	print initialized variable information	:global myVar true; :environment print;
terminal		terminal related commands	
error	:error <output>	Generate console error and stop executing the script	
parse	:parse <expression>	parse string and return parsed console commands. Can be used as function.	:global myFunc [:parse ":put hello!"]; $myFunc;
resolve	:resolve <arg>	return IP address of given DNS name	:put [:resolve "www.mikrotik.com"];
toarray	:toarray <var>	convert variable to array	
tobool	:tobool <var>	convert variable to boolean	
toid	:toid <var>	convert variable to internal ID	
toip	:toip <var>	convert variable to IP address	
toip6	:toip6 <var>	convert variable to IPv6 address	
tonum	:tonum <var>	convert variable to integer	
tostr	:tostr <var>	convert variable to string	
totime	:totime <var>	convert variable to time	

Menu specific commands

Following commands available from most sub-menus:

Command	Syntax	Description
add	add <param>=<value>..<param>=<value>	add new item
remove	remove <id>	remove selected item
enable	enable <id>	enable selected item
disable	disable <id>	disable selected item
set	set <id> <param>=<value>..<param>=<value>	change selected items parameter, more than one parameter can be specified at the time
get	get <id> <param>=<value>	get selected items parameter value
print	print <param><param>=[<value>]	print menu items. Output depends on print parameters specified. Most common print parameters are described here
export	export [file=<value>]	export configuration from current menu and its sub-menus (if present). If file parameter is specified output will be written to file with extension '.rsc', otherwise output will be printed to console. Exported commands can be imported by import command
edit	edit <id> <param>	edit selected items property in built-in text editor
find	find <expression>	find items by given expression.

import

Import command is available from root menu and is used to import configuration from files created by export command or written manually by hand.

print parameters

Several parameters are available for print command:

Parameter	Description	Example
append		
as-value	print output as array of parameters and its values	`:put [/ip address print as-value]`
brief	print brief description	
detail	print detailed description, output is not as readable as brief output, but may be useful to view all parameters	
count-only	print only count of menu items	
file	print output to file	
follow	print all current entries and track new entries until ctrl-c is pressed, very useful when viewing log entries	`/log print follow`
follow-only	print and track only new entries until ctrl-c is pressed, very useful when viewing log entries	`/log print follow-only`
from	print parameters only from specified item	`/user print from=admin`
interval	continuously print output in selected time interval, useful to track down changes where `follow` is not acceptable	`/interface print interval=2`
terse	show details in compact and machine friendly format	
value-list	show values one per line (good for parsing purposes)	
without-paging	If output do not fit in console screen then do not stop, print all information in one piece	
where	expressions followed by where parameter can be used to filter out matched entries	`/ip route print where interface="ether1"`

More than one parameter can be specified at a time, for example, `/ip route print count-only interval=1 where interface="ether1"`

Loops and conditional statements

Command	Syntax	Description
do..while	`:do { <commands> } while=( <conditions> ); :while ( <conditions> ) do={ <commands> };`	execute commands until given condition is met.
for	`:for <var> from=<int> to=<int> step=<int> do={ <commands> }`	execute commands over a given number of iterations
foreach	`:foreach <var> in=<array> do={ <commands> };`	execute commands for each elements in list

Command	Syntax	Description
if	`:if(<condition>) do={<commands>} else={<commands>} <expression>`	If a given condition is `true` then execute commands in the `do` block, otherwise execute commands in the `else` block if specified.

Example:

```
{
   :local myBool true;
   :if ($myBool = false) do={ :put "value is false" } else={ :put "value is true" }
}
```

Script repository

Sub-menu level: `/system script`

Contains all user created scripts. Scripts can be executed in several different ways:

- **on event** - scripts are executed automatically on some facility events (scheduler, netwatch, VRRP)
- **by another script** - running script within script is allowed
- **manually** - from console executing run command or in winbox

Property	Description
name (*string*; Default: *"Script[num]"*)	name of the script
policy (*string*; Default:)	list of applicable policies: • **ftp** - user can log on remotely via ftp and send and retrieve files from the router • **local** - user can log on locally via console • **policy** - manage user policies, add and remove user • **read** - user can retrieve the configuration • **reboot** - user can reboot the router • **ssh** - user can log on remotely via secure shell • **telnet** - user can log on remotely via telnet • **test** - user can run ping, traceroute, bandwidth test • **web** - user can log on remotely via http • **write** - user can retrieve and change the configuration
source (*string*;)	Script source code

Read only status properties:

Property	Description
last-started (*date*)	Date and time when the script was last invoked.
owner (*string*)	User who created the script
run-count (*integer*)	Counter that counts how many times script has been executed

Menu specific commands

Command	Description
run (*run [id|name]*)	Execute specified script by ID or name

Environment

Sub-menu level:

- `/system script environment`
- `/environment`

Contains all user defined variables and their assigned values.

```
[admin@MikroTik] > :global example;
[admin@MikroTik] > :set example 123
[admin@MikroTik] > /environment print
"example"=123
```

Read only status properties:

Property	Description
name (*string*)	Variable name
user (*string*)	User who defined variable
value ()	Value assigned to variable

Job

Sub-menu level: `/system script job`

Contains list of all currently running scripts.
Read only status properties:

Property	Description
owner (*string*)	User who is running script
policy (*array*)	List of all policies applied to script
started (*date*)	Local date and time when script was started

Manual:Scripting-examples

CMD Scripting examples

This section contains some useful scripts and shows all available scripting features. Script examples used in this section were tested with the latest 3.x version.

Create a file

In v3.x it is not possible to create file directly, however there is a workaround

```
/file print file=myFile
/file set myFile.txt contents=""
```

Check if IP on interface have changed

Sometimes provider gives dynamic IP addresses. This script will compare if dynamic IP address is changed.

```
:global currentIP;

:local newIP [/ip address get [find interface="ether1"] address];

:if ($newIP != $currentIP) do={
    :put "ip address $currentIP changed to $newIP";
    :set currentIP $newIP;
}
```

Strip netmask

This script is useful if you need ip address without netmask (for example to use it in firewall), but "`/ip address get [id] address`" returns ip address and netmask.

Code:

```
:global ipaddress 10.1.101.1/24

:for i from=( [:len $ipaddress] - 1) to=0 do={
      :if ( [:pick $ipaddress $i] = "/") do={
            :put [:pick $ipaddress 0 $i]
      }
}
```

Resolve host-name

Many users are asking feature to use dns names instead of IP address for radius servers, firewall rules, etc.

So here is an example how to resolve RADIUS server's IP.

Lets say we have radius server configured:

```
/radius
add address=3.4.5.6 comment=myRad
```

And here is a script that will resolve ip address, compare resolved ip with configured one and replace if not equal:

```
/system script add name="resolver" source= {

:local resolvedIP [:resolve "server.example.com"];
:local radiusID [/radius find comment="myRad"];
:local currentIP [/radius get $radiusID address];

:if ($resolvedIP != $currentIP) do={
   /radius set $radiusID address=$resolvedIP;
   /log info "radius ip updated";
}

}
```

Add this script to scheduler to run for example every 5 minutes

```
/system scheduler add name=resolveRadiusIP on-event="resolver" interval=5m
```

Write simple queue stats in multiple files

Lets consider queue namings are "some text.1" so we can search queues by last number right after the dot.

```
:local entriesPerFile 10;
:local currentQueue 0;
:local queuesInFile 0;
:local fileContent "";
#determine needed file count
:local numQueues [/queue simple print count-only] ;
:local fileCount ($numQueues / $entriesPerFile);
:if ( ($fileCount * $entriesPerFile) != $numQueues) do={
   :set fileCount ($fileCount + 1);
}

#remove old files
/file remove [find name~"stats"];

:put "fileCount=$fileCount";

:for i from=1 to=$fileCount do={
#create file
   /file print file="stats$i.txt";
#clear content
   /file set [find name="stats$i.txt"] contents="";

   :while ($queuesInFile < $entriesPerFile) do={
     :if ($currentQueue < $numQueues) do={
         :set currentQueue ($currentQueue +1);
         :put $currentQueue ;
         /queue simple
         :local internalID [find name~"\\.$currentQueue\$"];
         :put "internalID=$internalID";
```

```
          :set fileContent ($fileContent . [get $internalID target-address] . \
            " " . [get $internalID total-bytes] . "\r\n");
      }
      :set queuesInFile ($queuesInFile +1);

   }
   /file set "stats$i.txt" contents=$fileContent;
   :set fileContent "";
   :set queuesInFile 0;

}
```

Generate backup and send it by e-mail

This script generates backup file and sends it to specified e-mail address. Mail subject contains router's name, current date and time.

Note that smtp server must be configured before this script can be used. See `/tool e-mail` for configuration options.

Script:

```
/system backup save name=email_backup
/tool e-mail send file=email_backup.backup to="me@test.com" body="See attached file" \
   subject="$[/system identity get name] $[/system clock get time] $[/system clock get date] Backup")
```

Note: backup file contains sensitive information like passwords. So to get access to generated backup file, script or scheduler must have **'sensitive'** policy.

Use string as function

Code:

```
:global printA [:parse ":local A; :put \$A;" ];
$printA
```

Check bandwidth and add limitations

This script checks if download on interface is more than 512kbps, if true then queue is added to limit speed to 256kbps.

Code:

```
:foreach i in=[/interface find] do={
    /interface monitor-traffic $i once do={
        :if ($"received-bits-per-second" > 0 ) do={
            :local tmpIP [/ip address get [/ip address find interface=$i] address] ;
#            :log warning $tmpIP ;
            :for j from=( [:len $tmpIP] - 1) to=0 do={
                :if ( [:pick $tmpIP $j] = "/") do={
                    /queue simple add name=$i max-limit=256000/256000 dst-address=[:pick $tmpIP 0 $j] ;
                }
            }
        }
```

```
    }
}
```

Block access to specific websites

This script is useful if you want to block certain web sites but you don't want to use web proxy.

This example looks entries "rapidshare" and "youtube" in dns cache and adds IPs to address list named "restricted".

Before you begin, you must set up router to catch all dns requests:

```
/ip firewall nat
add action=redirect chain=dstnat comment=DNS dst-port=53 protocol=tcp to-ports=53
add action=redirect chain=dstnat dst-port=53 protocol=udp to-ports=53
```

and add firewall

```
/ip firewall filter
add chain=forward dst-address-list=restricted action=drop
```

Now we can write a script and schedule it to run, lets say, every 30 seconds.

Script Code:

```
:foreach i in=[/ip dns cache find] do={
    :local bNew "true";
    :local cacheName [/ip dns cache all get $i name] ;
#    :put $cacheName;

    :if (([:find $cacheName "rapidshare"] != 0) || ([:find $cacheName "youtube"] != 0)) do={

        :local tmpAddress [/ip dns cache get $i address] ;
#      :put $tmpAddress;

# if address list is empty do not check
        :if ( [/ip firewall address-list find ] = "") do={
            :log info ("added entry: $[/ip dns cache get $i name] IP $tmpAddress");
            /ip firewall address-list add address=$tmpAddress list=restricted comment=$cacheName;
        } else={
            :foreach j in=[/ip firewall address-list find ] do={
                :if ( [/ip firewall address-list get $j address] = $tmpAddress ) do={
                    :set bNew "false";
                }
            }
            :if ( $bNew = "true" ) do={
                :log info ("added entry: $[/ip dns cache get $i name] IP $tmpAddress");
                /ip firewall address-list add address=$tmpAddress list=restricted comment=$cacheName;
            }
        }
    }
}
```

Parse file to add ppp secrets

This script requires that entries inside the file is in following format:

username,password,local_address,remote_address,profile,service

For example:

```
janis,123,1.1.1.1,2.2.2.1,ppp_profile,myService
juris,456,1.1.1.1,2.2.2.2,ppp_profile,myService
aija,678,1.1.1.1,2.2.2.3,ppp_profile,myService
```

Code:

```
:global content [/file get [/file find name=test.txt] contents] ;
:global contentLen [ :len $content ] ;

:global lineEnd 0;
:global line "";
:global lastEnd 0;



:do {
       :set lineEnd [:find $content "\r\n" $lastEnd ] ;
       :set line [:pick $content $lastEnd $lineEnd] ;
       :set lastEnd ( $lineEnd + 2 ) ;

       :local tmpArray [:toarray $line] ;
      :if ( [:pick $tmpArray 0] != "" ) do={
      :put $tmpArray;
         /ppp secret add name=[:pick $tmpArray 0] password=[:pick $tmpArray 1] \
             local-address=[:pick $tmpArray 2] remote-address=[:pick $tmpArray 3] \
             profile=[:pick $tmpArray 4] service=[:pick $tmpArray 5];
}
} while ($lineEnd < $contentLen)
```

Detect new log entry

This script is checking if new log entry is added to particular buffer.

In this example we will use pppoe logs:

```
/system logging action
add name="pppoe"
/system logging
add action=pppoe topics=pppoe,info,!ppp,!debug
```

Log buffer will look similar to this one:

```
[admin@mainGW] > /log print where buffer=pppoe
13:11:08 pppoe,info PPPoE connection established from 00:0C:42:04:4C:EE
```

Now we can write a script to detect if new entry is added.

Code:

```
:global lastTime;

:global currentBuf [ :toarray [ /log find buffer=pppoe  ] ] ;
:global currentLineCount [ :len $currentBuf ] ;
:global currentTime [ :totime [/log get [ :pick $currentBuf ($currentLineCount -1) ] time   ] ];

:global message "";

:if ( $lastTime = "" ) do={
      :set lastTime $currentTime ;
      :set message [/log get [ :pick $currentBuf ($currentLineCount-1) ] message];

} else={
      :if ( $lastTime < $currentTime ) do={
            :set lastTime $currentTime ;
            :set message [/log get [ :pick $currentBuf ($currentLineCount-1) ] message];
      }
}
```

After new entry is detected, it is saved in "message" variable, which you can use later to parse log message, for example, to get pppoe clients mac address.

Allow use of ntp.org pool service for NTP

This script resolves the hostnames of two NTP servers, compares the result with the current NTP settings and changes the addresses if they're different. This script is required as RouterOS does not allow hostnames to be used in the NTP configuration. Two scripts are used. The first defines some system variables which are used in other scripts and the second does the grunt work:

```
# System configuration script - "GlobalVars"

:put "Setting system globals";

# System name
:global SYSname [/system identity get name];

# E-mail address to send notifications to
:global SYSsendemail "mail@my.address";

# E-mail address to send notifications from
:global SYSmyemail "routeros@my.address";

# Mail server to use
:global SYSemailserver "1.2.3.4";

# NTP pools to use (check www.pool.ntp.org)
:global SYSntpa "0.uk.pool.ntp.org";
:global SYSntpb "1.uk.pool.ntp.org";
```

```
# Check and set NTP servers - "setntppool"

# We need to use the following globals which must be defined here even
# though they are also defined in the script we call to set them.
:global SYSname;
:global SYSsendemail;
:global SYSmyemail;
:global SYSmyname;
:global SYSemailserver;
:global SYSntpa;
:global SYSntpb;

# Load the global variables with the system defaults
/system script run GlobalVars

# Resolve the two ntp pool hostnames
:local ntpipa [:resolve $SYSntpa];
:local ntpipb [:resolve $SYSntpb];

# Get the current settings
:local ntpcura [/system ntp client get primary-ntp];
:local ntpcurb [/system ntp client get secondary-ntp];

# Define a variable so we know if anything's changed.
:local changea 0;
:local changeb 0;

# Debug output
:put ("Old: " . $ntpcura . " New: " . $ntpipa);
:put ("Old: " . $ntpcurb . " New: " . $ntpipb);

# Change primary if required
:if ($ntpipa != $ntpcura) do={
    :put "Changing primary NTP";
    /system ntp client set primary-ntp="$ntpipa";
    :set changea 1;
    }

# Change secondary if required
:if ($ntpipb != $ntpcurb) do={
    :put "Changing secondary NTP";
    /system ntp client set secondary-ntp="$ntpipb";
    :set changeb 1;
    }

# If we've made a change, send an e-mail to say so.
:if (($changea = 1) || ($changeb = 1)) do={
```

```
    :put "Sending e-mail.";
    /tool e-mail send \
        to=$SYSsendemail \
        subject=($SYSname . " NTP change") \
        from=$SYSmyemail \
        server=$SYSemailserver \
        body=("Your NTP servers have just been changed:\n\nPrimary:\nOld: " . $ntpcura . "\nNew: " \
          . $ntpipa . "\n\nSecondary\nOld: " . $ntpcurb . "\nNew: " . $ntpipb);
    }
```

Scheduler entry:

```
/system scheduler add \
  comment="Check and set NTP servers" \
  disabled=no \
  interval=12h \
  name=CheckNTPServers \
  on-event=setntppool \
  policy=read,write,test \
  start-date=jan/01/1970 \
  start-time=16:00:00
```

Auto upgrade script

- Auto_upgrade_script_V3.x

Other scripts known to work with latest v3.x

- Dynamic_DNS_Update_Script_for_EveryDNS
- Dynamic_DNS_Update_Script_for_ChangeIP.com
- UPS Script

LUA Scripting examples

NOTE!

After RouterOS v4.0beta4, Lua support is removed until further notice

In v4.0beta3 Lua scripting language is integrated in console. This integration allows users to create their own functions and bypass several command line scripting limitations.
All examples below require at least basic knowledge of Lua scripting language. Good tutorials can be found here [1] as a starting point.

Print function

As stated in Lua documentation, 'print' command is not available in RouterOS compared to standard Lua release. This example will show you how to get back 'print' command

```
-- ----------------------------------------------------------------------
-- Print function
-- ----------------------------------------------------------------------
function print (...)
   local strPrintResult = ""
```

```
   if ... then
      local targs = {...}
      for i,v in ipairs(targs) do
        strPrintResult = strPrintResult .. tostring(v) .. "  "
      end
      strPrintResult = strPrintResult .. "\r\n"
      io.write(strPrintResult)
   end
end
```

Now you can include this custom function to other scripts and use this cool custom print function :)
You can also modify this function to write messages in RouterOS log.

Read and write large files

Many users requested ability to work with files. Now you can do it without limitations.
Create and write to file:

```
:global newContent "new file content\r\nanother line\r\n";
[/lua "local f=assert(io.open('/test.txt', 'w+')); f:write(newContent); f:close()" ];
```

Read file content to variable:

```
:global cnt ""
[/lua "local f=assert(io.open('/test.txt', 'r')); cnt=f:read('*all'); f:close()" ];
:put $cnt
```

Include custom function in another script

This example will show where to store and how to include your cool custom created functions into another scripts

- In router's file root directory create subdirectory named 'lua'
- On your PC create new file named customprint.lua and write this function in it.
- Upload newly created file in router's 'lua' directory that we made in first step
- Now you can test your custom lua function

```
[:lua "require 'customprint'\n print('hello from custom print function')"]
```

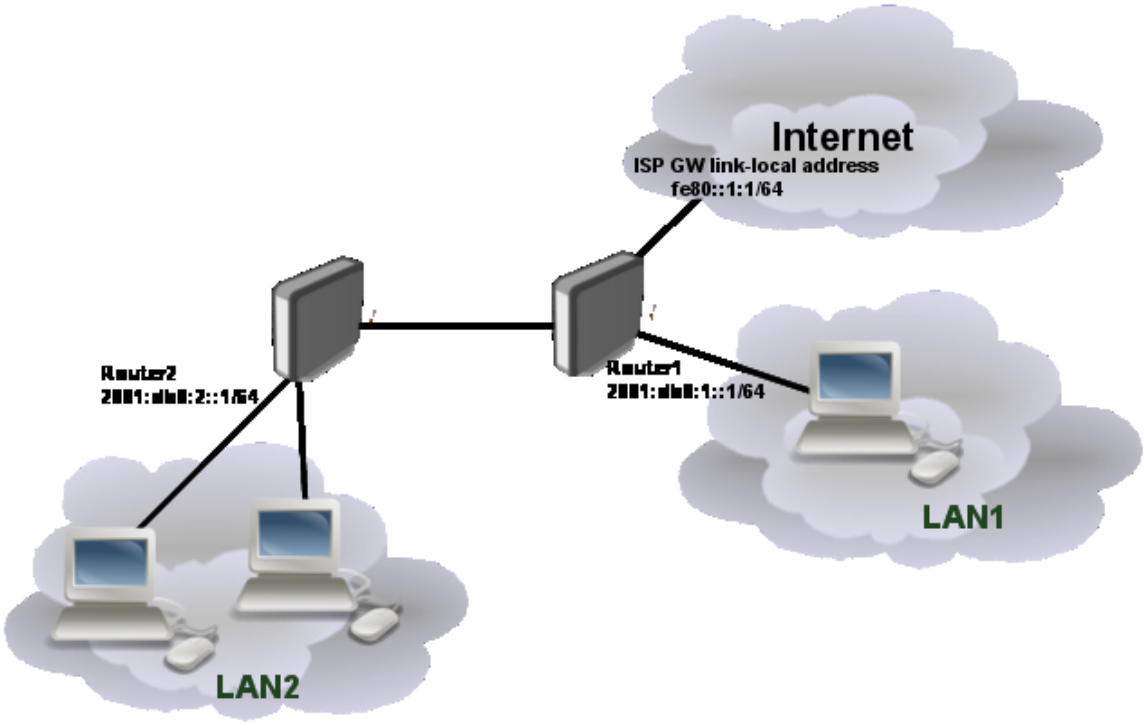
References

[1] http://lua-users.org/wiki/TutorialDirectory

Manual:Simple Static IPv6 Routing

Introduction

Lets make a simple routing setup illustrated in image below



Lets consider ISP is giving us prefix 2001:db8::/62 and prefix is routed to us with link-local address (fe80::1:1).

Ether1 of Router1 is connected to ISP and will be the gateway of our networks. Router2 is connected to ether2 of Router1 and will act as a gateway for clients connected to it from LAN2. Router1 also connects one client to ether3. Our goal is to create setup so that clients from LAN1 can reach clients from LAN2 and all of them can connect to the internet.

Configuration

At first we need to find what link-local addresses are on Router1 and on Router's 2 ether1 for routing. We can do IPv6 routing without globally configuring addresses on every link that way addresses are not wasted. In current setup there is no global addresses even between ISP and our gateway.

```
[admin@R1] /ipv6 address> print
Flags: X - disabled, I - invalid, D - dynamic, G - global, L - link-local
 #    ADDRESS                                     FROM-POOL INTERFACE     ADVERTISE
 0 DL fe80::219:d1ff:fe00:3511/64                           ether1        no
 1 DL fe80::219:d1ff:fe00:3512/64                           ether2        no
 1 DL fe80::219:d1ff:fe00:3513/64                           ether3        no
```

```
[admin@R2] /ipv6 address> print
Flags: X - disabled, I - invalid, D - dynamic, G - global, L - link-local
```

```
 #    ADDRESS                                          FROM-POOL INTERFACE     ADVERTISE
 0 DL fe80::219:d1ff:fe39:3535/64                                 ether1        no
 1 DL fe80::219:d1ff:fe39:3536/64                                 ether2        no
```

Now we can start configuration.

Router1

```
/ipv6 address
add address=2001:db8:1::1/64 interface=ether3 advertise=yes

/ipv6 route
add gateway=fe80::1:1%ether1
add dst-address=2001:db8:2::/64 gateway=fe80::219:d1ff:fe39:3535%ether2
```

Router2

```
/ipv6 address
add address=2001:db8:2::1/64 interface=ether2 advertise=yes

/ipv6 route
add gateway=fe80::219:d1ff:fe00:3512%ether1
```

Notice how link local addresses are configured as gateways. We provide directly connected neighbour routers link-local address and explicitly specify on which interface ll address is reachable.

Added global addresses are with advertise flag meaning that RA will be used to automatically configure IPv6 addressing on the client PCs. Read more>>

That is all required configuration. At this point all clients are directly reachable from remote locations.

Note: Since IPv6 does not have NAT all clients have direct connection to the Internet. IPv6 firewall rules are required to protect the clients from unwanted access or attacks

See Also

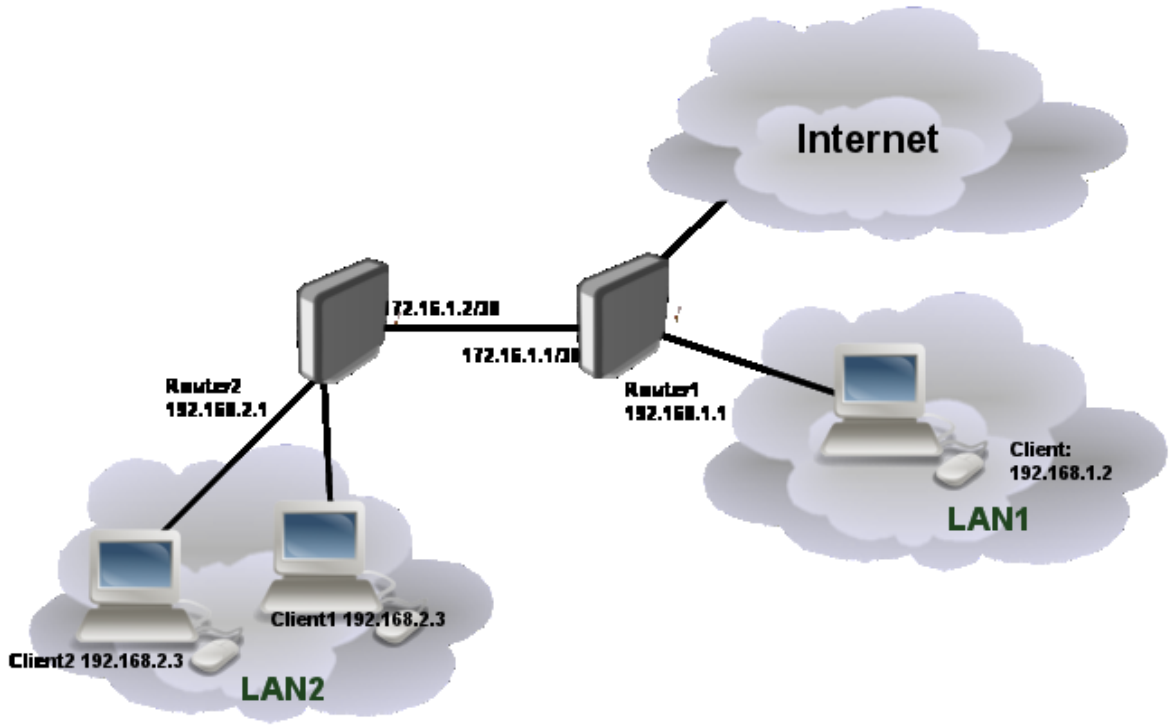
- IPv6 routing example with tunnel broker

[Top | Back to Content]

Manual:Simple Static Routing

Introduction

Lets make a simple routing setup illustrated in image below



Ether1 of Router1 is connected to ISP and will be the gateway of our networks. Router2 is connected to ether2 of Router1 and will act as a gateway for clients connected to it from LAN2. Router1 also connects one client to ether3. Our goal is to create setup so that clients from LAN1 can reach clients from LAN2 and all of them can connect to internet.

Configuration

Lets consider that ISP gave us an address 10.1.1.2/30 and gateway is 10.1.1.1 Router1:

```
/ip address
add address=10.1.1.2 interface=ether1
add address=172.16.1.1/30 interface=ether2
add address=192.168.1.1/24 interface=ether3

/ip route
add gateway=10.1.1.1
add dst-address=192.168.2.0/24 gateway=172.16.1.2
```

Router2:

```
/ip address
add address=172.16.1.2/30 interface=ether1
```

```
add address=192.168.2.1/24 interface=ether2

/ip route
add gateway=172.16.1.1
```

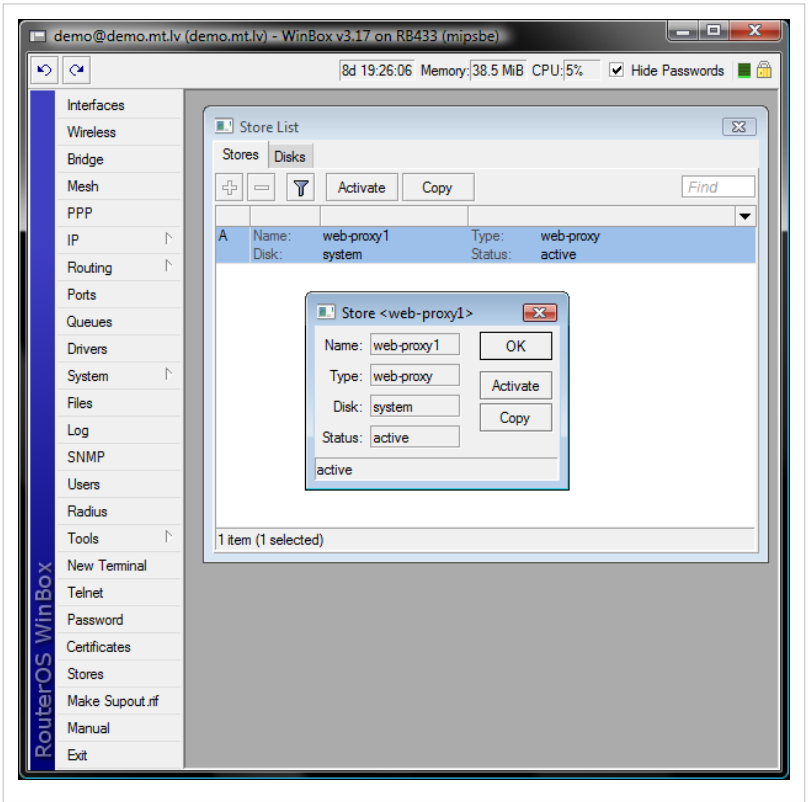
If you look at configuration then you will see that on Router1 we added route to destination 182.168.2.0/24. It is required for clients from LAN1 to be able to reach clients on LAN2. On Router2 such route is not required since LAN1 can be reached by default route.

[Top | Back to Content]

Manual:Store

Applies to RouterOS: v3, v4+

Store manages storage devices used by RouterOS various facilities.



Currently Store can be used for:

- **Webproxy**
- **User Manager**
- **the Dude**

This is especially useful for RouterBOARD devices with **SD/CF** slots - as the built-in storage is quite small, an external drive comes in very handy when you want a big User Manager database.

You can add as many external or secondary drives as you want, and select any number of them for each of the mentioned feature use. For example User Manager could be used on 3 disks, one of them would be the active database, and the rest would be backups. You can then add a fourth disk, copy the active data to it - unplug it - and move to another server, to keep using the actual database.

This means migration and backup made easy!

Note: For proper operation router needs to be rebooted after adding or removing external storage.

Creating a Store instance

```
[normis@demo.mt.lv] /store> print
Flags: X - disabled, A - active
 #   NAME                      TYPE                    DISK                    STATUS
 0 A web-proxy1                web-proxy               system                  active
[normis@demo.mt.lv] /store> add
 activate  comment  copy-from  disabled  name  disk  type

[normis@demo.mt.lv] /store> add name=webproxy_backup disk=disk1 type=web-proxy activate=no
```

This will add a new storage place for Webproxy on disk1, and will set it as inactive.

Activate new store instance to save proxy cache on secondary disk (other proxy settings configured separately from **/ip proxy menu**),

```
[normis@demo.mt.lv] > store activate webproxy-backup
```

E.g. RB1000 with populated CF Card slot and User Manager, one can add the CF card for use by User manager to store all it's data as follows

```
/store add disk=CF1 type=user-manager activate=yes
```

Store management

Sub-menu: `/store`

Properties

Property	Description
activate (*yes* | *no*; Default: **no**)	Whether to activate this store as primary.
comment (*string*; Default:)	Short description of an item.
disabled (*yes* | *no*; Default: **no**)	Whether to disable store.
disk (*string*; Default:)	Name of the disk (from `/store disk` menu) used by this store.
name (*string*; Default:)	Descriptive name of the store
type (*user-manager* | *web-proxy*; Default:)	Configured type of the store. Two options are available, either store is used by web-proxy or by user-manager.

Read-only Properties

Property	Description
status (*backup* | *active* | *invalid*)	Current status of the store. Shows whether store is used as backup,active or some of the config is invalid.

Menu specific commands

Property	Description
activate (<*store_name*>)	Makes specified store as active if previously was in backup state.

Disk management

Sub-menu: `/store disk`

Read-only Properties

Property	Description
free-space (*integer [KiB]*)	Shows the free space left on the disk.
name (*string*)	Name of the disk
status (*strung*)	Shows the current status of the disk, can be **ready**, **formating** etc.
system (*yes* | *no*)	Shows whether disk is used as system drive
total-space (*integer [KiB]*)	Shows total available disk space

Menu specific commands

Property	Description
check-drive (<*drive_name*>)	Check the drive for errors.
clean-drive (<*drive_name*>)	Clean the drive
format-drive (<*drive_name*>)	Format the file system in usable by RouterOS file system.

Note: Before using drive as a storage device it must be formatted. Before doing so, make sure that all sensitive data is backed up.

Manual:Support Output File

What is a supout.rif file?

Applies to RouterOS: ALL

'The support file is used for debugging MikroTik RouterOS and to solve the support questions faster. All MikroTik Router information is saved in a binary file, which is stored on the router and can be downloaded from the router using ftp.'

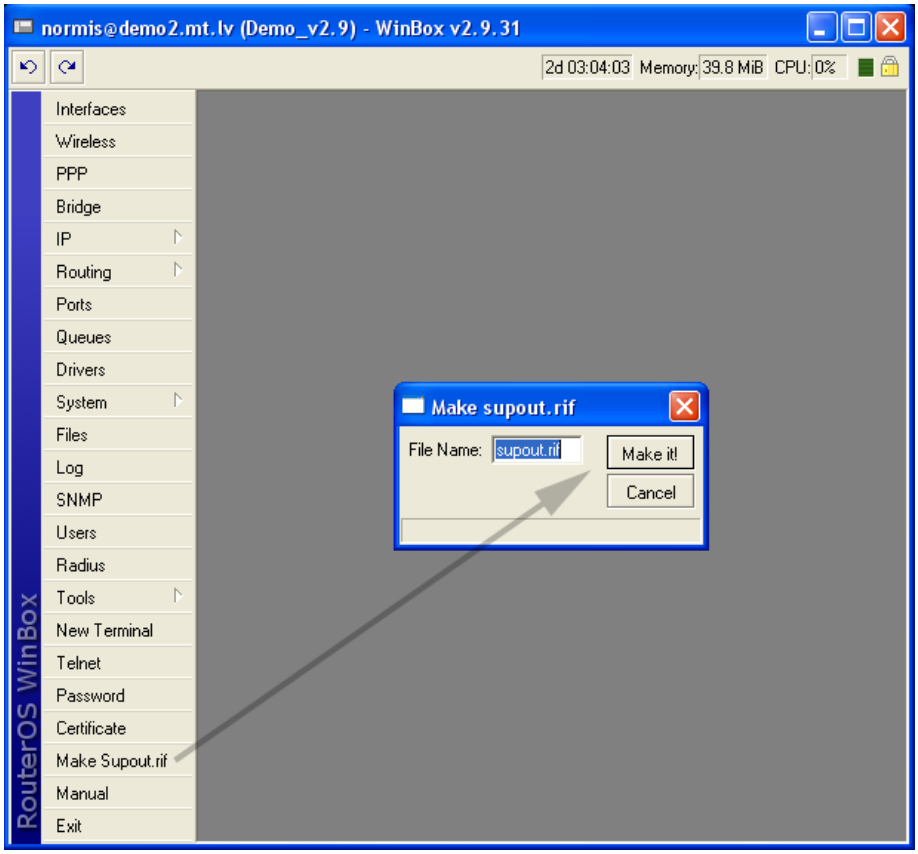
You can view the contents of this file in your Mikrotik account [1], simply to to the Supout.rif section and upload the file.

This file contains all your routers configuration, logs and some other details that will help the MikroTik Support to solve your issue.

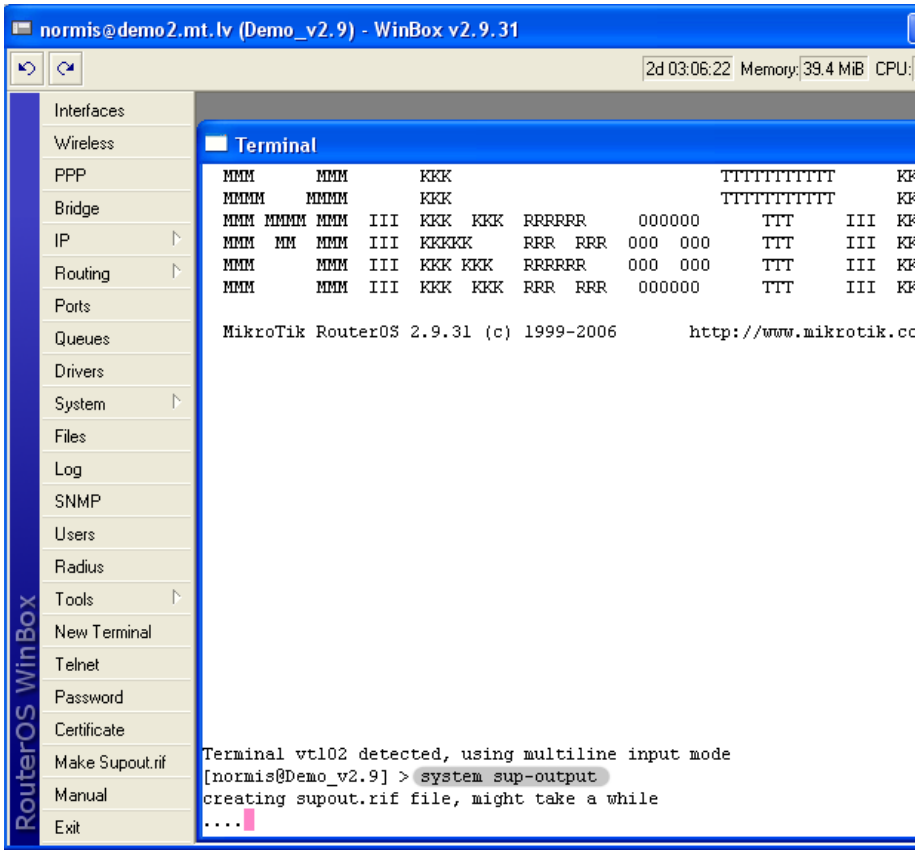
To generate this file, you must type:

```
/system sup-output
```

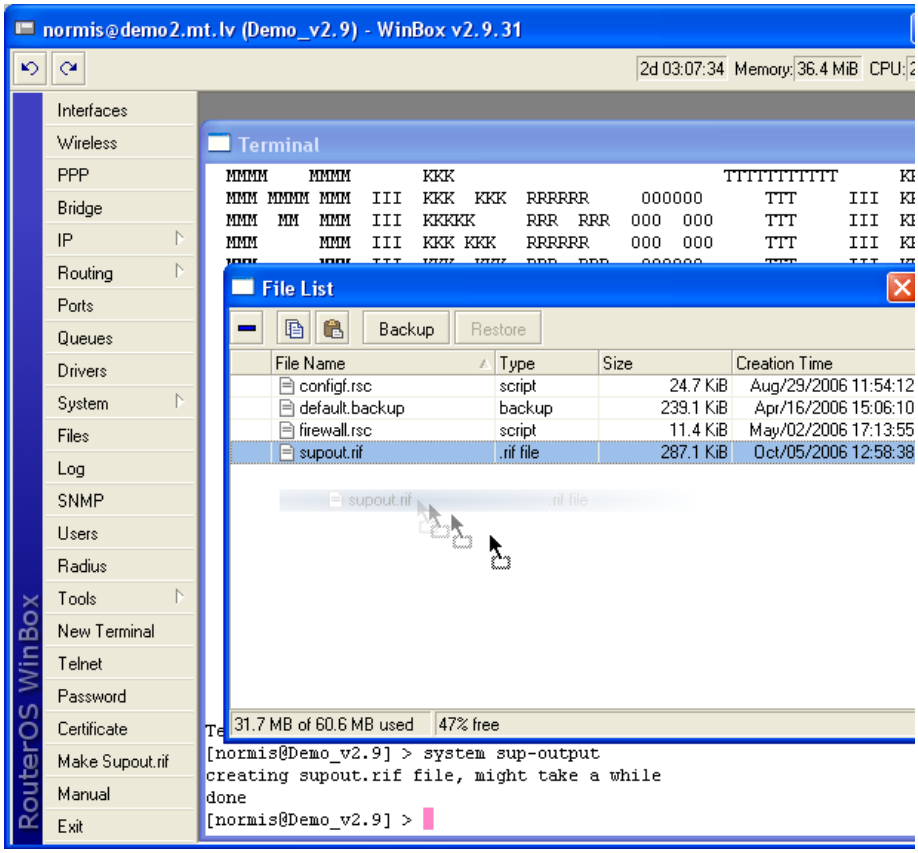
In command line, or use winbox:



You can also use the terminal in Winbox:



To save the file direcly from Winbox, simply drag the file to your desktop:



Of course, it is also possible to download the file with FTP/SFTP or to automate this process with scripting, and have the file emailed to you.

[Top | Back to Content]

References

[1] http://www.mikrotik.com

Manual:System

List of reference sub-pages **Case studies** **List of examples**

<splist showparent=yes />

Manual:System/Scheduler

Applies to RouterOS: 2.9, v3, v4

Summary

The scheduler can trigger script execution at a particular time moment, after a specified time interval, or both.

Properties

- **interval** *(time; default: 0s)* - interval between two script executions, if time interval is set to zero, the script is only executed at its start time, otherwise it is executed repeatedly at the time interval is specified
- **name** *name)* - name of the task
- **on-event** *(name)* - name of the script to execute. It must be presented at /system script
- **run-count** *(read-only: integer)* - to monitor script usage, this counter is incremented each time the script is executed
- **start-date** *(date)* - date of the first script execution
- **start-time** *(time)* - time of the first script execution
- **startup** - execute the script 3 seconds after the system startup.

Notes

Rebooting the router will reset run-count counter.

If more than one script has to be executed simultaneously, they are executed in the order they appear in the scheduler configuration. This can be important if one scheduled script is used to disable another one. The order of scripts can be changed with the move command.

If a more complex execution pattern is needed, it can usually be done by scheduling several scripts, and making them enable and disable each other.

if scheduler item has start-time set to startup, it behaves as if start-time and start-date were set to time 3 seconds after console starts up. It means that all scripts having start-time=startup and interval=0 will be executed once each time router boots.

Examples

We will add a task that executes the script log-test every hour:

```
[admin@MikroTik] system script> add name=log-test source=:log message=test
[admin@MikroTik] system script> print
    0 name="log-test" source=":log messgae=test" owner=admin run-count=0
[admin@MikroTik] system script> .. scheduler
[admin@MikroTik] system scheduler> add name=run-1h interval=1h
on-event=log-test
[admin@MikroTik] system scheduler> print

Flags: X - disabled
 #   NAME      ON-EVENT  START-DATE  START-TIME INTERVAL     RUN-COUNT
 0   run-1h    log-test  mar/30/2004 06:11:35   1h           0
[admin@MikroTik] system scheduler>
```

In another example there will be two scripts added that will change the bandwidth setting of a queue rule "Cust0". Every day at 9AM the queue will be set to 64Kb/s and at 5PM the queue will be set to 128Kb/s. The queue rule, the scripts, and the scheduler tasks are below:

```
 [admin@MikroTik] queue simple> add name=Cust0 interface=ether1 \
\... dst-address=192.168.0.0/24 limit-at=64000
[admin@MikroTik] queue simple> print
Flags: X - disabled, I - invalid
  0   name="Cust0" target-address=0.0.0.0/0 dst-address=192.168.0.0/24
      interface=ether1 limit-at=64000 queue=default priority=8 bounded=yes

[admin@MikroTik] queue simple> /system script
[admin@MikroTik] system script> add name=start_limit source={/queue simple set \
\... Cust0 limit-at=64000}
[admin@MikroTik] system script> add name=stop_limit source={/queue simple set \
\... Cust0 limit-at=128000}
[admin@MikroTik] system script> print
  0 name="start_limit" source="/queue simple set Cust0 limit-at=64000"
    owner=admin run-count=0

  1 name="stop_limit" source="/queue simple set Cust0 limit-at=128000"
    owner=admin run-count=0

[admin@MikroTik] system script> .. scheduler
[admin@MikroTik] system scheduler> add interval=24h name="set-64k" \
\... start-time=9:00:00 on-event=start_limit
[admin@MikroTik] system scheduler> add interval=24h name="set-128k" \
\... start-time=17:00:00 on-event=stop_limit
[admin@MikroTik] system scheduler> print
Flags: X - disabled
  #   NAME      ON-EVENT START-DATE  START-TIME INTERVAL             RUN-COUNT
  0   set-64k   start... oct/30/2008 09:00:00   1d                   0
```

```
  1   set-128k  stop_... oct/30/2008 17:00:00   1d                  0
[admin@MikroTik] system scheduler>
```

The following example schedules a script that sends each week a backup of router configuration by e-mail.

```
[admin@MikroTik] system script> add name=e-backup source={/system backup
{... save name=email; /tool e-mail send to="root@host.com" subject=([/system
{... identity get name] . " Backup") file=email.backup}
[admin@MikroTik] system script> print
  0 name="e-backup" source="/system backup save name=ema... owner=admin
    run-count=0

[admin@MikroTik] system script> .. scheduler
[admin@MikroTik] system scheduler> add interval=7d name="email-backup" \
\... on-event=e-backup
[admin@MikroTik] system scheduler> print
Flags: X - disabled
  #   NAME      ON-EVENT START-DATE  START-TIME INTERVAL            RUN-COUNT
  0   email-... e-backup oct/30/2008 15:19:28   7d                  1
[admin@MikroTik] system scheduler>
```

Do not forget to set the e-mail settings, i.e., the SMTP server and From: address under /tool e-mail. For example:

```
[admin@MikroTik] tool e-mail> set server=159.148.147.198 from=SysAdmin@host.com
[admin@MikroTik] tool e-mail> print
    server: 159.148.147.198
      from: SysAdmin@host.com
[admin@MikroTik] tool e-mail>
```

Example below will put 'x' in logs each hour from midnight till noon:

```
[admin@MikroTik] system script> add name=enable-x source={/system scheduler
{... enable x}
[admin@MikroTik] system script> add name=disable-x source={/system scheduler
{... disable x}
[admin@MikroTik] system script> add name=log-x source={:log message=x}
[admin@MikroTik] system script> .. scheduler
[admin@MikroTik] system scheduler> add name=x-up start-time=00:00:00 \
\... interval=24h on-event=enable-x
[admin@MikroTik] system scheduler> add name=x-down start-time=12:00:00
\... interval=24h on-event=disable-x
[admin@MikroTik] system scheduler> add name=x start-time=00:00:00 interval=1h \
\... on-event=log-x
[admin@MikroTik] system scheduler> print
Flags: X - disabled
  #   NAME      ON-EVENT START-DATE  START-TIME INTERVAL            RUN-COUNT
  0   x-up      enable-x oct/30/2008 00:00:00   1d                  0
  1   x-down    disab... oct/30/2008 12:00:00   1d                  0
  2   x         log-x    oct/30/2008 00:00:00   1h                  0
[admin@MikroTik] system scheduler>
```

Manual:System/SSH client

Overview

RouterOS provides SSH client that supports SSHv2 logins to SSH servers reachable from the router.

Requirements

For this command to be available router has to have *system* and *security* packages installed.

Available features

Simple log-in to remote host

It is able to connect to remote host and initiate ssh session. IP address supports both IPv4 and IPv6.

```
/system ssh 192.168.88.1
/system ssh 2001:db8:add:1337::beef
```

In this case user name provided to remote host is one that has logged into the router. If other value is required, then *user=<username>* has to be used.

```
/system ssh 192.168.88.1 user=lala
/system ssh 2001:db8:add:1337::beef user=lala
```

Log-in from certain IP address of the router

For testing or security reasons it may be required to log-in to other host using certain source address of the connection. In this case *src-address=<ip address>* argument has to be used. Note that IP address in this case supports both, IPv4 and IPv6.

```
/system ssh 192.168.88.1 src-address=192.168.89.2
/system ssh 2001:db8:add:1337::beef src-address=2001:db8:bad:1000::2
```

in this case, ssh client will try to bind to address specified and then initiate ssh connection to remote host.

Log-in using certificate

For this to work user has to set up public key on remote end where ssh will connect to. How to do that on RouterOS you can read here. On local end router, public and private keys have to be uploaded to be used in */user ssh-keys private* when adding private key and user name that will be able to use this key.

Warning: User with full rights on the router can change 'user' attribute value under */user ssh-keys privte*

Here private key is created for use for user *lala*

```
/user ssh-keys private import private-key-file=id_dsa public-key-file=id_dsa.pub user=lala
```

Executing remote commands

To execute remote command it has to be supplied at the end of log-in line

```
/system ssh 192.168.88.1 "/ip address print"
/system ssh 192.168.88.1 command="/ip address print"
/system ssh 2001:db8:add:1337::beef "/ip address print"
/system ssh 2001:db8:add:1337::beef command="/ip address print"
```

Warning: If server does not support pseudo-tty (ssh -T or ssh host command), like mikrotik ssh server, then it is not possible to send multiline commands via SSH

For example, sending command `"/ip address \n add address=1.1.1.1/24"` to Mikrotik router will fail.

[Top | Back to Content]

Manual:Tools/Sigwatch

Applies to RouterOS: All

The Sigwatch utility monitors state of attached asynchronous serial ports and generates system events upon state change.

Requirements

Sigwatch is available only on X86 (PC) platform. *Advanced Tools* package is required.

Settings

- **count** (read-only: integer) - how many times the event for this item was triggered. Count is reset on reboot and on most item configuration changes
- **log** (yes | no; default: no) - whether to add a message in form of name-of-sigwatch-item: signal changed [to high | to low] to System-Info facility whenever this sigwatch item is triggered
- **name** (name) - name of the sigwatch item
- **on-condition** (on | off | change; default: on) - on what condition to trigger action of this item
  - on - trigger when state of pin changes to high
  - off - trigger when state of pin changes to low
  - change - trigger whenever state of pin changes. If state of pin changes rapidly, there might be triggered only one action for several state changes
- **port** (name) - serial port name to monitor
- **script** (name) - script to execute when this item is trigered
- **signal** (dtr | rts | cts | dcd | ri | dsr; default: rts) - name of signal of number of pin (for standard 9-pin connector) to monitor
  - dtr - Data Terminal Ready (pin #4)
  - rts - Request To Send (pin #7)
  - cts - Clear To Send (pin #8)
  - dcd - Data Carrier Detect (pin #1)
  - ri - Ring Indicator (pin #9)

- dsr - Data Set Ready (pin #6)

- **state** (read-only: text) - last remembered state of monitored signalcount (read-only: integer) - how many times the event for this item was triggered. Count is reset on reboot and on most item configuration changes

Note: You can type actual script source instead of the script name from /system script list.

Example

In the following example we will add a new sigwatch item that monitors whether the port serial1 has cts signal.

```
[admin@MikroTik] tool sigwatch> print
Flags: X - disabled
  #   NAME                                PORT    SIGNAL     ON-CONDITION LOG
  0   test                                serial1 cts        change       no
[admin@MikroTik] tool sigwatch>
```

By typing a command print detail interval=1s, we can check whether a cable is connected or it is not. See the state argument - if the cable is connected to the serial port, it shows on, otherwise it will be off.

```
[admin@MikroTik] tool sigwatch> print detail
Flags: X - disabled
  0   name="test" port=serial1 signal=cts on-condition=change log=no script=""
      count=1 state=on

[admin@MikroTik] tool sigwatch> print detail
Flags: X - disabled
  0   name="test" port=serial1 signal=cts on-condition=change log=no script=""
      count=1 state=on

[admin@MikroTik] tool sigwatch> print detail
Flags: X - disabled
  0   name="test" port=serial1 signal=cts on-condition=change log=no script=""
      count=2 state=off

[admin@MikroTik] tool sigwatch> print detail
Flags: X - disabled
  0   name="test" port=serial1 signal=cts on-condition=change log=no script=""
      count=2 state=off

[admin@MikroTik] tool sigwatch>
```

In the port menu you can see what signal is used by serial cable. For example, without any cables it looks like this:

```
[admin@MikroTik] port> print stats
  0 name="serial0" line-state=dtr,rts

  1 name="serial1" line-state=dtr,rts
[admin@MikroTik] port>
```

But after adding a serial cable to the serial port:

```
[admin@MikroTik] port> print stats
  0 name="serial0" line-state=dtr,rts

  1 name="serial1" line-state=dtr,rts,cts
[admin@MikroTik] port>
```

This means that the line-state besides the dtr and rts signals has also cts when a serial cable is connected. The example below will execute a script whenever on-condition changes to off:

```
[admin@MikroTik] tool sigwatch> print detail
Flags: X - disabled
  0   name="cts_rest" port=serial1 signal=cts on-condition=off log=no
      script=/system shutdown count=0 state=on
[admin@MikroTik] tool sigwatch>
```

It means that if a serial cable is connected to the serial port, all works fine, but as soon as it is disconnected, the router shuts down. It will continue all the time until the serial cable will not be connected again.

Manual:Tools/Sms

Applies to RouterOS: v3, v4 +

Summary

Sub-menu: `/tool sms`
Package: `advanced-tools`
Standards:

It is possible to connect GSM modem to RouterOS device and use it to send and receive SMS messages. RouterOS lists such modem as serial port that appears in '/port print' listing. GSM standard defines AT commands for sending SMS messages, and defines how messages should be encoded in these commands.

'advanced-tools' package provides command `'/tool sms send'` that uses standard GSM AT commands to send SMS.

Sending

Command: `/tool sms send`

Example

```
/tool sms send usb3 "20000000" \
   message="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz!@#\$%^&*(){}[]\"'~"
```

Send command takes following parameters:

Parameter	Description
port (*string*)	Name of port from */port* list that GSM modem is attached to.
dst (*string*)	Recepient phone number. Allowed characters are "0123456789*#abc". If first character is "+" then phone number type is set to *international*, otherwise it is set to *unknown*.
channel (*integer*)	Which modem channel to use for sending.
message (*string*)	Message contents. It is encoded using GSM 7 encoding (UCS2 currently is not supported), so message length is limited to 160 characters (characters ^{}\[]~
smsc (*string*)	
type (*string*)	If set to *class-0*, then send class 0 SMS message. It is displayed immedeately and not stored in phone.

Note: Since V3.23 there is one port per modem, and modem has channels used for commands and data. Channels have numbers 0,1,2, etc. Some modems may have just two channels, some have more. The SMS tool has channel support since v3.28

Receiving

Since v3.24 RouterOS also supports receiving of SMS messages, and can exectue scripts, and even respond to the sender.

Before router can receive SMS, relevant configuration is required in general **/tool sms** menu. Following parameters are configurable:

Parameter	Description
allowed-number (*string*; Default: "")	Sender number that will be allowed to run commands, must specify country code ie. +371XXXXXXX
channel (*integer*; Default: **0**)	Which modem channel to use for receiving.
keep-max-sms (*integer*; Default: **0**)	Maximum number of messages that will be saved. If you set this bigger than SIM supports, new messages will not be received!'
port (*string*; Default: **(unknown)**)	Modem port (modem can be used only by one process "/port> print")
receive-enabled (*yes* | *no*; Default: **no**)	Must be turned on to receive messages
secret (*string*; Default: "")	the secret password, mandatory

Inbox

Sub-menu: `/tool sms inbox`

If you have enabled the reader, you will see incoming messages in this submenu.

Read-only properties:

Property	Description
src (*string*)	Senders phone number.
text (*string*)	Message body
timestamp (*time*)	Time when message was received
type (*string*)	Message type

Syntax

```
 :cmd SECRET script NAME [[ VAR[=VAL] ] ... ]
```

- **SECRET** - the password
- **NAME** - name of the script that's available in "/system script"
- **VAR** - variables that will be passed to the script (can be passed as VAR or as VAR=value), separated by spaces.

Other things to remember:

- *Parameters can be put into quotes "VAR"="VAL" if necessary.
- *Escaping of values is not supported (VAR="\"").
- *Combined SMS are not supported, every SMS will be treated separately
- * 16Bit unicode messages are not supported
- * SMS are decoded with the standard GSM7 alphabet, so you can't send in other encodings, otherwise it will be decoded incorrectly

Examples

Wrong:

```
:cmd script mans_skripts
:cmd slepens script mans skripts
:cmd slepens script mans_skripts var=
:cmd slepens script mans_skripts var= a
:cmd slepens script mans_skripts var=a a
```

Right:

```
:cmd slepens script mans_skripts
:cmd slepens script "mans skripts"
:cmd slepens script mans_skripts var
:cmd slepens script mans_skripts var=a
:cmd slepens script mans_skripts var="a a"
```

Debugging

/tool sms send command is logging data that is written and read. It is logged with tags *gsm,debug,write* and *gsm,debug,read* For more information see system logging.

Implementation details

AT+CMGS and *AT+CMGF* commands are used. Port is acquired for the duration of the command and cannot be used concurently by another RouterOS component. Message sending process can take a long time, it times out after a minute and after two seconds during initial AT command exchange.

Manual:Using scope and target-scope attributes

The problem

No all routes in present in routing table are active. One of thing neccessary for a route to become active, is that it's nexthop can be resolved. Inactive routes are not used for packet forwarding and are not redistributed to routing protocols. They do nothing useful, just waste memory :)

Route scope and target scope attributes can be used to change nexthop resolving. Normally nexthops can be resolved only through routes that are on link. On the other hand, routes in BGP updates frequently has nexthops from networks that are not directly connected. By default, these routes will be installed in routing table but will not be active:

```
[admin@A] > ip route pr detail
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf,
B - blackhole, U - unreachable, P - prohibit
 0  Db  dst-address=3.0.0.0/8 gateway=192.65.184.3 interface=""
        gateway-state=unreachable distance=20 scope=255 target-scope=30
        bgp-as-path="513,8220,7018,701,703,80" bgp-local-pref=100 bgp-origin=igp
        received-from=10.0.0.128

 1  Db  dst-address=4.0.0.0/8 gateway=192.65.184.3 interface=""
        gateway-state=unreachable distance=20 scope=255 target-scope=30
        bgp-as-path="513,8220,3356" bgp-local-pref=100 bgp-atomic-aggregate=yes
        bgp-origin=igp received-from=10.0.0.128

 2  Db  dst-address=4.21.104.0/24 gateway=192.65.184.3 interface=""
        gateway-state=unreachable distance=20 scope=255 target-scope=30
        bgp-as-path="513,8220,7018,26207,26207,26207,26207" bgp-local-pref=100
        bgp-origin=igp received-from=10.0.0.128

 3  Db  dst-address=4.21.112.0/23 gateway=192.65.184.3 interface=""
        gateway-state=unreachable distance=20 scope=255 target-scope=30
        bgp-as-path="513,8220,7018,26207,26207,26207,26207" bgp-local-pref=100
        bgp-origin=igp received-from=10.0.0.128
```

Solution using scope attribute

One way to make all routes active is to allow to resolve nexthops through default route. To do that, you can make use of recursive nexthop resolving. Add default route with scope < target-scope of BGP routes:

```
[admin@A] > ip route add gateway=10.0.0.1 scope=10
[admin@A] > ip route pr detail
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf,
B - blackhole, U - unreachable, P - prohibit
 0 A S  dst-address=0.0.0.0/0 gateway=10.0.0.1 interface=ether1
        gateway-state=reachable distance=1 scope=10 target-scope=10
```

```
 1 ADb  dst-address=3.0.0.0/8 gateway=192.65.184.3 interface=ether1
        gateway-state=recursive distance=20 scope=255 target-scope=30
        bgp-as-path="513,8220,7018,701,703,80" bgp-local-pref=100 bgp-origin=igp
        received-from=10.0.0.128

 2 ADb  dst-address=4.0.0.0/8 gateway=192.65.184.3 interface=ether1
        gateway-state=recursive distance=20 scope=255 target-scope=30
        bgp-as-path="513,8220,3356" bgp-local-pref=100 bgp-atomic-aggregate=yes
        bgp-origin=igp received-from=10.0.0.128
```

Solution using target-scope attribute

When there is need to change target-scope? Possible problems with previously described approach are that *all* routes in the table always will be active. This may be not what you want.

An example: router with two interfaces, ethernet and wireless. All BGP routes are resolved through ethernet; wireless interface has some additional static routes. You want these static routes to be active only when wireless interface is in running state. Normally this is the case. However, when there is a default route with low enough scope, all routes will be switched to ethernet interface after wireless interface loses it's running bit.

One possible solution is to leave the scope of default route intact and modify the target-scope of BGP routes.

```
[admin@A] > ip route set 0 scope=255
[admin@A] > routing filter add chain=bgp-in set-target-scope=255
[admin@A] > routing bgp peer set peer1 in-filter=bgp-in
[admin@A] > ip route pr detail
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf,
B - blackhole, U - unreachable, P - prohibit
 0 A S  dst-address=0.0.0.0/0 gateway=10.0.0.1 interface=ether1
        gateway-state=reachable distance=1 scope=255 target-scope=10

 1 ADb  dst-address=3.0.0.0/8 gateway=192.65.184.3 interface=ether1
        gateway-state=recursive distance=200 scope=255 target-scope=255
        bgp-as-path="513,8220,7018,701,703,80" bgp-local-pref=100 bgp-origin=igp
        received-from=10.0.0.128

 2 ADb  dst-address=4.0.0.0/8 gateway=192.65.184.3 interface=ether1
        gateway-state=recursive distance=200 scope=255 target-scope=255
        bgp-as-path="513,8220,3356" bgp-local-pref=100 bgp-atomic-aggregate=yes
        bgp-origin=igp received-from=10.0.0.128
```

How *not* to use them

Possibility to set both scope and target scope of nexthops is a powerful feature and as such can be easily abused. It is possible to create *nexthop resolving loops*. If there will be a logical loop in the routing table, RouterOS will not freeze, it will simply stop nexthop resolving at some point.

Simple loop example (three routes, each one wanting to resolve through another):

```
[admin@A] > /ip route add dst-address=1.1.1.0/24 gateway=2.2.2.2 scope=10 target-scope=10
[admin@A] > /ip route add dst-address=2.2.2.0/24 gateway=3.3.3.3 scope=10 target-scope=10
[admin@A] > /ip route add dst-address=3.3.3.0/24 gateway=1.1.1.1 scope=10 target-scope=10
[admin@A] > /ip route pr
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf,
B - blackhole, U - unreachable, P - prohibit
  #      DST-ADDRESS        PREF-SRC        G GATEWAY         DISTANCE INTERFACE
 0   S  1.1.1.0/24                           2.2.2.2         1
 1   S  2.2.2.0/24                           3.3.3.3         1
 2   S  3.3.3.0/24                           1.1.1.1         1
 3 ADC  10.0.0.0/24        10.0.0.133                        0        ether1
```

Change the gateway of any of the first three routes to 10.0.0.x and they all will become active.

More complex loop example:

```
[admin@A] > ip route add dst-address=1.1.1.0/24 gateway=3.3.3.3 scope=10 target-scope=10
[admin@A] > ip route add dst-address=1.1.1.0/24 gateway=10.0.0.1 scope=10 target-scope=10 distance=3
[admin@A] > ip route add dst-address=3.3.3.0/24 gateway=1.1.1.1 scope=10 target-scope=10
[admin@A] > ip route pr detail
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf,
B - blackhole, U - unreachable, P - prohibit
 0   S  dst-address=1.1.1.0/24 gateway=3.3.3.3 interface=ether1
        gateway-state=recursive distance=1 scope=10 target-scope=10

 1 A S  dst-address=1.1.1.0/24 gateway=10.0.0.1 interface=ether1
        gateway-state=reachable distance=3 scope=10 target-scope=10

 2 A S  dst-address=3.3.3.0/24 gateway=1.1.1.1 interface=ether1
        gateway-state=recursive distance=1 scope=10 target-scope=10

 3 ADC  dst-address=10.0.0.0/24 pref-src=10.0.0.133 interface=ether1 distance=0
        scope=10 target-scope=0
```

Note that now the active route has larger (i.e. worse) distance.

Interface routes, unreachable routes and nexhops

Nexthops cannot be resolved through interface routes (i.e. routes that have interface index instead of gateway address as nexthop). Nexthops also cannot be resolved through unreachable routes (with type B, U, or P) even when they are active. They also do not have nexthops themselves.

Manual:SimTest

Configuration Syntax Explained

Version definition

Version always should be defined at the beginning of the file.

Syntax:

```
version 1;
```

Description

Description is used to show detailed description of what this simulation do.

Syntax:

```
desc {
 ...
}
```

Include

Allows to include parts of configuration located in another files. For example include common host information or common network configuration.

Syntax:

```
#include "host.cfg"
```

include works only from 'tests' directory. So, for example, if file test.cfg is in 'tests/lala/ and 'host.cfg' is in the same dir, then include should be `#include "lala/host.cfg"`

Note: './' and '../' does not work

Host

Defines host login information, where tester application should log in and add virtual network.

Host should always be at the same broadcast network as tester, since ipv6 link-local addresses are used to connect to virtual guests from the tester.

Syntax:

```
host <name> {
    <param> <value>;
}
```

Host configuration allows following parameters:

Param	Description
desc (*string*)	Simple description, can be left blank
addr (*ipv4* | *ipv6*)	IP/IPv6 address of the host
port (*integer* | *string*)	API port on the host
user (*string*)	Username used to login
pass (*string*)	Password used to login
bridge (*string*)	Name of the bridge where management interfaces of guests are added. This is needed for tester to be able to connect to the guest routers and apply configuration.

Network

Define all routers and links between routers in virtual network.

```
network <name> {
  <param> <value>;

  router <ref_name> {
    META {
        <api_commands>
    }
  }
}
```

Network configuration allows following parameters:

Param	Description
desc (*string*)	Simple description, can be left blank
host (*string*)	Which of the defined hosts to use
vlans (*integer [0..1]*)	Whether to use vlans for link simulations instead of virtual interfaces. Vlans should be used in MetaROUTER mode, since metarouters has limited amount of available virtual ethernets
routers (*list of router names separated by comma (R1, R2;)*)	Define all routers in virtual network. All routers are separated by comma
link (*list of routers (R1 R2;)*)	Define links between routers. It is possible to define point to point and broadcast links. Link definition will also make an interface name for the guest interface. For example `link R1 R2` will make point to point link between router with name R1 and router with name R2: • on router R1 interface will be called **R1_R2** • on router R2 interface will be called **R2_R1** Another example, `link R1 R2 R3` will make broadcast network between three routers. Interfaces on routers will be named as follows: • on R1 interface will be called **R1_R2_R3** • on R2 interface will be called **R2_R1** • on R3 interface will be called *'R3_R2*

router scope allows to define two sub-scopes:

* META - this scope should contain configuration in API syntax which will be applied to the router.

* KVM - this scope should contain configuration in RouterOS CLI syntax which will be applied during KVM image creation process.

Simulation

Defines actual simulation process.

```
sim <name> {
  <param> <value>;

  events {
    ...
  }
}
```

Sim configuration allows following parameters:

Param	Description
desc (*string*)	Simple description, can be left blank
network (*string*)	Which network to use from all defined networks.

Simulation Events

Event scope defines all events during simulation in actual order.

Possible events:

print

Output to the screen;

```
print "Hello World\n";
```

delay

Delay in seconds before moving to the next event.

```
delay 4;
```

wait

Pause simulation and timer until manually continued.

reboot

Reboot the router

```
reboot <router_name> ;
```

reconnect

Reconnect to the router, necessary after virtual router reboot

```
reconnect <router_name> ;
```

ping

Run the ping command and check the result. If received packet count < configured count, event returns failure.

```
ping <router_name> {
  <param> <value>;
  ...
```

```
}
```

Currently ping allows following parameters:

Param	Description
src (*IP*)	Used Source address of the ICMP packet
dst (*IP*)	Destination where to send ping.
count (*integer*)	Number of ping packets to send. NOTE: if count is not specified infinite loop will occur.
size (*integer [64..1500]*)	Size of the packet
table (*string*)	Name of the routing table which will be used for nexthop lookup.
status (*string*)	Status returned at least on one response. If specified status is not found then event exits with failure.

For example:

```
ping "R1" {
      dst      10.5.101.1;
      src      10.5.101.21;
      count      1;
      size      64;
      table      main;
}
```

trace

Run the traceroute command and check the result. If recorded path is not the same as specified path, event returns failure.

```
trace <router_name> {
  <param> <value>;
  ...
}
```

Currently trace allows following parameters:

Param	Description
src (*IP*)	Used Source address of the ICMP packet
dst (*IP*)	Destination where to send ping.
size (*integer [64..1500]*)	Size of the packet
table (*string*)	Name of the routing table which will be used for nexthop lookup.
path (*string*)	List of the hops in the path. Hops should be separated by comma (e.g. "1.1.1.1,2.2.2.2")

For example:

```
trace "R1" {
      dst      10.5.101.1;
      src      10.1.101.21;
      path      "1.1.1.1,2.2.2.10,10.5.101.1";
      size      64;
      table      main;
}
```

config_set

Add or modify configuration of the specified virtual router

```
config_set <router_name> {
  id {
    <API CMD>
  }
  cmd {
    <API CMD>
  }
}
```

Sometimes it is needed to get ID of the item which should be modify. In this case use **id** scope to get ID, which will be automatically set in **cmd** scope. If multiple IDs are returned then config is applied to all of them.

For example:

```
config_set "R2" {
      id {
            /ip/route/print
            ?dst-address=0.0.0.0/0
            =.proplist=.id
      }
      cmd {
            /ip/route/set
             =disabled=yes
      }
}
```

config_check

Check parameters on the specified virtual router. Scope can have multiple 'cmd' sections, if any of those fails event will return failure.

```
config_check <router_name> {
  cmd {
    <API CMD>
  }
  <param> <value>
}
```

Config check allows following parameters:

Param	Description
data (*yes* | *no*)	What to do if API command returns data. If not set then it is considered as `data no;` If set to 'yes', event will return failure if API command returns non empty data. If set to 'no', event will return failure if API command returns empty data.
status (*inconclusive* | *failed* | *completed*)	What status to set if event returns failure. • inconclusive - something failed before actual test is possible. For example, BGP failed to establish connection in the test which should test routing filters. • failed - actual test failed • completed - test completed. If nothing failed, then 'completed' is always returned at the end of simulation.

Sample Configuration

host.cfg

```
version 1;

// host login information
host "default" {
      desc "Default host";
      addr "10.5.101.14";
      port "8728";
      user "admin";
      pass "";

      bridge "bypass";
}
```

network_2_routers.cfg

```
/* Configuration template for two interconnected routers
*
*              R1 ---- R2
*
*/

version 1;

#include "host.cfg"

network "2_router_network" {
      desc            "";
      host             "default";

      vlans            0;

      routers       R1, R2;

      link             R1 R2;
```

```
      router "R1" {
            META {
                  /ip/address/add
                  =address=1.1.1.1/30
                  =interface=R1_R2
            }
      }

      router "R2" {
            META {
                  /ip/address/add
                  =address=1.1.1.2/30
                  =interface=R2_R1
            }
      }

}
```

bgp/001-auth.cfg

```
version 1;

desc {
+=====================================================================
| BGP connection establishment with authentication
|
+=====================================================================
.
.Test setup creates a simple link between 2 routers
.
.
.              R1------R2
.
.  Simulator actions:
.      * Check if BGP connection is established
.      * Change to incorrect MD5 key
.      * Check if BGP connection is NOT established
.      * Change back to correct MD5 key
.      * Check if BGP connection is established
}

#include "network_2_routers.cfg"


sim "simple"{
      desc            "MD5 authentication test";
      network       "2_router_network";
```

```
	events {
		print "Set initial config...\n";
		config_set "R1" {
			cmd {
				/routing/bgp/peer/add
				=name=R2
				=remote-address=1.1.1.2
				=remote-as=65530
				=tcp-md5-key=helloworld
			}
		}
		config_set "R2" {
			cmd {
				/routing/bgp/peer/add
				=name=R1
				=remote-address=1.1.1.1
				=remote-as=65530
				=tcp-md5-key=helloworld
			}
		}
		delay 7;

		print "Check if BGP session is established..\n";
		config_check "R1" {
			cmd {
				/routing/bgp/peer/print
				?state=established
				=.proplist=.id
			}
			status	inconclusive;
		}

		print "Change MD5 key..\n";
		config_set "R1" {
			// id can be only one and should contain only one command
			id {
				/routing/bgp/peer/print
				?name=R2
				=.proplist=.id
			}
			cmd {
				/routing/bgp/peer/set
				=tcp-md5-key=lalala
			}
		}

		print "Waiting...\n";
```

```
            delay 3;

            print "Check if BGP session is NOT established..\n";
            config_check "R1" {
                  cmd {
                        /routing/bgp/peer/print
                        ?state=established
                        =.proplist=.id
                  }
                  data      yes; // failure if returns any entries
                  status      failed;
            }


            print "Change MD5 key..\n";
            config_set "R1" {
                  // id can be only one and should contain only one command
                  id {
                        /routing/bgp/peer/print
                        ?name=R2
                        =.proplist=.id
                  }
                  cmd {
                        /routing/bgp/peer/set
                        =tcp-md5-key=helloworld
                  }
            }

            print "Waiting...\n";
            delay 3;

            print "Check if BGP session is established..\n";
            config_check "R1" {
                  cmd {
                        /routing/bgp/peer/print
                        ?state=established
                        =.proplist=.id
                  }
                  data      no;
                  status      failed;
            }

      }
}
```

Manual:SNMP

Applies to RouterOS: v5

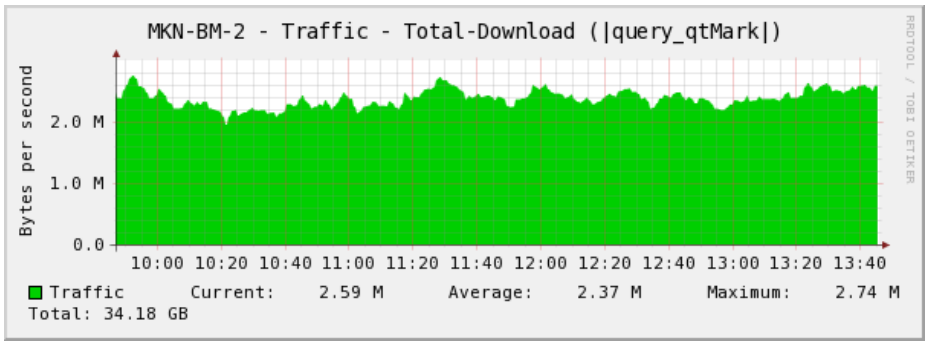
Overview

Standards: RFC 1157 RFC 3414 RFC 3416

Package: system

Simple Network Management Protocol (SNMP) is an Internet-standard protocol for managing devices on IP networks. SNMP can be used to graph various data with tools such as CACTI, MRTG or The Dude [1]

SNMP write support is only available for some OIDs. For supported OIDs SNMP v1, v2 or v3 write is supported



Quick Configuration

To enable SNMP in RouterOS:

```
[admin@MikroTik] /snmp> print
       enabled: no
       contact:
       location:
       engine-id:
       trap-community: (unknown)
       trap-version: 1
[admin@MikroTik] /snmp>  set enabled yes
```

You can also specify administrative contact information in the above settings. All SNMP data will be available to communities configured in *community* menu.

General Properties

Sub-menu: `/snmp`

This sub menu allows to enable SNMP and to configure general settings.

Property	Description
contact (*string*; Default: "")	Contact information
enabled (*yes* | *no*; Default: **no**)	Used to disable/enable SNMP service
engine-id (*string*; Default: "")	for SNMP v3, used as part of identifier. You can configure suffix part of engine id using this argument.
location (*string*; Default: "")	Location information
trap-community (*string*; Default: **public**)	Which communities configured in *community* menu to use when sending out the trap.
trap-generators (*interfaces* | *start-trap*; Default:)	What action will generate traps: • interfaces - interface changes; • start-trap - snmp server starting on the router
trap-interfaces (*string* | *all*; Default:)	List of interfaces that traps are going to be sent out.
trap-target (*list of IP/IPv6*; Default: **0.0.0.0**)	IP (IPv4 or IPv6) addresses of SNMP data collectors that have to receive the trap
trap-version (*1|2|3*; Default: **1**)	Version of SNMP protocol to use for trap

Community

Sub-menu: `/snmp community`

This sub-menu allows to set up access rights for the SNMP data.

There is little security in v1 and v2c, just Clear text community string („username“) and ability for Limiting access by IP adress.

Since SNMP v3, better options have been introduced - Authorisation (User + Pass) with MD5/SHA1, Encryption with DES.

```
[admin@MikroTik] /snmp community> print value-list
                     name: public
                  address: 0.0.0.0/0
                 security: none
              read-access: yes
             write-access: no
  authentication-protocol: MD5
      encryption-protocol: DES
  authentication-password: *****
      encryption-password: *****
```

Warning: Default settings only have one community named *public* without any additional security settings. These settings should be considered insecure and should be adjusted according required security profile.

Properties

Property	Description
address (*IP/IPv6 address*; Default: **0.0.0.0/0**)	Addresses from which connections to SNMP server is allowed
authentication-password (*string*; Default: "")	Password used to authenticate connection to the server (SNMPv3)
authentication-protocol (*MD5* | *SHA1*; Default: **MD5**)	Protocol used for authentication (SNMPv3)
encryption-password (*string*; Default: "")	password used for encryption (SNMPv3)
encryption-protocol (*DES*; Default: **DES**)	encryption protocol to be used to encrypt the communication (SNMPv3)
name (*string*; Default:)	
read-access (*yes* | *no*; Default: **yes**)	Whether read access is enabled for this community
security (*authorized* | *none* | *private*; Default: **none**)	
write-access (*yes* | *no*; Default: **no**)	Whether write access is enabled for this community. Read more >>

Management information base (MIB)

The Management Information Base (MIB) is the database of information maintained by the agent that the manager can query. You can download the latest MikroTik RouterOS MIB [2] file.

MIBs used in RouterOS v5.x:

- MIKROTIK-MIB
- MIB-2
- HOST-RESOURCES-MIB
- IF-MIB
- IP-MIB
- IP-FORWARD-MIB
- IPV6-MIB
- BRIDGE-MIB
- DHCP-SERVER-MIB
- CISCO-AAA-SESSION-MIB
- ENTITY-MIB
- UPS-MIB
- SQUID-MIB

Object identifiers (OID)

Each OID identifies a variable that can be read via SNMP. Although the MIB file contains all the needed OID values, you can also print individual OID information in the console with the **print oid** command at any menu level:

```
[admin@MikroTik] /interface> print oid

Flags: D - dynamic, X - disabled, R - running, S - slave
 0  R  name=.1.3.6.1.2.1.2.2.1.2.1 mtu=.1.3.6.1.2.1.2.2.1.4.1
       mac-address=.1.3.6.1.2.1.2.2.1.6.1 admin-status=.1.3.6.1.2.1.2.2.1.7.1
       oper-status=.1.3.6.1.2.1.2.2.1.8.1 bytes-in=.1.3.6.1.2.1.2.2.1.10.1
       packets-in=.1.3.6.1.2.1.2.2.1.11.1 discards-in=.1.3.6.1.2.1.2.2.1.13.1
       errors-in=.1.3.6.1.2.1.2.2.1.14.1 bytes-out=.1.3.6.1.2.1.2.2.1.16.1
       packets-out=.1.3.6.1.2.1.2.2.1.17.1 discards-out=.1.3.6.1.2.1.2.2.1.19.1
       errors-out=.1.3.6.1.2.1.2.2.1.20.1
```

Traps

SNMP traps enable router to notify data collector of interface changes and SNMP service status changes by sending traps. It is possible to send out traps with security features to support SNMPv1 (no security). SNMPv2 and variants and SNMPv3 with encryption and authorization.

For SNMPv2 and v3 you have to set up appropriately configured community as a *trap-community* to enable required features (password or encryption/authorization)

SNMP write

Since RouterOS v3, SNMP write is supported for some functions. SNMP write allows to change router configuration with SNMP requests. Consider to secure access to router or to router's SNMP, when SNMP and write-access are enabled.

To change settings by SNMP requests, use the command below to allow SNMP write for the selected community, Write-access option for SNMP is available from v3.14,

```
/snmp community set <number> write-access=yes
```

System Identity

It's possible to change router system identity by SNMP set command,

```
snmpset -c public -v 1 192.168.0.0 1.3.6.1.2.1.1.5.0  s New_Identity
```

- *snmpset* - SNMP application used for SNMP SET requests to set information on a network entity;
- *public* - router's community name;
- *192.168.0.0* - IP address of the router;
- *1.3.6.1.2.1.1.5.0* - SNMP value for router's identity;

SNMPset command above is equal to the RouterOS command,

```
/system identity set identity=New_Identity
```

Reboot

It's possible to reboot the router with SNMP set commamd, you need to set value for reboot SNMP settings, which is not equal to 0,

```
snmpset -c public -v 1 192.168.0.0 1.3.6.1.4.1.14988.1.1.7.1.0 s 1
```

- **1.3.6.1.4.1.14988.1.1.7.1.0**, SNMP value for the router reboot;
- **s 1**, snmpset command to set value, value should not be equal to 0;

Reboot snmpset command is equal to the RouterOS command,

```
/system reboot
```

Run Script

SNMP write allows to run scripts on the router from **system script** menu, when you need to set value for SNMP setting of the script,

```
snmpset -c public -v 1 192.168.0.0 1.3.6.1.4.1.14988.1.1.8.1.1.3.X s 1
```

- **X**, script number, numeration starts from 1;
- **s 1**, snmpset command to set value, value should not be equal to 0;

The same command on RouterOS,

```
/system script> print
Flags: I - invalid
 0   name="kaka" owner="admin" policy=ftp,reboot,read,write,policy,
test,winbox,password,sniff last-started=jan/01/1970
01:31:57 run-count=23 source=:beep

/system script run 0
```

See Also

- SNMP MRTG

[Top | Back to Content]

References

[1] http://www.mikrotik.com/thedude.php
[2] http://mikrotik.com/download/Mikrotik.mib

Manual:IP/SOCKS

Applies to RouterOS: 2.9, v3, v4

MikroTik RouterOS supports SOCKS proxy server, version 4 [1].

About SOCKS

SOCKS is a proxy server that allows TCP based application data to relay across the firewall, even if the firewall would block the packets. The SOCKS protocol is independent from application protocols, so it can be used for many services, e.g, WWW, FTP, TELNET, and others.

At first, an application client connects to the SOCKS proxy server, then the proxy server looks in its access list to see whether the client is permited to access the remote application resource or not, if it is permitted, the proxy server relies the packet to the application server and creates a connection between the application server and client.

Remember to configure your application client to use SOCKS version 4.

You should secure the SOCKS proxy using its access list and/or firewall to disallow access from outisde. Failing to secure the proxy server may introduce security issues to your network, and may provide a way for spammers to send junk mail through the router.

Property Description

- **connection-idle-timeout** (time; default: 2m) - time after which idle connections are terminated
- **enabled** (yes | no; default: no) - whether to enable or no the SOCKS proxy
- **max-connections** (integer: 1..500; default: 200) - maxumum number of simultaneous connections
- **port** (integer: 1..65535; default: 1080) - TCP port on which the SOCKS server listens for connections

Access List

Submenu level: /ip socks access

Description

In the SOCKS access list you can add rules which will control access to SOCKS server. This list is similar to firewall lists.

Property Description

- action (allow | deny; default: allow) - action to be performed for this rule
- allow - allow packets, matching this rule, to be forwarded for further processing
- deny - deny access for packets, matching this rule
- dst-address (IP address/netmask) - destination (server's) address
- dst-port (port) - destination TCP port
- src-address (IP address/netmask) - source (client's) address for a packet
- src-port (port) - source TCP port

Active Connections

Submenu level: /ip socks connections

Description

The Active Connection list shows all established TCP connections, which are maintained through the SOCKS proxy server.

Property Description

- **dst-address** (read-only: IP address) - destination (application server) IP address
- **rx** (read-only: integer) - bytes received
- **src-address** (read-only: IP address) - source (application client) IP address
- **tx** (read-only: integer) - bytes sent
- **type** (read-only: in | out | unknown) - connection type
  - in - incoming connection
  - out - outgoing connection
  - unknown - connection has just been initiated

Example

To see current TCP connections:

```
[admin@MikroTik] ip socks connections> print
 # SRC-ADDRESS                DST-ADDRESS                TX         RX
 0 192.168.0.2:3242           159.148.147.196:80         4847       2880
 1 192.168.0.2:3243           159.148.147.196:80         3408       2127
 2 192.168.0.2:3246           159.148.95.16:80           10172      25207
 3 192.168.0.2:3248           194.8.18.26:80             474        1629
 4 192.168.0.2:3249           159.148.95.16:80           6477       18695
 5 192.168.0.2:3250           159.148.95.16:80           4137       27568
 6 192.168.0.2:3251           159.148.95.16:80           1712       14296
 7 192.168.0.2:3258           80.91.34.241:80            314        208
 8 192.168.0.2:3259           80.91.34.241:80            934        524
 9 192.168.0.2:3260           80.91.34.241:80            930        524
10 192.168.0.2:3261           80.91.34.241:80            312        158
11 192.168.0.2:3262           80.91.34.241:80            312        158
[admin@MikroTik] ip socks connections>
```

Application Examples

FTP service through SOCKS server

Let us consider that we have a network 192.168.0.0/24 which is masqueraded, using a router with a public IP 10.1.0.104/24 and a private IP 192.168.0.1/24. Somewhere in the network is an FTP server with IP address 10.5.8.8. We want to allow access to this FTP server for a client in our local network with IP address 192.168.0.2/24.

We have already masqueraded our local network:

```
[admin@MikroTik] ip firewall nat> print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=srcnat action=masquerade src-address=192.168.0.0/24
```

```
[admin@MikroTik] ip firewall nat>
```

And the access to public FTP servers is denied in firewall:

```
[admin@MikroTik] ip firewall filter> print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=forward action=drop src-address=192.168.0.0/24 dst-port=21 protocol=tcp
[admin@MikroTik] ip firewall filter>
```

We need to enable the SOCKS server:

```
[admin@MikroTik] ip socks> set enabled=yes
[admin@MikroTik] ip socks> print
                    enabled: yes
                       port: 1080
    connection-idle-timeout: 2m
            max-connections: 200
[admin@MikroTik] ip socks>
```

Add access to a client with an IP address 192.168.0.2/32 to SOCKS access list, allow data transfer from FTP server to client (allow destionation ports from 1024 to 65535 for any IP address), and drop everything else:

```
[admin@MikroTik] ip socks access> add src-address=192.168.0.2 dst-port=21 \
\... action=allow
[admin@MikroTik] ip socks access> add dst-port=1024-65535 action=allow
[admin@MikroTik] ip socks access> add action=deny
[admin@MikroTik] ip socks access> print
Flags: X - disabled
 0   src-address=192.168.0.2 dst-port=21 action=allow
 1   dst-port=1024-65535 action=allow
 2   action=deny
[admin@MikroTik] ip socks access>
```

That's all - the SOCKS server is configured. To see active connections and data transmitted and received:

```
[admin@MikroTik] ip socks connections> print
 # SRC-ADDRESS                DST-ADDRESS                TX         RX
 0 192.168.0.2:1238           10.5.8.8:21                1163       4625
 1 192.168.0.2:1258           10.5.8.8:3423              0          3231744
[admin@MikroTik] ip socks connections>
```

Note! In order to use SOCKS proxy server, you have to specify its IP address and port in your FTP client. In this case IP address would be 192.168.0.1 (local IP address of the router/SOCKS server) and TCP port 1080.

References

[1] http://archive.socks.permeo.com/protocol/socks4.protocol

Manual:Special Login

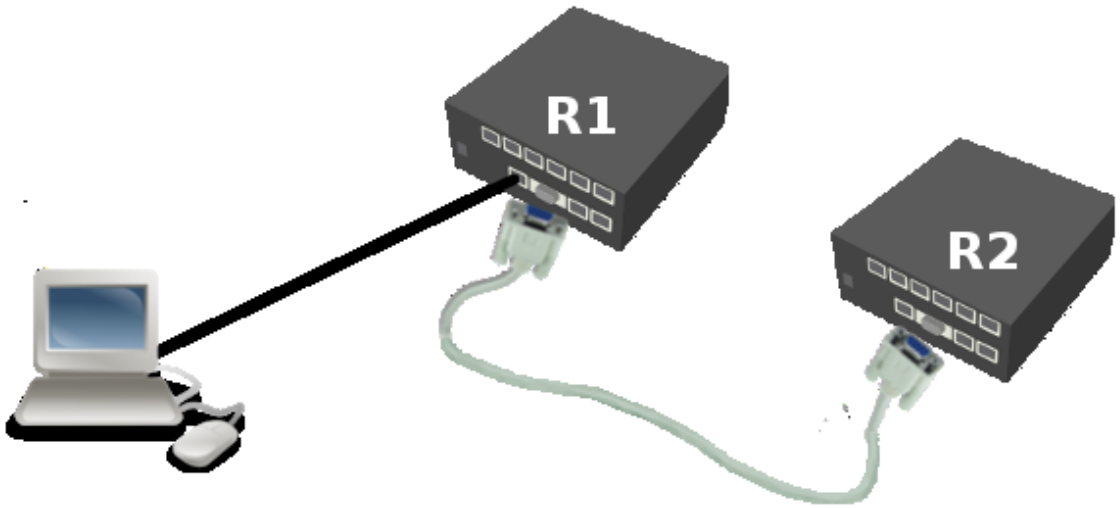
Applies to RouterOS: v3, v4, v5

Description

Special login can be used to access another device (like a switch, for example) that is connected through a serial cable by opening a telnet/ssh session that will get you directly on this device (without having to login to RouterOS first).

Setup

For demonstration we will use two RouterBoards and one PC.



Routers R1 and R2 are connected with serial cable and PC is connected to R1 via ethernet. Lets say we want to access router R2 via serial cable from our PC. To do this you have to set up serial interface proxy on R1. It can be done by feature called **special-login**.

Note: that by default console is bound to serial port.

First task is to unbind console from serial simply by disabling entry in `/system console` menu:

```
[admin@MikroTik] /system console> print
Flags: X - disabled, U - used, F - free
```

```
 #   PORT                                                                    TERM
 0 X serial0                                                                 vt102
```

Next step is to add new user, in this case *serial*, and bind it to the serial port

```
[admin@MikroTik] > /user add name=serial group=full
[admin@MikroTik] > /special-login add user=serial port=serial0 disabled=no
[admin@MikroTik] > /special-login print
Flags: X - disabled
 #   USER                                                                    PORT
 0   serial                                                                  serial0
```

Now we are ready to access R2 from our PC.

```
maris@bumba:/$ ssh serial@10.1.101.146

[Ctrl-A is the prefix key]
R2 4.0beta4
R2 Login:

[admin@R2] >
```

To exit special login mode press Ctrl+A and Q

```
[admin@MikroTik] >
[Q - quit connection]      [B - send break]
[A - send Ctrl-A prefix]   [R - autoconfigure rate]


Connection to 10.1.101.146 closed.
```

See More

- Serial Console
- Sigwatch

[Top | Back to Content]

Manual:Spectral scan

Applies to RouterOS: v4.3+

The spectral scan can scan all frequencies supported by your wireless card, and plot them directly in console. Exact frequency span depends on card. Allowed ranges on r52n: [4790; 6085], [2182; 2549].

Wireless card can generate 4us long spectral snapshots for any 20mhz wide channel. This is considered a single spectral sample.

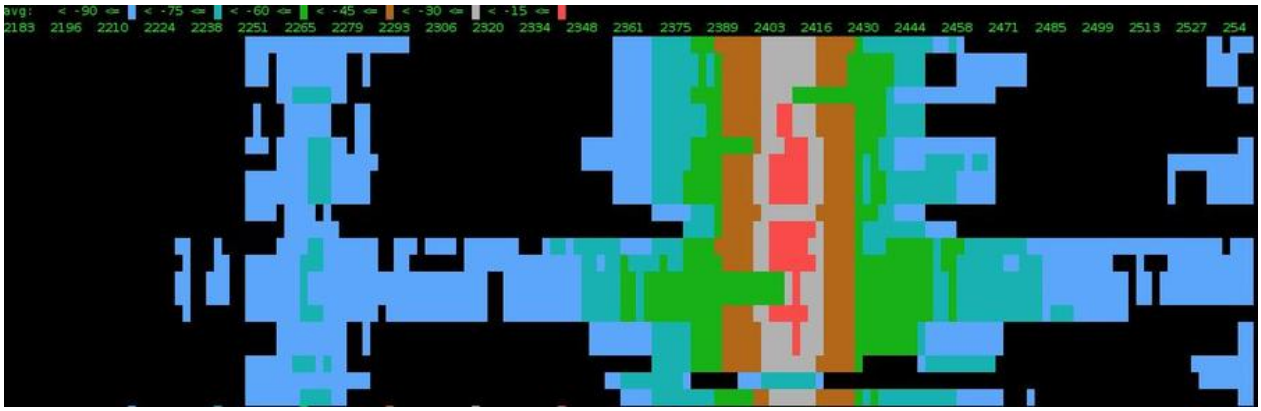
To improve data quality spectrum is scanned with 10mhz frequency increments, which means doubled sample coverage at each specific frequency (considering 20mhz wide samples).

Currently this feature is supported only for Atheros Merlin chips. (ie. AR9220, AR9280, AR9223).

Currently tested models: RouterBOARD R52N and R2N only.

Console

Spectral History




```
/interface wireless spectral-history <wireless interface name>
```

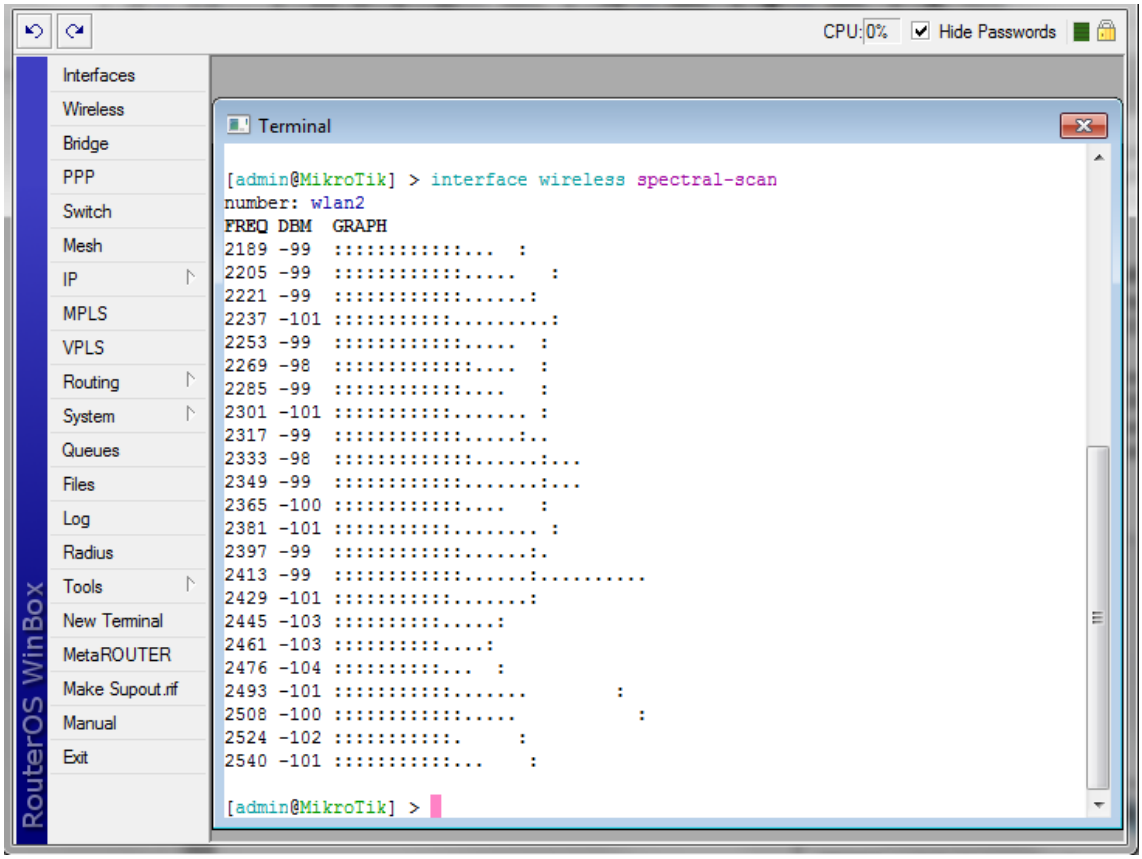
Plots spectrogram. Legend and frequency ruler is printed every 24 lines. Numbers in the ruler correspond to the value at their leftmost character position. Power values that fall in different ranges are printed as different colored characters with the same foreground and background color, so it is possible to copy and paste terminal output of this command.

- *value* -- select value that is plotted on the output. 'interference' is special as it shows detected interference sources (affected by 'classify-samples' parameter) instead of power readings, and cannot be made audible.
- *interval* -- interval at which spectrogram lines are printed.
- *duration* -- terminate command after specified time. default is indefinite.
- *buckets* -- how many values to show in each line of spectrogram. This value is limited by the number of columns in terminal. It is useful to reduce this value if using 'audible'.
- *average-samples* -- Number of 4us spectral snapshots to take at each frequency, and calculate average and maximum energy over them. (default 10)
- *classify-samples* -- Number of spectral snapshots taken at each frequency and processed by interference classification algorithm. Generally more samples gives more chance to spot certain type of interference (default 50)
- *range* --
  - 2.4ghz - scan whole 2.4ghz band
  - 5ghz - scan whole 5ghz band

- current-channel - scan current channel only (20 or 40 mhz wide)
- range - scan specific range

- *audible=yes* -- play each line as it is printed. There is a short silence between lines. Each line is played from left to right, with higher frequencies corresponding to higher values in the spectrogram.

Spectral Scan




```
/interface wireless spectral-scan <wireless interface name>
```

Continuously monitor spectral data. This command uses the same data source as 'spectral-history', and thus shares many parameters.

Each line displays one spectrogram bucket -- frequency, numeric value of power average, and a character graphic bar. Bar shows average power value with ':' characters and average peak hold with '.' characters. Maximum is displayed as a lone floating ':' character.

- *show-interference* -- add column that shows detected interference sources.

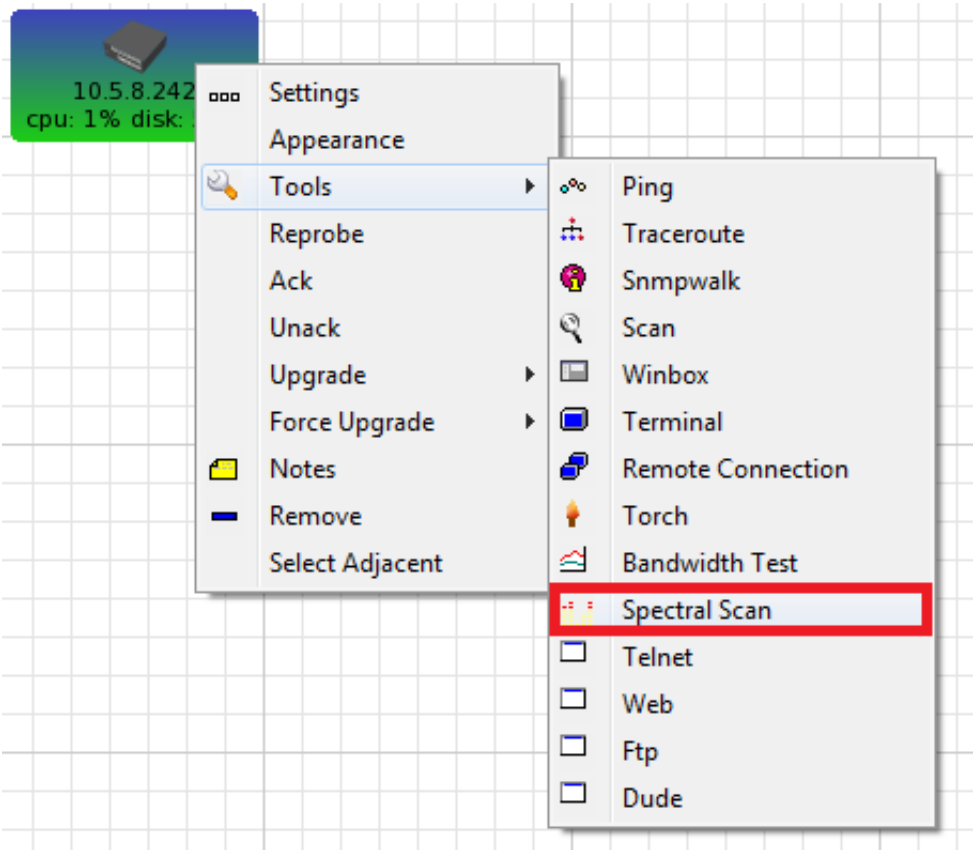
Types of possibly classified interference:

- bluetooth-headset
- bluetooth-stereo
- cordless-phone
- microwave-oven
- cwa
- video-bridge
- wifi

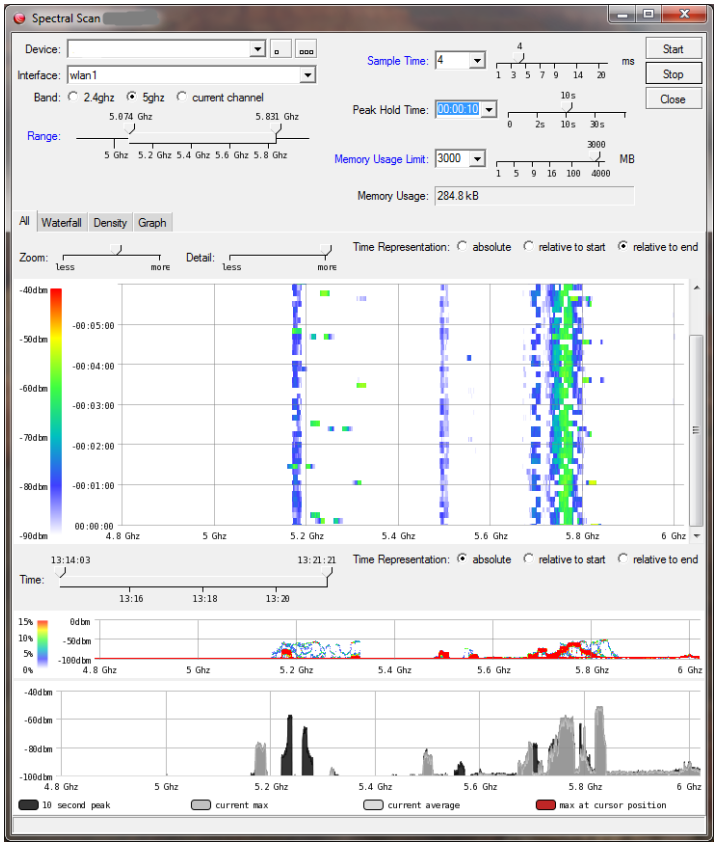
The Dude

The Dude is a free network monitoring and management program by MikroTik. You can download it here [1].

The Dude has a built-in capability to run graphical Spectral Scan from any of your RouterOS devices with a supported wireless card. Simply select this device in your Dude map, right click and choose Tools -> Spectral Scan.



This will bring up the Spectral Scan GUI with various options and different view modes:



MikroTik Support

MikroTik Product Support Service

Always remember, that the most questions are answered and easily explained in the Reference Manual!

1. If you have bought at least a Level 4 license, you can get limited support service by e-mail for 30 days after the purchase: support[at]mikrotik.com, if you obtained your license from a reseller - please contact your reseller for support
2. If you have a problem with hardware that you have purchased from us, please make a clear description of what exactly happened. Write that to support[at]mikrotik.com and include your invoice number
3. You can hire a certified consultant for detailed configurations or network diagnostics
4. Don't forget to log on to our Forum [1]

Support Request Instructions

When contacting us at support[at]mikrotik.com:

1. Make sure that you have the latest version of the RouterOS. If you don't have it, upgrade your router to the latest version and check if the problem still persists!
2. Read the FAQ page, maybe the answer to your question is already there!
3. If you have trouble making some setup, check the WIKI for answers, maybe you will find an example for your configuration there!
4. Give us your Invoice number, if you purchased MikroTik RouterOS online, or information about preinstalled system you purchased from us. (Support is only for paying customers). Free Demo License users should pay for the license to receive support.
5. Give brief problem description, including information about your network setup.

6. When the problem appears execute the /system sup-output command to create support output file. Get the supout.rif file from your router using ftp BINARY mode and attach it to your e-mail message with the support request.
7. Make sure that you include the previous conversation in the body of the email message when replying and do not delete the ticket number from the message subject

Please note, that support does not include training on TCP/IP, You should have read the Manual and have at least basic knowledge about networking (what is IP address, what is network address, how to use subnets, how to use 'traceroute' for troubleshooting, etc.).Picking up a resource from this list [2] is a good start.

If you are in need of **immediate assistance**, you can hire a certified consultation specialist [3].

Problems contacting support

To make sure we receive your email, check if:

- your SMTP server isn't blacklisted in international spam databases
- you have configured your e-mail client properly (missing From name?)
- your e-mail doens't contain improper HTML or attachments that could be suspicious
- the outgoing mailserver is configured properly, and has a valid host name. More info in **RFC 2821**

Popular Issues

Before contacting us, always check the following:

- If you have purchased your MikroTik product from a DEALER or RESELLER, please contact themfor support.
- There might be no answer to your ticket, if it has been submitted without a supout.rif file attached. To get your request answered, please reply to this e-mail with file attached.
- Maybe your problem has already been fixed in a more recent version, so upgrade [1] before contacting support
- Many RouterOS questions are already answered, please look at our Wiki [4], Documentation [5] and Forum [1]
- If there are problems with a wireless connection.
  - check that the antenna connector is connected to the 'main' antenna connector
  - check that all screw connectors are tight
  - check that there is no water or moisture in the cable and that there has never been water in the cable (replace the cable if needed)
  - check that the default rate settings for the radio are being used (use '/system reset' as a final resort -- you must be connected to the console or have keyboard and monitor attached to the router)

If you are NOT a MikroTik product user

Sometimes Mikrotik gets emails from people that think Mikrotik is attacking them, sending spam, etc. Please note, MikroTik is a manufacturer of Networking Devices that are used around the world. Just because the *problem* seems to be coming from a MikroTik device, doesn't mean that MikroTik Ltd. is involved. Please always check the owner of the IP address before writing to MikroTik Ltd.

References

[1] http://forum.mikrotik.com
[2] http://www.mikrotik.com/resources.html
[3] http://www.mikrotik.com/consultants.html
[4] http://wiki.mikrotik.com
[5] http://www.mikrotik.com/docs/ros/3.0/

Manual:Switch Chip Features

Applies to RouterOS: 4

Introduction

In this topic we will talk about switch chip features that are implemented in RouterOS (complete set of features implemented starting v4.0)

Presently we use several types of switch chips on our Routerboards and they have a different set of features. Most of them (from now on "Other") have only basic "Port Switching" feature, but there are few with more features:

Capabilities of switch chips:

Feature	Atheros8316	Atheros8327	Atheros7240	ICPlus175D	Other
Port Switching	yes	yes	yes	yes	yes
Port Mirroring	yes	yes	yes	yes	no
Host table	2k entries	2k entries	2k entries	no	no
Vlan table	4096 entries	4096 entries	16 entries	no	no
Rule table	32 rules	92 rules	no	no	no

Atheros8316 is present on **RB493G**(ether1+ether6-ether9, ether2-ether5), **RB1200**(ether1-ether5), **RB450G**(all ports with ether1 optional[more [1]]), **RB435G**(all ports with ether1 optional[more [1]]), **RB750G** and **RB1100**(ether1-ether5, ether6-ether10).

Atheros8327 is present on **RB2011 series**(ether1-ether5+sfp1) **RB750GL**, **RB751G-2HnD** and **RB1100AH, RB1100AHx2**(ether1-ether5, ether6-ether10).

Atheros8227 is present on **RB2011 series**(ether6-ether10).

Atheros7240 is present on **RB750**(ether2-ether5), **RB750UP**(ether2-ether5) and **RB751U-2HnD**(ether2-ether5).

ICPlus175D is present on newest versions of **RB450**(ether2-ether5) and **RB433 series**(ether2-ether3).

ICPlus175C is present on some **RB450**(ether2-ether5) and some **RB433 series**(ether2-ether3).

ICPlus178C is present on **RB493 series**(ether2-ether9) and **RB816**.

Command line config is under `/interface ethernet switch` menu. This menu contains a list of all switch chips present in system, and some sub-menus as well. `/interface ethernet switch` menu list item represents a switch chip in system:

```
[admin@MikroTik] /interface ethernet switch> print
Flags: I - invalid
 #   NAME     TYPE         MIRROR-SOURCE   MIRROR-TARGET
 0   switch1  Atheros-8316 ether2          none
```

Depending on switch type there might be available or not available some configuration capabilities.

Atheros8316 packet flow diagram [2]

Features

Port Switching

Switching feature allows wire speed traffic passing among a group of ports, like the ports were a regular ethernet switch. You configure this feature by setting a "master-port" property to one ore more ports in `/interface ethernet` menu. A 'master' port will be the port through which the RouterOS will communicate to all ports in the group. Interfaces for which the 'master' port is specified become inactive - no traffic is received on them and no traffic can be sent out.

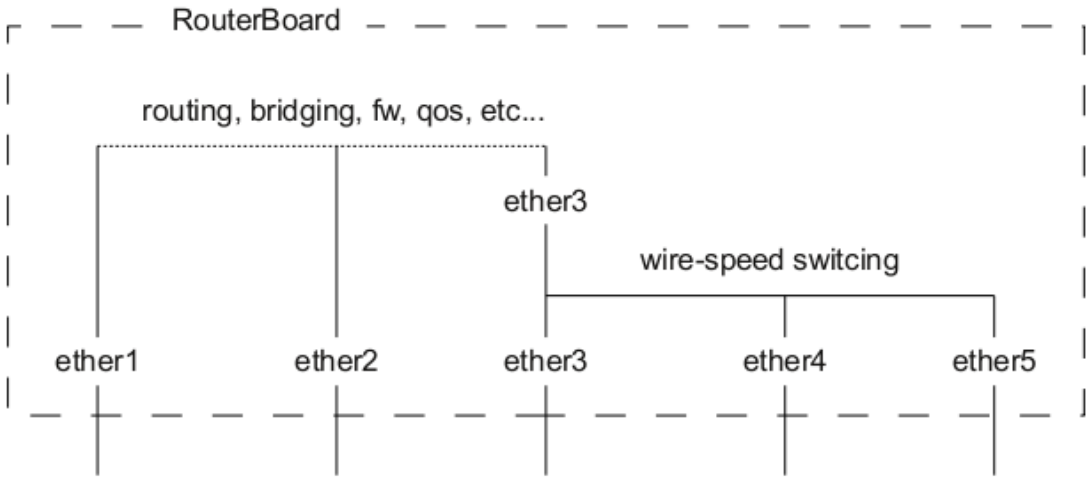
For example consider a router with five ethernet interfaces:

```
[admin@MikroTik] > interface ethernet print
Flags: X - disabled, R - running, S - slave
 #    NAME      MTU   MAC-ADDRESS       ARP        MASTER-PORT      SWITCH
 0 R  ether1    1500  00:0C:42:3E:5D:BB enabled
 1    ether2    1500  00:0C:42:3E:5D:BC enabled    none             switch1
 2    ether3    1500  00:0C:42:3E:5D:BD enabled    none             switch1
 3    ether4    1500  00:0C:42:3E:5D:BE enabled    none             switch1
 4 R  ether5    1500  00:0C:42:3E:5D:BF enabled    none             switch1
```

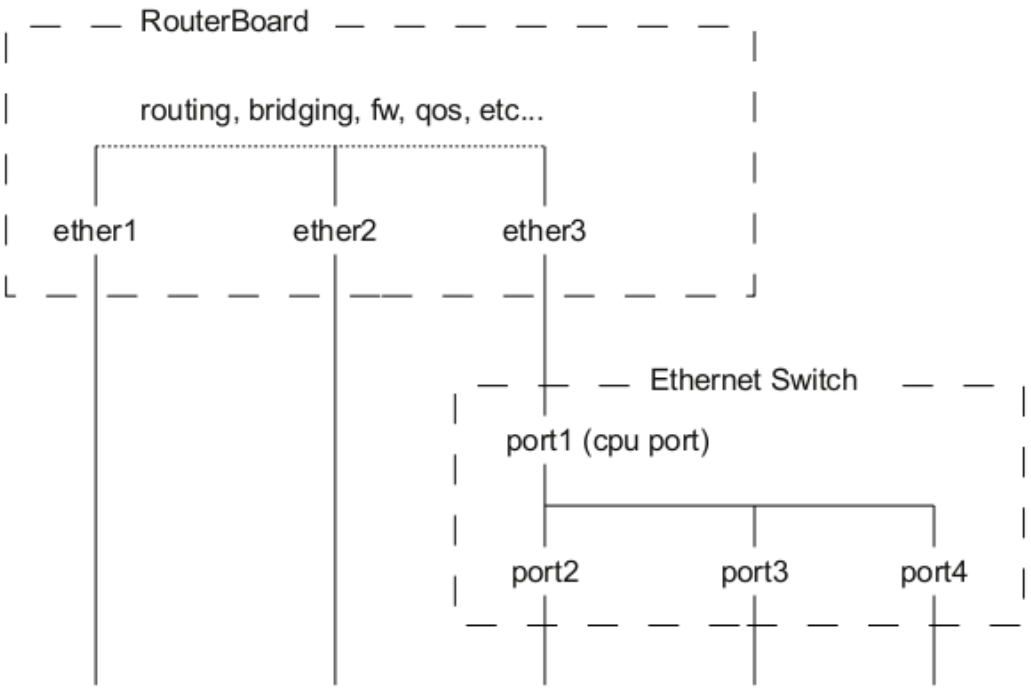
And you configure a switch containing three ports ether3, ether4 and ether5:

```
[admin@MikroTik] /interface ethernet> set ether4,ether5 master-port=ether3
[admin@MikroTik] /interface ethernet> print
Flags: X - disabled, R - running, S - slave
 #    NAME      MTU   MAC-ADDRESS       ARP        MASTER-PORT      SWITCH
 0 R  ether1    1500  00:0C:42:3E:5D:BB enabled
 1    ether2    1500  00:0C:42:3E:5D:BC enabled    none             switch1
 2 R  ether3    1500  00:0C:42:3E:5D:BD enabled    none             switch1
 3  S ether4    1500  00:0C:42:3E:5D:BE enabled    ether3           switch1
 4 RS ether5    1500  00:0C:42:3E:5D:BF enabled    ether3           switch1
```

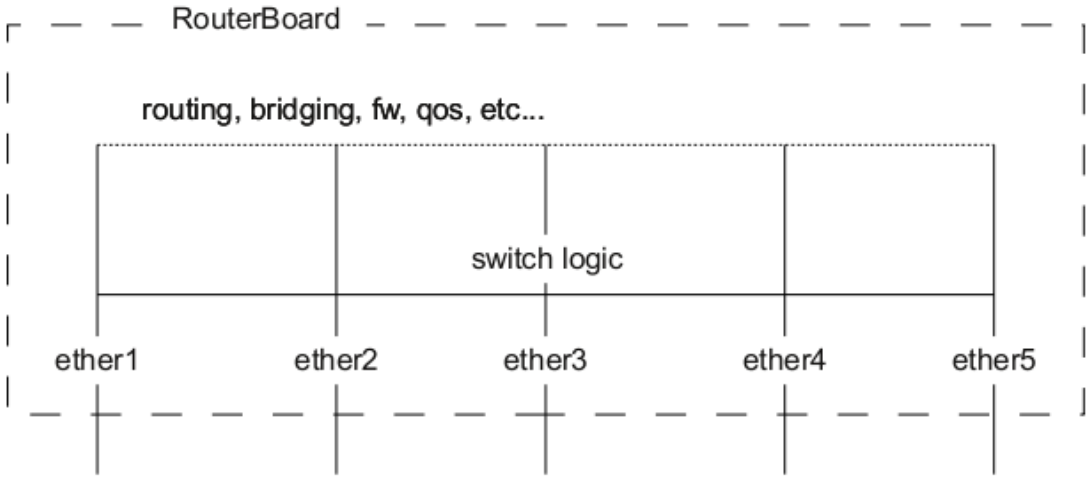
ether3 is now the master port of the group. Note: you can see that previously a link was detected only on ether5, but now as the ether3 is a 'master' the running flag is propagated to master port.



In essence this configuration is the same as if you had a RouterBoard with 3 ethernet interfaces with ether3 connected to ethernet switch that has 4 ports:



A more general diagram of RouterBoard with switch chip that has 5 port switch chip:



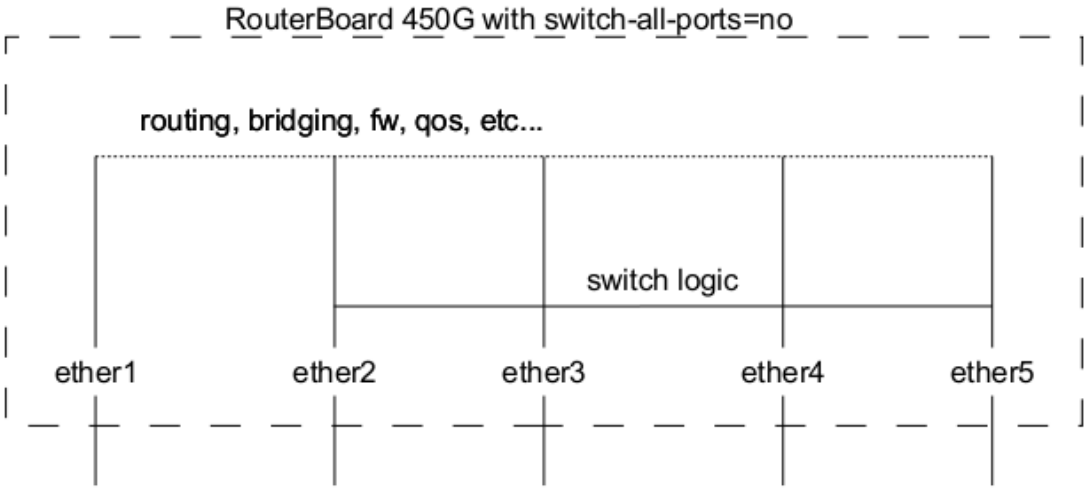
Here you can see that, a packet that gets received by one of the ports always passes through the switch logic at first. Switch logic decides to which ports the packet should be going to. Passing packet 'up' or giving it to RouterOS is also called sending it to switch chips 'cpu' port. That means that at the point switch forwards the packet to cpu port the packet starts to get processed by RouterOS as some interfaces incoming packet. While the packet does not have to go to cpu port it is handled entirely by switch logic and does not require any cpu cycles and happen at wire speed for any frame size.

Ether1 port on RB450G has a feature that allows it to be removed/added to the default switch group. By default ether1 port will be included in the switch group. This configuration can be changed with `/interface ethernet switch set switch1 switch-all-ports=no`

- **switch-all-ports**=yes/no -

"yes" means ether1 is part of switch and supports switch grouping, and all other advanced Atheros8316 features including extended statistics (`/interface ethernet print stats`).

"no" means ether1 is not part of switch, effectivly making it as stand alone ethernet port, this way increasing its troughtput to other ports in bridged, and routed mode, but removing the switching possibility on this port.



Port Mirroring

Port mirroring lets switch 'sniff' all traffic that is going in and out of one port (mirror-source) and send a copy of those packets out of some other port (mirror-target). This feature can be used to easily set up a 'tap' device that receives all traffic that goes in/out of some specific port. Note that mirror-source and mirror-target ports have to belong to same switch. (See which port belong to which switch in `/interface ethernet switch port` menu). Also mirror-target can have a special 'cpu' value, which means that 'sniffed' packets should be sent out of switch chips cpu port. Port mirroring happens independently of switching groups that have or have not been set up.

Host Table

Basically the table represents switch chips internal mac address to port mapping. It can contain two kinds of entries: dynamic and static. Dynamic entries get added automatically, this is also called a learning process: when switch chip receives a packet from certain port, it adds the packets source mac address X and port it received the packet from to host table, so when a packet comes in with destination mac address X it knows to which port it should forward the packet. If the destination mac address is not present in host table then it forwards the packet to all ports in the group. Dynamic entries take about 5 minutes to time out. Learning is enabled only on ports that are configured as part of switch group. So you won't see dynamic entries if you have not specified some 'master-ports'. Also you can add static entries that take over dynamic if dynamic entry with same mac-address already exists. Also by adding a static entry you get access to some more functionality that is controlled via following params:

- **copy-to-cpu**=yes/no - a packet can be cloned and sent to cpu port
- **redirect-to-cpu**=yes/no - a packet can be redirected to cpu port
- **mirror**=yes/no - a packet can be cloned and sent to mirror-target port configured in "/interface ethernet switch"
- **drop**=yes/no - a packet with certain mac address coming from certain ports can be dropped

copy-to-cpu, redirect-to-cpu, mirror actions are performed for packets which destination mac matches mac address specified in entry drop action is performed for packets which source mac address matches mac address specified in entry

Another possibility for static entries is that mac address can be mapped to more that one port, including 'cpu' port.

Vlan Table

Vlan tables specifies certain forwarding rules for packets that have specific 802.1q tag. Those rules are of higher priority than switch groups configured using 'master-port' property. Basically the table contains entries that map specific vlan tag ids to a group of one or more ports. Packets with vlan tags leave switch chip through one or more ports that are set in corresponding table entry. The exact logic that controls how packets with vlan tags are treated is controlled by vlan-mode parameter that is changeable per switch port in `/interface ethernet switch port` menu. Vlan-mode can take following values:

- **disabled** - ignore vlan table, treat packet with vlan tags just as if they did not contain a vlan tag;
- **fallback** - the default mode - handle packets with vlan tag that is not present in vlan table just like packets without vlan tag. Packets with vlan tags that are present in vlan table, but incoming port does not match any port in vlan table entry does not get dropped.
- **check** - drop packets with vlan tag that is not present in vlan table. Packets with vlan tags that are present in vlan table, but incoming port does not match any port in vlan table entry does not get dropped.
- **secure** - drop packets with vlan tag that is not present in vlan table. Packets with vlan tags that are present in vlan table, but incoming port does not match any port in vlan table entry get dropped.

Vlan tag id based forwarding also take into account the mac addresses learned or manually added in host table.

Packets without vlan tag are treated just like if they had a vlan tag with vlan id = 0. This means that if "vlan-mode=check or secure" to be able to forward packets without vlan tags you have to add a special entry to vlan table with vlan id set to 0.

Vlan-header option (configured in `/interface ethernet switch port`) sets the VLAN tag mode on egress port. This option works only with Atheros 8316 switch chip and takes the following values:

- **leave-as-is** - packet remains unchanged on egress port;
- **always-strip** - if VLAN header is present it is removed from the packet;
- **add-if-missing** - if VLAN header is not present it is added to the packet.

Rule Table

Rule table is very powerful tool allowing wire speed packet filtering, forwarding and vlan tagging based on L2,L3,L4 protocol header field condition.

Each rule contains a conditions part and an action part. Action part is controlled by following parameters:

- **copy-to-cpu**=yes/no - clones matching packets and sends them to cpu port;
- **redirect-to-cpu**=yes/no - redirects matching packets to cpu port;
- **mirror**=yes/no - clones matching packets and send them to mirror-target port;
- **new-dst-ports** - if set forces the destination port to be as specified, multiple ports allowed, including cpu port. Non obvious feature of this parameter is to pass empty list of ports to drop matching packets;
- **new-vlan-id** (only applies to Atheros8316) - if specified changes the vlan tag id, or add new vlan tag if one was not present;
- **new-vlan-priority** - if specified changes the vlan tag priority bits;
- **rate** (only applies to Atheros8327) - Sets limitation (bits per second) for all matched traffic. Can only be applied to first 32 rule slots.

Conditions part is controlled by rest of parameters:

- **ports** - match port that packet came in from (multiple ports allowed);
- **mac layer conditions**
  - *dst-mac-address* - match by destination mac address and mask;
  - *src-mac-address* - ...;

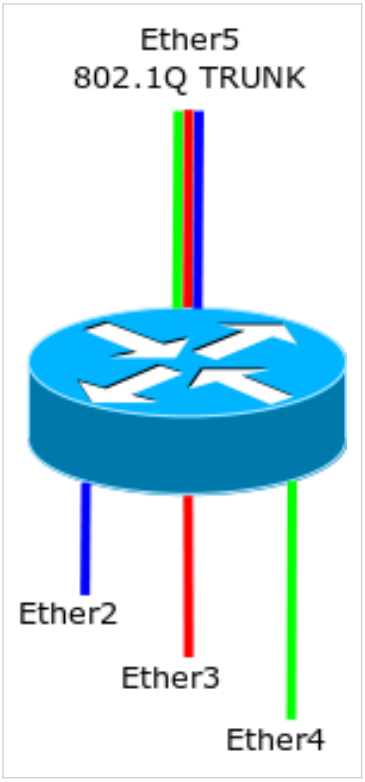
- *vlan-header* - match by vlan header presence;
- *vlan-id* (only applies to Atheros8316) - match by vlan tag id;
- *vlan-priority* (only applies to Atheros8316) - match by priority in vlan tag;
- *mac-protocol* - match by mac protocol (skips vlan tags if any);

- **ip conditions**
  - *dst-address* - match by destination ip and mask;
  - *src-address* - match by source ip and mask;
  - *dscp* - match by ip dscp field;
  - *protocol* - match by ip protocol;
- **ipv6 conditions**
  - *dst-address6* - match by destination ip and mask;
  - *src-address6* - match by source ip and mask;
  - *flow-label* - match by ipv6 flow label;
  - *traffic-class* - match by ipv6 traffic class;
  - *protocol* - match by ip protocol;
- **L4 conditions**
  - *src-port* - match by tcp/udp source port range;
  - *dst-port* - match by tcp/udp destination port range;

IPv4 and IPv6 specific conditions cannot be present in same rule. Menu contains ordered list of rules just like in `/ip firewall filter`. Due to the fact that the rule table is processed entirely in switch chips hardware there is limitation to how many rules you may have. Depending on the amount of conditions (MAC layer, IP layer, IPv6, L4 layer) you use in your rules the amount of active rules may vary from 8 to 32 for Atheros8316 switch chip and from 24 to 96 for Atheros8327 switch chip. You can always do `/interface ethernet switch rule print` after modifying your rule set to see that no rules at the end of the list are 'invalid' which means those rules did not fit into the switch chip.

Example - 802.1Q Trunking with Atheros 8316

Routerboards with Atheros 8316 switch chips can be used for 802.1Q Trunking. In this example ether2,ether3 and ether4 interfaces are access ports, while ether5 is trunk port. VLAN-IDs for each access port: ether2 - 200, ether3 - 300, ether4 - 400.



- Assign vlan-mode and vlan-header mode on egress for each port. Set vlan-mode=secure to ensure strict use of VLAN table. Set vlan-header=always-strip for access ports - it removes VLAN header from frame when it leaves the switch chip. Set vlan-header=add-if-missing for trunk port - it adds VLAN header to untagged frames.

```
/interface ethernet switch port
set 1 vlan-header=always-strip vlan-mode=secure
set 2 vlan-header=always-strip vlan-mode=secure
set 3 vlan-header=always-strip vlan-mode=secure
set 4 vlan-header=add-if-missing vlan-mode=secure
```

- Specify forwarding rules for access and trunk ports. Default VLAN-ID will be changed for untagged frames entering access port before they will be forwarded to trunk port. Frames received on trunk port will be forwarded to appropriate access ports according to their VLAN-ID.

```
/interface ethernet switch rule
add new-dst-ports=ether2 ports=ether5 switch=switch1 vlan-header=present vlan-id=200
add new-dst-ports=ether3 ports=ether5 switch=switch1 vlan-header=present vlan-id=300
add new-dst-ports=ether4 ports=ether5 switch=switch1 vlan-header=present vlan-id=400
add new-dst-ports=ether5 new-vlan-id=200 ports=ether2 switch=switch1 vlan-header=not-present
add new-dst-ports=ether5 new-vlan-id=300 ports=ether3 switch=switch1 vlan-header=not-present
add new-dst-ports=ether5 new-vlan-id=400 ports=ether4 switch=switch1 vlan-header=not-present
```

- Add VLAN table entries to allow frames with specific VLAN-IDs on ingress port.

```
/interface ethernet switch vlan
add ports=ether5 switch=switch1 vlan-id=200
add ports=ether5 switch=switch1 vlan-id=300
add ports=ether5 switch=switch1 vlan-id=400
add ports=ether2,ether3,ether4 switch=switch1 vlan-id=0
```

References

[1] http://wiki.mikrotik.com/wiki/Manual:Switch_Chip_Features#switch-all-ports
[2] http://wiki.mikrotik.com/wiki/Manual:Packet_flow_through_Atheros8316

Manual:System/File

Applies to RouterOS: v3, v4 +

Summary

Sub-menu level: `/file`

File menu shows all user space files on the router. It is possible to see and edit file content or delete file. file creation is not possible from this menu, to create files see scripting examples for workaround.

If RouterOS ".npk" package is uploaded, file menu will also show package specific information, like architecture, build date and time, etc.

File content example:

```
[admin@dzeltenais_burkaans] /file> print
 # NAME                          TYPE                    SIZE                CREATION-TIME
 0 autosupout.rif                .rif file               357368              oct/05/2010 09:47:01
 1 sample.txt                    .txt file               230                 oct/11/2010 12:14:43
[admin@dzeltenais_burkaans] /file> set 1 contents=Hello

[admin@dzeltenais_burkaans] /file> print detail where name~"sample"
 1 name="sample.txt" type=".txt file" size=5 creation-time=oct/11/2010 12:15:38 contents=Hello
[admin@dzeltenais_burkaans] /file>
```

Package example:

```
[admin@493G] /file> print detail
 0 name="multicast-5.0rc2-mipsbe.npk" type="package" size=245643 creation-time=jan/05/1970 21:44:25
   package-name="multicast" package-version="5.0rc2" package-build-time=oct/11/2010 06:34:02
   package-architecture="mips"
```

Properties

Property	Description
contents (*string*; Default:)	Actual content of the file.

Read-only properties

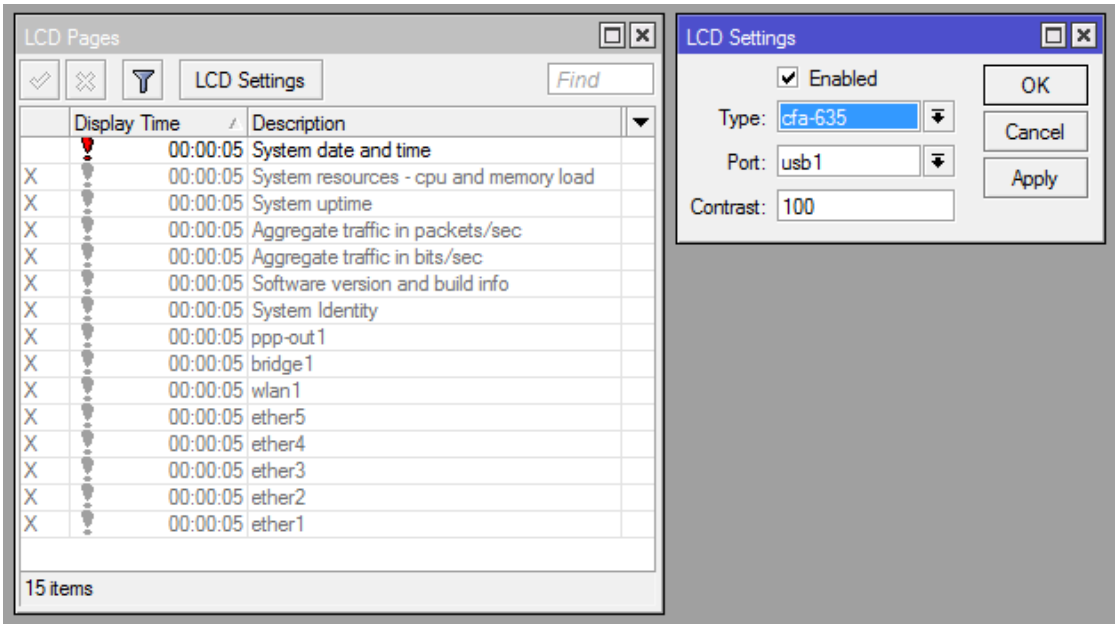
Property	Description
creation-time (*time*)	Time when file was created
name (*string*)	Name of the file
package-architecture (*string*)	Architecture that package is built for. Applies only to RouterOS ".npk" files.
package-built-time (*string*)	Time when package was built. Applies only to RouterOS ".npk" files.
package-name (*string*)	Name of the installable package that. Applies only to RouterOS ".npk" files.
package-version (*string*)	version of the installable package that. Applies only to RouterOS ".npk" files.
size (*integer*)	File size in bytes
file type (*string*)	Type of the file. For folders file type is *directory*

read more

- Scripting examples
- RouterOS upgrade

Manual:System/LCD

Applies to RouterOS: v3, v4, v5+



Summary

Sub-menu: `/system lcd`

Package: `lcd`

LCDs are used to display system information.

The MikroTik RouterOS supports the following LCD hardware:

- 16x2
- 20x4
- ax89063
- cfa-631
- mtb-134
- vitek-vc2025-1
- 16x4
- 24x2
- ax93304
- cfa-633
- nexcom
- vitek-vc2025-2
- 20x2
- 24x4
- ax93304n
- cfa-635
- tw-rc

- Crystalfontz (http://www.crystalfontz.com) Intelligent Serial LCD Module 632 (16x2 characters) and 634 (20x4 characters)
- Powertip (http://www.powertip.com.tw) PC1602 (16x2 characters), PC1604 (16x4 characters), PC2002 (20x2 characters), PC2004 (20x4 characters), PC2402 (24x2 characters) and PC2404 (24x4 characters)
- Portwell (http://www.portwell.com.tw) EZIO-100 (16x2 characters)
- Townet (http://www.townet.it/prodotti/remote-control/tw-rc.html) TW-RC REMOTE CONTROL (16x2)
- ax93304n (serial port) new ax93304 model with smaller screen buffer (since v5.8)
- nexcom (parallel port) (since v5.8)

Properties

Property	Description
contrast (*integer [0..255]*; Default: **0**)	Contrast setting, sent to the LCD, if it contrast regulation is supported
enabled (*yes* | *no*; Default: **no**)	Turn the LCD on/off
port (*string* | *parallel*; Default: **parallel**)	Name of the port where the LCD is connected. May be either one of the serial ports, or the first parallel port
type (*16x2* | *16x4* | *20x2* | *20x4* | *24x2* | *24x4* | *ax89063* | *ax93304* | *ax93304n* | *cfa-631* | *cfa-633* | *cfa-635* | *mtb-134* | *tw-rc* | *vitek-vc2025-1* | *vitek-vc2025-2*; Default: **24x4**)	Sets the type of the LCD • **cfa-631/633/635** - Crystalfontz • **mtb-134** - Portwell EZIO-100 • **tw-rc** - TowNet TW-RC • **vitek-vc2025-1/2** - Vitek parallel port LCDs • **ax89063/ax93304/ax93304n** - Axiomtek LCDs

Example

To enable Powertip parallel port LCD:

```
[admin@MikroTik] system lcd> print
   enabled: no
      type: 24x4
      port: parallel
  contrast: 0
[admin@MikroTik] system lcd> set enabled=yes
[admin@MikroTik] system lcd> print
   enabled: yes
      type: 24x4
      port: parallel
  contrast: 0
[admin@MikroTik] system lcd>
```

To enable Crystalfontz serial LCD on serial1:

```
[admin@MikroTik] system lcd> set port=serial1
[admin@MikroTik] system lcd> print
   enabled: yes
      type: 24x4
      port: serial1
  contrast: 0
[admin@MikroTik] system lcd>
```

LCD Information Display Configuration

Sub-menu: `/system lcd page`

The submenu is used to configure LCD information display: what pages and how long they will be shown. You cannot neither add your own pages (they are created dynamically depending on the configuration) nor change pages' description.

Pages will be displayed for specified amount of time starting from the first one.

Example

To display all the pages:

```
[admin@MikroTik] system lcd page> print
Flags: X - disabled
  #   DISPLAY-TIME         DESCRIPTION
  0 X 5s               System date and time
  1 X 5s               System resources - cpu and memory load
  2 X 5s               System uptime
  3 X 5s               Aggregate traffic in packets/sec
  4 X 5s               Aggregate traffic in bits/sec
  5 X 5s               Software version and build info
  6 X 5s               ether1
  7 X 5s               prism1
[admin@MikroTik] system lcd page> enable [find]
[admin@MikroTik] system lcd page> print
```

```
Flags: X - disabled
  #   DISPLAY-TIME         DESCRIPTION
  0   5s               System date and time
  1   5s               System resources - cpu and memory load
  2   5s               System uptime
  3   5s               Aggregate traffic in packets/sec
  4   5s               Aggregate traffic in bits/sec
  5   5s               Software version and build info
  6   5s               ether1
  7   5s               prism1
[admin@MikroTik] system lcd page>
```

To set "System date and time" page to be displayed for 10 seconds:

```
[admin@MikroTik] system lcd page> set 0 display-time=10s
[admin@MikroTik] system lcd page> print
Flags: X - disabled
  #   DISPLAY-TIME         DESCRIPTION
  0   10s               System date and time
  1   5s               System resources - cpu and memory load
  2   5s               System uptime
  3   5s               Aggregate traffic in packets/sec
  4   5s               Aggregate traffic in bits/sec
  5   5s               Software version and build info
  6   5s               ether1
  7   5s               prism1
[admin@MikroTik] system lcd page>
```

Troubleshooting

LCD doesn't work, cannot be enabled by the '/system lcd set enabled=yes' command.

> Probably the selected serial port is used by PPP client or server, or by the serial console. Check the availability and use of the ports by examining the output of the /port print command. Alternatively, select another port for connecting the LCD, or free up the desired port by disabling the related resource

LCD doesn't work, does not show any information.

> Probably none of the information display items have been enabled. Use the /system lcd page set command to enable the display.

See Also

- LCD Touch Screen control

[Top | Back to Content]

Manual:System/Note

Applies to RouterOS: v3, v4, v5

Summary

Sub-menu level: `/system note`

System note feature allows you to assign arbitrary text notes or messages that will be displayed on each login right after banner. For example, you may distribute warnings between system administrators this way, or describe what does that particular router actually do. To configure system note, you may upload a plain text file named **sys-note.txt** on the router's FTP server, or, additionally, edit the settings in this menu

Properties

Property	Description
note (*string*; Default:)	Note that will be displayed.
show-at-login (*yes* | *no*; Default: **yes**)	whether to show system note on each login

Example

It is possible to add multi-line notes using embedded text editor (*/system note edit note*), for example add ASCII art to your home router:

```
[admin@RB493G] /system note> edit note

                            (   )
                           (    )
                            (    )
                           (    )
                             )  )
                            (  (                    /\
                              (_)                  /  \  /\
                        _________[_]_________      /\/    \/  \
           /\       /\         _______    \    /   /\/\  /\/\
          /  \     //_\         \     /\    \  /\/\/    \/    \
   /\    / /\/\  //___\         \__/  \    \/
  /  \  /\/    \//_____\         \ |[]|     \
 /\/\/\/        //_______\         \|__|      \
/       \       /XXXXXXXXXX\                   \
         \     /_I_II  I__I_\_____________________\
                I_I|  I__I_____[]_|_[]_____I
                I_II  I__I_____[]_|_[]_____I
                I II__I  I     XXXXXXX     I
```

```
            ~~~~~"   "~~~~~~~~~~~~~~~~~~~~~~~~

C-c quit C-o save&quit C-u undo C-k cut line C-y paste F5 repaint
```

Save changes by pressing **Ctrl+o**. Now next time you log in, console will print your piece of art:

```
  MMMM    MMMM       KKK                          TTTTTTTTTTT      KKK
  MMM MMMM MMM  III  KKK  KKK  RRRRRR     OOOOOO      TTT     III  KKK  KKK
  MMM  MM  MMM  III  KKKKK     RRR  RRR  OOO  OOO     TTT     III  KKKKK
  MMM      MMM  III  KKK KKK   RRRRRR    OOO  OOO     TTT     III  KKK KKK
  MMM      MMM  III  KKK  KKK  RRR  RRR   OOOOOO      TTT     III  KKK  KKK

  MikroTik RouterOS 5.3 (c) 1999-2011       http://www.mikrotik.com/









                          (   )
                         (    )
                          (    )
                         (    )
                           )  )
                          (  (                  /\
                            (_)                /  \  /\
                   _________[_]_________      /\/    \/  \
          /\      /\        _______    \    /   /\/\  /\/\
         /  \    //_\       \    /\    \  /\/\/    \/    \
   /\    / /\/\  //___\       \__/  \    \/
  /  \  /\/    \//_____\       \ |[]|     \
 /\/\/\/       //_______\       \|__|      \
/      \      /XXXXXXXXXX\                  \
        \    /_I_II  I__I_\_____________________\
               I_I|  I__I_____[]_|_[]_____I
               I_II  I__I_____[]_|_[]_____I
               I II__I  I     XXXXXXX     I
            ~~~~~"   "~~~~~~~~~~~~~~~~~~~~~~~~
[admin@RB493G] >
```

[Top | Back to Content]

Manual:System/Resource

Applies to RouterOS: v3, v4 +

General

Sub-menu level: `/system resource`

General resource menu shows overall resource usage and router statistics like uptime, memory usage, disk usage, version etc.

It also has several sub-menus for more detailed hardware statistics like PCI, IRQ and USB.

```
[admin@RB1100test] /system resource> print
                   uptime: 2w1d23h34m57s
                  version: "5.0rc1"
             free-memory: 385272KiB
            total-memory: 516708KiB
                      cpu: "e500v2"
                cpu-count: 1
            cpu-frequency: 799MHz
                 cpu-load: 9%
           free-hdd-space: 466328KiB
          total-hdd-space: 520192KiB
  write-sect-since-reboot: 1411
         write-sect-total: 70625
               bad-blocks: 0.2%
        architecture-name: "powerpc"
               board-name: "RB1100"
                 platform: "MikroTik"
```

Properties

All properties are read-only

Property	Description
architecture-name (*string*)	CPU architecture. Can be *powerpc*, *x86*, *mipsbe* or *mipsle*.
bad-blocks (*percent*)	Shows percentage of bad blocks on the NAND.
board-name (*string*)	RouterBOARD model name
cpu (*string*)	Cpu model that is on the board.
cpu-count (*integer*)	Number of CPUs present on the system. Each core is separate CPU, Intel HT is also separate CPU.
cpu-frequency (*string*)	Current CPU frequency.
cpu-load (*percent*)	Percentage of used CPU resources. Combines all CPUs. Per-core CPU usage can be see in CPU submenu
free-hdd-space (*string*)	Free space on hard drive or NAND

free-memory (*string*)	Unused amount of RAM
platform (*string*)	Platform name, usually it is "MikroTik"
total-hdd-space (*string*)	Size of the hard drive or NAND
total-memory (*string*)	Amount of installed RAM
uptime (*time*)	Time interval passed since boot-up.
version (*string*)	Installed RouterOS version number.
write-sect-since-reboot (*integer*)	Number of sector writes in HDD or nand since router was last time rebooted.
write-sect-total (*integer*)	Number of sector writes in total.

CPU

Sub-menu level: `/system resource cpu`

This submenu shows per-cpu usage, as long as IRQ and Disk usage.

```
[admin@RB1100test] /system resource cpu> print
CPU LOAD         IRQ          DISK
0   5%           0%           0%
[admin@RB1100test] /system resource cpu>
```

(needs editing)

Properties

Read-only properties

Property	Description
cpu (*integer*)	Identification number of CPU which usage is shown.
load (*percent*)	CPU usage in percents
irq (*percent*)	IRQ usage in percents
disk (*percent*)	Disk usage in percents

IRQ

Sub-menu level: `/system resource irq`

Menu shows all used IRQs on the router. It is possible to set up IRQ load balancing on mulicore systems by assigning IRQ to specific core. IRQ assignments are done by hardware and cannot be changed from RouterOS. For example, if all Ethernets are assigned to one IRQ, then you have to deal with hardware: upgrade motherboards BIOS, reassign IRQs manually in BIOS, if none of above helps then change the hardware.

Properties

Property	Description
cpu (*auto* | *integer*; Default:)	Specifies which CPU is assigned to the IRQ. • **auto** - pick CPU based on number of interrupts. Uses NAPI [1] to optimize interrupts.

Read-only properties

Property	Description
active-cpu (*integer*)	Shows active CPU in multicore systems.
count (*integer*)	Number of interrupts. On ethernet interfaces interrupt=packet.
irq (*integer*)	IRQ identification number
users (*string*)	Process assigned to IRQ

RPS

Sub-menu level: `/system resource rps`

This menu allows to enable Receive Packet Steering (RPS) to reduce single core usage.

NAPI [1] can become a bottleneck under high packet load, because of serialization per device queue. RPS distributes the load of received packet processing across multiple cores.

USB

Sub-menu level: `/system resource usb`

This menu displays all available USB controllers on the board. Menu is available only if at least one USB controller is present.

```
[admin@MikroTik] /system resource usb> print detail
 0 device="2:1" name="RB400 EHCI" serial-number="rb400_usb" vendor-id="0x1d6b"
   device-id="0x0002" speed="480 Mbps" ports=2 usb-version="2.00"

 1 device="1:1" name="RB400 OHCI" serial-number="rb400_usb" vendor-id="0x1d6b"
   device-id="0x0001" speed="12 Mbps" ports=2 usb-version="1.10"
```

Properties

Property	Description
device (*string*)	
device-id (*hex*)	Hexadecimal device ID
name (*string*)	Descriptive name of the device retrieved from driver
ports (*integer*)	How many ports are supported by usb controller
serial-number (*string*)	
speed (*string*)	Max USB speed that can be used (480Mbps for USBv2 and 12Mbps for USBv1)
usb-version (*string*)	Identifies max spported USB version
vendor (*string*)	Device manufacturer's name.

vendor-id (*hex*)	Hexadecimal vendor ID	

PCI

Sub-menu level: `/system resource pci`

PCI submenu shows the information about all PCI devices on the board

```
[admin@RB1100test] /system resource pci> print
 # DEVICE   VENDOR                        NAME                        IRQ
 0 06:00.0  Attansic Technology Corp.     unknown device (rev: 192)  18
 1 05:00.0  Freescale Semiconductor Inc   MPC8544 (rev: 17)          0
 2 04:00.0  Attansic Technology Corp.     unknown device (rev: 192)  17
 3 03:00.0  Freescale Semiconductor Inc   MPC8544 (rev: 17)          0
 4 02:00.0  Attansic Technology Corp.     unknown device (rev: 192)  16
 5 01:00.0  Freescale Semiconductor Inc   MPC8544 (rev: 17)          0
 6 00:00.0  Freescale Semiconductor Inc   MPC8544 (rev: 17)          0
```

Properties

All properties are read-only

Property	Description
category (*string*)	PCI device type, for example *ethernet controller*
device (*string*)	
device-id (*hex*)	Hexadecimal device ID
io (*hex-hex*)	I/O memory range
irq (*integer*)	IRQ asigned to the device
memory (*hex-hex*)	Memory range
name (*string*)	Descriptive name of the device retrieved from driver
vendor (*string*)	Device manufacturer's name.
vendor-id (*hex*)	Hexadecimal vendor ID

References

[1] http://www.linuxfoundation.org/collaborate/workgroups/networking/napi

Manual:System/Serial Console

Applies to RouterOS: v3, v4, v5+

Overview

Sub-menu: `/system console, /system serial-terminal`
Standards: `RS-232`

The Serial Console and Terminal are tools, used to communicate with devices and other systems that are interconnected via serial port. The serial terminal may be used to monitor and configure many devices - including modems, network devices (including MikroTik routers), and any device that can be connected to a serial (asynchronous) port.

The Serial Console feature is for configuring direct-access configuration facilities (monitor/keyboard and serial port) that are mostly used for initial or recovery configuration.

If you do not plan to use a serial port for accessing another device or for data connection through a modem, you can configure it as a serial console. The first serial port is configured as a serial console, but you can choose to unconfigure it to free it for other applications. A free serial port can also be used to access other routers' (or other equipment, like switches) serial consoles from a MikroTik RouterOS router. A special null-modem cable is needed to connect two hosts (like, two PCs, or two routers; not modems). Note that a terminal emulation program (e.g., HyperTerminal on Windows or minicom on linux) is required to access the serial console from another computer.

Several customers have described situations where the Serial Terminal (managing side) feature would be useful:

- on a mountaintop, where a MikroTik wireless installation sits next to equipment (including switches and Cisco routers) that can not be managed in-band (by telnet through an IP network)
- monitoring weather-reporting equipment through a serial port
- connection to a high-speed microwave modem that needed to be monitored and managed by a serial connection

With the serial-terminal feature of the MikroTik, up to 132 (and, maybe, even more) devices can be monitored and controlled.

Serial Console Configuration

A special null-modem cable should be used for connecting to the serial console from another computer. The Serial Console cabling diagram for DB9 connectors is as follows:

Router Side (DB9f)	Signal	Direction	Side (DB9f)
1, 6	CD, DSR	IN	4
2	RxD	IN	3
3	TxD	OUT	2
4	DTR	OUT	1, 6
5	GND	-	5
7	RTS	OUT	8
8	CTS	IN	7

Note that the above diagram will not work if the software is configured to do hardware flow control, but the hardware does not support it (e.g., some RouterBOARD models have reduced seral port functionality). If this is the case, either turn off the hardware flow control or use a null-modem cable with loopback, which will simulate the other device's handshake signals with it's own. The diagram for such cable is as follows:

Router Side (DB9f)	Signal	Direction	Side (DB9f)
1, 4, 6	CD, DTR, DSR	LOOP	1, 4, 6
2	RxD	IN	3
3	TxD	OUT	2
5	GND	-	5
7, 8	RTS, CTS	LOOP	7, 8

Note that although it is recommended to have 5-wire cable for this connection, in many cases it is enough to have 3 wires (for unlooped signals only), leaving both loops to exist only inside the connectors. Other connection schemes exist as well.

Configuring Console

Sub-menu: `/system console`

Properties

Property	Description
disabled (*yes* | *no*; Default: **no**)	Whether serial console is enabled or not.
port (*string*)	Which port should the serial terminal listen to
term (*string*)	Terminal type

Read-only properties

Property	Description
free (*yes* | *no*)	Console is ready for use.
used (*yes* | *no*)	Console is in use.
vcno (*integer*)	number of virtual console - [Alt]+[F1] represents '1', [Alt]+[F2] - '2', etc..
wedged (*yes* | *no*)	Console is currently not available

Example

To disable all virtual consoles (available through the direct connection with keyboard and monitor) extept for the first one:

```
[admin@MikroTik] system console> print
Flags: X - disabled, W - wedged, U - used, F - free
 #   PORT    VCNO       TERM
 0 F serial0            MyConsole
 1 U         1          linux
 2 F         2          linux
 3 F         3          linux
 4 F         4          linux
 5 F         5          linux
 6 F         6          linux
 7 F         7          linux
 8 F         8          linux
[admin@MikroTik] system console> disable 2,3,4,5,6,7,8
[admin@MikroTik] system console> print
Flags: X - disabled, W - wedged, U - used, F - free
 #   PORT    VCNO       TERM
 0 F serial0            MyConsole
 1 U         1          linux
 2 X         2          linux
 3 X         3          linux
 4 X         4          linux
 5 X         5          linux
 6 X         6          linux
 7 X         7          linux
 8 X         8          linux
[admin@MikroTik] system console>
```

To check if the configuration of the serial port:

```
[admin@MikroTik] system serial-console> /port print detail
  0 name=serial0 used-by=Serial Console baud-rate=9600 data-bits=8 parity=none
    stop-bits=1 flow-control=none

  1 name=serial1 used-by="" baud-rate=9600 data-bits=8 parity=none stop-bits=1
    flow-control=none

[admin@MikroTik] system serial-console>
```

Using Serial Terminal

Command: `/system serial-terminal`

The command is used to communicate with devices and other systems that are connected to the router via serial port.

All keyboard input is forwarded to the serial port and all data from the port is output to the connected device. *Ctrl-A* is the prefix key, which means that you will enter a small "menu" by pressing this combination of keys. The *Ctrl-A* character will not be sent to your device! If you need to send *Ctrl-A* character to remote device, press *Ctrl-A* twice. After exiting with *Ctrl-A* and then *Q*, the control signals of the port are lowered. The speed and other parameters of serial port may be configured in the /port directory of router console. No terminal translation on printed data is performed. It is possible to get the terminal in an unusable state by outputting sequences of inappropriate control characters or random data. Do not connect to devices at an incorrect speed and avoid dumping binary data.

Property	Description
port (*string*; Default:)	Port name to use

The serial port to be used as a serial terminal needs to be free (e.g., there should not be any serial consoles, LCD or other configuration). Chack the previous chapter to see how to disable serial console on a particular port. Use /port print command to see if some other application is still using the port.

Ctrl-A have special meaning and is used to provide a possibility of exiting from nested serial-terminal sessions:

To send *Ctrl-A* to to serial port, press *Ctrl-A Ctrl-A*

Note: When rebooting a RouterBoard the bootloader (RouterBOOT) will always use the serial console (serial0 on RouterBoards) to send out some startup messages and offer access to the RouterBOOT menu. Having text coming out of the serial port to the connected device might confuse your attached device and get stuck on boot loader. To avoid this you can reconfigure RouterBOOT to enter the RouterBOOT menu only when a **DEL** character is received.

Example

To connect to a device connected to the serial1 port:

```
[admin@dzeltenais_burkaans] > /system serial-terminal serial0

[Ctrl-A is the prefix key]

[admin@R2] /ip address>
```

Console Screen

Sub-menu: `/system console screen`

This facility is created to change line number per screen if you have a monitor connected to router.

Property	Description
line count (*25|40|50*; Default: **25**)	Number of lines on monitor

This parameter is applied only to a monitor, connected to the router.

Example

To set monitor's resolution from 80x25 to 80x40:

```
[admin@MikroTik] system console screen> set line-count=40
[admin@MikroTik] system console screen> print
    line-count: 40
[admin@MikroTik] system console screen>
```

See More

- Special Login
- Sigwatch

[Top | Back to Content]

Manual:IP/SSH

Applies to RouterOS: v5

Summary

This menu controls if ssh port forwarding is/is not allowed on the router. Enabling this feature remote hosts are able to create ssl encrypted connection from remote host to router port.

Settings

Property	Desciption
forwarding-enabled *(no|yes default:no)*	controls ssh port forwarding

Example

To use this feature from Linux host using OpenSSH client this command can be used:

```
 ssh reamoteuser@remotehost -L port:remotehost:remoteport
```

where:

- remoteuser - user of router
- remotehost - router address (if host name is used in -L settings, router should be able to resolve this name)
- port - local port that your host will listen on
- remoteport - port on the router

If user requires telnet to router, but you do not want to allow it to be plain text, Following can be done:

```
ssh admin@192.168.88.1 -L 3000:192.168.88.1:23
```

now when user uses *telnet localhost 3000" it will log in the router using telnet over encrypted tcp connection.*

Note: we fully support SFTP v3 as described in draft-ietf-secsh-filexfer-02.txt [1] other versions can cause problems

References

[1] http://tools.ietf.org/wg/secsh/draft-ietf-secsh-filexfer/draft-ietf-secsh-filexfer-02.txt

Manual:IP/TFTP

Summary

Standards: `RFC 1350 RFC 2348`

TFTP is a very simple protocol used to transfer files. It is from this that its name comes, Trivial File Transfer Protocol or TFTP. Each nonterminal packet is acknowledged separately. RouterOS has a built-in TFTP server since v3.22

Warning: Since version 4.4 to set up tftp rules you will **have to have** policy ***sensitive*** enabled for your account.

Note: since RouterOS 5.6 sequence number roll-over is supported by TFTP server. That has to be set specifically for TFTP rule that allows it.

TFTP access rules

Menu: ***/ip tftp*** This menu contains all TFTP access rules. If in this menu are no rules, TFTP server is not started when RouterOS boots. This menu only shows 1 additional attribute compared to what you can set when creating rule, see explanations of attribute descriptions lower.

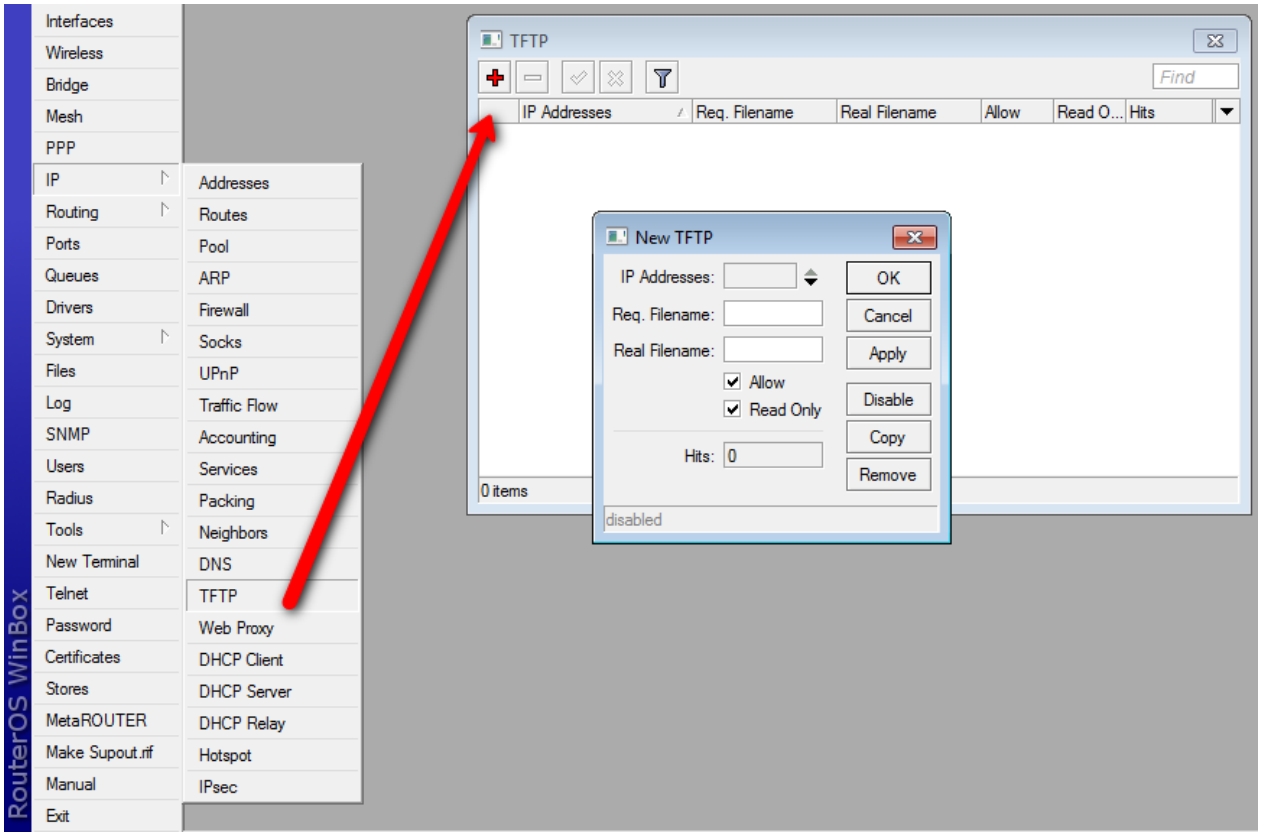
Property	Desciption
hits	how many times this access rule entry has been used (read-only)

Add new access rule

Expansion of command: */ip tftp add*

To add new tftp access rule you will have to issue command *add* under */ip tftp* menu with attributes as follows:

Property	Desciption
ip-address *(required)*	range of IP addresses accepted as clients if empty *0.0.0.0/0* will be used
allow-rollover *(Default:No)*	if set to yes TFTP server will allow sequence number to roll over when maximum value is reached. This is used to enable large downloads using TFTP server.
req-filename	requested filename as *regular expression (regex)* if field is left empty it defaults to .*
real-filename	if **req-filename** and **real-filename** values are set and valid, the requested filename will be replaced with matched file. This field has to be set. If multiple *regex* are specified in *req-filename*, with this field you can set which ones should match, so this rule is validated. *real-filename* format for using multiple *regex* is **filename\0\5\6**
allow (*default: yes*)	to allow connection if above fields are set. if *no*, connection will be interrupted
read-only (*default: no*)	sets if file can be written to, if set to "no" write attempt will fail with error



req-filename field allowed regexp

allowed regexps in this field are

- **brackets ()** - marking subsection

```
example 1 a(sd|fg) will match asd or afg
```

- **asterisk "*"** - match zero or more times preceding symbol,

```
example 1 a* will match any length name consisting purely of symbols a or no symbols at all
example 2 .* will match any length name, also, empty field
example 3 as*df will match adf, asdf, assdf, asssdf etc.
```

- **plus "+"** will match one or more times preceding symbol,

```
    example: as+df will match asdf, assdf etc.
```

- **dot "."** - matches any symbol

```
    example as.f will match asdf, asbf ashf etc.
```

- **square brackets []** - variation between

```
    example as[df] will match asd and asf
```

- **question mark "?"** will match one or none symbols,

```
    example asd?f will match asdf and asf
```

- **caret "^"** - used at the beginning of the line means that line starts with,
- **dollar "$"** - means at the end of the line

Examples

- **example 1** if *file* is requested return file from store called *sata1*:

```
/ip tftp add req-filename=file.txt real-filename=/sata1/file.txt allow=yes read-only=yes
```

- **example 2** if we want to give out one specific *file* no matter what user is requesting:

```
/ip tftp add req-filename=.* real-filename=/sata1/file.txt allow=yes read-only=yes
```

- **example 3** if user requests *aaa.bin* or *bbb.bin* then give them *ccc.bin*:

```
/ip tftp add req-filename="(aaa.bin)|(bbb.bin)" real-filename="/sata1/ccc.bin\\0" allow=yes read-only=yes
```

Manual:System/Time

Applies to RouterOS: v3, v4

Clock and Time zone configuration

RouterOS uses data from the tz database [1], Most of the time zones from this database are included, and have the same names. Because local time on the router is used mostly for timestamping and time-dependant configuration, and not for historical date calculations, time zone information about past years is not included. Currently only information starting from 2005 is included.

Following settings are available in the **/system clock** console path, and in the "Time" tab of the "System > Clock" WinBox window:

- **time** (*HH:MM:SS*, where *HH* - hour 00..24, *MM* - minutes 00..59, *SS* - seconds 00..59)
- **date** (*mmm/DD/YYYY*, where *mmm* - month, one of *jan*, *feb*, *mar*, *apr*, *may*, *jun*, *jul*, *aug*, *sep*, *oct*, *nov*, *dec*, *DD* - date, 00..31, *YYYY* - year, 1970..2037) : **date** and **time** show current local time on the router. These values can be adjusted using the **set** command. Local time cannot, however, be exported, and is not stored with the rest of the configuration.
- **time-zone-name** (*manual*, or name of time zone; default value: *manual*) : Name of time zone. Like most of the text values in RouterOS, this value is case sensitive. Special value *manual* applies manually configured GMT offset, which by default is *00:00* with no daylight saving time.

Startup date and time is *jan/02/1970 00:00:00 [+|-]gmt-offset*. If router has a battery (for example RB230), then BIOS stored time is used as a startup time.

Active time zone information

- **dst-active** (*yes* or *no*>; read-only property) : This property has value *yes* while daylight saving time of the current time zone is active.
- **gmt-offset** ([+|-]*HH:MM* - offset in hours and minutes; read-only property) : This is the current value of GMT offset used by the system, after applying base time zone offset and active daylight saving time offset.

Manual time zone configuration

These settings are available in **/system clock manual** console path, and in the "Manual Time Zone" tab of the "System > Clock" WinBox window. These settings have effect only when **time-zone-name=***manual*. It is only possible to manually configure single daylight saving time period.

- **time-zone**, **dst-delta** ([+|-]*HH:MM* - time offset in hours and minutes, leading plus sign is optional; default value: *+00:00*) : While DST is not active use GMT offset **time-zone**. While DST is active use GMT offset **time-zone** + **dst-delta**.
- **dst-start**, **dst-end** (*mmm/DD/YYYY HH:MM:SS* - date and time, either date or time can be ommited in the **set** command; default value: *jan/01/1970 00:00:00*) : Local time when DST starts and ends.

SNTP client

SNTP client is included in the *system* package. RouterOS implements SNTP protocol defined in RFC4330. Manycast mode is not supported. NTP server and a NTP client is included in the separate *ntp* package, that is not installed by default.

Client configuration is located in the **/system ntp client** console path, and the "System > NTP Client" WinBox window. This configuration is shared by the SNTP client implementation in the *system* package and the NTP client implementation in the *ntp* package. When *ntp* package is installed and enabled, the SNTP client is disabled automatically.

- **enabled** (*yes* or *no*; default value: *no*)
- **mode** (One of *broadcast* or *unicast*; default value: *broadcast*) : In broadcast mode, client does not send any requests, and listens for the broadcast messages sent by the NTP server. In unicast mode client periodically sends requests to the currently selected active server, and waits for a reply message from that server.
- **primary-ntp**, **secondary-ntp** (IP address) : IP addresses of the NTP servers. These properties have effect only when **mode**=*unicast*. Value *0.0.0.0* is ignored. If both values are zero and **mode** is *unicast* then SNTP client is disabled. If both values are non-zero, then SNTP client will alternate between the two server addresses, switching to the other when request to the current server times out or when the "KoD" packet is received, indicating that server is not willing to respond to requiests from this client.

Status

- **active-server** (IP address; read-only property) : Currently selected NTP server address. This value is equal to **primary-ntp** or **secondary-ntp**.
- **poll-interval** (Time interval; read-only property) : Current iterval between requests sent to the active server. Initial value is 16 seconds, and it is increased by doubling to 15 minutes.

Last received packet information

Values of the following properties are reset when the SNTP client is stopped or restarted, either because of a configuration change, or because of a network error.

- **last-update-from** (IP address; read-only property) : Source IP address of the last received NTP server packed that was successfully processed.
- **last-update-before** (Time interval; read-only property) : Time since the last successfully received server message.
- **last-adjustment** (Time interval; read-only property) : Amount of clock adjustment that was calculated from the last successfully received NTP server message.
- **last-bad-packet-from** (IP address; read-only property) : Source IP address of last received SNTP packed that was not successfully processed. Reason of the failure and time since this packet was received is available in the next two properties.
- **last-bad-packet-before** (Time interval; read-only property) : Time since the last receive failure.
- **last-bad-packet-reason** (Text; read-only property) : Text that describes reason of the last receive failure. Possible values are:
  - *bad-packet-length* - Packet length is not in the acceptable range.
  - *server-not-synchronized* - Leap Indicator field is set to "alarm condition" value, which means that clock on the server has not been synchronized yet.
  - *zero-transmit-timestamp* - Transmit Timestamp field value is 0.
  - *bad-mode* - Value of the Mode field is neither 'server' nor 'broadcast'.

- *kod-ABCD* - Received "KoD" (Kiss-o'-Death) response. *ABCD* is the short "kiss code" text from the Reference Identifier field.
- *broadcast* - Received proadcast message, but **mode=***unicast*.
- *non-broadcast* - Received packed was server reply, but **mode=***broadcast*.
- *server-ip-mismatch* - Received response from address that is not **active-server**.
- *originate-timestamp-mismatch* - Originate Timestamp field in the server response message is not the same as the one included in the last request.
- *roundtrip-too-long* - request/response roundtrip exceeded 1 second.

Log messages

SNTP client can produce the following log messages. See article "log" on how to set up logging and how to inspect logs.

- **ntp**,**debug** gradually adjust by *OFFS*
- **ntp**,**debug** instantly adjust by *OFFS*
- **ntp**,**debug** Wait for *N* seconds before sending next message
- **ntp**,**debug** Wait for *N* seconds before restarting
- **ntp**,**debug**,**packet** packet receive error, restarting
- **ntp**,**debug**,**packet** received *PKT*
- **ntp**,**debug**,**packet** ignoring received *PKT*
- **ntp**,**debug**,**packet** error sending to *IP*, restarting
- **ntp**,**debug**,**packet** sending to *IP PKT*

Explanation of log message fields

- *OFFS* - difference of two NTP timestamp values, in hexadecimal.
- *PKT* - dump of NTP packet. If packet is shorter than the minimum 48 bytes, it is dumped as a hexadecimal string. Otherwise, packet is dumped as a list of field names and values, one per log line. Names of fields follow RFC4330.
- *IP* - remote IP address.

NOTE: the above logging rules work only with the built-in SNTP client, the separate NTP package doesn't have any logging facilities.

NTP client and server

To use NTP client and server, *ntp* package must be installed and enabled.

Client settings

Client configuration is located in **/system ntp client**.

- **enabled** (*yes* or *no*; default value: *no*)
- **mode** (One of *broadcast*, *unicast*, *multicast* or *manycast*.)
- **primary-ntp**, **secondary-ntp** (IP address)

References

[1] http://www.twinsun.com/tz/tz-link.htm

Manual:Tools

List of reference sub-pages **Case studies** **List of examples**

<splist showparent=yes />

Manual:Tools/Traffic Generator

Applies to RouterOS: v5 +

Summary

Traffic Generator is a tool that allows to evaluate performance of DUT (Device Under Test) or SUT (System Under Test).

Tool can generate and send RAW packets over specific ports. It also collects latency and jitter values, tx/rx rates, counts lost packets and detects Out-of-Order (OOO) packets.

Traffic Generator can be used similar to bandwidth test tool as well as generate packets that will be routed back to packet generator for advanced status collection.

General

Sub-menu `/tool traffic-generator`

This menu allows to set general traffic generator properties and contains commands to quickly start and stop the tool.

Properties

Property	Description
latency-distribution-scale (*integer [0..28]*; Default: **10**)	
test-id (*integer [0..255]*; Default: **0**)	

Read-Only Properties

Property	Description
latency-distribution-samples (*integer*)	
latency-distribution-measure-interval (*time*)	
running (*yes* | *no*)	Shows whether traffic generator tool is started.

Commands

Property	Description
quick ()	This command allows to quickly start packet generator and print the stats output to the terminal. Command also accept several parameters that overrides settings in packet template and stream settings. Accepted parameters are **duration, entries-to-show, freeze-frame-interval, mbps, num, packet-size, port, pps, stream, test-id, tx-template** • `duration` - how long to run the test • `entries-to-show` - how many status lines print to the terminal • `freeze-frame-interval` - how often to update status to the terminal The rest of the parameters are not command specific and are described elsewhere. Parameters specified when running quick command overrides configured values. In case if parameter is specified only for one header then value is multiplied for all the other headers (if required).
start ()	Commands starts the traffic generator tool in the background. It accepts one parameter **test-id**
stop ()	Command stops the started traffic generator tool by **start** command.

Packet Template

Sub-menu `/tool traffic-generator packet-template`

This sub menu allows to build packet based on provided parameters. Based on parameters you can build ip packet with vlan tags and set udp ports. Raw packet template is generated based on provided parameters.

If you require more low level packet or take full advantage of traffic generator, then please use raw-packet-template builder to build the packet.

If same type of header is present in packet more than once then header field values are passed as comma separated list. (For example if there are two ip headers then source addresses are given like "ip-src=1.1.1.1,2.2.2.2").

For quicker header construction many of the header field values are assumed. For example if header stack is "mac,ip" then traffic generator can assume that mac-protocol value is "ip". Or if "port" or "interface" setting is specified traffic generator can assume "mac-src" to be MAC address of interface). Assumed values have distinct names that start with "assumed-" and are read only. Manually specified values override assumed ones.

Note: Assumed values are not automatically updated. New values are assumed after template edit. "packet-template set 0" is enough to trigger new assumed values

Properties

Property	Description
comment (*string*; Default:)	Short description of packet you are building.
data (*incrementing* | *random* | *specific-byte* | *uninitialized*; Default: **uninitialized**)	Specifies how packet payload will be filled: • **uninitialized** - packets data (after header) is uninitialized, but not zero. Fastest. • **specific-byte** - works together with setting data-byte • **incrementing** - packets data filled with "00 01 02 03" and so on • **random** - packets data filled with random bytes. Slowest.
data-byte (*hex [0..FF]]*; Default: **0**)	Byte that will be used to fill packet payload.
interface (*string*; Default:)	Optional parameter of packet template. This is mutually exclusive with "port" setting. Specifying "interface" allows user not to create a port entry for interface in port menu. In fact a port entry is created dynamically. This is useful for running quick tests.
ip-dscp (*list of integer[0..255] (max 16 times)*; Default:)	Single DSCP or list of DSCP values that will be set in IP header
ip-dst (*list of IP/Netmask (max 16 times)*; Default:)	List of destination IP addresses that will be used when generating IP headers.
ip-frag-off (*list of integer[0..65535] (max 16 times)*; Default:)	List of fragmentation offsets in IP header.
ip-gateway (*IP*; Default:)	In situations when sender and receiver is the same device, it is impossible to determine nexthop automatically from **ip-dst**. If ip-gateway is specified packet template will assume destination mac address based on resolved ip-gateway.
ip-id (*list of integer [0..65535]*; Default:)	
ip-protocol (*list of IP protocols (max 16 times)*; Default:)	
ip-src (*list of IP/Mask (max 16 times)*; Default:)	
ip-ttl (*list of integer [0..255] (max 16 times)*; Default:)	
mac-dst (*list of MAC/MASK (max 16 times)*; Default:)	
mac-protocol (*list of mac protocols (max 16 times)*; Default:)	
mac-src (*list of MAC/MASK (max 16 times)*; Default:)	
name (*string*; Default:)	Descriptive name of the template.
port (*string*; Default:)	Optional parameter of packet template. This suggests a port through which packets generated using this template should be sent out. Port can also be specified in other places such as in stream settings. This is mutually exclusive with interface setting.
raw-header (*string (max 16 times)*; Default:)	Raw packet header as string in hexadecimal format.
udp-dst-port (*list of port [0..65535]/mask [0..FFFF] (max 16 times)*; Default:)	
udp-src-port (*list of port [0..65535]/mask [0..FFFF] (max 16 times)*; Default:)	

vlan-id (; Default:)

vlan-priority (; Default:)

vlan-protocol (; Default:)

header-stack (*list of ip | mac | raw | udp | vlan (max 16 times)*; Default: **ip**)

Sequence of headers that a generated packet should have.

Currently supports:

- **mac** - Ethernet header (14 bytes)
- **vlan** - Ethernet VLAN tag (4 bytes)
- **ip** - IPv4 header (20 bytes)
- **udp** - UDP header (8 bytes)
- **raw** - arbitrary bytes specified as hex string

Most header types can be present in header multiple times. There can be only 2 ip headers and 1 udp header per packet. Some limitations are imposed on possible sequences of headers based on our practical experience with network protocols (for example vlan header can follow only a mac header or other vlan header).

Traffic generator suggests first header for a packet template (in port menu). But it is not enforced.

Port Configuration

Sub-menu `/tool packet-generator port`

This menu allows to configure ports that will be associated to specific interface and will be used to receive/send generated packets.

Properties

Property	Description
disabled (*yes* | *no*; Default: **no**)	Whether port is disabled and does not participate in receiving/sending of the packets
name (*string*; Default:)	Descriptive name of the port
interface (*string*; Default:)	Name of the interface associated with the port.

Read-Only Properties

Property	Description
dynamic (*yes* | *no*)	Whether port configuration is generated automatically.
first-header (*ip* | *mac* | *raw* | *udp* | *vlan*)	Shows suggested first header for packets to be sent out of specified interface. This is information can be used when creating packet templates.
inactive (*yes* | *no*)	Whether port is inactive and can't participate in tx/rx of the packets.

Stats

Sub-menu `/tool traffic-generator stats`

If traffic generator is not running in **quick** mode then all statistics about the test is stored in this menu.

Latency Distribution

Sub-menu `/tool traffic-generator stats latency-distribution`

This sub-menu shows how many packets are received in specific latency range. Latency range can be viewed by streams or by sequences (for example, **print stream-num=3**, **print seq=10**)

Here is an example output of the latency graph:

```
[admin@test-host] /tool traffic-generator stats latency-distribution> print
 # LATENCY                  COUNT        SHARE GRAPH
 0 0-15.5us                     0           0%
 1 15.5us-31us                  0           0%
 2 31us-46.5us                  0           0%
 3 46.5us-62.1us                0           0%
 4 62.1us-77.6us                0           0%
 5 77.6us-93.1us                0           0%
 6 93.1us-109us                 0           0%
 7 109us-124us                  0           0%
 8 124us-140us                  0           0%
 9 140us-155us                  0           0%
10 155us-171us                  0           0%
11 171us-186us                  4           0% *
12 186us-202us                 29           0% *
13 202us-217us                 90       0.001% *
14 217us-233us                302       0.004% *
15 233us-248us                630       0.009% *
16 248us-264us                789       0.011% *
17 264us-279us              1 384       0.021% -*
18 279us-295us              1 990        0.03% --*
19 295us-310us              2 966       0.045% ---*
20 310us-326us              4 089       0.062% -----*
21 326us-341us              4 958       0.075% ------*
22 341us-357us              6 059       0.091% -------*
23 357us-372us              6 660       0.101% --------*
24 372us-388us              8 320       0.126% ----------*
25 388us-403us              9 988       0.151% -------------*
26 403us-419us             11 781       0.178% ---------------*
27 419us-434us             12 512       0.189% ----------------*
28 434us-450us             13 836        0.21% -----------------*
29 450us-465us             15 681       0.238% ---------------------*
30 465us-481us             17 740       0.269% ------------------------*
31 481us-496us             19 913       0.302% --------------------------*
32 496us-512us             21 106        0.32% ----------------------------*
33 512us-528us             22 848       0.346% ------------------------------*
34 528us-543us             25 059        0.38% --------------------------------*
35 543us-559us             26 593       0.403% ----------------------------------*
36 559us-574us             27 663       0.419% ------------------------------------*
37 574us-590us             29 351       0.445% --------------------------------------*
38 590us-605us             31 265       0.474% ----------------------------------------*
39 605us-621us             33 224       0.504% ------------------------------------------*
40 621us-636us             34 464       0.523% --------------------------------------------*
41 636us-652us             35 630        0.54% ----------------------------------------------*
42 652us-667us             37 245       0.565% ------------------------------------------------*
43 667us-683us             38 158       0.579% --------------------------------------------------*
44 683us-698us             38 626       0.586% ---------------------------------------------------*
```

```
45 698us-714us            38 985       0.591% ------------------------------------------------*
46 714us-729us            39 061       0.592% ------------------------------------------------*
47 729us-745us            39 750       0.603% -------------------------------------------------*
48 745us-760us            39 145       0.594% ------------------------------------------------*
49 760us-776us            39 162       0.594% ------------------------------------------------*
50 776us-791us            38 197       0.579% -----------------------------------------------*
51 791us-807us            37 811       0.573% -----------------------------------------------*
52 807us-822us            37 364       0.567% ----------------------------------------------*
53 822us-838us            36 770       0.558% ----------------------------------------------*
54 838us-853us            35 831       0.543% ---------------------------------------------*
55 853us-869us            35 380       0.536% ---------------------------------------------*
56 869us-884us            34 472       0.523% --------------------------------------------*
57 884us-900us            33 672       0.511% -------------------------------------------*
58 900us-915us            33 799       0.513% -------------------------------------------*
59 915us-931us            32 754       0.497% ------------------------------------------*
60 931us-946us            32 339        0.49% ------------------------------------------*
61 946us-962us            32 419       0.492% ------------------------------------------*
62 962us-977us            32 107       0.487% -----------------------------------------*
63 977us-993us            31 552       0.478% -----------------------------------------*
64 0-993us             1 221 523       18.54%
```

Properties

Property	Description
count (*integer*)	Number of packets in current latency range
graph (*string*)	graphical representation of share
latency (*string*)	latency range
share (*precent*)	Percentage of packets falling in this latency range.

Stream Stats

Sub-menu `/tool traffic-generator stats stream`

This sub-menu stores statistics sorted by streams. Output is the same as in **quick** mode.

```
[admin@test-host] /tool traffic-generator stats stream> print
 # SEQ    NUM     TX-PACKET   TX-RATE     RX-PACKET   RX-RATE   LOST-PACKET LOST-RATE
 0 1      3          43 064 499.5Mbps        25 180 292.0Mbps        17 884 207.4Mbps
 1 1      4          43 062 499.5Mbps        39 946 463.3Mbps         3 116  36.1Mbps
 2 1      TOT        86 126 999.0Mbps        65 126 755.4Mbps        21 000 243.6Mbps
 3 2      3          43 544 505.1Mbps        30 449 353.2Mbps        13 095 151.9Mbps
 4 2      4          43 543 505.0Mbps        42 982 498.5Mbps           561   6.5Mbps
 5 2      TOT        87 087 1010.2...        73 431 851.7Mbps        13 656 158.4Mbps

...

59 20     TOT        87 277 1012.4...        73 755 855.5Mbps        13 522 156.8Mbps
60 21     3          43 546 505.1Mbps        30 605 355.0Mbps        12 941 150.1Mbps
61 21     4          43 546 505.1Mbps        42 682 495.1Mbps           864  10.0Mbps
```

```
62 21     TOT         87 092 1010.2...         73 287 850.1Mbps         13 805 160.1Mbps
63 TOT    3          913 942 504.8Mbps        629 210 347.5Mbps        284 732 157.2Mbps
64 TOT    4          913 939 504.8Mbps        898 374 496.2Mbps         15 565   8.5Mbps
65 TOT    TOT      1 827 881 1009.6...      1 527 584 843.8Mbps        300 297 165.8Mbps

[admin@test-host] /tool traffic-generator stats stre
```

Port Stats

Sub-menu `/tool traffic-generator stats port`

This sub-menu stores statistics sorted by rx/tx ports.

```
[admin@test-host] /tool traffic-generator stats port> print
 # SEQ    PORT        RX-UNK-PACKET    RX-UNK-BYTE RX-UNK...     TX-PACKET   TX-RATE     RX-PACKET
 0 1      port0:et...             0              0      0bps        43 064 499.5Mbps        39 946
 1 1      port1:et...             0              0      0bps        43 062 499.5Mbps        25 180
 2 1      TOT                     0              0      0bps        86 126 999.0Mbps        65 126
 3 2      port0:et...             0              0      0bps        43 544 505.1Mbps        42 982
 4 2      port1:et...             0              0      0bps        43 543 505.0Mbps        30 449
 5 2      TOT                     0              0      0bps        87 087 1010.2...        73 431
 6 3      port0:et...             0              0      0bps        43 540 505.0Mbps        42 615
 7 3      port1:et...             0              0      0bps        43 540 505.0Mbps        30 191
 8 3      TOT                     0              0      0bps        87 080 1010.1...        72 806
```

Raw Stats

Sub-menu `/tool traffic-generator stats raw`

This sub-menu stores raw statistics data.

```
[admin@test-host] /tool traffic-generator stats raw> print
 # SEQ    PORT       NUM     TX-PACKET   TX-RATE     RX-PACKET   RX-RATE   LOST-PACKET LOST-RATE
 0 1      port0:e... 3          43 064 499.5Mbps             0      0bps        43 064 499.5Mbps
 1 1      port1:e... 3               0      0bps        25 180 292.0Mbps       -25 180 292.0Mbps
 2 1      TOT        3          43 064 499.5Mbps        25 180 292.0Mbps        17 884 207.4Mbps
 3 1      port0:e... 4               0      0bps        39 946 463.3Mbps       -39 946 463.3Mbps
 4 1      port1:e... 4          43 062 499.5Mbps             0      0bps        43 062 499.5Mbps
 5 1      TOT        4          43 062 499.5Mbps        39 946 463.3Mbps         3 116  36.1Mbps
 6 1      port0:e... TOT        43 064 499.5Mbps        39 946 463.3Mbps         3 118  36.1Mbps
 7 1      port1:e... TOT        43 062 499.5Mbps        25 180 292.0Mbps        17 882 207.4Mbps
 8 2      port0:e... 3          43 544 505.1Mbps             0      0bps        43 544 505.1Mbps
 9 2      port1:e... 3               0      0bps        30 449 353.2Mbps       -30 449 353.2Mbps
10 2      TOT        3          43 544 505.1Mbps        30 449 353.2Mbps        13 095 151.9Mbps
```

Streams

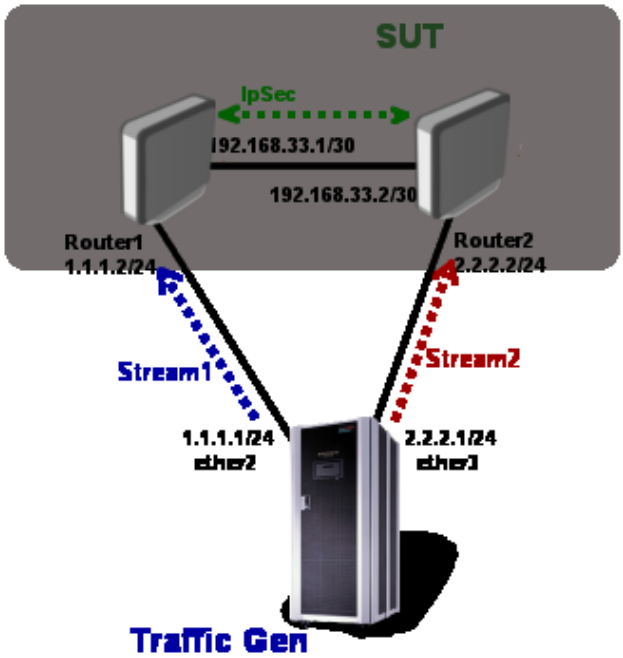
Properties

Property	Description
disabled (*yes* | *no*; Default: **no**)	Whether stream is disabled
mbps (*integer [0..4294967295]*; Default: **0**)	Value in Mega bits per second that stream will try to generate.
name (*string*; Default:)	Descriptive name of the stream.
num (*integer [0..15]*; Default: **0**)	
packet-size (*integer[1..65535] [-integer[1..65535]]*; Default: **0**)	Generated size of the packets in bytes. Can be set as the range for random packet size generation.
port (*string*; Default:)	Name of the port from Port menu that will be used to transmit packets.
pps (*integer [0..4294967295]*; Default: **0**)	Packets per second that stream will try to generate.
tx-template (*string*; Default:)	Name of the packet template from packet-template or raw-packet-template menus used as the packet content source.

Configuration Examples

IpSec tunnel performance test

Consider following test setup



System Under Test (SUT) consists of two routers connected to traffic generator server. Connection between both SUT routers are IPSec encrypted.

Traffic generator will run two streams:

- in direction from 1.1.1.0/24 network to 2.2.2.0/24 network
- in direction from 2.2.2.0/24 network to 1.1.1.0/24 network.

R1 routing and ipsec setup

```
/ip address
add address=192.168.33.1/30 interface=ether1
add address=1.1.1.2/24 interface=ether2

/ip route
add dst-address=2.2.2.0/24 gateway=192.168.33.2

/ip ipsec proposal
set default enc-algorithms=aes-128

/ip ipsec peer
add address=192.168.33.2 secret=123

/ip ipsec policy
add sa-src-address=192.168.33.1 sa-dst-address=192.168.33.2 \
    src-address=1.1.1.0/24 dst-address=2.2.2.0/24 tunnel=yes
```

R2 routing and ipsec setup

```
/ip address
add address=192.168.33.2/30 interface=ether1
add address=2.2.2.2/24 interface=ether2

/ip route
add dst-address=1.1.1.0/24 gateway=192.168.33.1

/ip ipsec proposal
set default enc-algorithms=aes-128

/ip ipsec peer
add address=192.168.33.1 secret=123

/ip ipsec policy
add sa-src-address=192.168.33.2 sa-dst-address=192.168.33.1 \
    src-address=2.2.2.0/24 dst-address=1.1.1.0/24 tunnel=yes
```

Traffig generator server setup

```
/ip address
add address=1.1.1.1/24 interface=ether2
add address=2.2.2.1/24 interface=ether3
```

First we will define which ports will be used as traffic generator tx/rx ports

```
/tool traffic-generator port
add disabled=no interface=ether2 name=port0
add disabled=no interface=ether3 name=port1
```

To construct actual packet that will be generated, packet-template is used.

```
/tool traffic-generator packet-template
add header-stack=mac,ip,udp ip-dst=2.2.2.1/32 ip-gateway=1.1.1.2 ip-src=1.1.1.1/32 \
    name=routing-1 port=port0
add header-stack=mac,ip,udp ip-dst=1.1.1.1/25 ip-gateway=2.2.2.2 ip-src=2.2.2.1/32 \
    name=routing-2 port=port1
```

Notice that mac addresses was not specified since template generator can assume next-hop mac address automatically by sending ARP messages. Since we are doing routing and destination IP is not directly reachable, we have set **ip-gateway** parameter to determine next-hop mac-address.

When running **print** you can see all assumed (detected) values including mac addresses.

```
[admin@test-host] /tool traffic-generator packet-template> print
 0 name="routing-1" header-stack=mac,ip,udp port=port0
   assumed-mac-dst=00:0C:42:00:38:9D assumed-mac-src=00:0C:42:40:94:25
   assumed-mac-protocol=ip assumed-ip-dscp=0 assumed-ip-id=0
   assumed-ip-frag-off=0 assumed-ip-ttl=64 assumed-ip-protocol=udp
   ip-src=1.1.1.1/32 ip-dst=2.2.2.1/32 assumed-udp-src-port=100
   assumed-udp-dst-port=200 ip-gateway=1.1.1.2 data=uninitialized data-byte=0

 1 name="routing-2" header-stack=mac,ip,udp port=port1
   assumed-mac-dst=00:0C:42:00:38:D1 assumed-mac-src=00:0C:42:40:94:26
   assumed-mac-protocol=ip assumed-ip-dscp=0 assumed-ip-id=0
   assumed-ip-frag-off=0 assumed-ip-ttl=64 assumed-ip-protocol=udp
   ip-src=2.2.2.1/32 ip-dst=1.1.1.1/32 assumed-udp-src-port=100
   assumed-udp-dst-port=200 ip-gateway=2.2.2.2 data=uninitialized data-byte=0
```

For example if any router in SUT change, assumed mac-addresses will not be updated automatically. To update packet templates simply issue command :

```
/tool traffic-generator packet-template set [find]
```

Last part is to configure streams

```
/tool traffic-generator stream
add disabled=no mbps=500 name=str1 num=3 packet-size=1450 port=port0 pps=0 \
    tx-template=routing-1
add disabled=no mbps=500 name=str3 num=4 packet-size=1450 port=port1 pps=0 \
    tx-template=routing-2
```

Notice that each stream has unique **num** value. This value identifies stream packets, otherwise traffic generator will not work.

Now we are ready to run the test. In this case **quick** mode will be used:

```
[admin@test-host] /tool traffic-generator> quick mbps=450
SEQ    NUM     TX-PACKET   TX-RATE     RX-PACKET   RX-RATE        RX-OOO   LOST-PACKET LOST-RATE
37     4          39 488 458.0Mbps        39 270 455.5Mbps        15 509           218   2.5Mbps
37     TOT        78 976 916.1Mbps        76 485 887.2Mbps        22 529         2 491  28.8Mbps
38     3          38 957 451.9Mbps        37 657 436.8Mbps         7 078         1 300  15.0Mbps
38     4          38 958 451.9Mbps        38 402 445.4Mbps        14 763           556   6.4Mbps
38     TOT        77 915 903.8Mbps        76 059 882.2Mbps        21 841         1 856  21.5Mbps
39     3          38 816 450.2Mbps        37 893 439.5Mbps         7 307           923  10.7Mbps
```

```
39     4          38 815 450.2Mbps        38 642 448.2Mbps        15 110          173   2.0Mbps
39     TOT        77 631 900.5Mbps        76 535 887.8Mbps        22 417        1 096  12.7Mbps
40     3          39 779 461.4Mbps        37 415 434.0Mbps         7 136        2 364  27.4Mbps
40     4          39 780 461.4Mbps        39 567 458.9Mbps        15 908          213   2.4Mbps
40     TOT        79 559 922.8Mbps        76 982 892.9Mbps        23 044        2 577  29.8Mbps
41     3          39 218 454.9Mbps        37 089 430.2Mbps         7 075        2 129  24.6Mbps
41     4          39 218 454.9Mbps        38 663 448.4Mbps        15 752          555   6.4Mbps
41     TOT        78 436 909.8Mbps        75 752 878.7Mbps        22 827        2 684  31.1Mbps
42     3          39 188 454.5Mbps        37 906 439.7Mbps         6 729        1 282  14.8Mbps
42     4          39 187 454.5Mbps        38 954 451.8Mbps        15 565          233   2.7Mbps
42     TOT        78 375 909.1Mbps        76 860 891.5Mbps        22 294        1 515  17.5Mbps
TOT    3       1 645 468 454.4Mbps     1 568 201 433.1Mbps       280 174       77 267  21.3Mbps
TOT    4       1 645 464 454.4Mbps     1 626 896 449.3Mbps       627 480       18 568   5.1Mbps
TOT    TOT     3 290 932 908.9Mbps     3 195 097 882.4Mbps       907 654       95 835  26.4Mbps
```

Stats shows throughput of each stream and total throughput of both streams, Out-of-order packet count, Lost rate, latency and jitter.

[Top | Back to Content]

Manual:Connection oriented communication (TCP/IP)

Connection oriented communication (TCP/IP)

The connection-oriented communication is a data communication mode in which you must first establish a connection with remote host or server before any data can be sent. It is similar with analog telephone network where you had to establish connection before you are able to communicate with a recipient. Connection establishment included operations such as dial number, receive dial tone, wait for calling signal etc.

TCP session establishment and termination

Process when transmitting device establishes a connection-oriented session with remote peer is called a **three-way handshake**. As the result end-to-end virtual (logical) circuit is created where flow control and acknowledgment for reliable delivery is used. TCP has several message types used in connection establishment and termination process (see Figure 2.1.).

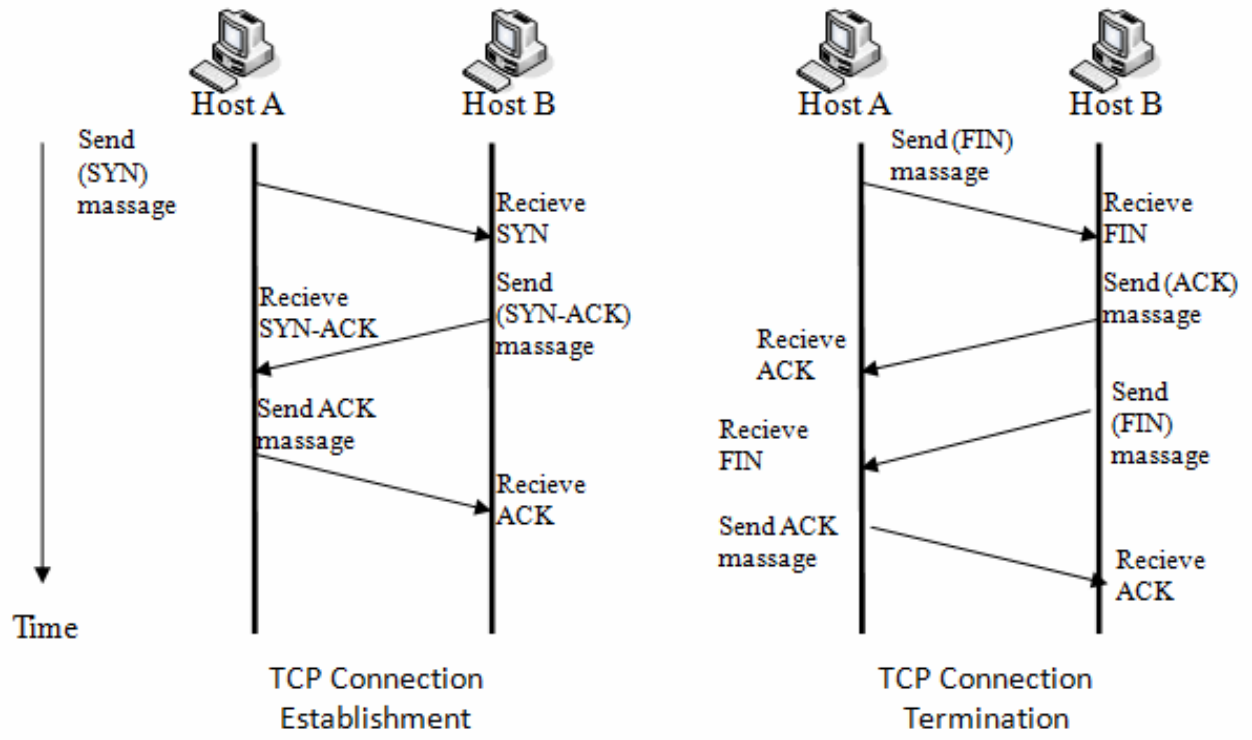


Figure 2.1. TCP session establishment and termination

Connection establishment process

1. The host A who needs to initialize a connection sends out a SYN (Synchronize) packet with proposed initial sequence number to the destination host B.
2. When the host B receives SYN message, it returns a packet with both SYN and ACK fags set in the TCP header (SYN-ACK).
3. When the host A receives the SYN-ACK, it sends back ACK (Acknowledgment) macket.
4. Host B receives ACK and at this stage the connection is ESTABLISHED.

Connection-oriented protocol services are often sending acknowledgments (ACKs) after successful delivery. After packet with data is transmitted, sender waits acknowledgement from receiver. If time expires and sender did not receive ACK, packet is retransmitted.

Connection termination

When the data transmission is complete and the host wants to terminate the connection, termination process is initiated. Unlike TCP Connection establishment, which uses three-way handshake, connection termination uses four-way massages. Connection is terminated when both sides have finished the shut down procedure by sending a FIN and receiving an ACK.

1. The host A, who needs to terminate the connection, sends a special message with the FIN (finish) flag, indicating that it has finished sending the data.
2. The host B, who receives the FIN segment, does not terminate the connection but enters into a "passive close" (CLOSE_WAIT) state and sends the ACK for the FIN back to the host A. Now the host B enters into LAST_ACK state. At this point host B will no longer accept data from host A, but can continue transmit data to host A. If host B does not have any data to transmit to the host A it will also terminate the connection by sending FIN segment.
3. When the host A receives the last ACK from the host B, it enters into a (TIME_WAIT) state, and sends an ACK back to the host B.

4. Host B gets the ACK from the host A and closes the connection.

Segments transmission (windowing)

Now that we know how the TCP connection is established we need to understand how data transmission is managed and maintained. In TCP/IP networks transmission between hosts is handled by TCP protocol.

Let's think about what happens when datagrams are sent out faster than receiving device can process. Receiver stores them in memory called a buffer. But since buffer space are not unlimited, when its capacity is exceeded receiver starts to drop the frames. All dropped frames must be retransmitted again which is the reason for low transmission performance.

To address this problem, TCP uses flow control protocol. *window* mechanism is used to control the flow of the data. When connection is established, receiver specifies window field (see, TCP header format, Figure 1.6.) in each TCP frame. Window size represents the amount of received data that receiver is willing to store in the buffer. *window size (in bytes)* is send together with acknowledgements to the sender. So the size of window controls how much information can be transmitted from one host to another without receiving an acknowledgment. Sender will send only amount of bites specified in window size and then will wait for acknowledgments with updated window size.

If the receiving application can process data as quickly as it arrives from the sender, then the receiver will send a positive window advertisement (increase the windows size) with each acknowledgement. It works until sender becomes faster than receiver and incoming data will eventually fill the receiver's buffer, causing the receiver to advertise acknowledgment with a zero window. A sender that receives a zero window advertisement must stop transmit until it receives a positive window. Windowing process is illustrated in Figure 2.2.

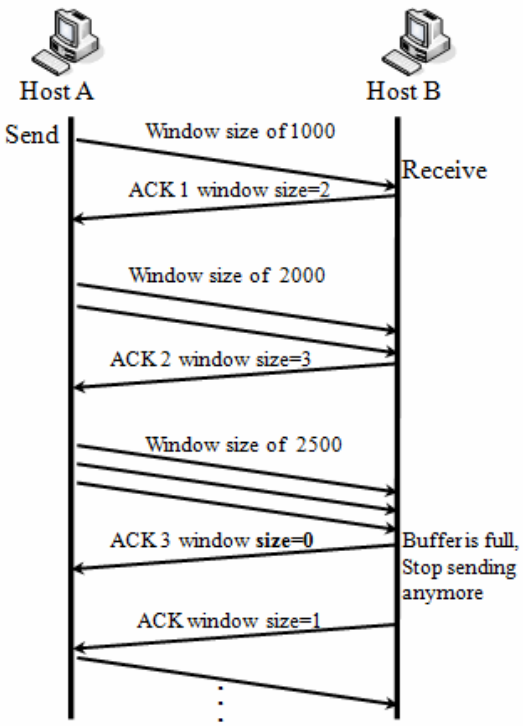


Figure 2.2. TCP flow control using *windowing*

The host A starts transmit with window size of 1000, one 1000byte frame is transmitted. Receiver (host B) returns ACK with window size to increase to 2000. The host A receives ACK and transmits two frames (1000 bytes each). After that receiver advertises an initial window size to 2500. Now sender transmits three frames (two containing 1,000 bytes and one containing 500 bytes) and waits for an acknowledgement. The first three segments fill the receiver's buffer faster than the receiving application can process the data, so the advertised window size reaches

zero indicating that it is necessary to wait before further transmission is possible.

The size of the window and how fast to increase or decrease the window size is available in various TCP congestion avoidance algorithms such as Reno, Vegas, Tahoe etc.

Ethernet networking

CSMA/CD

The Ethernet system consists of three basic elements:

- the physical medium used to carry Ethernet signals between network devices,
- medium access control system embedded in each Ethernet interface that allow multiple computers to fairly control access to the shared Ethernet channel,
- Ethernet frame that consists of a standardized set of bits used to carry data over the system.

Ethernet network uses Carrier Sense Multiple Access with Collision detection (CSMA/CD) protocol for data transmission. That helps to control and manage access to shared bandwidth when two or more devices want to transmit data at the same time. CSMA/CD is a modification of Carrier Sense Multiple Access. Carrier Sense Multiple Access with Collision Detection is used to improve CSMA performance by terminating transmission as soon as collision is detected, reducing the probability of a second collision on retry.

Before we discuss a little more about CSMA/CD we need to understand what is collision, collision domain and network segment. A collision is the result of two devices on the same Ethernet network attempting to transmit data at the same time. The network detects the "collision" of the two transmitted packets and discards both of them.

If we have one large network solution is to break it up into smaller networks – often called *network segmentation*. It is done by using devices like routers and switches - each of switch ports create separate network segment which result in separate collision domain. A collision domain is a physical network segment where data packets can *"collide"* with each other when being sent on a shared medium. Therefore on a hub, only one computer can receive data simultaneously otherwise collision can occur and data will be lost.

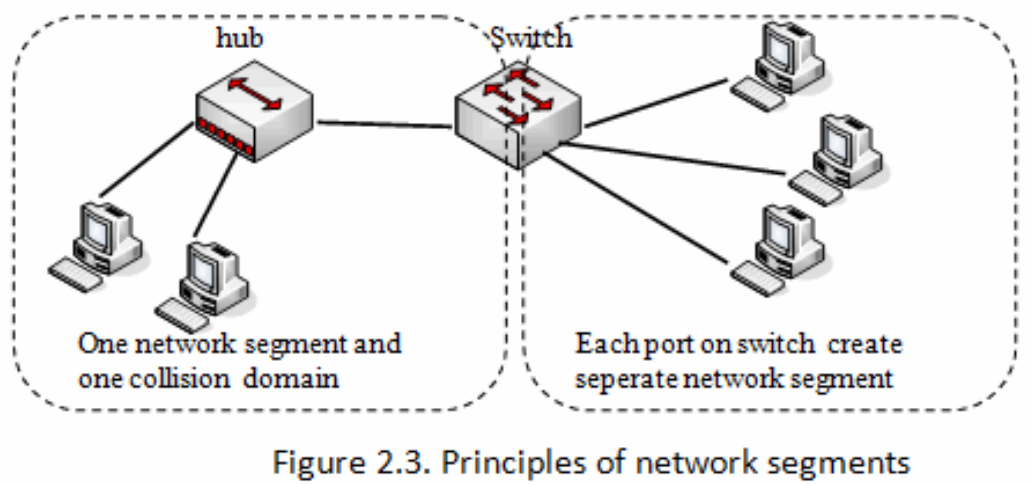


Figure 2.3. Principles of network segments

Hub (called also *repeater*) is specified in Physical layer of OSI model because it regenerates only electrical signal and sends out input signal to each of ports. Today hubs do not dominate on the LAN networks and are replaced with switches.

Carrier Sense – means that a transmitter listens for a carrier (encoded information signal) from another station before attempting to transmit.

Multiple Access – means that multiple stations send and receive on the one medium.

Collision Detection - involves algorithms for checking for collision and advertises about collision with collision response – *"Jam signal"*.

When the sender is ready to send data, it checks continuously if the medium is busy. If the medium becomes idle the sender transmits a frame.

Look at the Figure 2.4 bellow where simple example of CSMA/CD is explained.

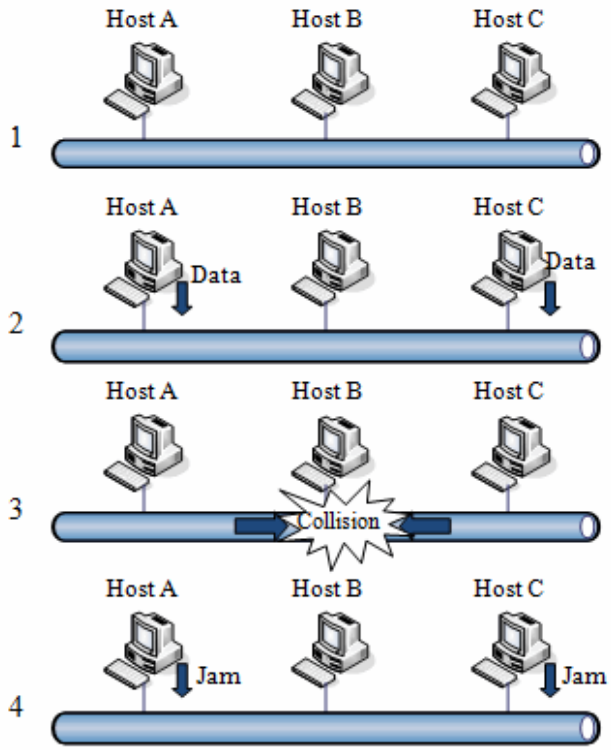


Figure 2.4. Principles of CSMA/CD

1. Any host on the segment that wants to send data “listens” what is happening on the physical medium(wire) an is checking whether someone else is not sending data already.
2. Host A and host C on shared network segment sees that nobody else is sending and tries to send frames.
3. Host A and Host C are listening at the same time so both of them will transmit at the same time and collision will occur. Collision results in what we refer to as "noise" - a change in the voltage of the signals in the line (wire).
4. Host A and Host B detect this collision and send out “jam” signal to tell other hosts not to send data at this time. Both Host A and Host C need to retransmit this data, but we don't want them to send frames simultaneously once again. To avoid this, host A and host B will start a *random timer (ms)* before attempting to start CSMA/CD process again by listening to the wire.

Each computer on Ethernet network operates independently of all other stations on the network.

Half and Full duplex Ethernet

Ethernet standards such as Ethernet II and Ethernet 802.3 are passed through formal IEEE (Institute of Electrical and Electronics Engineers) standardization process. The difference is that Ethernet II header includes Protocol type field whereas in Ethernet 802.3 this field was changed to length field. Ethernet is the standard CSMA/CD access method. Ethernet supports different data transfer rates Ethernet (10BaseT) – 10 Mbps, Fast Ethernet (100Base-TX) – 100 Mbps Gigabit Ethernet (1000Base-T) – 1000 Mbps through different types of physical mediums (twisted pairs (Copper), coaxial cable, optical fiber). Today Ethernet cables consist of four twisted pairs (8 wires). For example, 10Base-T uses only one of these wire pairs for running in both directions using half-duplex mode.

Half-duplex data transmission means that data can be transmitted in both directions between two nodes, but only one direction at the same time. Also in the Gigabit Ethernet is defined (Half-duplex) specifications, but it isn’t used in practice.

Full-duplex data transmission means that data can be transmitted in both directions using different twisted pairs for each of direction at the same time. Full Duplex Ethernet, collisions are not possible since data is transmitted and received on different wires, and each segment is connected directly to a switch. Full-duplex Ethernet offers performance in both directions for example, if your computer supports Gigabit Ethernet (full duplex mode) and your gateway (router) also support it then between your computer and gateway 2Gbps aggregated bandwidth is available.

Simple network communication example

ARP protocol operation

Address Resolution Protocol (ARP) is a protocol for mapping an Internet Protocol (IP) address of host in the local network to the hardware address (MAC address). The physical/hardware address is also known as a Media Access Control or MAC address. Each network device maintains ARP tables (cache) that contain list of MAC address and its corresponding IP address. MAC addresses uniquely identify every network interface in the network. IP addresses are used for path selection to destination (in the routing process), but frame forwarding process from one interface to another occur using MAC addresses.

When host on local area network wants to send IP packet to another host in this network, it must looks for Ethernet MAC address of destination host in its ARP cache. If the destination host's MAC address is not in ARP table, then ARP request is sent to find device with corresponding IP address. ARP sends broadcast request message to all devices on the LAN by asking the devices with the specified IP address to reply with its MAC address. A device that recognizes the IP address as its own returns ARP response with its own MAC address. Figure 2.5 shows how an ARP looks for MAC address on the local network.

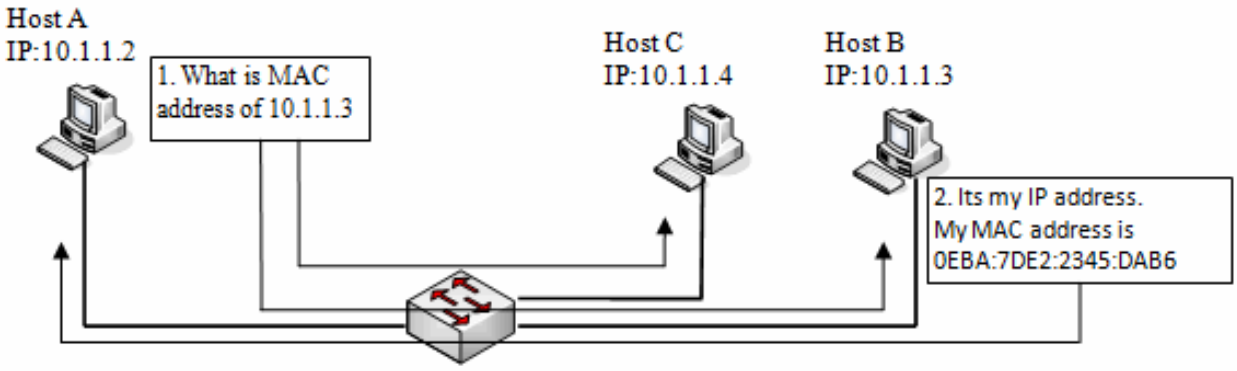


Figure 2.5. ARP operation on the LAN

Commands that displays current ARP entries on a PC (linux, DOS) and a MikroTik router (commands might do the same thing, but they syntax may be different):

For windows and Unix like machines: ***arp − a*** displays the list of IP addresses with its corresponding MAC addresses

ip arp print − same command as *arp − a* but display the ARP table on a MikroTik Router.

[Top | Back to Content]

Manual:TE tunnel auto bandwidth

Overview

By default MPLS TE tunnels do not apply any rate limitation on traffic that gets sent over tunnel. That way "bandwidth" settings for MPLS TE enabled interfaces and TE tunnels are only used for reservation accounting. There are also no means to adjust bandwidth that gets reserved for tunnel other than changing tunnel configuration no matter what is actual amount of traffic sent over tunnel. To make TE tunnels more flexible and easy to use, the following features have been introduced:

- Bandwidth limitation
- Automatic bandwidth adjustment

These features operate on tunnel head end (ingress) router. These features can either be used alone or in combination.

Bandwidth limitation

TE tunnel can be configured to limit the rate at which traffic is allowed to enter the tunnel. Limit is specified on ingress router in percent of tunnel bandwidth. E.g. creating the following tunnel:

```
[admin@R1] /interface traffic-eng> add name=te1 from-address=9.9.9.1 to-address=9.9.9.5 \
   bandwidth=100000 bandwidth-limit=120 primary-path=stat
```

means that tunnel will reserve bandwidth of 100 kilobits per second across MPLS backbone from 9.9.9.1 to 9.9.9.5 and that ingress router will limit the rate of traffic entering the tunnel to 120 kilobits per second (120% of 100 kilobits per second bandwidth). This can be confirmed by monitoring tunnel interface:

```
[admin@R1] /interface traffic-eng> monitor te1
              tunnel-id: 3
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 1
           active-label: 20
     reserved-bandwidth: 100.0kbps
             rate-limit: 120.0kbps
     rate-measured-last: 0bps
  rate-measured-highest: 0bps
```

Note that by default any limiting is disabled. By specifying limit as percentage of tunnel bandwidth, TE tunnel bandwith limits can be configured in rather flexible ways - some tunnels can be configured to hard limit while others can be configured with reasonable reserve, achieving different classes of service.

Automatic bandwidth adjustment

Auto bandwidth adjustment feature enables MPLS TE network to follow the changes of amount of data transmitted over tunnel. Bandwidth adjustment feature works as follows:

- Actual amount of data entering tunnel during averaging interval (**auto-bandwidth-avg-interval**) is measured, producing average rate.
- Tunnel keeps track of highest average rate seen during update interval (**auto-bandwidth-update-interval**)
- When update interval expires, TE tunnel bandwidth is updated to highest observed average rate, taking into account specified range over which bandwidth is allowed to change (**auto-bandwidth-range**)

Auto bandwidth adjustment feature gets enabled by specifying **auto-bandwidth-range**. For example, adding the following tunnel:

```
[admin@R1] /interface traffic-eng> add name=te1 from-address=9.9.9.1 to-address=9.9.9.5 \
  bandwidth=100000 primary-path=stat auto-bandwidth-range=10000-500000 \
  auto-bandwidth-avg-interval=10s auto-bandwidth-update-interval=1m
```

means that tunnel will measure average rate over 10 second periods and once per minute will update bandwidth in range from 10 to 500 kilobits per second. Tunnel **bandwidth** setting specifies the initial bandwidth of tunnel. The above tunnel in complete absence of data over it after 1 minute will change its bandwidth to specified minimum 10 kbps:

```
[admin@R1] /interface traffic-eng> monitor te1
               tunnel-id: 3
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 2
           active-label: 21
     reserved-bandwidth: 10.0kbps
             rate-limit: 12.0kbps
     rate-measured-last: 0bps
  rate-measured-highest: 0bps
```

Additionally, tunnel can be configured to reserve more bandwidth than measured. This can be achieved with **auto-bandwidth-reserve** setting which specifies percentage of additional bandwidth to reserve - so setting **auto-bandwith-reserve** to 10 means that tunnel will reserve 10% more bandwidth than measured (but will still obey the **auto-bandwidth-range**). For example changing above tunnel and running constant stream of 50kbps through it will yield the following results:

```
[admin@R1] /interface traffic-eng> set te1 auto-bandwidth-reserve=30
```

In the beginning tunnel reserves its initially specified bandwidth:

```
[admin@R1] /interface traffic-eng> monitor te1
               tunnel-id: 6
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 1
```

```
           active-label: 27
     reserved-bandwidth: 100.0kbps
              rate-limit: 120.0kbps
     rate-measured-last: 48.8kbps
  rate-measured-highest: 48.8kbps
```

After update period and after previous reservations are torn down notice how reserved bandwidth exceeds average rate by 30%. Also notice that **rate-limit** correctly changes to 120% of **reserved-bandwidth**:

```
[admin@R1] /interface traffic-eng> monitor te1
               tunnel-id: 6
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 2
           active-label: 28
     reserved-bandwidth: 64.4kbps
              rate-limit: 77.3kbps
     rate-measured-last: 48.8kbps
  rate-measured-highest: 48.8kbps
```

Note that in case reservation must be updated to lower value, brief period after update period **reserved-bandwidth** will still display previous reservation value. The reason for this is that new reservation is made without disrupting the previous tunnel and therefore shares its reservation until old reservation is torn down. **rate-limit** on turn is correctly updated to intended value. In the above example, after stopping the 50kbps stream and after update period will pass with tunnel being idle, for a brief period after update tunnel info can be:

```
[admin@R1] /interface traffic-eng> monitor te1
               tunnel-id: 6
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 2
           active-label: 34
     reserved-bandwidth: 63.4kbps
              rate-limit: 12.0kbps
     rate-measured-last: 0bps
  rate-measured-highest: 0bps
```

After previous reservation (63.4kbps) is torn down, **reserved-bandwidth** correctly changes to 10kbps:

```
[admin@R1] /interface traffic-eng> monitor 1
               tunnel-id: 6
     primary-path-state: established
           primary-path: stat
   secondary-path-state: not-necessary
            active-path: stat
           active-lspid: 2
```

```
           active-label: 34
     reserved-bandwidth: 10.0kbps
             rate-limit: 12.0kbps
     rate-measured-last: 0bps
 rate-measured-highest: 0bps
```

Note that **auto-bandwidth-reserve** is applied to actual measured bandwidth, before range checking according to **auto-bandwidth-range** - therefore 10kbps gets reserved, instead of 13kbps.

Combining bandwidth limitation with automatic bandwidth adjustment

Auto bandwidth adjustment can be used in combination with bandwidth limit feature - **bandwidth-limit** setting will apply to bandwidth actually reserved for tunnel. In order to successfully cobine both features, actual bandwidth must be allowed to fluctuate to some extent - e.g. if **bandwidth-limit** will be configured to 100% (this effectively means that rate will be limited to the bandwidth reserved for tunnel), tunnel will not have any chance to increase its reservation. Therefore either **bandwidth-limit** should be configured to more than 100%, or **auto-bandwidth-reserve** should be configured to more than 0%.

Manual:TE Tunnels

Overview

For MPLS overview and RouterOS supported MPLS features see MPLS Overview.

MPLS RSVP TE tunnels are a way to establish unidirectional label switching paths. In general RSVP TE serves similar purpose as label distribution using LDP protocol - establishing label switched path that ensures frame delivery from ingress to egress router, but with additional features:

- possibility to establish label switching path using either full or partial explicit route;
- constraint based LSP establishment - label switching path is established over links that fulfill requirements, such as bandwidth and link properties.

MPLS RSVP TE is based on RSVP protocol with extensions introduced by RFC 3209 that adds support for explicit route and label exchange.

Note that constraints for path establishment are purely controlled by administrator - for example, bandwidth of link participating in RSVP TE network is set by administrator and does not necessarily reflect real bandwidth of the link. The same way bandwidth reserved for tunnel is set by administrator and does not automatically imply any limits on traffic sent over tunnel. Therefore at any moment in time, bandwidth available on TE link is bandwidth configured for link minus sum of all reservations made on link, not physically available bandwidth which can be either less (in case data is forwarded over tunnels with rate that exceeds bandwidth reserved for tunnel or if non-RSVP tunnel data is forwarded over link as well) or more (in case data is forwarded over tunnels with rate smaller than allocated for tunnel) than bandwidth available for reservations.

RSVP TE tunnels are initiated by head-end (ingress router) of tunnel. Head-end router sends RSVP Path message containing necessary parameters towards tail-end of the tunnel. Routers along the path ensure that they can forward Path message towards next hop, taking into acount path constraints. Once Path message reaches tail-end of the tunnel, tail-end router sends RSVP Resv message in the opposite direction. Resv message hop by hop traverses exactly the same path that Path message, only in the opposite direction. Each router forwarding Resv message allocates necessary bandwith on appropriate downstream link if possible. Once head-end router succesfully receives Resv message that matches sent Path message, tunnel can be considered established. Tunnel is maintained by

periodically refreshing its state using Path and Resv messages.

RSVP TE tunnels can be established with number of path options:

- along path that data from head-end of tunnel is routed to tail-end - in this case each router along tunnel path figures out next hop of tunnel based on routing table. If at some point usable route is not found or downstream interface does not meet constraints (for example if requested bandwidth exceeds available bandwidth), tunnel can not be established.
- along statically configured explicit path - in this case each router along tunnel path figures out next hop of tunnel based on explicit route specified in Path message. This explicit route can be either complete (specifies all routers along the path in the order they must be traversed) or partial (specifies only some routers that must be traversed). To decide next hop router, each router along the path look up route to next router specified in explicit route. If no usable route is found or downstream interface does not meet constraints, tunnel can not be established
- Constrained Shortest Path First - in this case head-end router calculates path to tail-end using its knowledge of network state - properties of links and available bandwidth. This option needs assistance from IGP routing protocol (such as OSPF) to distribute bandwidth information throughout the network. This is implemented in OSPF by means of opaque LSAs. When using CSPF, head-end router calculates path that satisfies the requirements and produces explicit path for Path message. If path that matches constraints can not be calculated, tunnel can not be established. Dynamically calculated path can also be partially explicit - in this case CSPF seeks for shortest path matching constraints between every two explicit hops. If explicit path is specified completely and CSPF is used, CSPF just checks if this path meets the constraints taking into account knowledge about link states in network - so instead of failure to establish tunnel while forwarding Path message in network, Path message is not even sent as it is clear that establishing tunnel will fail.

Forwarding traffic onto TE tunnels

RSVP TE tunnel head-end appears as interface in RouterOS. Note that RSVP TE tunnels are unidirectional - it is not necessary to have matching tunnel for reverse direction on tail-end router. When tail-end router receives data sent over tunnel, it either receives it with TE tunnel label stripped off by penultimate hop (non-default behaviour) or with explicit-null label, which gets stripped and packet is further inspected (if tunnel label is last label in stack, packet gets routed, otherwise it is processed based on next label in stack, for example, as VPLS packet). Bidirectional tunnel can be simulated by creating one tunnel in one direction and other in other direction between the same endpoints. Still no data will be accounted as received over TE tunnel, as in reality both tunnels are unrelated.

One way to forward traffic onto tunnel is to use routing, but this limits TE tunnel to be used only for routing IP packets.

Additionally, several types of traffic can be forwarded onto TE tunnel automatically, if it is known to be destined to the endpoint of tunnel and if tunnel is active:

- traffic that is routed using route route learned from BGP, if BGP NextHop is tunnel endpoint (this default behaviour can be changed by setting route porperty "use-te-nexthop" to "no"), both - regular IP and VPNv4 (MP-BGP IP VPN) routes fit in this category;
- traffic for VPLS interfaces, if remote endpoint of VPLS pseudowire is the same as TE tunnel endpoint.

For example, for IP BGP route having BGP NextHop x.x.x.x, forwarding method will be chosen according to the following rules:

- if TE tunnel with endpoint x.x.x.x is active, use it;
- otherwise if LDP label mapping from next hop towards x.x.x.x is received, use it;
- otherwise use regular routing (no MPLS encapsulation).

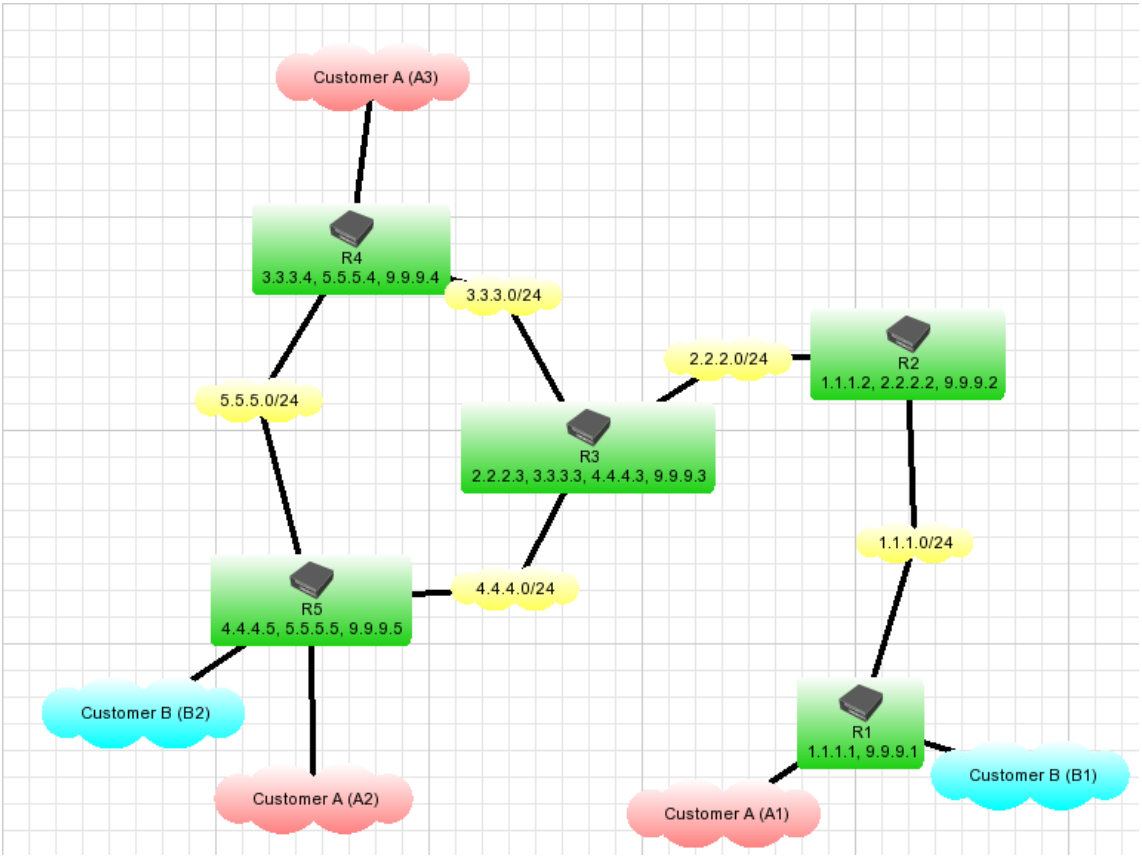
In similar way, if remote address of VPLS pseudowire is x.x.x.x, forwarding method will be chosen in the following order:

- if TE tunnel with endpoint x.x.x.x is active, use it;
- otherwise if LDP label mapping from next hop towards x.x.x.x is received, use it;
- otherwise VPLS tunnel can not be active.

Note that RSVP TE tunnels as a way to establish LSPs can be used together with LDP. Using RSVP TE does not replace or disable LDP, but LSP established by TE is usually preferred over one established using LDP.

Example network

Consider the same network as used for LDP signaled VPLS example in MPLSVPLS:



Customer A wants to establish IP VPN between his 3 sites and Customer B wants to transparent connection for ethernet segments at his sites.

Prerequisites for MPLS TE

In general, prerequisites for using MPLS TE are the same as mentioned in MPLSVPLS, but there are a few details:

- by default TE tunnel tail-end router advertises explicit null label, therefore penultimate hop popping does not happen (the purpose of using explicit null label is to communicate QoS information in MPLS label Exp field), so main purpose of having "loopback" IP address for every router is to have tunnel endpoints unaffected by link state changes;
- in order to use CSPF path selection for tunnels, OSPF must be configured and running in network.

Enabling TE support

In order for OSPF to distribute TE information, TE related OSPF parameters must be set:

```
[admin@R1] > /routing ospf set mpls-te-area=backbone mpls-te-router-id=lobridge
```

This instructs OSPF to distribute TE information in "backbone" area using IP address of "lobridge" as router ID.

In order for router to be able to participate in TE tunnel (either as head-end, tail-end or forwarding router), TE support must be enabled. TE support must be enabled on all interfaces that will receive and send RSVP TE protocol packets. On R1 it is done by commands (interface ether3 is facing network 1.1.1.0/24):

```
[admin@R1] > /mpls traffic-eng interface add interface=ether3 bandwidth=100000
```

This configures ether3 interface with TE support, having bandwidth 100000 Bps. Other routers are configured in similar way.

As soon as TE support is enabled on interface, appropriate opaque LSAs are distributed into OSPF area. For example, on R1 it can be seen, that there is total 15 opaque LSAs in LSA database:

```
[admin@R1] > /routing ospf lsa print
  ...
backbone              opaque-area  1.0.0.0        1.1.1.2        0x80000004      1038
backbone              opaque-area  1.0.0.0        2.2.2.3        0x80000004      1039
backbone              opaque-area  1.0.0.0        3.3.3.4        0x80000004      1038
backbone              opaque-area  1.0.0.0        4.4.4.5        0x80000004      1038
backbone              opaque-area  1.0.0.0        11.11.11.1     0x80000004      1037
backbone              opaque-area  1.0.0.1        1.1.1.2        0x80000004      1038
backbone              opaque-area  1.0.0.1        2.2.2.3        0x80000004      1039
backbone              opaque-area  1.0.0.1        3.3.3.4        0x80000004      1037
backbone              opaque-area  1.0.0.1        4.4.4.5        0x80000004      1038
backbone              opaque-area  1.0.0.2        1.1.1.2        0x80000004      1038
backbone              opaque-area  1.0.0.2        2.2.2.3        0x80000004      1039
backbone              opaque-area  1.0.0.2        3.3.3.4        0x80000004      1037
backbone              opaque-area  1.0.0.2        4.4.4.5        0x80000004      1038
backbone              opaque-area  1.0.0.3        2.2.2.3        0x80000004      1039
backbone              opaque-area  1.0.0.3        11.11.11.1     0x80000004      1037
  ...
```

Creating basic TE tunnel

Assume that we want to create TE tunnel from R1 to R5. In order to do this, tunnel path specification must be created:

```
[admin@R1] > /mpls traffic-eng tunnel-path add use-cspf=yes name=dyn
```

This creates path template for purely dynamic path that will use CSPF.

Next, TE tunnel itself must be created:

```
[admin@R1] /interface traffic-eng> add name=te1 bandwidth=1000 primary-path=dyn \
   from-address=9.9.9.1 to-address=9.9.9.5 disabled=no record-route=yes
```

We can monitor tunnel to see its state:

```
[admin@R1] /interface traffic-eng> monitor 0
            tunnel-id: 7
    primary-path-state: established
          primary-path: dyn
  secondary-path-state: not-necessary
           active-path: dyn
          active-lspid: 1
          active-label: 29
       explicit-route: "S:1.1.1.2/32,S:2.2.2.2/32,S:2.2.2.3/32,S:4.4.4.3/32,S:4.4.4.5/32"
       recorded-route: "1.1.1.2[30],2.2.2.3[29],4.4.4.5[0]"
```

Notice, that CSPF has created explicit route that traverses R2, R3 and R5 (tail-end). TE tunnel was requested to record route it is traversing (by "record-route=yes" setting), recorded route is displayed in status along with labels that particular router has allocated for this tunnel.

Once TE tunnel is established, VPLS interface from R1 to R5 automatically switches to use this TE tunnel:

```
[admin@R1] /interface vpls> monitor 0
       remote-label: 24
        local-label: 25
      remote-status:
          transport: te1
  transport-nexthop: 1.1.1.2
     imposed-labels: 30,24
```

On routers in between R1 and R5, RSVP path and reservation state can be monitored, for example on R2:

```
[admin@R2] > /mpls traffic-eng path-state print
Flags: L - locally-originated, E - egress, F - forwarding, P - sending-path, R - sending-resv
 #      SRC             DST             BANDWIDTH  OUT-INTERFACE                      OUT-NEXT-HOP
 0  FPR 9.9.9.1:1       9.9.9.5:2       1000       ether2                             2.2.2.3
[admin@R2] > /mpls traffic-eng resv-state print
Flags: E - egress, A - active, N - non-output, S - shared
 #    SRC             DST             BANDWIDTH  LABEL               INTERFACE         NEXT-HOP
 0 AS 9.9.9.1:1       9.9.9.5:7       1000       30                  ether2            2.2.2.3
```

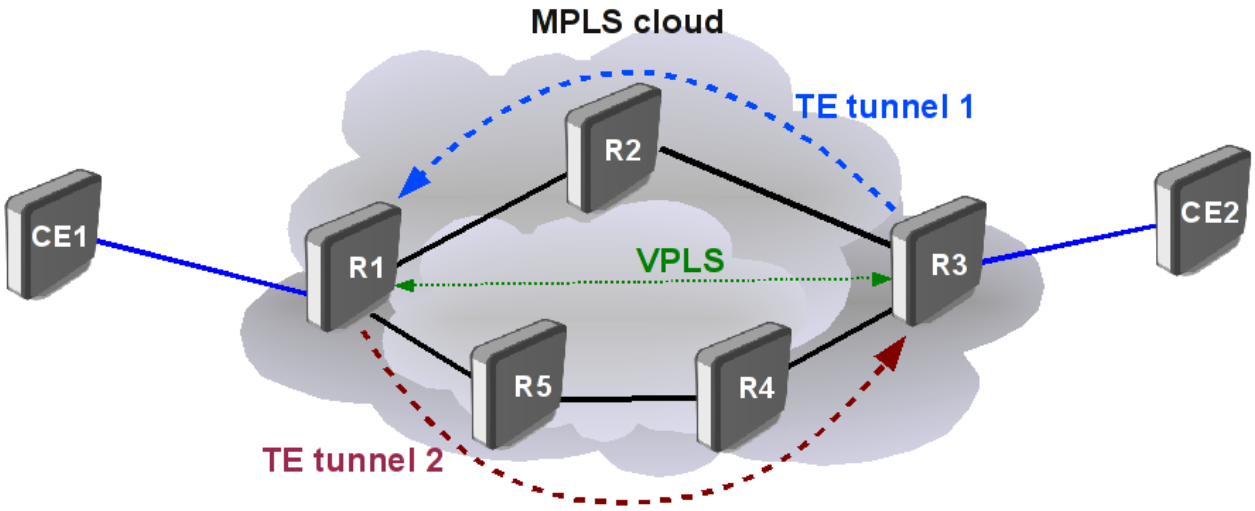
Note, that available bandwidth on ether2 interface (connected to R3) on R2 has changed:

```
[admin@R2] > /mpls traffic-eng interface print
Flags: X - disabled, I - invalid
 #   INTERFACE                                    BANDWIDTH  TE-METRIC  REMAINING-BW
 0   ether1                                       100000     1          100000
 1   ether2                                       100000     1          99000
```

Manual:TE Tunnels Example

Application example

Consider following setup:



IP Connectivity and LDP

R1

ether1 connects to R2, ether2 connects to R5

```
/system identity set name=R1

/interface bridge add name=lo0

/ip address
add address=192.168.55.1/30 interface=ether1
add address=192.168.55.18/30 interface=ether2
add address=10.255.1.1/32 interface=lo0


/routing ospf instance
set default router-id=10.255.1.1

/routing ospf network
add network=192.168.55.0/24 area=backbone
add network=10.255.1.0/24 area=backbone

/mpls ldp
set enabled=yes lsr-id=10.255.1.1 transport-address=10.255.1.1

/mpls ldp interface
add interface=ether1
add interface=ether2
```

R2

ether1 connects to R1, ether2 connects to R3

```
/system identity set name=R2

/interface bridge add name=lo0

/ip address
add address=192.168.55.2/30 interface=ether1
add address=192.168.55.5/30 interface=ether2
add address=10.255.1.2/32 interface=lo0



/routing ospf instance
set default router-id=10.255.1.2

/routing ospf network
add network=192.168.55.0/24 area=backbone
add network=10.255.1.0/24 area=backbone

/mpls ldp
set enabled=yes lsr-id=10.255.1.2 transport-address=10.255.1.2

/mpls ldp interface
add interface=ether1
add interface=ether2
```

R3

ether1 connects to R2, ether2 connects to R4

```
/system identity set name=R3

/interface bridge add name=lo0

/ip address
add address=192.168.55.6/30 interface=ether1
add address=192.168.55.9/30 interface=ether2
add address=10.255.1.3/32 interface=lo0



/routing ospf instance
set default router-id=10.255.1.3

/routing ospf network
add network=192.168.55.0/24 area=backbone
add network=10.255.1.0/24 area=backbone

/mpls ldp
```

```
set enabled=yes lsr-id=10.255.1.3 transport-address=10.255.1.3

/mpls ldp interface
add interface=ether1
add interface=ether2
```

R4

ether1 connects to R3, ether2 connects to R5

```
/system identity set name=R4

/interface bridge add name=lo0

/ip address
add address=192.168.55.10/30 interface=ether1
add address=192.168.55.13/30 interface=ether2
add address=10.255.1.4/32 interface=lo0


/routing ospf instance
set default router-id=10.255.1.4

/routing ospf network
add network=192.168.55.0/24 area=backbone
add network=10.255.1.0/24 area=backbone

/mpls ldp
set enabled=yes lsr-id=10.255.1.4 transport-address=10.255.1.4

/mpls ldp interface
add interface=ether1
add interface=ether2
```

R5

ether1 connects to R4, ether2 connects to R1

```
/system identity set name=R5

/interface bridge add name=lo0

/ip address
add address=192.168.55.14/30 interface=ether1
add address=192.168.55.17/30 interface=ether2
add address=10.255.1.5/32 interface=lo0


/routing ospf instance
set default router-id=10.255.1.5
```

```
/routing ospf network
add network=192.168.55.0/24 area=backbone
add network=10.255.1.0/24 area=backbone

/mpls ldp
set enabled=yes lsr-id=10.255.1.5 transport-address=10.255.1.5

/mpls ldp interface
add interface=ether1
add interface=ether2
```

After OSPF and LDP setup ensure that ospf is working properly

```
[admin@R1] /routing ospf neighbor> print
 0 instance=default router-id=10.255.1.5 address=192.168.55.17 interface=ether2
   priority=1 dr-address=192.168.55.17 backup-dr-address=192.168.55.18
   state="Full" state-changes=5 ls-retransmits=0 ls-requests=0 db-summaries=0
   adjacency=32m17s

 1 instance=default router-id=10.255.1.2 address=192.168.55.2 interface=ether1
   priority=1 dr-address=192.168.55.2 backup-dr-address=192.168.55.1
   state="Full" state-changes=5 ls-retransmits=0 ls-requests=0 db-summaries=0
   adjacency=32m17s
[admin@R1] /routing ospf neighbor>
```

```
[admin@R1] /ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 0 ADS  0.0.0.0/0                          10.1.101.1         0
 1 ADC  10.1.101.0/24      10.1.101.9      ether3             0
 2 ADC  10.255.1.1/32      10.255.1.1      lo0                0
 3 ADo  10.255.1.2/32                      192.168.55.2       110
 4 ADo  10.255.1.3/32                      192.168.55.2       110
 5 ADo  10.255.1.4/32                      192.168.55.17      110
 6 ADo  10.255.1.5/32                      192.168.55.17      110
 7 ADC  192.168.55.0/30    192.168.55.1    ether1             0
 8 ADo  192.168.55.4/30                    192.168.55.2       110
 9 ADo  192.168.55.8/30                    192.168.55.2       110
                                           192.168.55.17
10 ADo  192.168.55.12/30                   192.168.55.17      110
11 ADC  192.168.55.16/30   192.168.55.18   ether2             0
[admin@R1] /ip route>
```

Also make sure MPLS forwarding-table has label bindings

```
[admin@R1] /mpls forwarding-table> print
Flags: L - ldp, V - vpls, T - traffic-eng
 #   IN-LABEL      OUT-LABELS  DESTINATION                  I NEXTHOP
```

```
 0   expl-null
 1 L 16                        10.255.1.5/32                 e 192.168.55.17
 2 L 17            19          192.168.55.8/30               e 192.168.55.2
 3 L 18            19          10.255.1.4/32                 e 192.168.55.17
 4 L 19            21          10.255.1.3/32                 e 192.168.55.2
 5 L 20                        192.168.55.12/30              e 192.168.55.17
 6 L 21                        192.168.55.4/30               e 192.168.55.2
 7 L 22                        10.255.1.2/32                 e 192.168.55.2
```

VPLS tunnel

ether4 goes to CE routers

R1

```
/interface bridge add name=vpn

/interface vpls
add remote-peer=10.255.1.3 vpls-id=3:3

/interface bridge port
add interface=ether4 bridge=vpn
add interface=vpls1 bridge=vpn
```

R3

```
/interface bridge add name=vpn

/interface vpls
add remote-peer=10.255.1.1 vpls-id=3:3

/interface bridge port
add interface=ether4 bridge=vpn
add interface=vpls1 bridge=vpn
```

Make sure that VPLS tunnel is established and running

```
[admin@R1] /interface vpls> monitor 0 once
       remote-label: 23
        local-label: 23
      remote-status:
          transport: 10.255.1.3/32
  transport-nexthop: 192.168.55.2
     imposed-labels: 21,23

[admin@R1] /interface vpls>
```

TE Support

Traffic engineering needs RSVP protocol enabled on head end, tail end and forwarding routers. And additional setup to use CSPF.

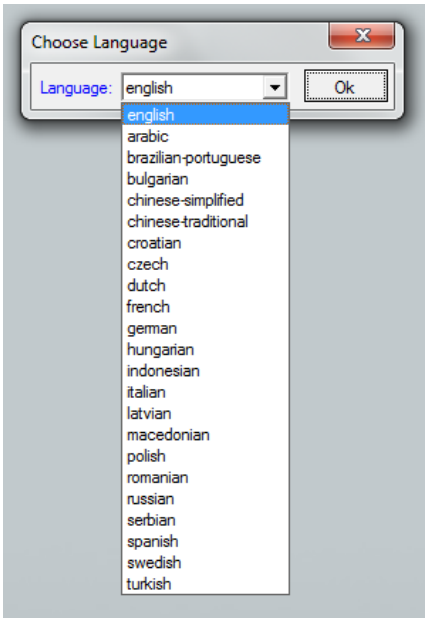
In our example all routers have the same configuration:

```
# set up CSPF
/routing ospf instance
set default mpls-te-area=backbone mpls-te-router-id=lo0

# add interfaces on which to run RSVP
/mpls traffic-eng interface
add interface=ether1 bandwidth=10Mbps
add interface=ether2 bandwidth=10Mbps
```

TE Tunnels

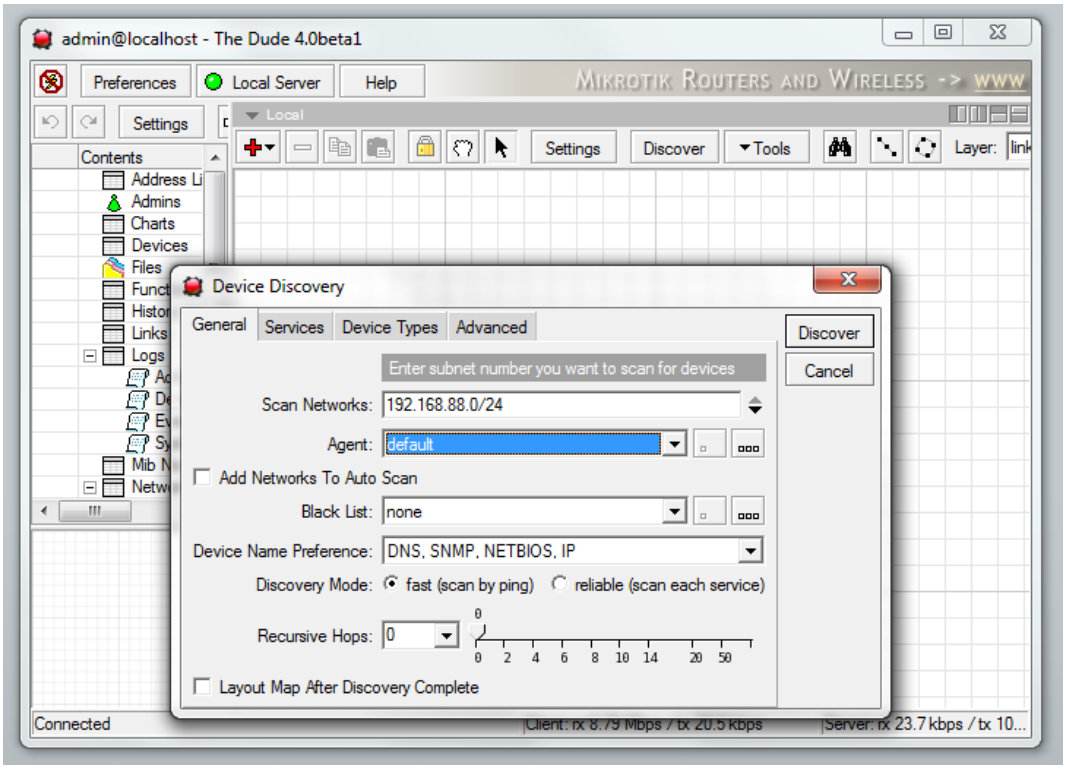
Manual:The Dude/First use

When first launching The Dude, you will be given the choice of the program language. Language translations are provided by other users of the program.



Note: You can also create and submit your own language translation, more instructions are avialable here.

After selecting the desired language, the Dude program will open, will automatically connect to the Localhost service, and will present you the Discovery window



Read more

- The Dude/Interface
- The Dude/Device discovery

Manual:The Dude/Installation

The Dude is free software, no purchase is necessary. You can download The Dude from the MikroTik webpage, in the Software section. On the Dude page, you will see **Stable** and **Beta** versions of the Dude, as well as special **NPK files** for The Dude support inside RouterOS.

Note: Generally Beta versions include more features, but could contain yet undiscovered issues. Stable versions are recommended for critical installations.

- The Dude changelog provides information about feature changes and bug fixes between versions.
- The Dude license provides legal information regarding the use of The Dude

System requirements

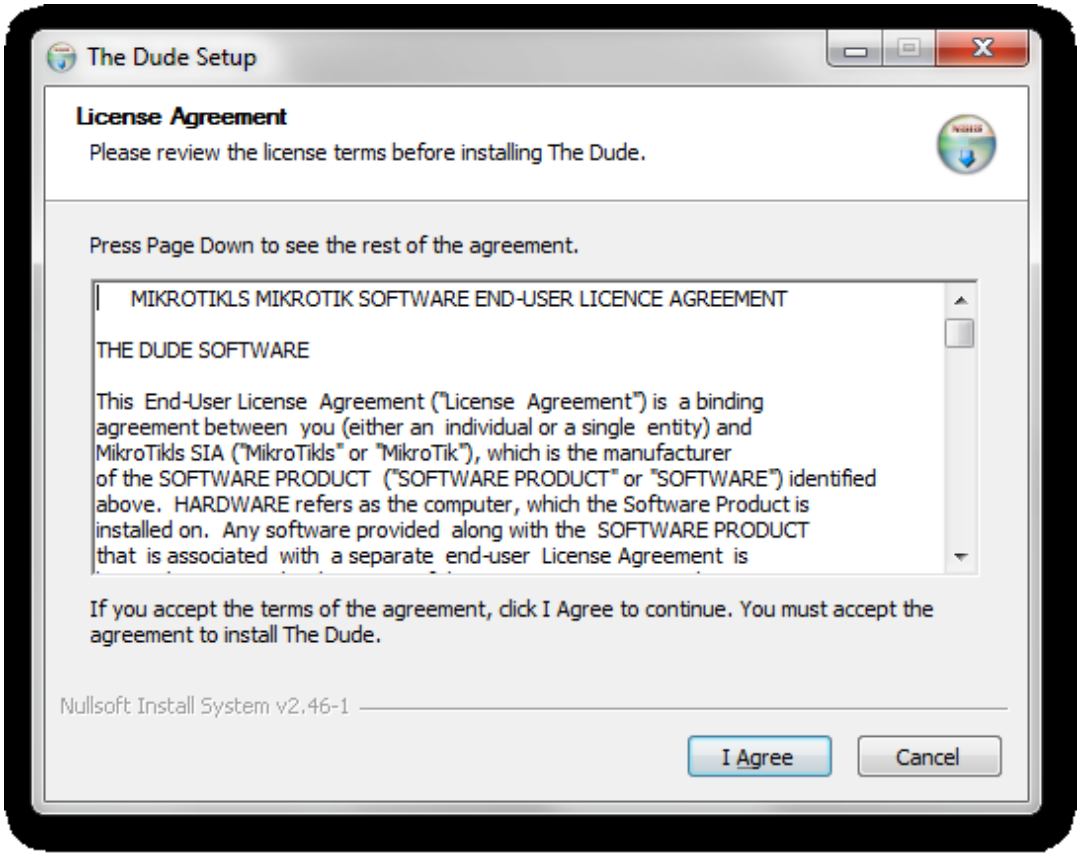
The Dude runs on most versions of Microsoft Windows. It is recommended to use Windows 2000 or newer. We have successfully used The Dude even on very low power machines, so generally, any system which can acceptably run Windows 2000 or Windows XP will be able to run The Dude.

The program can also be used on Linux and MacOS if using Wine [1] or Darwine [2] respectively.

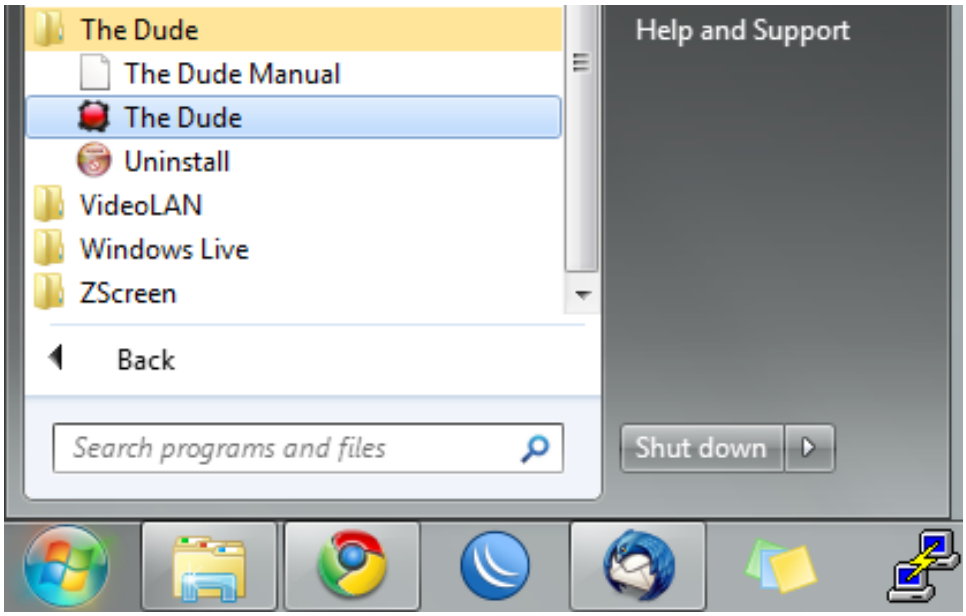
Installation process

- Download The Dude installation file

- After downloading the Dude installation file, run it to start the installation.



- After the installation process is complete, The Dude start menu item group will be created, and The Dude will be ready to use



Read more

- The Dude/First use

References

[1] http://www.winehq.org/
[2] http://winebottler.kronenberg.org/

Manual:TOC

[See Also TOC by Menus]

Basic

- First Time Startup
- Initial Configuration using WebFig
- Console Login Process
- Troubleshooting Tools
- Support output file
- RouterOS features
- RouterOS FAQ
- Connection Oriented Communication (TCP/IP)

Management tools

- Console
- Winbox
- WebFig

RouterOS Licensing

- License
- Purchasing a License for RouterOS
- Entering a RouterOS License key
- Replacement Key

RouterOS Installation and packages

- Default Configurations on RouterBOARDS
- RouterOS package types
- Upgrading RouterOS
- CD Install
- Netinstall
- Configuration Management

Interfaces

General interface list

- Ethernet
- Bonding (Link Aggregation)
- Bridging
- VRRP (High Availability)

Examples

- Bonding Examples
- VRRP Examples

Misc

- Switch Chip Features
- Maximum Transmission Units (MTU) on RouterBOARDs

Wireless

General reference and protocols

- Wireless Interface Reference
- Wireless AP Client
- Wireless Station Modes
- NV2 protocol
- WMM
- Spectrum Analyzer
- Wireless Advanced Channels
- HWMP+

Configuration examples

- Making A Simple Wireless AP

Misc

- Wireless FAQ
- Wireless Debug Logs

VPN

- Virtual Lan Network (VLAN)
- IP Security (Ipsec)

Point to point Tunnels

- Ethernet Over IP (EoIP)
- GRE tunnel
- IPIP tunnel

PPP tunnels

- PPPoE
- PPTP
- L2TP
- SSTP
- OpenVPN
- PPP tunnel bridging protocol (BCP)
- MLPPP

MPLS Based VPNs

- VPLS

IP/ IPv6 Addressing

Simple IPv4/IPv6 Routing

DHCP

- Ip security settings

IPv4

- Ip address
- ARP
- Load Balancing Multiple Same Subnet Links

IPv6

- IPv6 Address
- Neighbor Discovery and Stateless Auto Configuration
- My First IPv6 Network
- Creating IPv6 Loopback Address

IPv4

- Routes in general
- Simple Static Routing
- VRF

IPv6

- IPv6 Routing in general
- Simple IPv6 routing example

- DHCP Server
- DHCP Client
- DHCP Relay
- IPv4 address pool
- DHCPv6 Server
- DHCPv6 Client
- IPv6 address pool

IP/IPv6 Firewall

IP Firewall

- Filters
- NAT
- Mangle
- Address Lists
- Layer 7 (L7) rules
- Connection tracking

IPv6 Firewall

- Filters
- Mangle
- Address Lists

Misc

- RouterOS and firewall services
- Per connection classifier (PCC)
- Connection Rate
- NTH Matcher
- Routing Table Matcher

Dynamic Routing

- Routing filters

OSPF

- OSPF Case Studies
- OSPF Exampes
- OSPF and Point-to-Point Interfaces

OSPFv3

- OSPFv3 with Quagga

BGP

- BGP HowTo & FAQ
- BGP Soft Reconfiguration
- BGP Load Balancing
- Simple BGP Multihoming
- Using Scope and Target-scope Attributes

RIP

- Prefix list

MME

- MME Case Studies

Multicast Routing

Traffic control

Queue

- HTB
- Queue Size
- Bursting
- PCQ
- PCQ Examples
- Packet Flow Diagram

MPLS Based Traffic control

- Traffic Engineering Tunnels
- TE Tunnel Auto Bandwidth
- Simple TE tunnel Example
- TE Tunnels Example Setup for VPLS
- Traffic Engineering reference

MPLS

User Management

Virtualization

MPLS in General

- MPLS Overview
- MPLS Over PPPoE
- EXP Bit Behaviour

LDP

- LDP
- LDP Based VPLS
- Cisco VPLS

BGP VPLS

- BGP Based VPLS
- VPLS Control Word

L3VPN

- Virtual Routing And Forwarding
- Layer3 MPLS VPN Example
- EBGP as PE-CE Routing Protocol
- OSPF as PE-CE Routing Protocol

Traffic Engineering

- TE Tunnels
- TE Tunnel Auto Bandwidth

Reference

- mpls/traffic-eng
- interface/traffic-engineering

- Router AAA
- PPP AAA
- RADIUS Client
- User Manager

Hotspot

- Hotspot Introduction
- Customizing Hotspot
- Hotspot Reference

Virtualization in general

- KVM
- Metarouter
- XEN
- Virtual Ethernets

Console

- Serial and USB port configuration
- Console in general
- Console Login Process
- Line Editor

Console Access Methods

- Special Login
- Serial Console

Scripting

- Console Scripting
- Scripting Examples
- LUA Scripting

SSH

- SSH Client
- SSH Forwarding

Monitoring

- System Logging
- UPS Control and Monitoring
- LCD Display Control
- LCD Touch Screen Control
- GPS
- Traffic Flow (NetFlow)
- SNMP
- Graphing
- CPU Profiler
- Packet Sniffer
- Other Diagnostic Tools

Hardware

- Grounding
- Wireless Card Diagnostics
- RouterBOARD Bad Blocks
- Password Reset
- Flashfig
- Bootloader Upgrade

Other

- Certificates
- Create Certificates
- Advanced Traffic Generator
- Bandwidth Test tool
- LED configuration
- Administrator Notes
- File List
- Resource Monitoring
- Health Monitoring
- Store
- Watchdog

- Scheduler
- System Time
- API
- Web Proxy
- Fast Path

Manual:Tools/Bandwidth Test

Applies to RouterOS: v2.9, v3, v4+

Summary

Sub-menu: `/tool`
Packages required: `system`

The Bandwidth Tester can be used to measure the throughput to another MikroTik router (either wired or wireless) and thereby help to discover network "bottlenecks".

The TCP test uses the standard TCP protocol with acknowledgments and follows the TCP algorithm on how many packets to send according to latency, dropped packets, and other features in the TCP algorithm. Please review the TCP protocol for details on its internal speed settings and how to analyze its behavior. Statistics for throughput are calculated using the entire size of the TCP data stream. As acknowledgments are an internal working of TCP, their size and usage of the link are not included in the throughput statistics. Therefore this statistic is not as reliable as the UDP statistic when estimating throughput.

The UDP tester sends 110% or more packets than currently reported as received on the other side of the link. To see the maximum throughput of a link, the packet size should be set for the maximum MTU allowed by the links which is usually 1500 bytes. There is no acknowledgment required by UDP; this implementation means that the closest approximation of the throughput can be seen.

Warning: Bandwidth Test uses all available bandwidth (by default) and may impact network usability.

Note: Bandwidth Test uses a lot of resources. If you want to test real throughput of a router, you should run bandwidth test through the tested router not from or to it. To do this you need at least 3 routers connected in chain: the Bandwidth Server, the router being tested and the Bandwidth Client.

Note: If you use UDP protocol then Bandwidth Test counts IP header+UDP header+UDP data. In case if you use TCP then Bandwidth Test counts only TCP data (TCP header and IP header are not included).

Bandwidth Test Server

Sub-menu: `/tool bandwidth-server`

Property	Description
allocate-udp-ports-from (*integer 1000..64000*; Default: **2000**)	Beginning of UDP port range
authenticate (*yes* | *no*; Default: **yes**)	Communicate only with authenticated clients
enabled (*yes* | *no*; Default: **yes**)	Defines whether bandwidth server is enabled or not
max-sessions (*integer 1..1000*; Default: **100**)	Maximal simultaneous test count

Bandwidth Server:

```
[admin@MikroTik] /tool bandwidth-server> print
                  enabled: yes
             authenticate: yes
  allocate-udp-ports-from: 2000
             max-sessions: 100
[admin@MikroTik] /tool bandwidth-server>
```

Active sessions:

```
[admin@MikroTik] /tool bandwidth-server session> print
  # CLIENT          PROTOCOL DIRECTION USER
  0 35.35.35.1      udp      send      admin
  1 25.25.25.1      udp      send      admin
  2 36.36.36.1      udp      send      admin
[admin@MikroTik] /tool bandwidth-server session>
```

To enable **bandwidth-test** server without client authentication:

```
[admin@MikroTik] /tool bandwidth-server> set enabled=yes authenticate=no
[admin@MikroTik] /tool bandwidth-server> print
                  enabled: yes
             authenticate: no
  allocate-udp-ports-from: 2000
             max-sessions: 100
[admin@MikroTik] /tool bandwidth-server>
```

Bandwidth Test Client

Command name: `/tool bandwidth-test`

Property	Description
address (*IP address* | *IPv6 prefix[%interface]*; Default:)	IP address of host
direction (*both* | *receive* | *transmit*; Default: **receive**)	Direction of data flow
duration (*time*; Default:)	Duration of the test
interval (*time: 20ms..5s*; Default: **1s**)	Delay between reports (in seconds)
local-tx-speed (*integer 0..4294967295*; Default:)	Transfer test maximum speed (bits per second)
local-udp-tx-size (*integer: 28..64000*)	Local transmit packet size in bytes
password (*string*; Default: "")	Password for the remote user
protocol (*udp* | *tcp*; Default: **udp**)	Protocol to use
random-data (*yes* | *no*; Default: **no**)	If random-data is set to yes, the payload of the bandwidth test packets will have incompressible random data stream so that links that use data compression will not distort the results (this is CPU intensive and random-data should be set to no for low speed CPUs)
remote-tx-speed (*integer 0..4294967295*; Default:)	Receive test maximum speed (bits per second)
remote-udp-tx-size (*integer: 28..64000*)	Remote transmit packet size in bytes
tcp-connection-count (*integer 1..100*; Default: **20**)	Number of TCP connections to use
user (*string*; Default: "")	Remote user

Example

To run 15-second long bandwidth-test to the **10.0.0.32** host sending and receiving **1000**-byte UDP packets and using username **admin** to connect:

```
[admin@MikroTik] /tool> bandwidth-test 10.0.0.32 duration=15s \
\... direction=both local-udp-tx-size=1000 protocol=udp \
\... remote-udp-tx-size=1000 user=admin
                  status: done testing
                duration: 15s
              tx-current: 272.8Mbps
    tx-10-second-average: 200.3Mbps
        tx-total-average: 139.5Mbps
              rx-current: 169.6Mbps
    rx-10-second-average: 164.8Mbps
        rx-total-average: 117.0Mbps
            lost-packets: 373
             random-data: no
               direction: both
                 tx-size: 1000
                 rx-size: 1000
[admin@MikroTik] /tool>
```

Link-local IPv6 example:

```
[admin@dzeltenais_burkaans] > /tool bandwidth-test fe80::34:23ff:fe6a:570c%local

                  status: running
                duration: 5s
              rx-current: 23.9Mbps
    rx-10-second-average: 15.1Mbps
        rx-total-average: 15.1Mbps
            lost-packets: 0
             random-data: no
               direction: receive
                 rx-size: 1500
```

[Top | Back to Content]

Manual:Tools/Packet Sniffer

Applies to RouterOS: v5.8+

Summary

Sub-menu: `/tool sniffer`
Packages required: `system`

Packet sniffer is a tool that can capture and analyze packets that are going to, leaving or going through the router (except the traffic that passes only through the switch chip).

Packet Sniffer Configuration

Sub-menu: `/tool sniffer`

Property**`file-limit`** (*integer 10..4294967295[KiB]*; Default: **1000KiB**)**`file-name`** (*string*; Default:)**`filter-ip-address`** (*ip/mask[,ip/mask] (max 16 items)*; Default:)**`filter-mac-address`** (*mac/mask[,mac/mask] (max 16 items)*; Default:)**`filter-port`** (*[!]port[,port] (max 16 items)*; Default:)**`filter-ip-protocol`** (*[!]protocol[,protocol] (max 16 items)*; Default:)**`filter-mac-protocol`** (*[!]protocol[,protocol] (max 16 items)*; Default:)**`filter-stream`** (*yes* | *no*; Default: **yes**)**`filter-direction`** (*any* | *rx* | *tx*; Default:)**`interface`** (*all* | *name*; Default: **all**)**`memory-limit`** (*integer 10..4294967295[KiB]*; Default: **100KiB**)**`memory-scroll`** (*yes* | *no*; Default: **yes**)**`only-headers`** (*yes* | *no*; Default: **no**)**`streaming-enabled`** (*yes* | *no*; Default: **no**)**`streaming-server`** (*IP*; Default: **0.0.0.0**)DescriptionFile size limit. Sniffer will stop when limit is reached.Name of the file where sniffed packets will be saved.Up to 16 ip addresses used as a filterUp to 16 MAC addresses and MAC address masks used as a filterUp to 16 comma separated entries used as a filterUp to 16 comma separated entries used as a filter IP protocols (instead of protocol names, protocol number can be used)

- **ipsec-ah** - IPsec AH protocol
- **ipsec-esp** - IPsec ESP protocol

- **ddp** - datagram delivery protocol
- **egp** - exterior gateway protocol
- **ggp** - gateway-gateway protocol
- **gre** - general routing encapsulation
- **hmp** - host monitoring protocol
- **idpr-cmtp** - idpr control message transport
- **icmp** - internet control message protocol
- **icmpv6** - internet control message protocol v6
- **igmp** - internet group management protocol
- **ipencap** - ip encapsulated in ip
- **ipip** - ip encapsulation
- **encap** - ip encapsulation
- **iso-tp4** - iso transport protocol class 4
- **ospf** - open shortest path first
- **pup** - parc universal packet protocol
- **pim** - protocol independent multicast
- **rspf** - radio shortest path first
- **rdp** - reliable datagram protocol
- **st** - st datagram mode
- **tcp** - transmission control protocol
- **udp** - user datagram protocol
- **vmtp** - versatile message transport
- **vrrp** - virtual router redundancy protocol
- **xns-idp** - xerox xns idp
- **xtp** - xpress transfer protocol

Up to 16 comma separated entries used as a filter. Mac protocols (instead of protocol names, protocol number can be used):

- **arp** - Address Resolution Protocol
- **ip** - Internet Protocol
- **ipv6** - Internet Protocol next generation
- **ipx** - Internetwork Packet Exchange
- **rarp** - Reverse Address Resolution Protocol

Sniffed packets that are devised for sniffer server are ignoredSpecifies om which direction filtering will be applied.Interface name on which sniffer will be running. **all** indicates that sniffer will sniff packets on all interfaces.Memory amount used to store sniffed data.Whether to rewrite older sniffed data when memory limit is reached.Save in the memory only packet's headers not the whole packet.Defines whether to send sniffed packets to streaming serverTazmen Sniffer Protocol (TZSP) stream receiver

Example

In the following example **streaming-server** will be added, streaming will be enabled, **file-name** will be set to test and packet sniffer will be started and stopped after some time:

```
[admin@MikroTik] tool sniffer> set streaming-server=192.168.0.240 \
\... streaming-enabled=yes file-name=test.pcap
[admin@MikroTik] tool sniffer> print
            interface: all
         only-headers: no
```

```
         memory-limit: 100KiB
        memory-scroll: yes
            file-name: test.pcap
           file-limit: 1000KiB
    streaming-enabled: yes
     streaming-server: 192.168.0.240
        filter-stream: yes
   filter-mac-address:
  filter-mac-protocol:
    filter-ip-address:
   filter-ip-protocol:
          filter-port:
     filter-direction: any
              running: no
[admin@MikroTik] tool sniffer> start
[admin@MikroTik] tool sniffer> stop
```

Running Packet Sniffer

Commands: `/tool sniffer start, /tool sniffer stop, /tool sniffer save`

The commands are used to control runtime operation of the packet sniffer. The **start** command is used to start/reset sniffering, **stop** - stops sniffering. To save currently sniffed packets in a specific file **save** command is used.

It is also possible to use quick mode.

Example

In the following example the packet sniffer will be started and after some time - stopped:

```
[admin@MikroTik] tool sniffer> start
[admin@MikroTik] tool sniffer> stop
```

Below the sniffed packets will be saved in the file named test:

```
[admin@MikroTik] tool sniffer> save file-name=test
[admin@MikroTik] tool sniffer> /file print
  # NAME                            TYPE         SIZE       CREATION-TIME
  0 test                            unknown      1350       apr/07/2003 16:01:52
[admin@MikroTik] tool sniffer>
```

Sniffed Packets

Sub-menu: `/tool sniffer packet`

This sub-menu allows to see the list of sniffed packets.

```
[admin@SXT test] /tool sniffer packet> print
 #    TIME INTERFACE SRC-ADDRESS                                DST-ADDRESS
120   1.993 ether1    10.5.101.1:646                            224.0.0.2:646                          >
121   2.045 ether1    10.5.101.15:8291 (winbox)                 10.5.101.10:36771                      >
122   2.046 ether1    10.5.101.15:8291 (winbox)                 10.5.101.10:36771                      >
123   2.255 ether1    fe80::20c:42ff:fe49:fcec                  ff02::5                                >
```

Property	Description
data (*read-only: text*)	Specified data inclusion in packets
direction (*read-only: in | out*)	Indicates whether packet is entering (**in**) or leaving (**out**) the router
dscp (*read-only: integer*)	IP DSCP field value
dst-address (*read-only: IP address*)	Destination IP address
fragment-offset (*read-only: integer*)	IP fragment offset
identification (*read-only: integer*)	IP identification
interface (*read-only: name*)	Name of the interface the packet has been captured on
ip-header-size (*read-only: integer*)	The size of IP header
ip-packet-size (*read-only: integer*)	The size of IP packet
ip-protocol (*read-only: ddp | egp | encap | ggp | gre | hmp | icmp | icmpv6 | dpr-cmt | igmp | ip | ipencap | ipip | ipsec-ah | ipsec-esp | iso-tp4 | ospf | pim | pup | rdp | rspft | st | tcp | udp | vmtp | vrrp | xns-idp | xtp*)	The name/number of IP protocol
protocol (*read-only: ip | arp | rarp | ipx | ipv6*)	The name/number of ethernet protocol
size (*read-only: integer*)	Size of packet
src-address (*read-only: IP address*)	Source IP address
src-mac (*read-only: MAC address*)	Source MAC address
data (*read-only: string*)	IP data
tcp-flags (*read-only: ack | cwr | ece | fin | psh | rst | syn | urg*)	TCP flags
time (*read-only: time*)	Time when packet arrived
ttl (*read-only: integer*)	IP Time To Live
vlan-id (*read-only: integer*)	VLAN-ID of the packet
vlan-priority (*read-only: integer*)	VLAN-Priority of the packet

Packet Sniffer Protocols

Sub-menu: `/tool sniffer protocol`

In this submenu you can see all sniffed protocols and their share of the whole sniffed amount.

```
[admin@SXT test] /tool sniffer protocol> print
 # PROTOCOL IP-PROTOCOL PORT                                          PACKETS       BYTES         SHARE
 0 802.2                                                                    1          60         0.05%
 1 ip                                                                     215      100377        99.04%
 2 arp                                                                      2         120         0.11%
 3 ipv6                                                                     6         788         0.77%
 4 ip       tcp                                                           210       99981        98.65%
 5 ip       udp                                                             3         228         0.22%
 6 ip       ospf                                                            2         168         0.16%
 7 ip       tcp         8291 (winbox)                                     210       99981        98.65%
 8 ip       tcp         36771                                             210       99981        98.65%
 9 ip       udp         646                                               3         228         0.22%
```

Property	Description
bytes (*read-only: integer*)	Total number of data bytes
ip-protocol (*read-only: ddp | egp | encap | ggp | gre | hmp | icmp | icmpv6 | dpr-cmt | igmp | ip | ipencap | ipip | ipsec-ah | ipsec-esp | iso-tp4 | ospf | pim | pup | rdp | rspft | st | tcp | udp | vmtp | vrrp | xns-idp | xtp*)	IP protocol
packets (*read-only: integer*)	The number of packets
port (*read-only: integer*)	The port of TCP/UDP protocol
protocol (*read-only: ip | arp | rarp | ipx | ipv6*)	The name/number of the protocol
share (*read-only: decimal*)	Specific type of traffic compared to all traffic in bytes

Packet Sniffer Host

Sub-menu: `/tool sniffer host`

The submenu shows the list of hosts that were participating in data excange you've sniffed.

```
[admin@SXT test] /tool sniffer host> print
 # ADDRESS         RATE                PEEK-RATE           TOTAL
 0 10.5.101.3      0bps/0bps           0bps/720bps         0/90
 1 10.5.101.10     0bps/0bps           175.0kbps/19.7kbps  61231/7011
 2 10.5.101.13     0bps/0bps           0bps/608bps         0/76
 3 10.5.101.14     0bps/0bps           0bps/976bps         0/212
 4 10.5.101.15     0bps/0bps           19.7kbps/175.0kbps  7011/61231
 5 224.0.0.2       0bps/0bps           608bps/0bps         76/0
 6 224.0.0.5       0bps/0bps           1440bps/0bps        302/0
[admin@SXT test] /tool sniffer host>
```

Property	Description
address (*read-only: IP address*)	IP address of the host
peek-rate (*read-only: integer/integer*)	The maximum data-rate received/transmitted
rate (*read-only: integer/integer*)	Current data-rate received/transmitted
total (*read-only: integer/integer*)	Total packets received/transmitted

Packet Sniffer Connections

Sub-menu: `/tool sniffer connection`

Here you can get a list of the connections that have been watched during the sniffing time.

```
[admin@MikroTik] tool sniffer connection> print
Flags: A - active
  #   SRC-ADDRESS       DST-ADDRESS             BYTES     RESENDS   MSS
  0 A 10.0.0.241:1839   10.0.0.181:23 (telnet)  6/42      60/0      0/0
  1 A 10.0.0.144:2265   10.0.0.181:22 (ssh)     504/252   504/0     0/0
[admin@MikroTik] tool sniffer connection>
```

Property	Description
active (*read-only: yes* | *no*)	Indicates whether connection is active or not
bytes (*read-only: integer/integer*)	Bytes in the current connection
dst-address (*read-only: IP address:port*)	Destination address
mss (*read-only: integer/integer*)	Maximum segment size
resends (*read-only: integer/integer*)	The number of packets resends in the current connection
src-address (*read-only: IP address:port*)	Source address

Quick mode

Quick mode will display results as they are filtered out with limited size buffer for packets. There are several attributes that can be set up filtering. If no attributes are set current configuration will be used.

Property**durationfreeze-frame-intervalinterfaceip-addressip-protocolmac-addressmac-protocolp** of the test in secondstime between data printoutintarface name or **all**up to 16 addresses to filter one of listed protocols, up to 16 entries

- **ipsec-ah** - IPsec AH protocol ***ipsec-esp** - IPsec ESP protocol
- **ddp** - datagram delivery protocol
- **egp** - exterior gateway protocol
- **ggp** - gateway-gateway protocol
- **gre** - general routing encapsulation
- **hmp** - host monitoring protocol
- **idpr-cmtp** - idpr control message transport
- **icmp** - internet control message protocol
- **icmpv6** - internet control message protocol v6
- **igmp** - internet group management protocol
- **ipencap** - ip encapsulated in ip
- **ipip** - ip encapsulation
- **encap** - ip encapsulation
- **iso-tp4** - iso transport protocol class 4
- **ospf** - open shortest path first
- **pup** - parc universal packet protocol
- **pim** - protocol independent multicast
- **rspf** - radio shortest path first
- **rdp** - reliable datagram protocol
- **st** - st datagram mode
- **tcp** - transmission control protocol
- **udp** - user datagram protocol
- **vmtp** - versatile message transport
- **vrrp** - virtual router redundancy protocol
- **xns-idp** - xerox xns idp
- **xtp** - xpress transfer protocol

up to 16 MAC addresses to filterup 16 entries

- **arp** - Address Resolution Protocol
- **ip** - Internet Protocol
- **ipv6** - Internet Protocol next generation

- **ipx** - Internetwork Packet Exchange
- **rarp** - Reverse Address Resolution Protocol

up to 16 entries to filter by

```
[admin@SXT test] /tool sniffer> quick interface=ether1
INTERFACE  TIME  NUM DI SRC-MAC           DST-MAC           VLAN SRC-ADDRESS
ether1    3.145  210 <- 00:24:1D:17:81:F7 00:0C:42:CB:DE:62      10.5.101.10:36771
ether1    3.145  211 -> 00:0C:42:CB:DE:62 00:24:1D:17:81:F7      10.5.101.15:8291 (winbox)
ether1    3.183  212 <- 00:24:1D:17:81:F7 00:0C:42:CB:DE:62      10.5.101.10:36771
ether1    3.184  213 -> 00:0C:42:CB:DE:62 00:24:1D:17:81:F7      10.5.101.15:8291 (winbox)
ether1    3.195  214 -> 00:0C:42:CB:DE:62 00:24:1D:17:81:F7      10.5.101.15:8291 (winbox)
ether1    3.195  215 -> 00:0C:42:CB:DE:62 00:24:1D:17:81:F7      10.5.101.15:8291 (winbox)
ether1    3.195  216 <- 00:24:1D:17:81:F7 00:0C:42:CB:DE:62      10.5.101.10:36771
ether1    3.217  217 <- 00:24:1D:17:81:F7 00:0C:42:CB:DE:62      10.5.101.10:36771
ether1    3.218  218 -> 00:0C:42:CB:DE:62 00:24:1D:17:81:F7      10.5.101.15:8291 (winbox)
ether1     3.22  219 <- 00:0C:42:49:FC:EC 33:33:00:00:00:05      fe80::20c:42ff:fe49:fcec
ether1    3.255  220 <- 00:0C:42:A1:6E:47 01:00:5E:00:00:05      192.168.15.6
ether1    3.256  221 <- 00:24:1D:17:81:F7 00:0C:42:CB:DE:62      10.5.101.10:36771
ether1    3.259  222 <- 00:24:1D:17:81:F7 00:0C:42:CB:DE:62      10.5.101.10:36771
ether1    3.294  223 -> 00:0C:42:CB:DE:62 00:24:1D:17:81:F7      10.5.101.15:8291 (winbox)
ether1    3.325  224 -> 00:0C:42:CB:DE:62 00:24:1D:17:81:F7      10.5.101.15:8291 (winbox)
ether1    3.325  225 -> 00:0C:42:CB:DE:62 00:24:1D:17:81:F7      10.5.101.15:8291 (winbox)
ether1    3.326  226 <- 00:24:1D:17:81:F7 00:0C:42:CB:DE:62      10.5.101.10:36771
ether1     3.35  227 <- 00:0C:42:A8:ED:C2 33:33:00:00:00:01      fe80::20c:42ff:fea8:edc2
ether1    3.391  228 <- 00:24:1D:17:81:F7 00:0C:42:CB:DE:62      10.5.101.10:36771
ether1    3.392  229 -> 00:0C:42:CB:DE:62 00:24:1D:17:81:F7      10.5.101.15:8291 (winbox)
```

Download Sniffer Results

Sub-menu: `/tool sniffer`

Packet Sniffer results could be downloaded and viewed as file by specific program (for example Wireshark [1]).

Property	Description
file-name (*string*; Default: "")	The name of the file where the sniffed packets will be saved to

To save sniffed result to file set,

```
[admin@MikroTik] /tool sniffer set file-name=example
```

Run sniffer with required settings,

```
[admin@MikroTik] /tool sniffer start
```

Do not forget to stop sniffer after sniffing is done,

```
[admin@MikroTik] /tool sniffer stop
```

Sniffed results could be downloaded from /file by FTP client or Windows Drag-n-Drop (do not forget to use **binary** mode, when file is downloaded by FTP).

```
[admin@MikroTik] /file print
 # NAME               TYPE             SIZE                 CREATION-TIME
```

```
 0 example            file              44092                jan/02/2010 01:11:59
```

[Top | Back to Content]

References

[1] http://www.wireshark.org/

Manual:Tools/Traffic Monitor

Applies to RouterOS: v2.9, v3, v4+

Summary

Sub-menu: `/tool traffic-monitor`
Packages required: `advanced-tools`

The traffic monitor tool is used to execute console scripts when interface traffic crosses a given threshold. Each item in traffic monitor list consists of its name (which is useful if you want to disable or change properties of this item from another script), some parameters, specifying traffic condition, and the pointer to a script or scheduled event to execute when this condition is met.

Properties

Property	Description
interface (*name*; Default:)	Interface to monitor
name (*string*; Default:)	Name of the traffic monitor item
on-event (*string*; Default:)	Script source
threshold (*integer: 0..4294967295*; Default: **1000000**)	Traffic threshold, in bits per second
traffic (*received* | *transmitted*; Default: **transmitted**)	Type of traffic to monitor • **received** - Received packets • **transmitted** - Transmitted packets
trigger (*above* | *always* | *below*; Default: **above**)	Condition on which to execute script • **above** -- The script will be run each time traffic exceeds the threshold • **below** -- Triggers script in the opposite condition, when traffic drops under the threshold • **always** -- Triggers script on both **above** and **below** conditions

Example

In this example the traffic monitor enables the interface ether2, if the received traffic exceeds 15kbps on ether1, and disables the interface ether2, if the received traffic falls below 12kbps on ether1.

```
[admin@MikroTik] > system script
[admin@MikroTik] /system script> add name=eth-down source="/interface disable \
ether2"
[admin@MikroTik] /system script> add name=eth-up source="/interface enable \
ether2"

[admin@MikroTik] > tool traffic-monitor
[admin@MikroTik] /tool traffic-monitor> add disabled=no interface=ether1 \
name=turn_on on-event=eth-up threshold=15000 traffic=received trigger=above
[admin@MikroTik] /tool traffic-monitor> add disabled=no interface=ether1 \
name=turn_off on-event=eth-down threshold=12000 traffic=received trigger=below

[admin@MikroTik] /tool traffic-monitor> print
Flags: X - disabled, I - invalid
 #   NAME         INTERFACE       TRAFFIC     TRIGGER THRESHOLD  ON-EVENT
 0   turn_on      ether1          received    above   15000      eth-up
 1   turn_off     ether1          received    below   12000      eth-down
[admin@MikroTik] /tool traffic-monitor>
```

[Top | Back to Content]

Manual:Troubleshooting tools

Troubleshooting tools

Before, we look at the most significant commands for connectivity checking and troubleshooting, here is little reminder on how to check host computer's network interface parameters on .

The Microsoft windows have a whole set of helpful command line tools that helps testing and configuring LAN/WAN interfaces. We will look only at commonly used Windows networking tools and commands.

All of the tools are being ran from windows terminal. Go to ***Start/Run*** and enter "cmd" to open a Command window.

Some of commands on windows are:

ipconfig – used to display the TCP/IP network configuration values. To open it, enter "`ipconfig`" in the command prompt.

```
C:\>ipconfig
Windows IP Configuration
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . : mshome.net
   Link-local IPv6 Address . . . . . : fe80::58ad:cd3f:f3df:bf18%8
   IPv4 Address. . . . . . . . . . . : 173.16.16.243
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 173.16.16.1
```

There are also a variety of additional functions for *ipconfig*. To obtain a list of additional options, enter "`ipconfig /?`" or "`ipconfig -?`".

netstat – displays the active TCP connections and ports on which the computer is listening, Ethernet statistics, the IP routing table, statistics for the IP, ICMP, TCP, and UDP protocols. It comes with a number of options for displaying a variety of properties of the network and TCP connections "netstat –?".

nslookup – is a command-line administrative tool for testing and troubleshooting DNS servers. For example, if you want to know what IP address is "www.google.com", enter "*nslookup www.google.com*" and you will find that there are more addresses 74.125.77.99, 74.125.77.104, 74.125.77.147.

netsh – is a tool an administrator can use to configure and monitor Windows-based computers at a command prompt. It allows configure interfaces, routing protocols, routes, routing filters and display currently running configuration.

Very similar commands are available also on unix-like machines. Today in most of Linux distributions network settings can be managed via GUI, but it is always good to be familiar with the command-line tools. Here is the list of basic networking commands and tools on Linux:

ifconfig – it is similar like *ipconfig* commands on windows. It lets enable/disable network adapters, assigned IP address and netmask details as well as show currently network interface configuration.

iwconfig - iwconfig tool is like ifconfig and ethtool for wireless cards. That also view and set the basic Wi-Fi network details.

nslookup – give a host name and the command will return IP address.

netstat – print network connections, including port connections, routing tables, interface statistics, masquerade connections, and more. (*netstat – r, netstat - a*)

ip – show/manipulate routing, devices, policy routing and tunnels on linux-machine.

For example, check IP address on interface using *ip* command:

```
$ip addr show
```

You can add static route using *ip* following command:

ip route add {NETWORK address} via {next hop address} dev {DEVICE}, for example:

```
$ip route add 192.168.55.0/24 via 192.168.1.254 dev eth1
```

mentioned tools are only small part of networking tools that is available on Linux. Remember if you want full details on the tools and commands options use ***man*** command. For example, if you want to know all options on ***ifconfig*** write command ***man ifconfig*** in terminal.

Check network connectivity

Using the *ping* command

Ping is one of the most commonly used and known commands. Administration utility used to test whether a particular host is reachable across an Internet Protocol (IP) network and to measure the round-trip time for packets sent from the local host to a destination host, including the local host's own interfaces.

Ping uses Internet Control Message Protocol (ICMP) protocol for echo response and echo request. Ping sends ICMP *echo* request packets to the target host and waits for an ICMP response. Ping output displays the minimum, average and maximum times used for a ping packet to find a specified system and return.

From PC:

Windows:

```
C:\>ping 10.255.255.4
 Pinging 10.255.255.4 with 32 bytes of data:
Reply from 10.255.255.4: bytes=32 time=1ms TTL=61
Reply from 10.255.255.4: bytes=32 time<1ms TTL=61
Reply from 10.255.255.4: bytes=32 time<1ms TTL=61
Reply from 10.255.255.4: bytes=32 time<1ms TTL=61
Ping statistics for 10.255.255.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0%
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

Unix-like:

```
andris@andris-desktop:/$ ping 10.255.255.6
PING 10.255.255.6 (10.255.255.6) 56(84) bytes of data.
64 bytes from 10.255.255.6: icmp_seq=1 ttl=61 time=1.23 ms
64 bytes from 10.255.255.6: icmp_seq=2 ttl=61 time=0.904 ms
64 bytes from 10.255.255.6: icmp_seq=3 ttl=61 time=0.780 ms
64 bytes from 10.255.255.6: icmp_seq=4 ttl=61 time=0.879 ms
^C
--- 10.255.255.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.780/0.948/1.232/0.174 ms
```

Press Ctrl-C to stop ping process.

From MikroTik:

```
[admin@MikroTik] > ping 10.255.255.4
10.255.255.4 64 byte ping: ttl=62 time=2 ms
10.255.255.4 64 byte ping: ttl=62 time=8 ms
10.255.255.4 64 byte ping: ttl=62 time=1 ms
10.255.255.4 64 byte ping: ttl=62 time=10 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 1/5.2/10 ms
```

Press Ctrl-C to stop ping process.

Using the *traceroute* command

Traceroute displays the list of the routers that packet travels through to get to a remote host. The **traceroute** or **tracepath** tool is available on practically all Unix-like operating systems and **tracert** on Microsoft Windows operating systems.

Traceroute operation is based on TTL value and ICMP “Time Exceeded” massage. Remember that TTL value in IP header is used to avoid routing loops. Each hop decrements TTL value by 1. If the TTL reaches zero, the packet is discarded and ICMP Time Exceeded message is sent back to the sender when this occurs.

Initially by traceroute, the TTL value is set to 1 when next router finds a packet with TTL = 1 it sets TTL value to zero, and responds with an ICMP "time exceeded" message to the source. This message lets the source know that the packet traverses that particular router as a hop. Next time TTL value is incremented by 1 and so on. Typically, each router in the path towards the destination decrements the TTL field by one unit TTL reaches zero.

Using this command you can see how packets travel through the network and where it may fail or slow down. Using this information you can determine the computer, router, switch or other network device that possibly causing network issues or failures.

From Personal computer:

Windows:

```
C:\>tracert 10.255.255.2
Tracing route to 10.255.255.2 over a maximum of 30 hops
  1    <1 ms    <1 ms    <1 ms  10.13.13.1
  2     1 ms     1 ms     1 ms  10.255.255.2
Trace complete.
```

Unix-like:

Traceroute and *tracepath* is similar, only tracepath does not not require superuser privileges.

```
andris@andris-desktop:~$ tracepath 10.255.255.6
 1:  andris-desktop.local (192.168.10.4)                   0.123ms pmtu 1500
 1:  192.168.10.1 (192.168.10.1)                           0.542ms
 1:  192.168.10.1 (192.168.10.1)                           0.557ms
 2:  192.168.1.2 (192.168.1.2)                             1.213ms
 3:  no reply
 4:  10.255.255.6 (10.255.255.6)                           2.301ms reached
     Resume: pmtu 1500 hops 4 back 61
```

From MikroTik:

```
[admin@MikroTik] > tool traceroute 10.255.255.1
     ADDRESS                                    STATUS
   1       10.0.1.17 2ms 1ms 1ms
   2    10.255.255.1 5ms 1ms 1ms
[admin@MikroTik] >
```

Log Files

System event monitoring facility allows to debug different problems using Logs. Log file is a text file created in the server/router/host capturing different kind of activity on the device. This file is the primary data analysis source. RouterOS is capable of logging various system events and status information. Logs can be saved in routers memory (RAM), disk, file, sent by email or even sent to remote syslog server.

All messages stored in routers local memory can be printed from `/log` menu. Each entry contains time and date when event occurred, topics that this message belongs to and message itself.

```
[admin@MikroTik] /log> print
15:22:52 system,info device changed by admin
16:16:29 system,info,account user admin logged out from 10.13.13.14 via winbox
16:16:29 system,info,account user admin logged out from 10.13.13.14 via telnet
16:17:16 system,info filter rule added by admin
16:17:34 system,info mangle rule added by admin
16:17:52 system,info simple queue removed by admin
16:18:15 system,info OSPFv2 network added by admin
```

Read more about logging on RouterOS here>>

Torch (/tool torch)

Torch is realtime traffic monitoring tool that can be used to monitor the traffic flow through an interface.

You can monitor traffic classified by protocol name, source address, destination address, port. Torch shows the protocols you have chosen and tx/rx data rate for each of them.

Example:

The following example monitor the traffic generated by the telnet protocol, which passes through the interface ether1.

```
[admin@MikroTik] tool> torch ether1 port=telnet
SRC-PORT                     DST-PORT                      TX         RX
1439                         23 (telnet)                   1.7kbps    368bps

[admin@MikroTik] tool>
```

To see what IP protocols are sent via ether1:

```
[admin@MikroTik] tool> torch ether1 protocol=any-ip
PRO.. TX         RX
tcp   1.06kbps   608bps
udp   896bps     3.7kbps
icmp  480bps     480bps
ospf  0bps       192bps

[admin@MikroTik] tool>
```

In order to see what protocols are linked to a host connected to interface 10.0.0.144/32 ether1:

```
[admin@MikroTik] tool> torch ether1 src-address=10.0.0.144/32 protocol=any

 PRO.. SRC-ADDRESS     TX         RX
 tcp   10.0.0.144      1.01kbps   608bps
 icmp  10.0.0.144      480bps     480bps
[admin@MikroTik] tool>
```

IPv6

Starting from v5RC6 torch is capable of showing IPv6 traffic. Two new parameters are introduced *src-address6* and *dst-address6*. Example:

```
admin@RB1100test] > /tool torch interface=bypass-bridge src-address6=::/0 ip-protocol=any sr
c-address=0.0.0.0/0
MAC-PROTOCOL    IP-PROT... SRC-ADDRESS                                TX         RX
ipv6            tcp        2001:111:2222:2::1                         60.1kbps   1005.4kbps
ip              tcp        10.5.101.38                                18.0kbps   3.5kbps
ip              vrrp       10.5.101.34                                0bps       288bps
ip              udp        10.5.101.1                                 0bps       304bps
ip              tcp        10.0.0.176                                 0bps       416bps
ip              ospf       224.0.0.5                                  544bps     0bps
                                                                      78.7kbps   1010.0kbps
```

To make */ping* tool to work with domain name that resolves IPv6 address use the following:

```
/ping [:resolve ipv6.google.com]
```

By default *ping* tool will take IPv4 address.

Winbox

More attractive Torch interface is available from Winbox (Tool>Torch). In Winbox you can also trigger a Filter bar by hitting the F key on the keyboard.

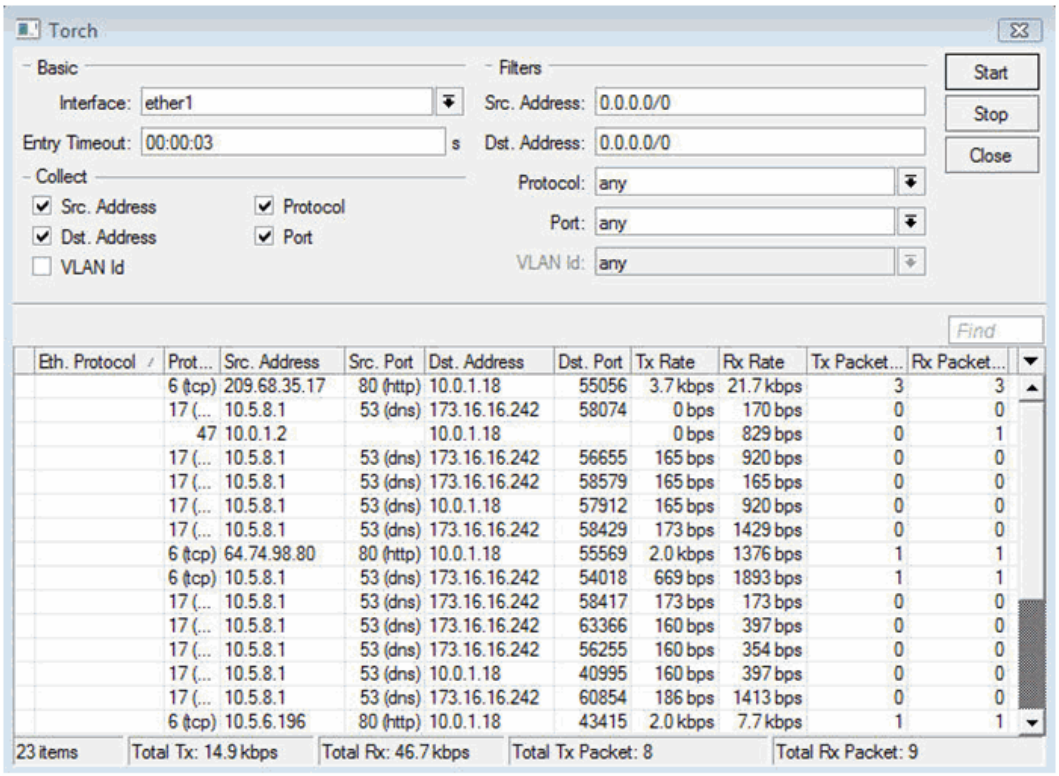


Figure 11.1 Torch tool GUI by winbox

Packet Sniffer (/tool sniffer)

Packet sniffer is a tool that can capture and analyze packets sent and received by specific interface. packet sniffer uses *libpcap* format.

Packet Sniffer Configuration

In the following example **streaming-server** will be added, streaming will be enabled, **file-name** will be set to test and packet sniffer will be started and stopped after some time:

```
[admin@MikroTik] tool sniffer> set streaming-server=192.168.0.240 \
\... streaming-enabled=yes file-name=test
[admin@MikroTik] tool sniffer> print
            interface: all
         only-headers: no
         memory-limit: 10
            file-name: "test"
           file-limit: 10
    streaming-enabled: yes
     streaming-server: 192.168.0.240
        filter-stream: yes
```

```
       filter-protocol: ip-only
       filter-address1: 0.0.0.0/0:0-65535
       filter-address2: 0.0.0.0/0:0-65535
               running: no
 [admin@MikroTik] tool sniffer> start
 [admin@MikroTik] tool sniffer> stop
```

Here you can specify different *packet sniffer* parameters, like maximum amount of used memory, file size limit in KBs.

Running Packet Sniffer Tool

There are three commands that are used to control runtime operation of the packet sniffer:

/tool sniffer start, ***/tool sniffer stop***, ***/tool sniffer save.***

The **start** command is used to start/reset sniffing, **stop** - stops sniffing. To save currently sniffed packets in a specific file **save** command is used.

```
In the following example the packet sniffer will be started and after some time - stopped:
```

```
 [admin@MikroTik] tool sniffer> start
 [admin@MikroTik] tool sniffer> stop
```

Below the sniffed packets will be saved in the file named *test*:

```
 [admin@MikroTik] tool sniffer> save file-name=test
```

View sniffed packets

There are also available different submenus for viewing sniffed packets.

- ***/tool sniffer packet – show*** the list of sniffed packets
- ***/tool sniffer protocol – show*** all kind of protocols that have been sniffed
- ***/tool sniffer host –*** shows the list of hosts that were participating in data exchange you've sniffed

For example:

```
 [admin@MikroTik] tool sniffer packet> print

  # TIME    INTERFACE   SRC-ADDRESS
  0 1.697    ether1    0.0.0.0:68 (bootpc)
  1 1.82     ether1    10.0.1.17
  2 2.007    ether1    10.0.1.18
  3 2.616    ether1    0.0.0.0:68 (bootpc)
  4 2.616    ether1    10.0.1.18:45630
  5 5.99     ether1    10.0.1.18
  6 6.057    ether1    159.148.42.138
  7 7.067    ether1    10.0.1.5:1701 (l2tp)
  8 8.087    ether1    10.0.1.18:1701 (l2tp)
  9 9.977    ether1    10.0.1.18:1701 (l2tp)
 -- more
```

Figure below shows sniffer GUI in Winbox, which is more user-friendly.

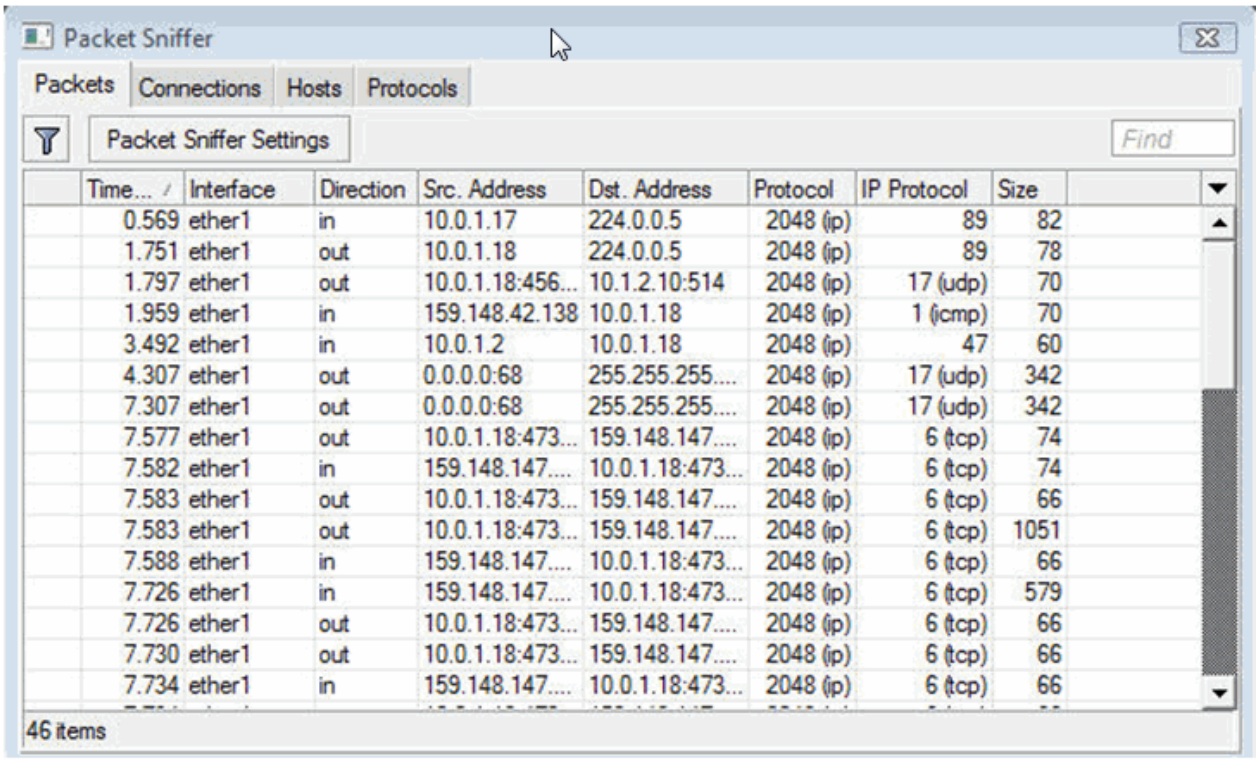


Figure 11.2 Packet Sniffer GUI by winbox

Detailed commands description can be found in the manual >>

Bandwidth test

The Bandwidth Tester can be used to measure the throughput (Mbps) to another MikroTik router (either wired or wireless network) and thereby help to discover network "*bottlenecks*"- network point with lowest throughput.

BW test uses two protocols to test bandwidth:

- TCP – uses the standard TCP protocol operation principles with all main components like connection initialization, packets acknowledgments, *congestion window* mechanism and all other features of TCP algorithm. Please review the TCP protocol for details on its internal speed settings and how to analyze its behavior. Statistics for throughput are calculated using the entire size of the TCP data stream. As acknowledgments are an internal working of TCP, their size and usage of the link are not included in the throughput statistics. Therefore statistics are not as reliable as the UDP statistics when estimating throughput.
- UDP traffic – sends 110% or more packets than currently reported as received on the other side of the link. To see the maximum throughput of a link, the packet size should be set for the maximum MTU allowed by the links which is usually 1500 bytes. There is no acknowledgment required by UDP; this implementation means that the closest approximation of the throughput can be seen.

Remember that *Bandwidth Test* uses all available bandwidth (by default) and may impact network usability.

If you want to test real throughput of a router, you should run bandwidth test through the router not from or to it. To do this you need at least 3 routers connected in chain:

Bandwidth Server – router under test – Bandwidth Client.

Note: If you use UDP protocol then Bandwidth Test counts IP header+UDP header+UDP data. In case if you use TCP then Bandwidth Test counts only TCP data (TCP header and IP header are not included).

Configuration example:

Server

To enable **bandwidth-test** server with client authentication:

```
[admin@MikroTik] /tool bandwidth-server> set enabled=yes authenticate=yes
[admin@MikroTik] /tool bandwidth-server> print
                  enabled: yes
             authenticate: yes
  allocate-udp-ports-from: 2000
             max-sessions: 100

[admin@MikroTik] /tool bandwidth-server>
```

Client

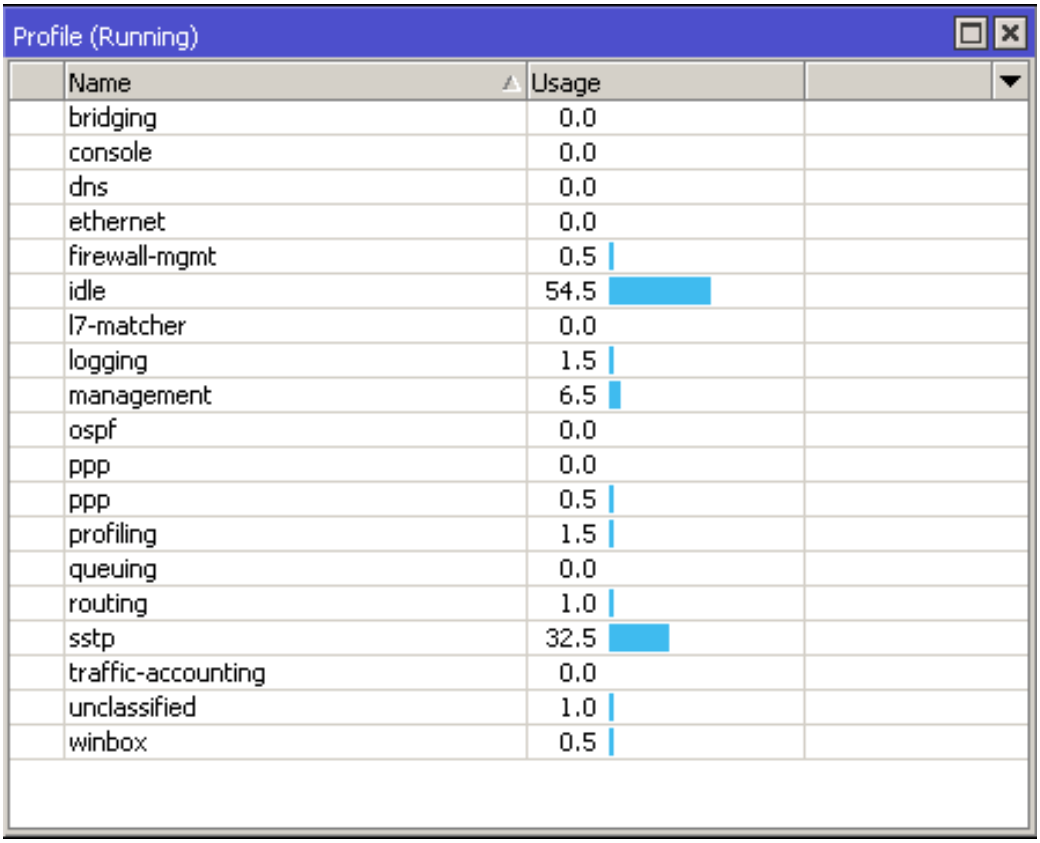
Run UDP bandwidth test in both directions, user name and password depends on remote Bandwidth Server. In this case user name is ‘admin’ without any password.

```
[admin@MikroTik] > tool bandwidth-test protocol=udp user=admin password="" direction=both \
address=10.0.1.5
                status: running
              duration: 22s
            tx-current: 97.0Mbps
  tx-10-second-average: 97.1Mbps
      tx-total-average: 75.2Mbps
            rx-current: 91.7Mbps
  rx-10-second-average: 91.8Mbps
      rx-total-average: 72.4Mbps
          lost-packets: 294
           random-data: no
             direction: both
               tx-size: 1500
               rx-size: 1500

-- [Q quit|D dump|C-z pause]
```

More information and all commands description can be found in the manual>>

Profiler

Profiler is a tool that shows CPU usage for each process running on RouterOS. It helps to identify which process is using most of the CPU resources.



Read more >>

[Top | Back to Content]

Manual:Interface/Traffic Engineering

Applies to RouterOS: v3, v4+

Properties

Sub-menu: `/interface traffic-eng`

Property	Description
affinity-exclude (*integer*; Default:)	Do not use interface if resource-class matches any of specified bits.
affinity-include-all (*integer*; Default:)	Use interface only if resource-class matches all of specified bits.
affinity-include-any (*integer*; Default:)	Use interface if resource-class matches any of specified bits.
auto-bandwidth-avg-interval (*time*; Default: **5m**)	Interval in which actual amount of data is measured, from which average bandwidth is calculated.
auto-bandwidth-range (*Disabled* | *Min[bps][-Max[bps]]*; Default: **0bps**)	Auto bandwidth adjustment range. `Read more >>`
auto-bandwidth-reserve (*integer[%]*; Default: **0%**)	Specifies percentage of additional bandwidth to reserve. `Read more >>`
auto-bandwidth-update-interval (*time*; Default: **1h**)	Interval during which tunnel keeps track of highest average rate.
bandwidth (*integer[bps]*; Default: **0bps**)	How much bandwidth to reserve for TE tunnel. Value is in bits per second. `Read more >>`
bandwidth-limit (*disabled* | *integer[%]*; Default: **disabled**)	Defines actual bandwidth limitation of TE tunnel. Limit is configured in percent of specified tunnel `bandwidth`. `Read more >>`
comment (*string*; Default:)	Short description of the item
disable-running-check (*yes* | *no*; Default: **no**)	Specifies whether to detect if interface is running or not. If set to **no** interface will always have `running` flag.
disabled (*yes* | *no*; Default: **yes**)	Defines whether item is ignored or used.
from-address (*auto* | *IP*; Default: **auto**)	Ingress address of the tunnel. If set to **auto** least IP address is picked.
holding-priority (*integer [0..7]*; Default:)	Is used to decide whether this session can be preempted by another session. 0 sets the highest priority.
mtu (*integer*; Default:)	
name (*string*; Default:)	Name of the interface
primary-path (*string*; Default:)	Primary label switching paths defined in `/mpls traffic-eng tunnel-path` menu.
primary-retry-interval (*time*; Default: **1m**)	Interval after which tunnel will try to use primary path.
record-route (*yes* | *no*; Default:)	If enabled, the sender node will receive information about the actual route that the LSP tunnel traverses. Record Route is analogous to a path vector, and hence can be used for loop detection.
reoptimize-interval (*time*; Default:)	Interval after which tunnel will re-optimize current path. If current path is not the best path then after optimization best path will be used. `Read more >>`

secondary-path (*string[,string]*; Default:)	List of label switching paths used by TE tunnel if primary path fails. Paths are defined in `/mpls traffic-eng tunnel-path` menu.
setup-priority (*integer[0..7]*; Default:)	Parameter is used to decide whether this session can preempt another session. 0 sets the highest priority.
to-address (*IP*; Default: **0.0.0.0**)	Remote end of TE tunnel.

Monitoring

To verify TE tunnel's status **monitor** command can be used.

```
[admin@R3] /interface traffic-eng> monitor 0
             tunnel-id: 12
    primary-path-state: on-hold
  secondary-path-state: established
        secondary-path: static
           active-path: static
          active-lspid: 3
          active-label: 66
        explicit-route: "S:192.168.55.10/32,L:192.168.55.13/32,L:192.168.55.17/32"
        recorded-route: "192.168.55.13[66],192.168.55.17[59],192.168.55.18[3]"
    reserved-bandwidth: 5.0Mbps
```

Reoptimization

Path can be re-optimized manually by entering following command `/interface traffic-eng reoptimize [id]`. It allows network administrators to reoptimize the LSPs that have been established based on changes in bandwidth, traffic, management policy, or other factors.

Lets say TE tunnel chose another path after link failure on best path. You can verify optimization by looking at **explicit-route** or **recorded-route** if `record-route` parameter is enabled.

```
[admin@R3] /interface traffic-eng> monitor 0
             tunnel-id: 12
    primary-path-state: established
          primary-path: dyn
  secondary-path-state: not-necessary
           active-path: dyn
          active-lspid: 1
          active-label: 67
        explicit-route: "S:192.168.55.10/32,S:192.168.55.13/32,S:192.168.55.14/32,
                         S:192.168.55.17/32,S:192.168.55.18/32"
        recorded-route: "192.168.55.13[67],192.168.55.17[60],192.168.55.18[3]"
    reserved-bandwidth: 5.0Mbps
```

Whenever the link comes back, TE tunnel will use the same path even it is not the best path (unless `reoptimize-interval` is configured). To fix it we can manually reoptimize tunnel path.

```
[admin@R3] /interface traffic-eng> reoptimize 0

[admin@R3] /interface traffic-eng> monitor 0
              tunnel-id: 12
```

```
   primary-path-state: established
         primary-path: dyn
secondary-path-state: not-necessary
          active-path: dyn
         active-lspid: 2
         active-label: 81
       explicit-route: "S:192.168.55.5/32,S:192.168.55.2/32,S:192.168.55.1/32"
       recorded-route: "192.168.55.2[81],192.168.55.1[3]"
  reserved-bandwidth: 5.0Mbps
```

Notice how explicit-route and recorded-route changed to shorter path.

See Also

- TE Tunnel Auto Bandwidth
- TE tunnels explained

[Top | Back to Content]

Manual:IP/Traffic Flow

Applies to RouterOS: 2.9, v3, v4 +

Summary

Sub-menu: `/ip traffic-flow`

MikroTik Traffic-Flow is a system that provides statistic information about packets which pass through the router. Besides network monitoring and accounting, system administrators can identify various problems that may occur in the network. With help of Traffic-Flow, it is possible to analyze and optimize the overall network performance. As Traffic-Flow is compatible with Cisco NetFlow, it can be used with various utilities which are designed for Cisco's NetFlow.

Traffic-Flow supports the following NetFlow formats:

- **version 1** - the first version of NetFlow data format, do not use it, unless you have to
- **version 5** - in addition to version 1, version 5 has possibility to inlude BGP AS and flow sequence number information. Currently RouterOS does not include BGP AS numbers.
- **version 9** - a new format which can be extended with new fields and record types thank's to its template-style design

General

Sub-menu: `/ip traffic-flow`

This section lists the configuration properties of Traffic-Flow.

Property	Description
interfaces (*string* | *all*; Default: **all**)	Names of those interfaces which will be used to gather statistics for traffic-flow. To specify more than one interface, separate them with a comma.
cache-entries (*128k* | *16k* | *1k* | *256k* | *2k* | ... ; Default: **4k**)	Number of flows which can be in router's memory simultaneously.
active-flow-timeout (*time*; Default: **30m**)	Maximum life-time of a flow.
inactive-flow-timeout (*time*; Default: **15s**)	How long to keep the flow active, if it is idle. If connection does not see any packet within this timeout, then traffic-flow will send packet out as new flow. If this timeout is too small it can create significant amount of flows and overflow the buffer.

Targets

Sub-menu: `/ip traffic-flow target`

With Traffic-Flow targets we specify those hosts which will gather the Traffic-Flow information from router.

Property	Description
address (*IP:port*; Default:)	IP address and port (UDP) of the host which receives Traffic-Flow statistic packets from the router.
v9-template-refresh (*integer*; Default: **20**)	Number of packets after which the template is sent to the receiving host (only for NetFlow version 9)
v9-template-timeout (*time*; Default:)	After how long to send the template, if it has not been sent.
version (*1* | *5* | *9*; Default:)	Which version format of NetFlow to use

Notes

By looking at packet flow diagram you can see that traffic flow is at the end of input, forward and output chain stack. It means that traffic flow will count only traffic that reaches one of those chains.

For example, you set up mirror port on switch, connect mirror port to router and set traffic flow to count mirrored packets. Unfortunately such setup will not work, because mirrored packets are dropped before they reach input chain.

Other interfaces will appear in report if traffic is passing thorugh them and monitored interface.

Examples

This example shows how to configure Traffic-Flow on a router

Enable Traffic-Flow on the router:

```
[admin@MikroTik] ip traffic-flow> set enabled=yes
[admin@MikroTik] ip traffic-flow> print
                enabled: yes
             interfaces: all
          cache-entries: 1k
```

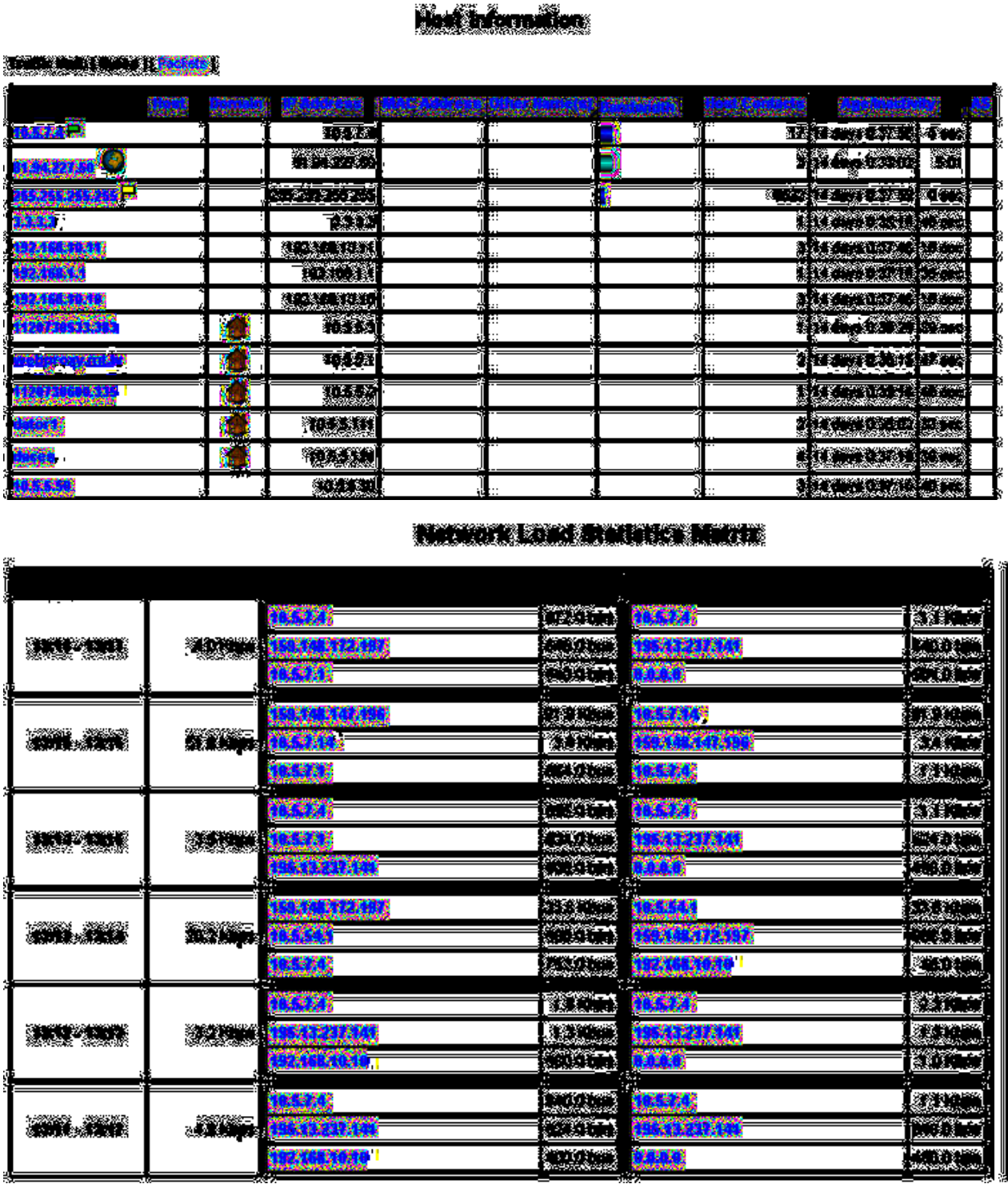
```
    active-flow-timeout: 30m
  inactive-flow-timeout: 15s
[admin@MikroTik] ip traffic-flow>
```

Specify IP address and port of the host, which will receive Traffic-Flow packets:

```
[admin@MikroTik] ip traffic-flow target> add address=192.168.0.2:2055 \
\... version=9
[admin@MikroTik] ip traffic-flow target> print
Flags: X - disabled
 #   ADDRESS               VERSION
 0   192.168.0.2:2055      9
[admin@MikroTik] ip traffic-flow target>
```

Now the router starts to send packets with Traffic-Flow information.

Some screenshots from NTop program [1], which has gathered Traffic-Flow information from our router and displays it in nice graphs and statistics. For example, where what kind of traffic has flown:


Host Information
Network Load Statistics Matrix

See more

- NetFlow Fundamentals [2]

[Top | Back to Content]

References

[1] http://www.ntop.org/download.html
[2] http://etutorials.org/Networking/network+management/Part+II+Implementations+on+the+Cisco+Devices/Chapter+7.+NetFlow/Fundamentals+of+NetFlow/

Manual:IP/UPnP

Applies to RouterOS: 2.9, v3, v4 +

Summary

Sub-menu: `/ip upnp`
Packages required: `system`

The MikroTik RouterOS supports Universal Plug and Play architecture for transparent peer-to-peer network connectivity of personal computers and network-enabled intelligent devices or appliances.

UPnP enables data communication between any two devices under the command of any control device on the network. Universal Plug and Play is completely independent of any particular physical medium. It supports networking with automatic discovery without any initial configuration, whereby a device can dynamically join a network. DHCP and DNS servers are optional and will be used if available on the network. UPnP implements simple yet powerfull NAT traversal solution, that enables the client to get full two-way peer-to-peer network support from behind the NAT.

There are two interface types for UPnP: internal (the one local clients are connected to) and external (the one the Internet is connected to). A router may only have one external interface with a 'public' IP address on it, and as many internal interfaces as needed, all with source-NATted 'internal' IP addresses.

The UPnP protocol is used for many modern applications, like most of DirectX games, as well as for various Windows Messenger features (remote asisstance, application sharing, file transfer, voice, video) from behind a firewall.

Additional Resources

UPnP Forum [1]

General Properties

Property	Description
allow-disable-external-interface (*yes* | *no* ; Default: **yes**)	whether or not should the users be allowed to disable router's external interface. This functionality (for users to be able to turn the router's external interface off without any authentication procedure) is required by the standard, but as it is sometimes not expected or unwanted in UPnP deployments which the standard was not designed for (it was designed mostly for home users to establish their ownlocal networks), you can disable this behavior
enabled (*yes* | *no* ; Default: **no**)	Enable uPnP service
show-dummy-rule (*yes* | *no* ; Default: **yes**)	Enable a workaround for some broken implementations, which are handling the absense of UPnP rules incorrectly (for example, popping up error messages). This option will instruct the server to install a dummy (meaningless) UPnP rule that can be observed by the clients, which refuse to work correctly otherwise

Warning: if you do not disable the allow-disable-external-interface, any user from the local network will be able (without any authentication procedures) to disable the router's external interface.

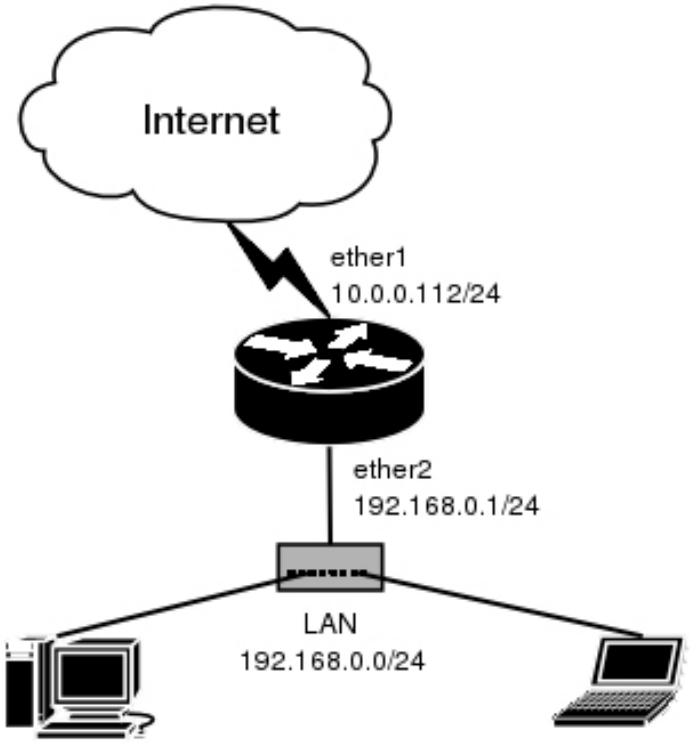
UPnP Interfaces

Sub-menu: `/ip upnp interfaces`

Property	Description
interface (*string*; Default:)	Interface name on which uPnP will be running
type (*external* | *internal*; Default: **no**)	uPnP interface type: • `external` - the interface a global IP address is assigned to • `internal` - router's local interface the clients are connected to

Note: It is highly recommended to upgrade DirectX runtime libraries to version DirectX 9.0c or higher and Windows Messenger to version Windows Messenger 5.0 or higher in order to get UPnP to work properly.

Configuration Example



We have masquerading already enabled on our router:

```
[admin@MikroTik] ip upnp> /ip firewall src-nat print
Flags: X - disabled, I - invalid, D - dynamic
  0   chain=srcnat action=masquerade out-interface=ether1
[admin@MikroTik] ip upnp>
```

To enable UPnP feature:

```
[admin@MikroTik] ip upnp> set enable=yes
[admin@MikroTik] ip upnp> print
                            enabled: yes
    allow-disable-external-interface: yes
                     show-dummy-rule: yes
[admin@MikroTik] ip upnp>
```

Now all we have to do is to add interfaces:

```
[admin@MikroTik] ip upnp interfaces> add interface=ether1 type=external
[admin@MikroTik] ip upnp interfaces> add interface=ether2 type=internal
[admin@MikroTik] ip upnp interfaces> print
Flags: X - disabled
  #   INTERFACE TYPE
  0 X ether1    external
  1 X ether2    internal

[admin@MikroTik] ip upnp interfaces> enable 0,1
[admin@MikroTik] ip upnp interfaces>
```

[Top | Back to Content]

References

[1] http://www.upnp.org/

Manual:User Manager

Introduction

- What is User Manager
- Requirements
- Supported browsers
- Demo
- Differences between version 3 and version 4-test

Getting started

- Download
- Install
- Create first subscriber
- First log on User Manager web

Quick start

- User Manager and HotSpot
- User Manager and PPP servers
- User Manager and DHCP
- User Manager and Wireless
- User Manager and RouterOS user

Concepts explained

Common

- Customers
- Users
- Routers
- Sessions
- Payments
- Reports
- Logs
- Customer permission levels
- Character constants
- Active sessions
- Active users
- Customer public ID

Version 4.x test package specific

- Profiles
- Limitations
- User data templates
- MAC binding
- Languages
- CoA (Radius incoming)

Version 3.x specific

- Subscribers
- Credits
- User prefix
- Time, traffic amount and rate limiting
- Prepaid and unlimited users
- Voucher template

Reference

Web interface

- Search patterns
- Tables:
  - Sorting
  - Filtering
  - Division in pages
  - Multiple object selection
  - Operations with selected objects
  - Minimization
  - Links to detail form
- Detail forms
- Page printing

Customer page

- Setup
- How to find it?
- Sections
  - Status
  - Routers
  - Credits
  - Users
  - Sessions
  - Customers
  - Reports
  - Logs

User page

- Setup
- How to find it?
- Link to user page
- Sections
  - Status
  - Payments
  - Settings

User sign-up

- Setup
- Sign-up steps
  - Creating account
  - Activating account
  - Login

User payments

- Authorize.Net
- PayPal

Questions and answers

- Quick introduction into User Manager setup
- How to separate users among customers?
- How to create a link to user page?
- How to create a link to user sign-up page?
- Visual bugs since upgrade
- Cannot log in User Manager
- Too many active sessions shown
- What does "active sessions" refer to?
- How to make Hotspot and User Manager on the same router?
- How to make MAC authentication in the User Manager?
- How to turn off logging for specific Routers?
- How to create timed Voucher?
- Cannot access User Manager WEB interface
- Incorrect time shown for sessions and credits
- User Manager does not allow to login due to expired uptime
- How to debug PayPal payments
- How to send logs to a remote host, using SysLog

Manual:Upgrading RouterOS

It is suggested to always keep your RouterOS installation up to date, MikroTik always keeps adding new functionality and improving performance and stability by releasing updates.

RouterOS versions are numbered sequentially, when a period is used to separate sequences, it does *not* represent a decimal point, and the sequences do *not* have positional significance. An identifier of 2.5, for instance, is not "two and a half" or "half way to version three", it is the fifth second-level revision of the second first-level revision. Therefore v5.2 is older than v5.18, which is newer.

Requirements and suggestions

In this article we assume that youre license allows upgrading. When using a RouterBOARD device, it is always suggested to upgrade it's RouterBOOT bootloader after RouterOS is upgraded. To do this, issue the command "*/system routerboard upgrade*"

Methods

You can upgrade RouterOS in the following ways:

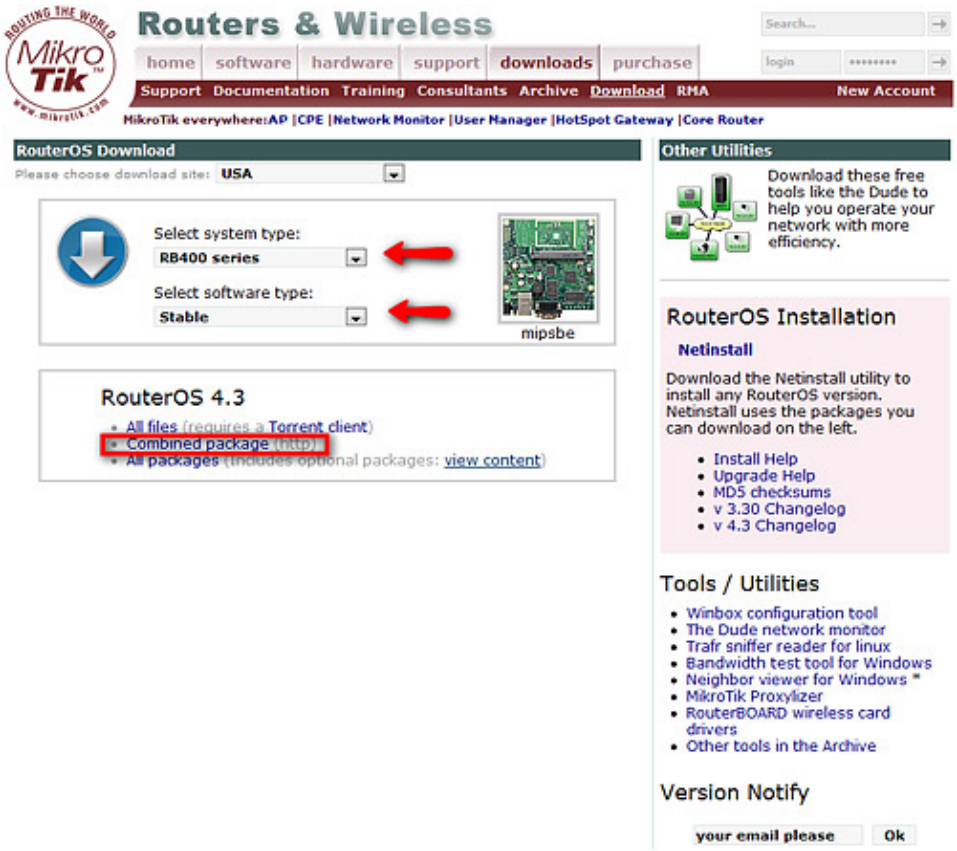
- Winbox – drag and drop files to the Files menu
- FTP - upload files to root directory
- The Dude – See manual here

Note: RouterOS cannot be upgraded through serial cable. Using this method only RouterBOOT can be upgraded.

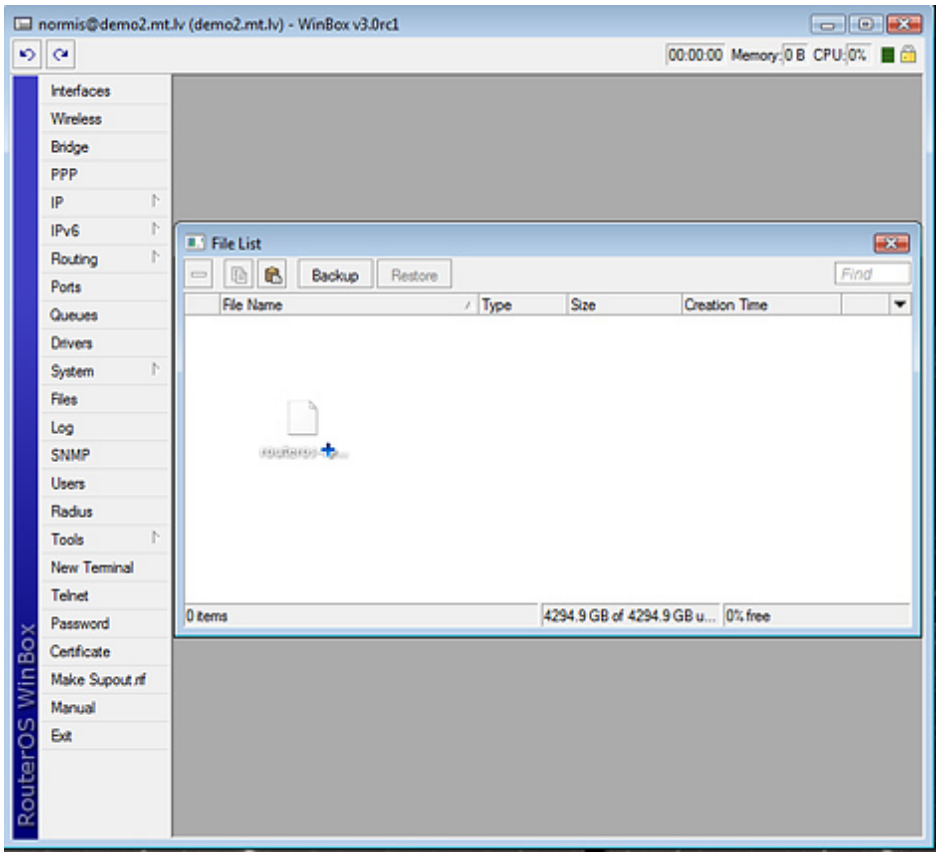
Upgrade process

- First step - visit www.mikrotik.com [1] and head to the download page, there choose the type of system you have the RouterOS installed on.
- Download the **Combined package**, it will include all the functionality of RouterOS:

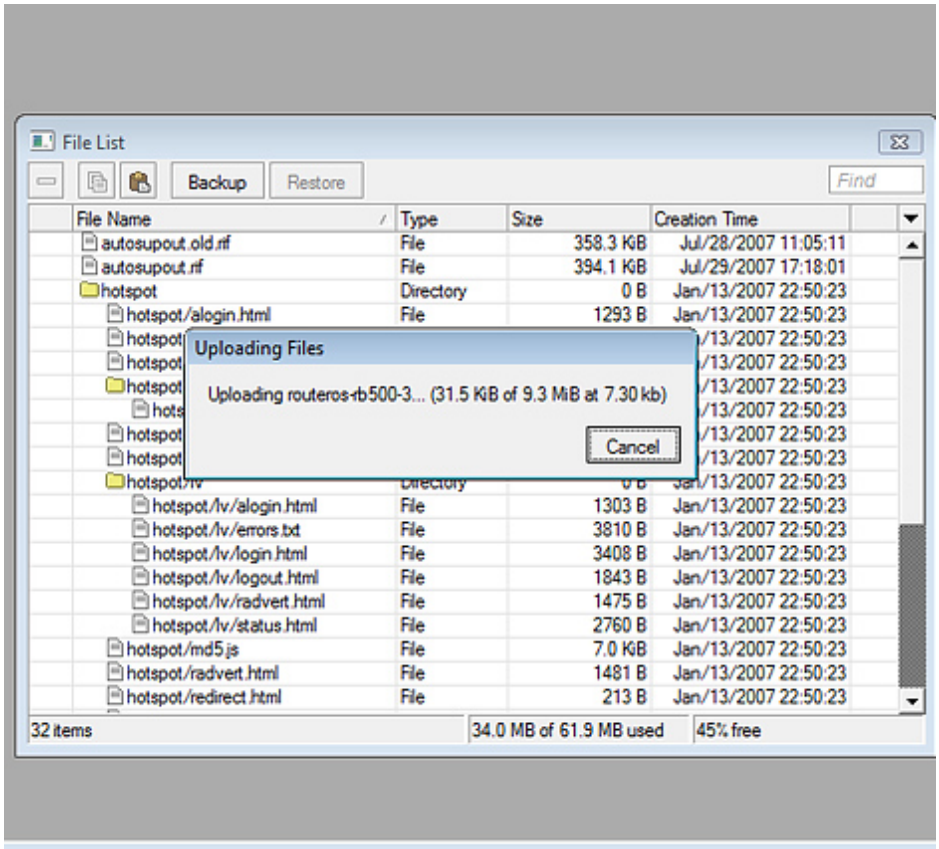
Using Winbox



- Connect to your router with Winbox, Select the downloaded file with your mouse, and drag it to the Files menu. **If there are some files already present, make sure to put the package in the root menu, not inside the hotspot folder!**:



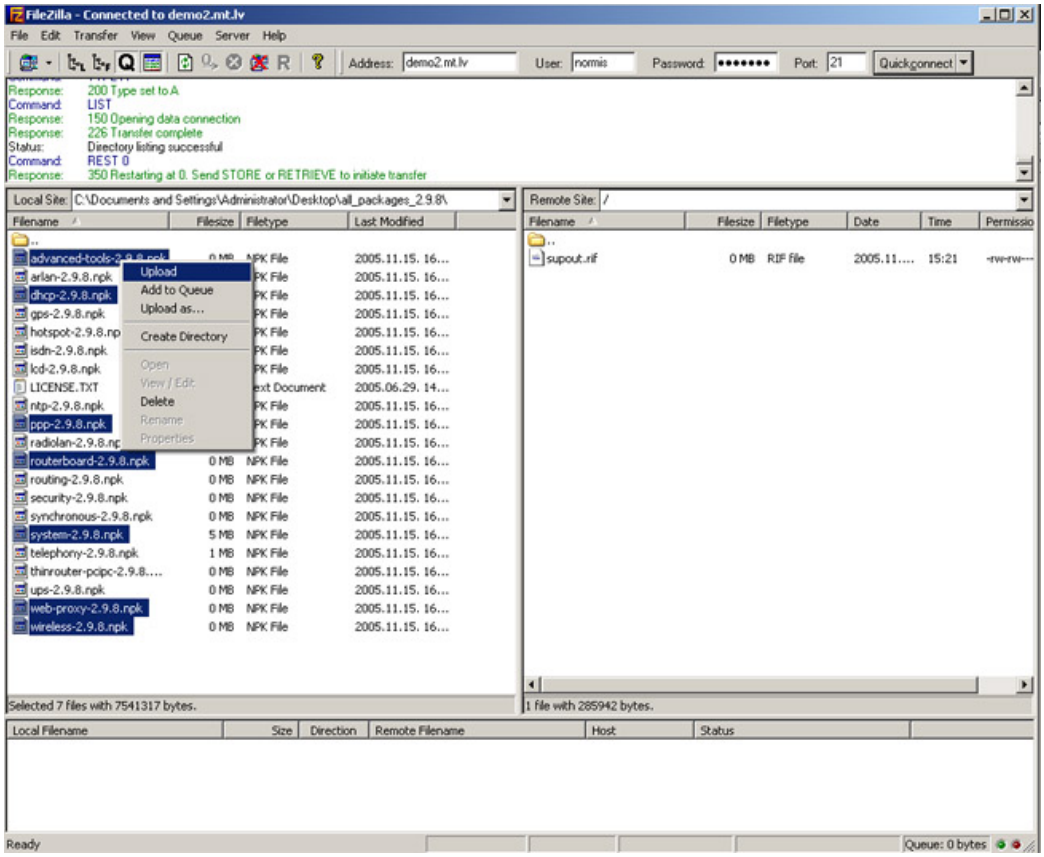
- The upload will start:



After it finishes - REBOOT and that's all! The New version number will be seen in the Winbox Title and in the Packages menu

Using FTP

- Open your favourite FTP program (in this case it is Filezilla [1]), select the package and upload it to your router (demo2.mt.lv is the address of my router in this example). note that in the image I'm uploading many packages, but in your case - you will have one file that contains them all



- if you wish, you can check if the file is successfully transferred onto the router (optional):

```
[normis@Demo_v2.9] > file print
 # NAME                    TYPE                SIZE       CREATION-TIME
 0 supout.rif              .rif file           285942     nov/24/2005 15:21:54
 1 dhcp-2.9.8.npk          package             138846     nov/29/2005 09:55:42
 2 ppp-2.9.8.npk           package             328636     nov/29/2005 09:55:43
 3 advanced-tools-2.9....  package             142820     nov/29/2005 09:55:42
 4 web-proxy-2.9.8.npk     package             377837     nov/29/2005 09:55:43
 5 wireless-2.9.8.npk      package             534052     nov/29/2005 09:55:43
 6 routerboard-2.9.8.npk   package             192628     nov/29/2005 09:55:45
 7 system-2.9.8.npk        package             5826498    nov/29/2005 09:55:54
```

- and reboot your router for the upgrade process to begin:

```
[normis@Demo_v2.9] > system reboot
Reboot, yes? [y/N]: y
```

- after the reboot, your router will be up to date, you can check it in this menu:

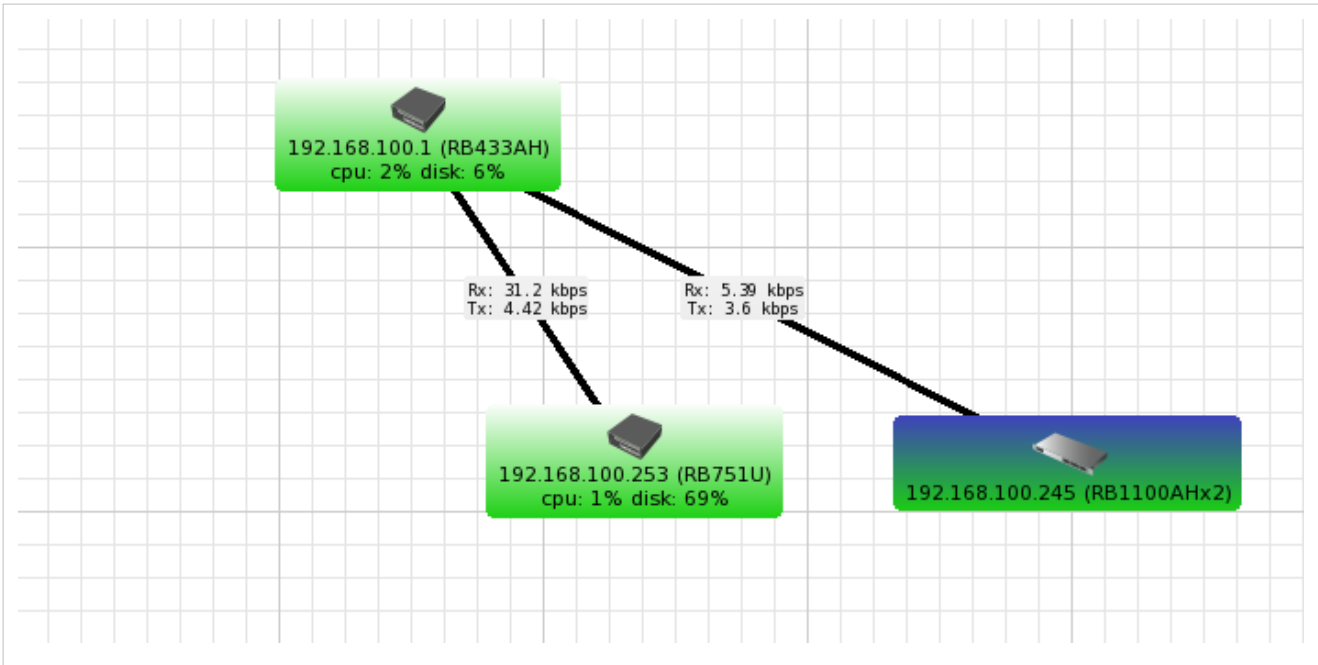
```
/system package print
```

- if your router did not upgrade correctly, make sure you check the **log**

```
/log print without-paging
```

RouterOS massive auto-upgrade

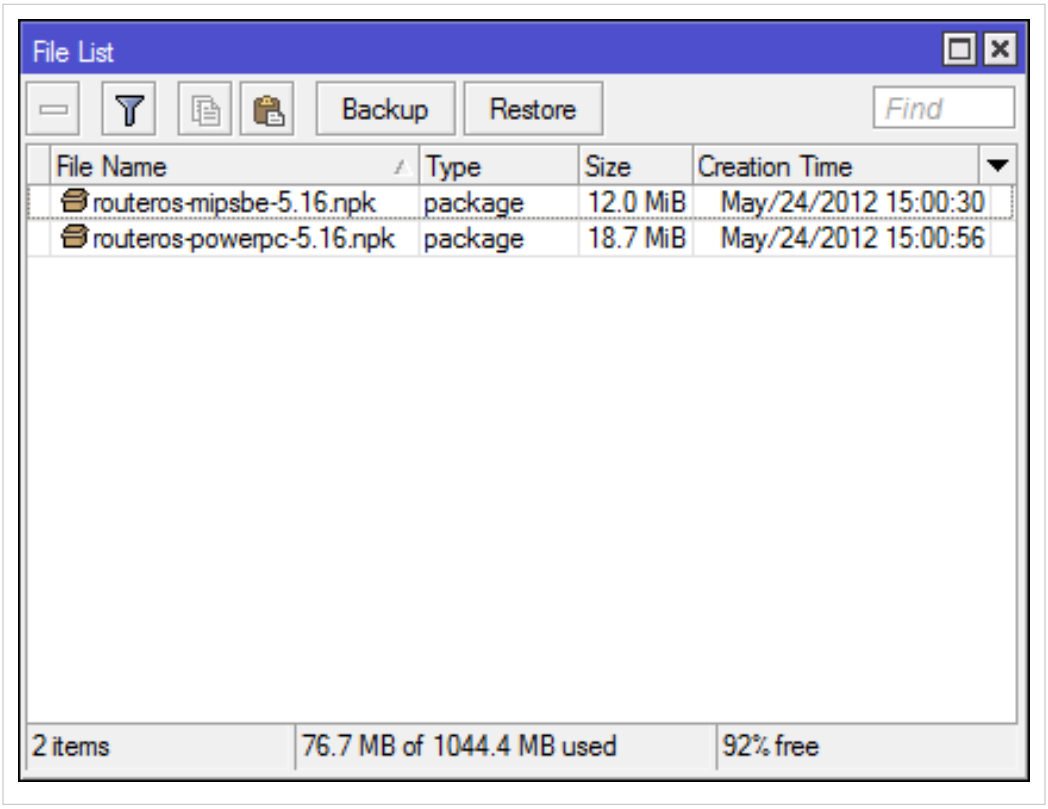
You can upgrade multiple MikroTik routers within few clicks. Let's have a look on simple network with 3 routers (the same method works on networks with infinite numbers of routers),



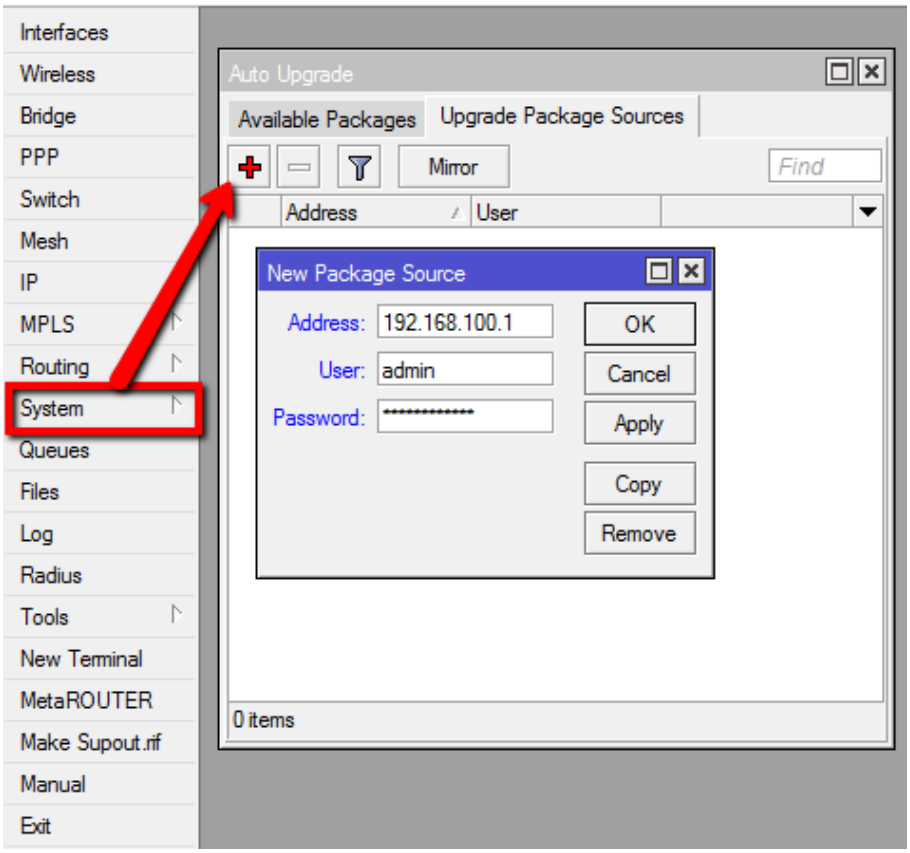
RouterOS auto-upgrade

RouterOS can download software packages from a remote MikroTik router.

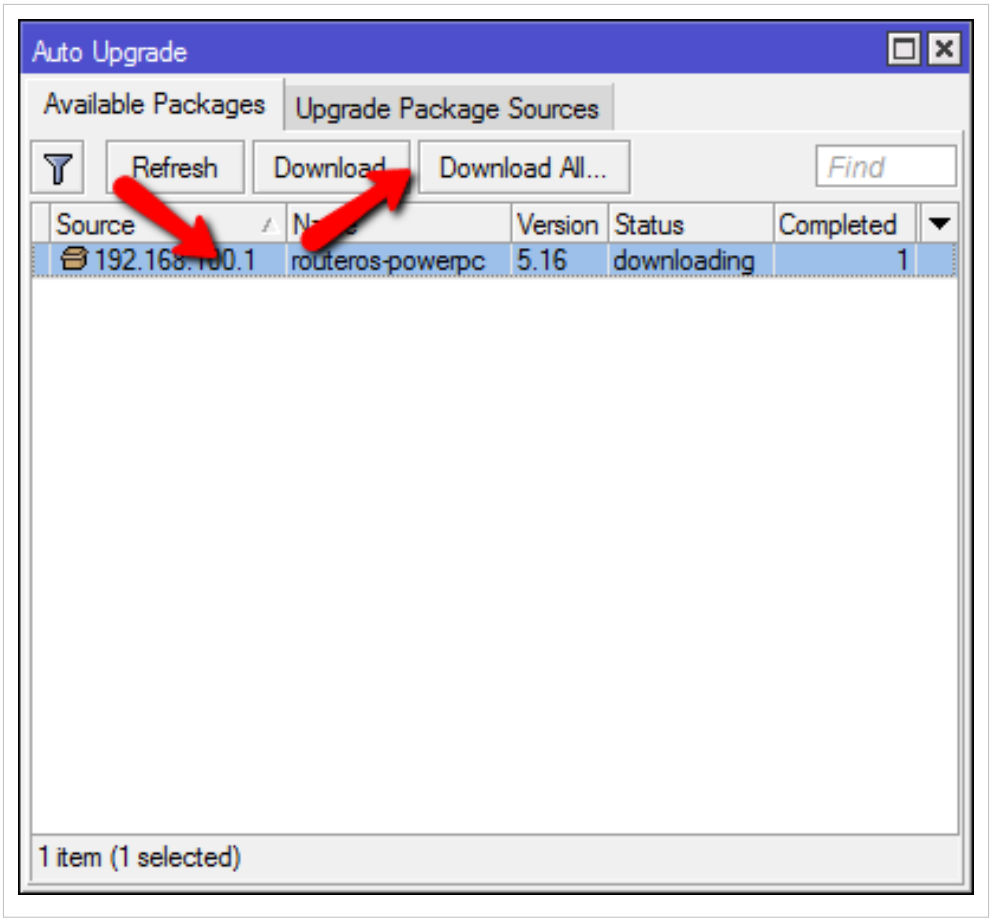
- Make one router as network upgrade central point, that will update MikroTik RouterOS on other routers.
- Upload necessary RouterOS packages to this router (in the example, mipsbe for RB751U and powerpc for RB1100AHx2).



- Add upgrade router (192.168.100.1) information to a router that you want to update (192.168.100.253), required settings IP address/Username/Password



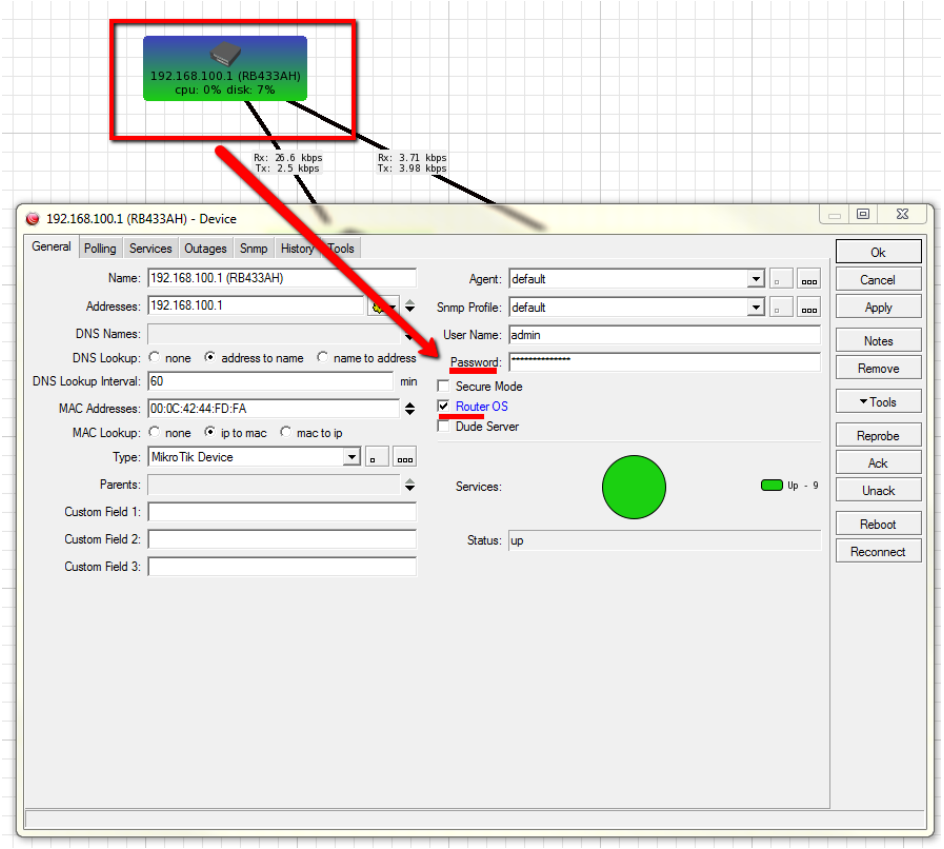
- Click on **Refresh** to see available packages, download newest packages and reboot the router to finalize the upgrade.


Auto Upgrade
Available Packages
Upgrade Package Sources
Refresh
Download
Download All...
Find
Source
Version
Status
Completed
routeros-powerpc
5.16
downloading
1
1 item (1 selected)

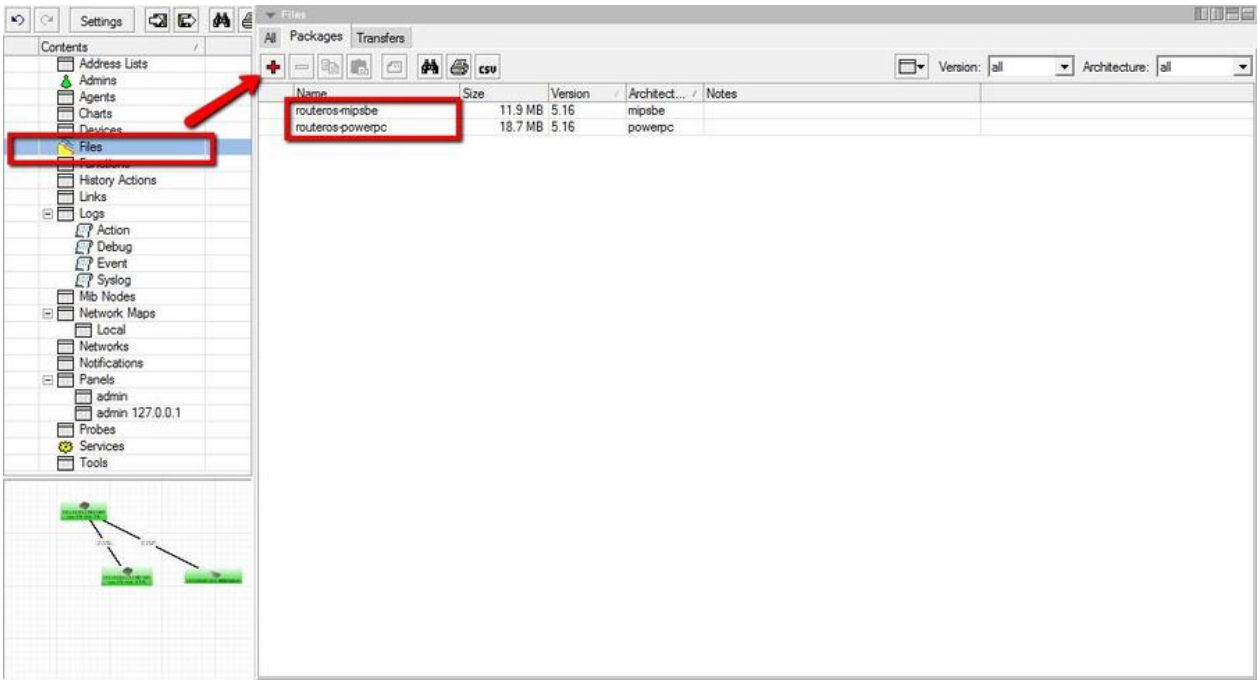
The Dude auto-upgrade

Dude application can help you to upgrade entire RouterOS network with one click per router.

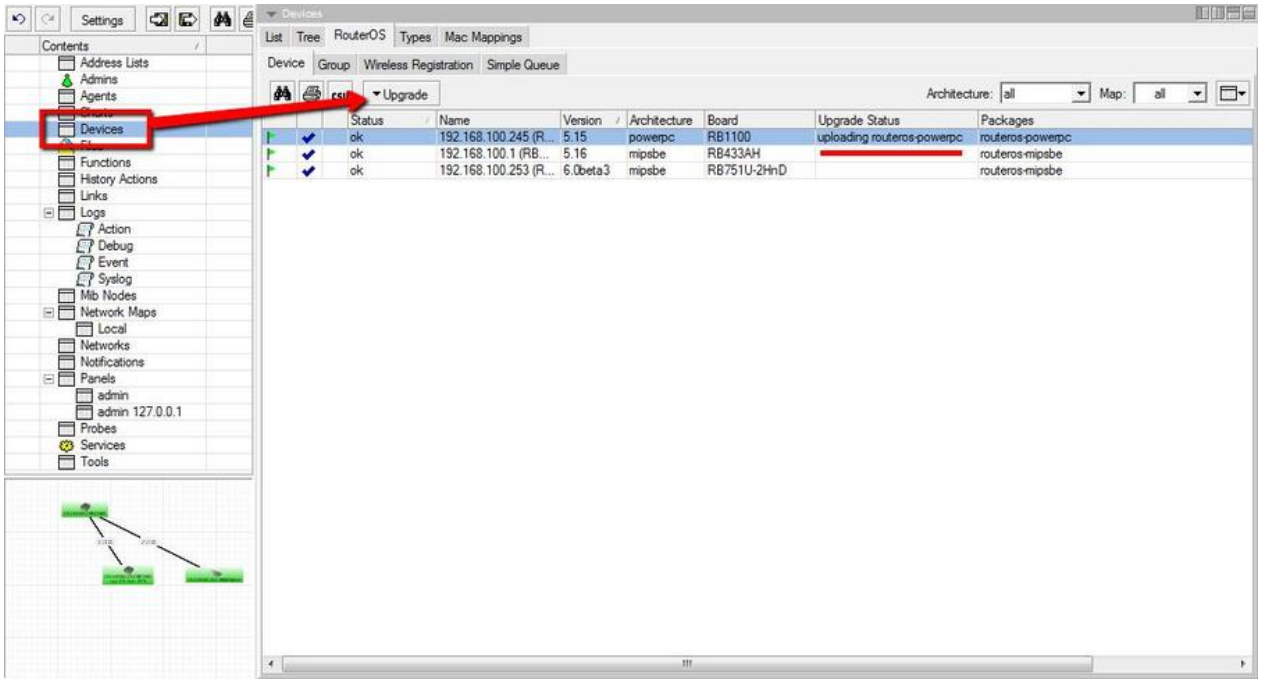
- Set type **RouterOS** and correct password for any device on your Dude map, that you want to upgrade automatically,



- Upload required RouterOS packages to Dude files,

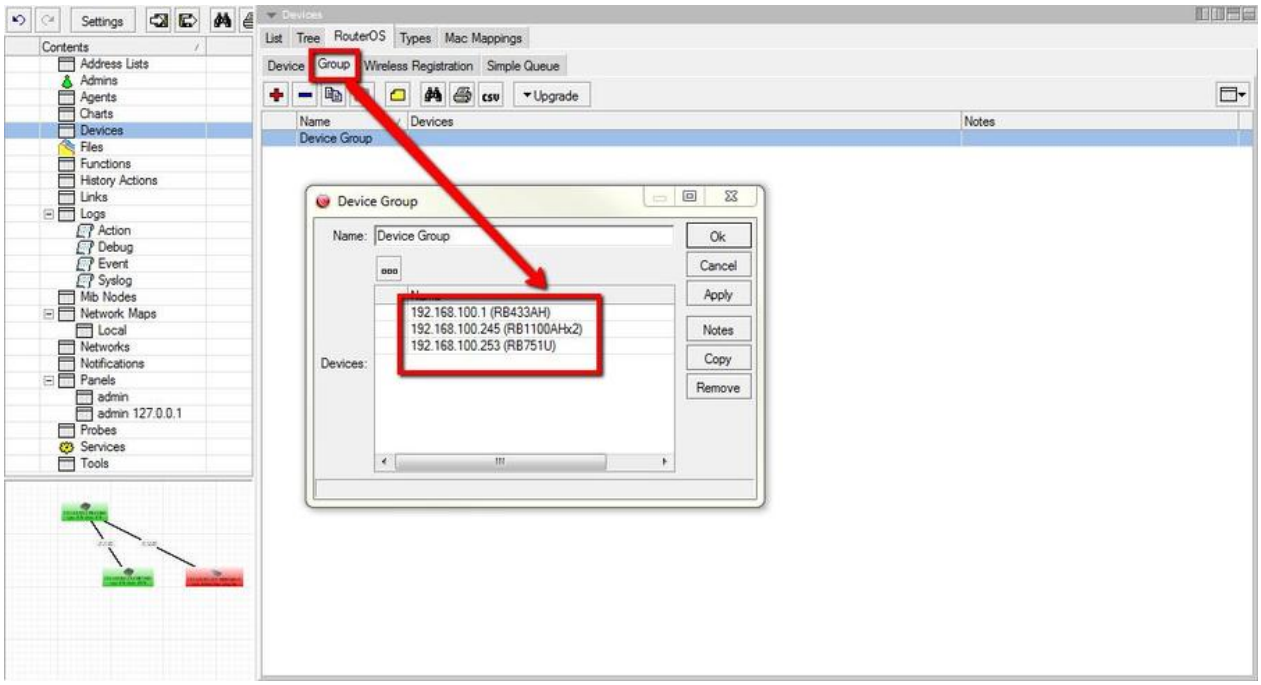


- Upgrade RouterOS version on devices from RouterOS list. Upgrade process is automatic, after click on upgrade (or force upgrade), package will be uploaded and router will be rebooted by the Dude automatically.

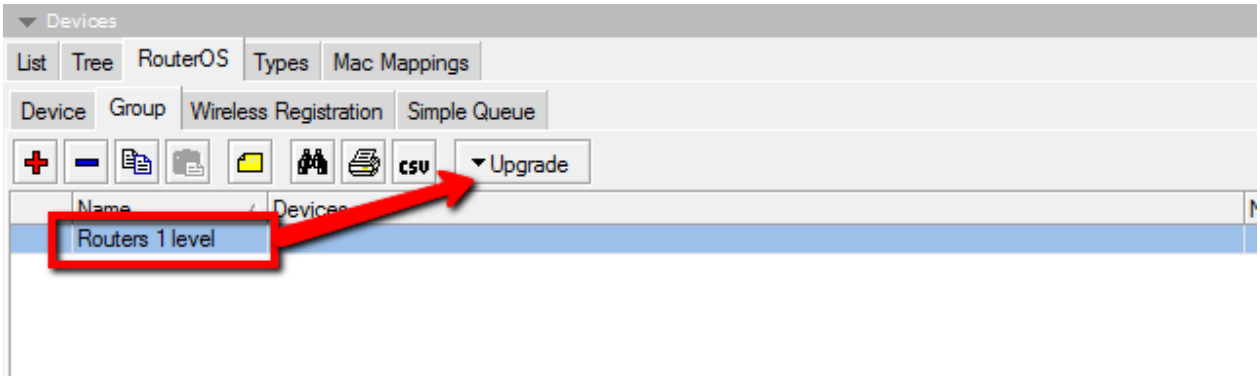


The Dude hierarchical upgrade

For complicated networks, when routers are connected sequentially, the simplest example is 1router-2router-3router connection. You might get an issue, 2router will go to reboot before packages are uploaded to the 3router. The solution is Dude groups, the feature allows to group routers and upgrade all of them by one click!



- Select group and click Upgrade (or Force Upgrade),



License issues

When upgrading from older versions, there could be issues with your license key. Possible scenarios:

- When upgrading from RouterOS v2.8 or older, the system might complain about expired upgrade time. To override this, use Netinstall to upgrade. Netinstall will ignore old license restriction and will upgrade
- When upgrading to RouterOS v4 or newer, the system will ask you to update license to a new format. To do this, ensure your Winbox PC (not the router) has a working internet connection without any restrictions to reach www.mikrotik.com and click "update license" in the license menu.

References

[1] http://filezilla.sourceforge.net/

Manual:Cisco VPLS

Applies to RouterOS: v3, v4

Overview

Since version 3.20 RouterOS implements features that provide compatibility with Cisco VPLS features:

- Cisco style static VPLS pseudowires (RFC 4447 FEC type 0x80)* Cisco VPLS BGP-based auto-discovery (draft-ietf-l2vpn-signaling-08)

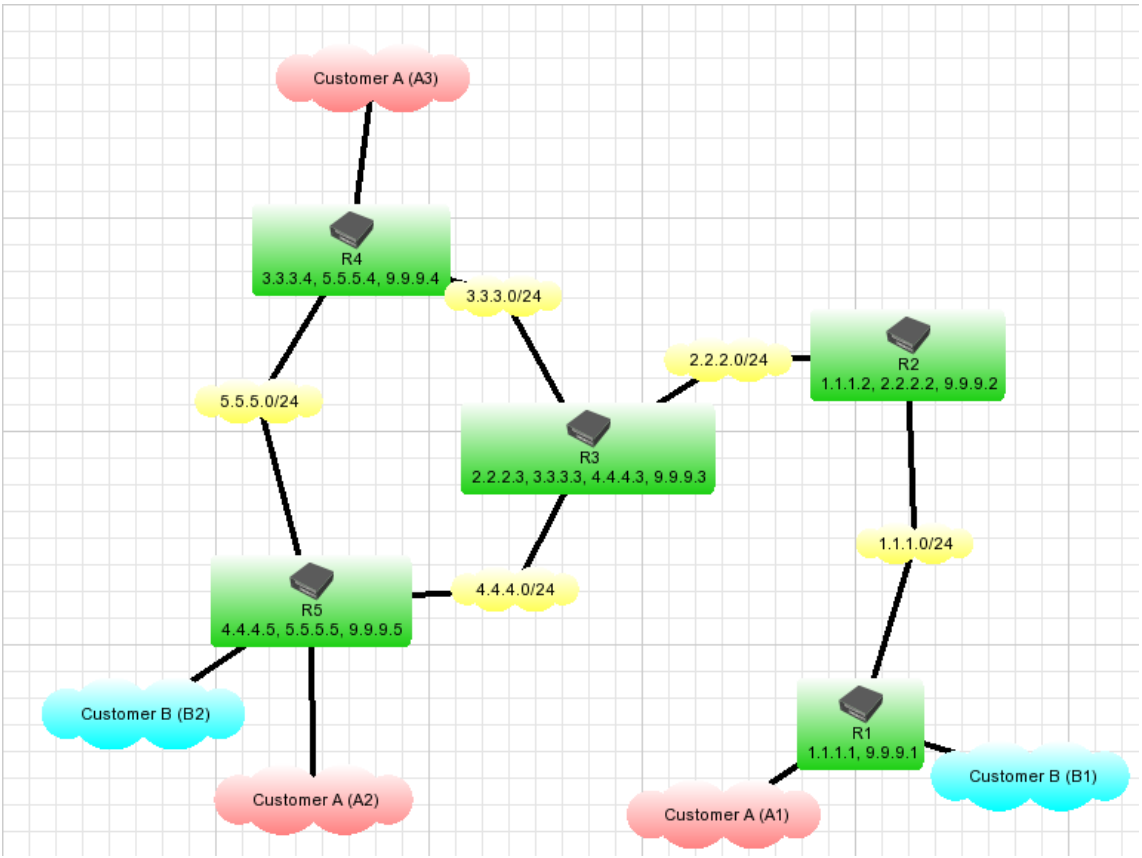
When signaling static VPLS tunnels (pseudowires) using LDP, Cisco does not use pseudowire endpoint identification as specified in RFC 4762 (FEC type 0x81), but uses other method - from RFC 4447 (FEC type 0x80). Such pseudowires can be configured in RouterOS by means of **cisco-style** and **cisco-style-id** settings.

Cisco does not implement BGP-based auto-discovery and signaling according to RFC 4671. Instead, Cisco implements BGP based auto-discovery (draft-ietf-l2vpn-signaling-08). This method specifies use of BGP only to auto-discover other peers that participate in VPLS. VPLS pseudowire signaling is done by LDP.

This document focuses on RouterOS configuration that is related to Cisco compatibility features, for general information on VPLS see MPLSVPLS, for information on RFC 4671 compatible BGP based VPLS, see BGP based VPLS.

Example network

The example network used throughout this document is the same as in MPLSVPLS.



The requirements of customers A and B are the same - ethernet segments must be transparently connected. Taking into account simplicity of given network topology Service Provider has decided to use R5 as route reflector and to have no backup route reflector. Consider that MPLS switching is configured and running, as discussed in MPLSVPLS, but no any VPLS configuration has been applied yet. the rest of this document deals with specifics that are introduced by using Cisco compatible VPLS features.

Customer's B networks are to be connected using static VPLS pseudowire, Customer's A networks are to be connected using VPLS BGP-based autodiscovery.

Configuring Cisco style static VPLS interface

Cisco compatible static VPLS interface is created by specifying appropriate settings: **cisco-style=yes** specifies that VPLS interface should use Cisco-like endpoint identification, parameter **cisco-style-id** specifies pseudowire ID to use. On R1:

```
[admin@R1] /interface vpls> add disabled=no cisco-style=yes cisco-style-id=666 remote-peer=9.9.9.5
```

and on R5:

```
[admin@R5] /interface vpls> add disabled=no cisco-style=yes cisco-style-id=666 remote-peer=9.9.9.1
```

This should result in establishment of targeted LDP session between R1 and R5 and VPLS interface becoming active:

```
[admin@R1] > mpls ldp neighbor print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                      SEND-TARGETED ADDRESSES
```

```
 0 DO   9.9.9.2         9.9.9.1         9.9.9.2:0                 no           1.1.1.2
                                                                               2.2.2.2
                                                                               9.9.9.2
 1 DOTV 9.9.9.5         9.9.9.1         9.9.9.5:0                 yes          4.4.4.5
                                                                               5.5.5.5
                                                                               9.9.9.5
[admin@R1] > /interface vpls print
Flags: X - disabled, R - running, D - dynamic, B - bgp-signaled, C - cisco-bgp-signaled
 0 R   name="vpls1" mtu=1500 mac-address=02:94:02:DB:60:6E arp=enabled disable-running-check=no
remote-peer=9.9.9.5 cisco-style=yes cisco-style-id=666
[admin@R1] > /interface vpls monitor vpls1 once
       remote-label: 29
        local-label: 31
      remote-status:
          transport: 9.9.9.5/32
  transport-nexthop: 1.1.1.2
     imposed-labels: 24,29
```

The rest of configuration to enable transparent bridging of Customer B networks (configuring bridging) is the same as described in MPLSVPLS

Configuring BGP for Cisco compatible VPLS

Configuring Cisco compatible BGP VPLS instance makes router advertise VPLS NLRI according to draft-ietf-l2vpn-signaling-08. Note that this NLRI uses the same BGP AFI/SAFI values as RFC 4762 compatible BGP VPLS. Therefore only one of those specifications (configured as BGP peer address-families) can be supported at a time.

In order to avoid configuring full mesh of BGP sessions between routers acting as attachment points for customer A networks (R1,R4,R5), R5 will be used as route reflector - the same as in BGP based VPLS.

On R5, make sure that client-to-client-reflection is enabled:

```
[admin@R5] /routing bgp instance> pr
Flags: X - disabled
 0   name="default" as=65530 router-id=0.0.0.0 redistribute-connected=no redistribute-static=no redistribute-rip=no
     redistribute-ospf=no redistribute-other-bgp=no out-filter="" client-to-client-reflection=yes ignore-as-path-len=no
```

Create BGP peers with support for **l2vpn-cisco** in peers **address-families**, on R5 configure route reflection:

```
[admin@R5] /routing bgp peer> add remote-as=65530 update-source=lobridge instance=default remote-address=9.9.9.1
address-families=l2vpn-cisco route-reflect=yes
[admin@R5] /routing bgp peer> add remote-as=65530 update-source=lobridge instance=default remote-address=9.9.9.4
address-families=l2vpn-cisco route-reflect=yes
```

```
[admin@R1] /routing bgp peer> add remote-as=65530 update-source=lobridge instance=default remote-address=9.9.9.5
address-families=l2vpn-cisco
```

```
[admin@R4] /routing bgp peer> add remote-as=65530 update-source=lobridge instance=default remote-address=9.9.9.5
address-families=l2vpn-cisco
```

Configuring Cisco compatible BGP VPLS instance

In order for full mesh of VPLS pseudowires to get established, appropriate Cisco compatible VPLS instance must be created. Creating such instance makes router to inject VPLS BGP NLRI in BGP network:

```
[admin@R1] /interface vpls cisco-bgp-vpls> add bridge=A bridge-horizon=1 export-route-targets=1:1 import-route-targets=1:1
l2router-id=9.9.9.1 route-distinguisher=1:1 vpls-id=1:1
```

```
[admin@R4] /interface vpls cisco-bgp-vpls> add bridge=A bridge-horizon=1 export-route-targets=1:1 import-route-targets=1:1
l2router-id=9.9.9.4 route-distinguisher=1:1 vpls-id=1:1
```

```
[admin@R5] /interface vpls cisco-bgp-vpls> add bridge=A bridge-horizon=1 export-route-targets=1:1 import-route-targets=1:1
l2router-id=9.9.9.5 route-distinguisher=1:1 vpls-id=1:1
```

This causes full mesh of targeted LDP sessions to get established and appropriate VPLS interfaces created, e.g. on R4:

```
[admin@R4] > /mpls ldp neighbor print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 ...
 1 DOTV 9.9.9.5         9.9.9.4         9.9.9.5:0                 no            4.4.4.5
                                                                                5.5.5.5
                                                                                9.9.9.5
 2 DOTV 9.9.9.1         9.9.9.4         9.9.9.1:0                 yes           1.1.1.1
                                                                                9.9.9.1
[admin@R4] > /interface vpls print
Flags: X - disabled, R - running, D - dynamic, B - bgp-signaled, C - cisco-bgp-signaled
 0 RDC name="vpls1" mtu=1500 mac-address=02:62:65:24:4C:FD arp=enabled disable-running-check=no remote-peer=9.9.9.1
       vpls-id=1:1 cisco-style=no cisco-style-id=0 vpls=cisco-bgp-vpls1

 1 RDC name="vpls2" mtu=1500 mac-address=02:58:9F:80:EB:94 arp=enabled disable-running-check=no remote-peer=9.9.9.5
       vpls-id=1:1 cisco-style=no cisco-style-id=0 vpls=cisco-bgp-vpls1
```

Manual:Interface/Virtual-ethernet

Applies to RouterOS: v4.x

Summary

To connect your virtual routers to RouterOS host system you either have to assign interface for your guest (possible only on MetaROUTER) or you can add virtual Ethernet interface that is described in this document.

May contain either static or dynamic interface. Static interfaces should be configured in virtual-ethernet menu and then assigned to virtual machine in **/kvm interface** (for KVM) or **/metarouter interface** (for MetaROUTER). Dynamic interfaces will be recreated automatically on each reboot and will contain new MAC address.

Note: Virtual-ethernets will be automatically removed even if configured as static in /kvm interface menu

Requirements

This menu becomes available:

- on *x86* architeecture you have to have *kvm* packge installed
- on *mipsbe* architecture RouterBOARDS
- on *ppc* architecture RouterBOARDS, except RB333, RB600 and variants.

Virtual Ethernet creation

Menu: */interface virtual-ethernet add*

Property	Desciption
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only, default: enabled*)	ARP protocol resolution mode: • *disabled* - interface is not replying to ARP requests • *enabled* - interface is replaying to all ARP requests on its MAC address • *proxy-arp* - interface is replying to all ARP requests even if it is not interface MAC address • *reply-only - interface replies only to known (static entries in ARP table) sources*
comment (*text*)	Short description of the item
copy-from (*number*)	Item number to copy settings from to create new item
disabled (*yes* | *no default: yes*)	identifies if entry is part of active coniguration
mac-address (*MAC address default: automatically generated*)	MAC address of interface. If automatically generated, then this pattern will be used *02:XX:XX:XX:XX:XX*
mtu (*0 65536 default: 1500*)	maximal transmission unit of the interface
name (*text default: tapX or vifX*)	Interface name where, if auto-generated, X is inreased if previous valid number already exists, starts with 1. tap is on x86 vif is on RouterBOARD platform.

See Also

- KVM
- MetaROUTER

Manual:Interface/VLAN

Applies to RouterOS: v3, v4+

Summary

Sub-menu: `/interface vlan`
Standards: `IEEE 802.1Q` [1]

Virtual Local Area Network (VLAN) is layer 2 method that allows you to have multiple Virtual LANs on a single physical interface (ethernet, wireless, etc.), giving the ability to segregate LANs efficiently.

You can use MikroTik RouterOS (as well as Cisco IOS, Linux and other router systems) to mark these packets as well as to accept and route marked ones.

As VLAN works on OSI Layer 2, it can be used just as any other network interface without any restrictions. VLAN successfully passes through regular Ethernet bridges.

You can also transport VLANs over wireless links and put multiple VLAN interfaces on a single wireless interface. Note that as VLAN is not a full tunnel protocol (i.e., it does not have additional fields to transport MAC addresses of sender and recipient), the same limitation applies to bridging over VLAN as to bridging plain wireless interfaces. In other words, while wireless clients may participate in VLANs put on wireless interfaces, it is not possible to have VLAN put on a wireless interface in station mode bridged with any other interface.

802.1Q

The most commonly used protocol for Virtual LANs (VLANs) is IEEE 802.1Q. It is standardized encapsulation protocol that defines how to insert a four-byte VLAN identifier into Ethernet header. (see Figure 12.1.)

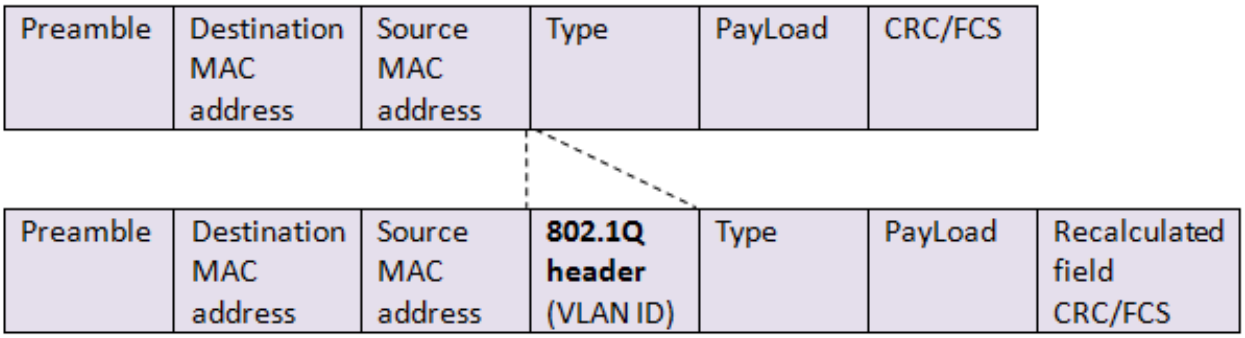


Figure 12.1. Insertion of 802.1Q Tag (VLAN ID) in Ethernet-II frame

Each VLAN is treated as separate subnet. It means that, by default, host in specific VLAN cannot communicate with host that is member of another VLAN, although they are connected in the same switch. So if you want inter-VLAN communication you need a router. RouterOS supports up to 4095 VLAN interfaces, each with a unique VLAN ID,

per interface. VLAN priorites may also be used and manipulated.

When the VLAN extends over more than one switch, the inter-switch link have to become *trunk*, where packets are tagged to indicate which VLAN they belong to. A trunk carries the traffic of multiple VLANs, it is like a point-to-point link that carries tagged packets between switches or between a switch and router.

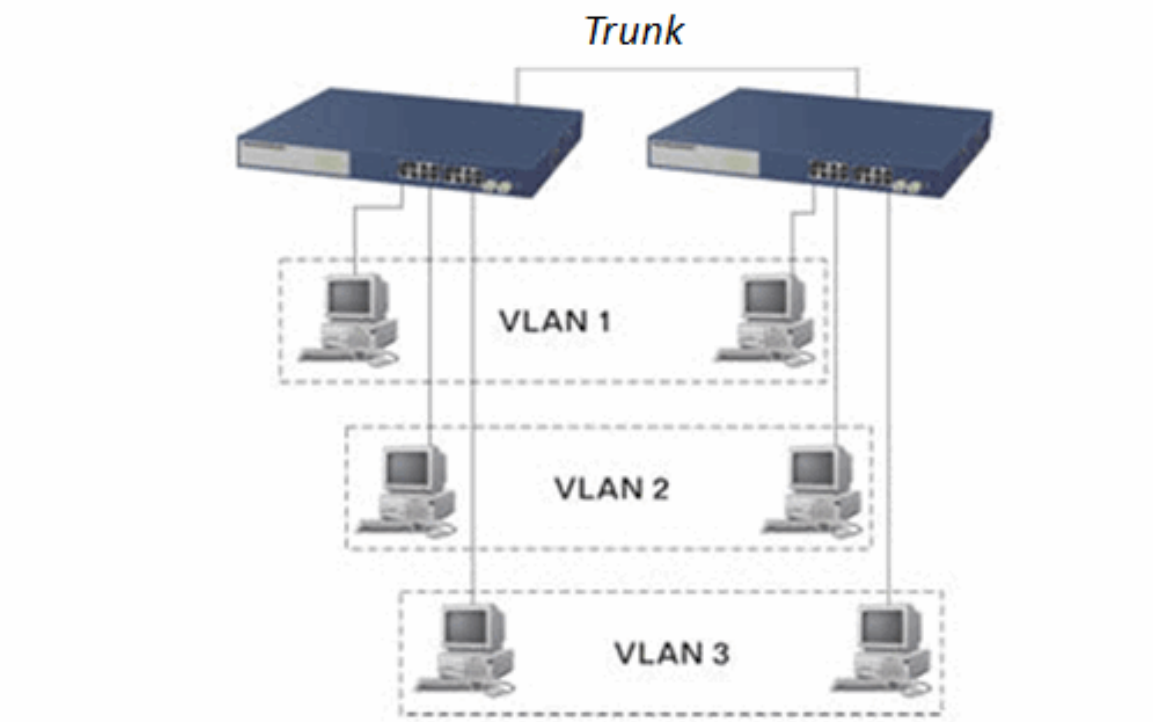


Figure 12.3. Inter-Switch link example using 802.1Q protocol

Q-in-Q

Original 802.1Q allows only one vlan header, Q-in-Q in the other hand allows two or more vlan headers. In RouterOS Q-in-Q can be configured by adding one vlan interface over another. Example:

```
/interface vlan
add name=vlan1 vlan-id=11 interface=ether1
add name=vlan2 vlan-id=12 interface=vlan1
```

If any packet is sent over "vlan2" interface, two vlan tags will be added to ethernet header - "11" and "12".

Properties

Property	Description
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; Default: **enabled**)	Address Resolution Protocol mode
interface (*name*; Default:)	Name of physical interface on top of which VLAN will work
l2mtu (*integer*; Default:)	Layer2 MTU. For VLANS this value is not configurable. Read more>>
mtu (*integer*; Default: **1500**)	Layer3 Maximum transmission unit
name (*string*; Default:)	Interface name
use-service-tag (*yes* | *no*; Default:)	802.1ad compatible Service Tag
vlan-id (*integer: 4095*; Default: **1**)	Virtual LAN identifier or tag that is used to distinguish VLANs. Must be equal for all computers that belong to the same VLAN.

Note: MTU should be set to 1500 bytes as on Ethernet interfaces. But this may not work with some Ethernet cards that do not support receiving/transmitting of full size Ethernet packets with VLAN header added (1500 bytes data + 4 bytes VLAN header + 14 bytes Ethernet header). In this situation MTU 1496 can be used, but note that this will cause packet fragmentation if larger packets have to be sent over interface. At the same time remember that MTU 1496 may cause problems if path MTU discovery is not working properly between source and destination.

Setup examples

Simple Example

Lets assume that we have several MikroTik routers connected to a hub. Remember that hub is OSI physical layer device (if there is a hub between routers, then from L3 point of view it is the same as Ethernet cable connection between them). For simplification assume that all routers are connected to the hub using ether1 interface and has assigned IP addresses as illustrated in figure below. Then on each of them the VLAN interface should be created.

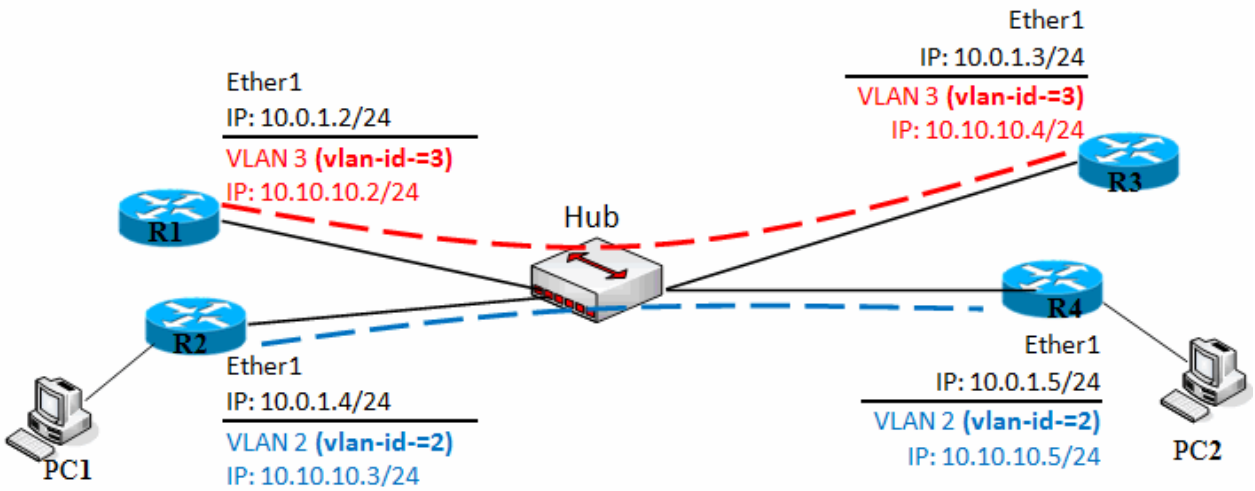


Figure 12.4. Network example with VLANs

Configuration for R2 and R4 is shown below:

R2:

```
[admin@MikroTik] /interface vlan> add name=VLAN2 vlan-id=2 interface=ether1 disabled=no

[admin@MikroTik] /interface vlan> print
Flags: X - disabled, R - running, S - slave
 #    NAME                MTU   ARP        VLAN-ID INTERFACE
0 R  VLAN2               1500  enabled    2       ether1
```

R4:

```
[admin@MikroTik] /interface vlan> add name=VLAN2 vlan-id=2 interface=ether1 disabled=no

[admin@MikroTik] /interface vlan> print
Flags: X - disabled, R - running, S - slave
 #    NAME                MTU   ARP        VLAN-ID INTERFACE
0 R  VLAN2               1500  enabled    2       ether1
```

The next step is to assign IP addresses to the VLAN interfaces.

R2:

```
 [admin@MikroTik] ip address> add address=10.10.10.3/24 interface=VLAN2
 [admin@MikroTik] ip address> print
 Flags: X - disabled, I - invalid, D - dynamic
   #   ADDRESS            NETWORK         BROADCAST       INTERFACE
   0   10.0.1.4/24        10.0.1.0        10.0.1.255      ether1
   1   10.20.0.1/24       10.20.0.0       10.20.0.255     pc1
   2   10.10.10.3/24      10.10.10.0      10.10.10.255    vlan2

 [admin@MikroTik] ip address>
```

R4:

```
 [admin@MikroTik] ip address> add address=10.10.10.5/24 interface=VLAN2
 [admin@MikroTik] ip address> print
 Flags: X - disabled, I - invalid, D - dynamic
   #   ADDRESS            NETWORK         BROADCAST       INTERFACE
   0   10.0.1.5/24        10.0.1.0        10.0.1.255      ether1
   1   10.30.0.1/24       10.30.0.0       10.30.0.255     pc2
   2   10.10.10.5/24      10.10.10.0      10.10.10.255    vlan2

[admin@MikroTik] ip address>
```

At this point it should be possible to ping router R4 from router R2 and vice versa:

```
 '''Ping from R2 to R4:'''

 [admin@MikroTik] ip address> /ping 10.10.10.5

 10.10.10.5 64 byte ping: ttl=255 time=4 ms

 10.10.10.5 64 byte ping: ttl=255 time=1 ms
```

```
2 packets transmitted, 2 packets received, 0% packet loss

round-trip min/avg/max = 1/2.5/4 ms


'''From R4 to R2:'''

[admin@MikroTik] ip address> /ping 10.10.10.3
10.10.10.3 64 byte ping: ttl=255 time=6 ms
10.10.10.3 64 byte ping: ttl=255 time=1 ms
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 1/3.5/6 ms
```

To make sure if VLAN setup is working properly, try to ping R1 from R2. If pings are timing out then VLANs are successfully isolated.

```
'''From R2 to R1:'''

[admin@MikroTik] ip address> /ping 10.10.10.2
10.10.10.2 ping timeout
10.10.10.2 ping timeout
3 packets transmitted, 0 packets received, 100% packet loss
```

Create *trunks* and implement routing between VLANs

If separate VLANs are implemented on a switch, then router is required to provide communication between VLANs. Switch works at OSI layer 2 so it uses only Ethernet header to forward and does not check IP header. For this reason we must use the router that is working as a gateway for each VLAN. Without a router host is unable to communicate outside its own VLAN. Routing process between VLANs described above is called inter-VLAN communication.

To illustrate inter-VLAN communication, we will create a trunk that will carry traffic from three VLANs (VLAN2 and VLAN3, VLAN4) across a single link between Mikrotik router and a manageable switch that supports VLAN trunking.

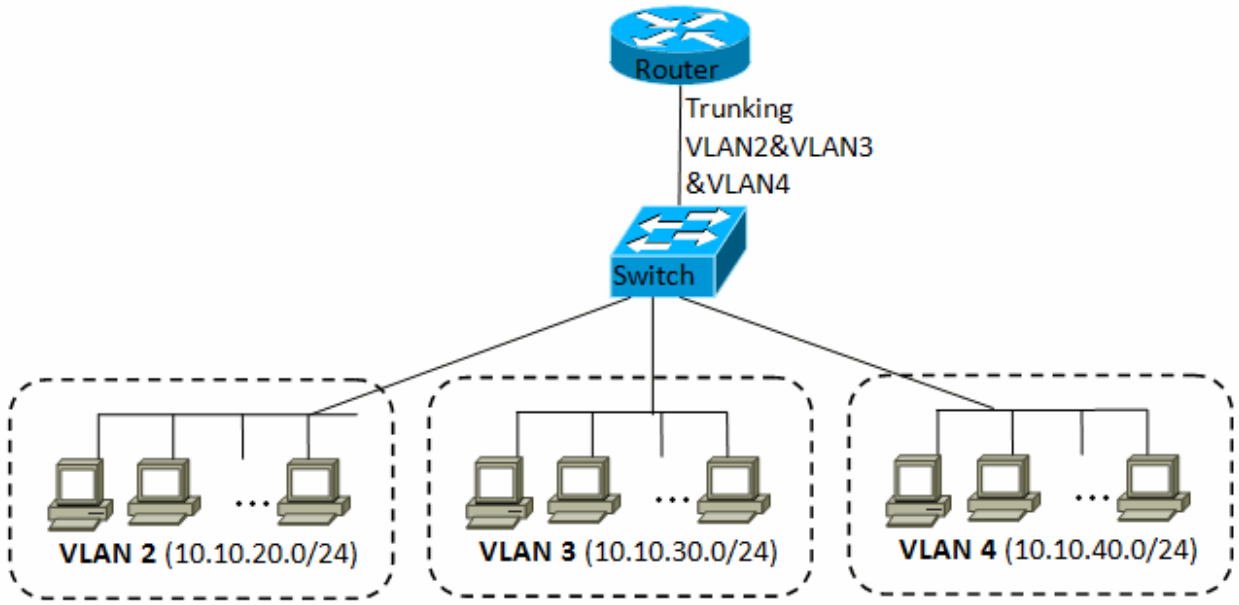


Figure 12.5. 802.1Q trunk between the router and the switch

Each VLAN has its own separate subnet (broadcast domain) as we see in figure above:

- VLAN 2 – 10.10.20.0/24;
- VLAN 3 – 10.10.30.0/24;
- VLAN 4 – 10.10.40.0./24.

VLAN configuration on most of switches is straightforward, basically we need to define which ports are members of VLAN and define "trunk" port that can carry tagged frames between switch and router.

Configuration example on MikroTik router:

Create VLAN interfaces:

```
/interface vlan
add name=VLAN2 vlan-id=2 interface=ether1 disabled=no
add name=VLAN3 vlan-id=3 interface=ether1 disabled=no
add name=VLAN4 vlan-id=4 interface=ether1 disabled=no
```

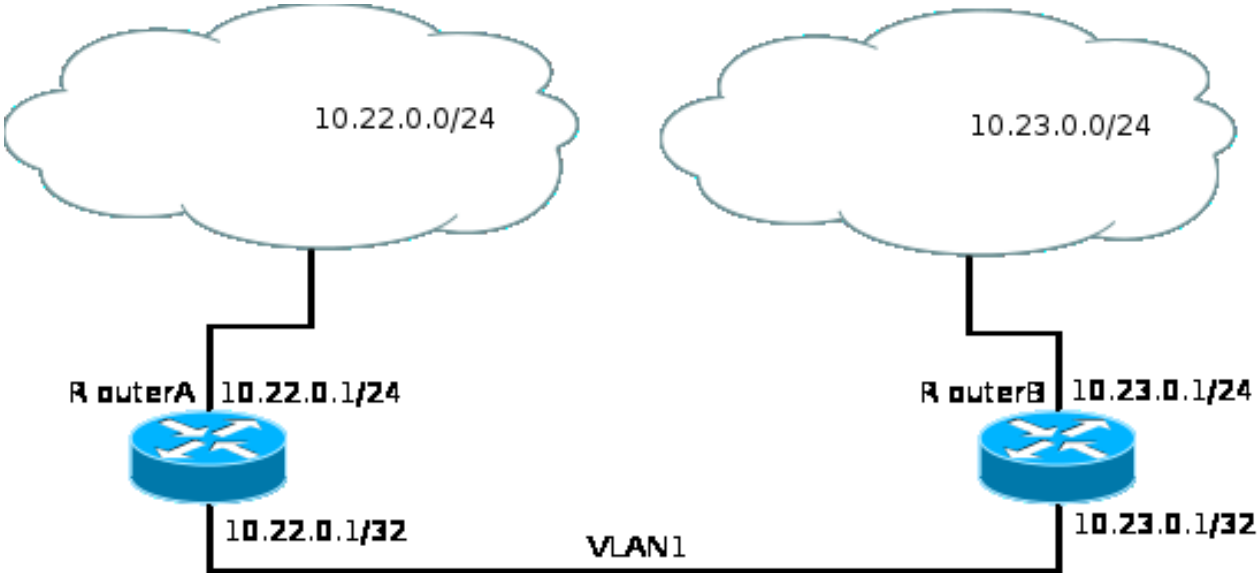
Add IP addresses to VLANs:

```
/ip address
add address=10.10.20.1/24 interface=VLAN2
add address=10.10.30.1/24 interface=VLAN3
add address=10.10.40.1/24 interface=VLAN4
```

RouterOS /32 and IP unnumbered addresses

In RouterOS to create point-to-point tunnel with addresses you have to use address with network mask /32 that effectively brings you same features as some vendors unnumbered IP address.

There are 2 routers RouterA and RouterB that each is part of networks 10.22.0.0/24 and 10.23.0.0/24 respectively, to connect these router using VLAN as carrier with the following configuration:



RouterA:

```
/ip address add address=10.22.0.1/24 interface=ether1
/interface vlan add interface=ether2 vlan-id=1 name=vlan1
/ip address add address=10.22.0.1/32 interface=vlan1 network=10.23.0.1
/ip route add gateway=10.23.0.1 dst-address=10.23.0.0/24
```

RouterB:

```
/ip address add address=10.23.0.1/24 interface=ether1
/interface vlan add interface=ether2 vlan-id=1 name=vlan1
/ip address add address=10.23.0.1/32 interface=vlan1 network=10.22.0.1
/ip route add gateway=10.22.0.1 dst-address=10.22.0.0/24
```

[Top | Back to Content]

References

[1] http://standards.ieee.org/getieee802/download/802.1Q-1998.pdf

Manual:Interface/VPLS

Applies to RouterOS: v3, v4 +

Summary

Virtual Private Lan Service (VPLS) interface can be considered tunnel interface just like EoIP interface. To achieve transparent ethernet segment forwarding between customer sites.

MikroTik RouterOS implements following VPLS features:

- LDP signaling (RFC 4762), see LDP based VPLS
- pseudowire fragmentation and reassembly (RFC 4623)
- MP-BGP based autodiscovery and signaling (RFC 4761), see BGP based VPLS

Since version 3.17:

- Cisco style static VPLS pseudowires (RFC 4447 FEC type 0x80), see static Cisco VPLS
- Cisco VPLS BGP-based auto-discovery (draft-ietf-l2vpn-signaling-08), see BGP based Cisco style VPLS
- support for multiple import/export route target extended communities for BGP based VPLS (both, RFC 4761 and draft-ietf-l2vpn-signaling-08)

General

Sub-menu: `/interface vpls`

List of all VPLS interfaces. This menu shows also dynamically created BGP based VPLS interfaces.

Properties

Property	Description
advertised-l2mtu (*integer*; Default: **1500**)	L2MTU value advertised to remote peer.
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; Default: **enabled**)	Address Resolution Protocol
cisco-style (*yes* | *no*; Default: **no**)	Specifies whether to use cisco style VPLS.
cisco-style-id (*integer*; Default: **0**)	VPLS tunnel ID, used if `cisco-style` is set to **yes**.
comment (*string*; Default:)	Short description of the item
disable-running-check (*yes* | *no*; Default: **no**)	Specifies whether to detect if interface is running or not. If set to **no** interface will always have `running` flag.
disabled (*yes* | *no*; Default: **yes**)	Defines whether item is ignored or used. By default VPLS interface is disabled.
l2mtu (*integer*; Default: **1500**)	
mac-address (*MAC*; Default:)	
mtu (*integer*; Default: **1500**)	
name (*string*; Default:)	Name of the interface
pw-type (*raw-ethernet* | *tagged-ethernet*; Default: **raw-ethernet**)	Pseudowire type.

remote-peer (*IP*; Default:)	IP address of remote peer.
use-control-word (*yes* | *no* | *default*; Default: **default**)	Enables/disables Control Word usage. Default values for regular and cisco style VPLS tunnels differ. Cisco style by default has control word usage disabled. `Read more >>`.
vpls-id (*AsNum* | *AsIp*; Default:)	Unique number that identifies VPLS tunnel.

Monitoring

Command `/interface vpls monitor [id]` will display current VPLS interface status

For example:

```
[admin@10.0.11.23] /interface vpls> monitor vpls2
       remote-label: 800000
        local-label: 43
      remote-status:
          transport: 10.255.11.201/32
  transport-nexthop: 10.0.11.201
     imposed-labels: 800000
```

Available read only properties:

Property	Description
imposed-label (*integer*)	VPLS imposed label
local-label (*integer*)	Local VPLS label
remote-group ()	
remote-label (*integer*)	Remote VPLS label
remote-status (*integer*)	
transport-nexthop (*IP prefix*)	Shows used transport address (typically Loopback address).
transport (*string*)	Name of the transport interface. Set if VPLS is running over Traffic Engineering tunnel.

BGP VPLS

Sub-menu: `/interface vpls bgp-vpls`

List of BGP signaled VPLS instances. Configured instance makes router advertise VPLS BGP NLRI that advertises that particular router belongs to some VPLS.

Properties

Property	Description
bridge (*none* | *string*; Default: **none**)	If set to **none** VPLS interface is not added to bridge ports.
bridge-cost (*integer*; Default: **50**)	
bridge-horizon (*none* | *integer*; Default: **none**)	If set to **none** bridge horizon will not be used.
comment (*string*; Default:)	Short description of the item
disabled (*yes* | *no*; Default: **no**)	Defines whether item is ignored or used.
export-route-target (*AsNum* | *AsIp*; Default:)	Setting is used to tag BGP NLRI with one or more route targets.
import-route-target (*AsNum* | *AsIp*; Default:)	Setting is used to determine if BGP NLRI is related to particular VPLS, by comparing route targets received from BGP NLRI.
name (*string*; Default:)	
pw-mtu (*integer[32..65535]*; Default: **1500**)	Advertised pseudowire MTU value.
pw-type (*raw-ethernet* | *tagged-ethernet* | *vpls*; Default: **vpls**)	Parameter is available starting from v5.16. It allows to choose advertised encapsulation in NLRI used only for comparison. It does not affect functionality of the tunnel. `See pw-type usage example >>`
route-distinguisher (*AsNum* | *AsIp*; Default:)	Specifies value that gets attached to VPLS NLRI so that receiving routers can distinguish advertisements that may otherwise look the same. This implies that unique route-distinguisher for every VPLS must be used. It is not necessary to use the same route distinguisher for some VPLS on all routers forming that VPLS as distinguisher is not used for determining if some BGP NLRI is related to particular VPLS (Route Target attribute is used for this), but it is mandatory to have different distinguishers for different VPLSes.
site-id (*integer*; Default: **1**)	Unique site identifier. Each site must have unique site-id.
use-control-word (*yes* | *no*; Default: **yes**)	Enables/disables Control Word usage. `Read more >>`

Cisco Style BGP VPLS

Sub-menu: `/interface cisco-bgp-vpls`

Properties

Property	Description
bridge (*none* | *string*; Default: **none**)	If set to **none** VPLS interface is not added to bridge ports.
bridge-cost (*integer*; Default: **50**)	
bridge-horizon (*none* | *integer*; Default: **none**)	If set to **none** bridge horizon will not be used.
comment (*string*; Default:)	Short description of the item
disabled (*yes* | *no*; Default: **no**)	Defines whether item is ignored or used.
export-route-target (*AsNum* | *AsIp*; Default:)	Setting is used to tag BGP NLRI with one or more route targets.

import-route-target (*AsNum* | *AsIp*; Default:)	Setting is used to determine if BGP NLRI is related to particular VPLS, by comparing route targets received from BGP NLRI.
l2router-id (*IP*; Default: **0.0.0.0**)	
name (*string*; Default:)	
pw-mtu (*integer[32..65535]*; Default: **1500**)	Advertised pseudo-wire MTU.
route-distinguisher (*AsNum* | *AsIp*; Default:)	Specifies value that gets attached to VPLS NLRI so that receiving routers can distinguish advertisements that may otherwise look the same. This implies that unique route-distinguisher for every VPLS must be used. It is not necessary to use the same route distinguisher for some VPLS on all routers forming that VPLS as distinguisher is not used for determining if some BGP NLRI is related to particular VPLS (Route Target attribute is used for this), but it is mandatory to have different distinguishers for different VPLSes.
use-control-word (*yes* | *no*; Default: **yes**)	Enables/disables Control Word usage. `Read more >>`
vpls-id (*integer*; Default:)	Unique number that identifies VPLS tunnel.

Manual:Interface/VRRP

Applies to RouterOS: v3, v4, v5

Summary

Sub-menu level: `/interface vrrp`
Standards: `RFC 5798, RFC 3768`

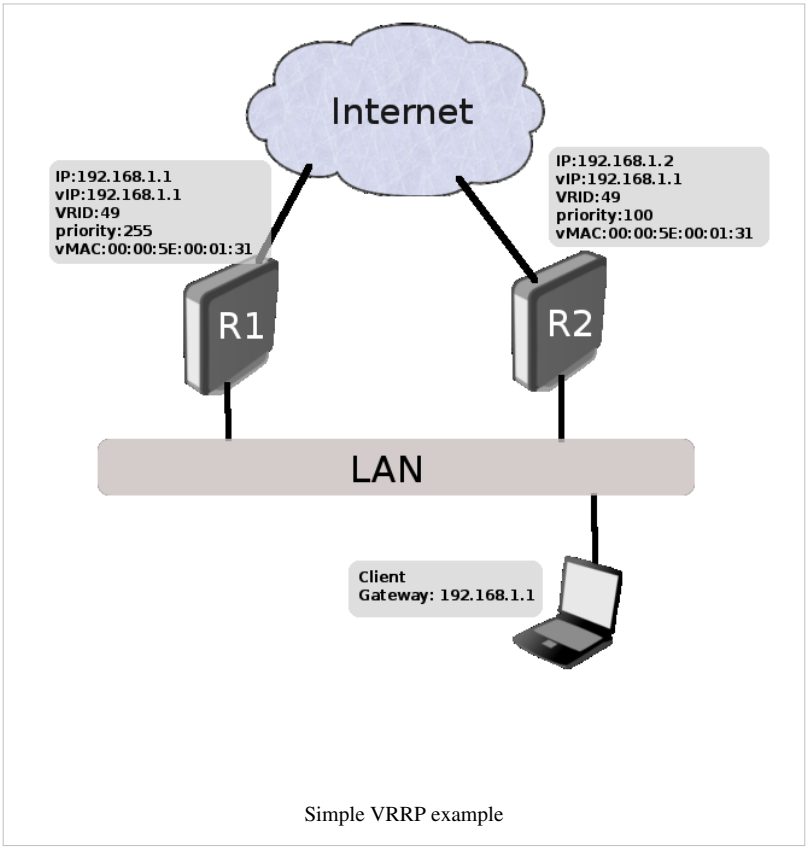
This chapter describes the Virtual Router Redundancy Protocol (VRRP) support in RouterOS.

Mostly on larger LANs dynamic routing protocols (OSPF or RIP) are used, however there are number of factors that may make undesirable to use dynamic routing protocols. One alternative is to use static routing, but if statically configured first hop fails, then host will not be able to communicate with other hosts.

In IPv6 networks, hosts learn about routers by receiving Router Advertisements used by Neighbor Discovery (ND) protocol. ND already has built in mechanism to determine unreachable routers. However it can take up to 38seconds to detect unreachable router. It is possible to change parameters and make detection faster, but it will increase overhead of ND traffic especially if there are a lot of hosts. VRRP allows to detect unreachable router within 3seconds without additional traffic overhead.

Virtual Router Redundancy Protocol (VRRP) provides a solution by combining number of routers into logical group called *Virtual Router* (VR). VRRP implementation in RouterOS is compliant to VRRPv2 RFC 3768 and VRRPv3 RFC 5798.

Protocol Overview



Simple VRRP example

The purpose of the VRRP is to communicate to all VRRP routers associated with the Virtual Router ID and support router redundancy through a prioritized election process among them.

All messaging is done by IPv4 or IPv6 multicast packets. Destination address of IPv4 packet is *224.0.0.12* and for IPv6 it is *FF02:0:0:0:0:0:0:12*. Source address of the packet is always the primary IP address of an interface from which the packet is being sent. In IPv6 networks source address is link-local address of an interface.

These packets are always sent with TTL=255 and are not forwarded by the router. If for any reason router receives a packet with lower TTL, packet is discarded.

Each VR node has a single assigned MAC address. This MAC address is used as a source for all periodic messages sent by Master.

Virtual Router is defined by VRID and mapped set of IPv4 or IPv6 addresses. Master router is said to be the **owner** of mapped IPv4/IPv6 addresses. There are no limits to use the same VRID for IPv4 and IPv6, however these will be two different Virtual Routers.

Only Master router is sending periodic Advertisement messages to minimize the traffic. Backup will try to preempt the Master only if it has the higher priority and preemption is not prohibited.

All VRRP routers belonging to the same VR must be configured with the same advertisement interval. If interval does not match router will discard received advertisement packet.

Virtual Router (VR)

A Virtual Router (VR) consists of one Owner router and one or more backup routers belonging to the same network.

VR includes:

- VRID configured on each VRRP router
- the same virtual IP on each router
- Owner and Backup configured on each router. On a given VR there can be only one Owner.

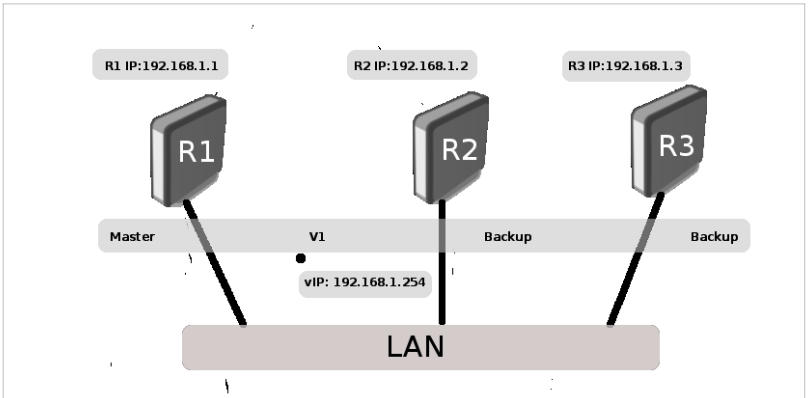
Virtual MAC address

VRRP automatically assigns MAC address to VRRP interface based on standard MAC prefix for VRRP packets and VRID number. First five octets are 00:00:5E:00:01 and last octet is configured VRID. For example, Virtual Routers VRID is 49, then virtual MAC address will be *00:00:5E:00:01:31*.

Note: Virtual mac address can not be manually set or edited.

Owner

An Owner router for a VR is default Master router and operates as the Owner for all subnets included in the VR. As mentioned before priority on an owner router must be the highest value (255). In example network **R1** is an Owner. It's priority is set to 255 and virtual IP is the same as real IP (owns the virtual IP address).



VRRP without Owner

All Virtual Router members can be configured so that virtual IP is not the same as physical IP. Such Virtual address can be called floating or pure virtual IP address.

Advantage of this setup is flexibility given to the administrator. Since the virtual IP address is not the real address of any one of the participant routers, the administrator can change these physical routers or their addresses without any need to reconfigure the virtual router itself.

Note: RouterOS can not be configured as Owner. Pure virtual IP configuration is the only valid configuration unless non-RouterOS device is set as owner.

Master

Master router in a VR operates as the physical gateway for the network for which it is configured. Selection of the Master is controlled by priority value. Master state describes behavior of Master router. In example network **R1** is the Master router. When R1 is no longer available R2 becomes master.

Backup

VR must contain at least one Backup router. Backup router must be configured with the same virtual IP as Master for that VR. Default priority for Backup routers is 100. When current master router is no longer available, backup router with highest priority will become current master. Every time when router with higher priority becomes available it is switched to master. Sometimes this behavior is not necessary. To override it preemption mode should be disabled.

Virtual Address

Virtual IP associated with VR must be identical and set on all VR nodes. On Owner router Virtual IP must be the same as real IP. For example on Owner router real IP and virtual IP is 192.168.1.1, on Backup router virtual IP is 192.168.1.1, but real IP is 192.168.1.2. All virtual and real addresses should be from the same network.

If the Master of VR is associated with multiple IP addresses, then Backup routers belonging to the same VR must also be associated with the same set of virtual IP addresses. If virtual address on the Master is not also on Backup a misconfiguration exists and VRRP advertisement packets will be discarded.

Note: It is not recommended to set up Mikrotik router as an Owner router. VRRP address and real IP address should not be the same.

In IPv6 networks first address is always link-local address associated to VR. If multiple IPv6 addresses are configured, then they are added in advertisement packet after the link-local address.

IPv4 ARP

The Master for a given VR responds to ARP requests with the VR's assigned MAC address. Virtual MAC address is also used as the source MAC address for advertisement packets sent by the Master. To ARP requests for non-virtual IP addresses router responds with the system MAC address. Backup routers are not responding to ARP requests for Virtual IPs.

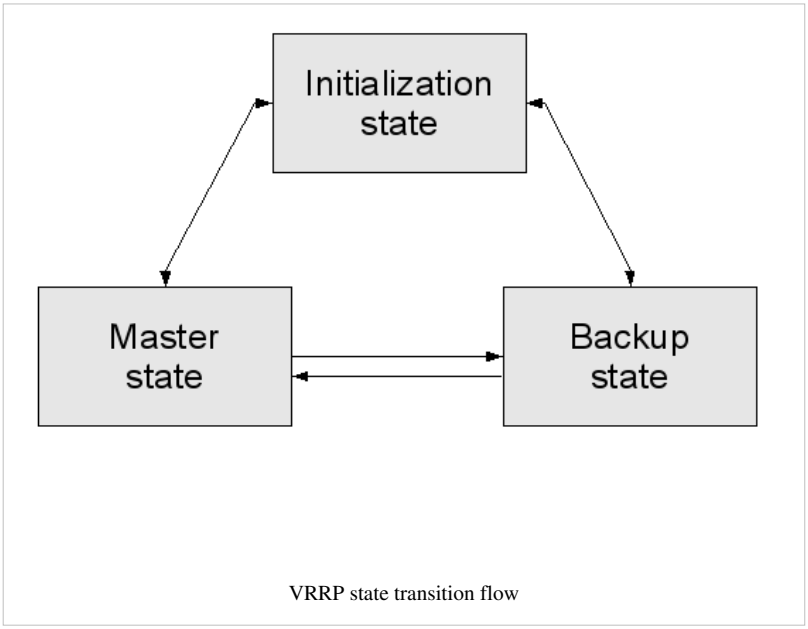
IPv6 ND

As you already know there are no ARP in IPv6 networks, routers are discovered by Neighbor Discovery protocol. When router becomes the Master, unsolicited ND Neighbor Advertisement with the Router Flag is sent for each IPv6 address associated with the virtual router.

VRRP state machine

As you can see from diagram, each VRRP node can be in one of three states:

- Init state
- Backup state
- Master state



VRRP state transition flow

Init state

The purpose of this state is to wait for a Startup event. When this event is received, then following actions are taken:

- if priority is 255,
- * for IPv4 send advertisement packet and broadcast ARP requests
- * for IPv6 send an unsolicited ND Neighbor Advertisement for each IPv6 address associated with the virtual router and set target address to link-local address associated with VR.
- * transit to MASTER state;
- else transit to BACKUP state.

Backup state

When in backup state,

- in IPv4 networks, node is not responding to ARP requests and is not forwarding traffic for the IP associated with the VR.
- in IPv6 networks, node is not responding to ND Neighbor Solicitation messages and is not sending ND Router Advertisement messages for VR associated IPv6 addresses.

Routers main task is to receive advertisement packets and check if master node is available.

Backup router will transit itself to master state in two cases:

- If priority in advertisement packet is 0;
- When Preemption_Mode is set to no, or Priority in the ADVERTISEMENT is greater than or equal to the local Priority

After transition to Master state node is:

- in IPv4 broadcasts gratuitous ARP request;
- in IPv6 sends an unsolicited ND Neighbor Advertisement for every associated IPv6 address.

In other cases advertisement packets will be discarded. When shutdown event is received, transit to Init state.

Note: Preemption mode is ignored if Owner router becomes available.

Master state

When MASTER state is set, node functions as a forwarding router for IPv4/IPv6 addresses associated with the VR.

In IPv4 networks Master node responds to ARP requests for the IPv4 address associated with the VR. In IPv6 networks Master node:

- responds to ND Neighbor Solicitation message for the associated IPv6 address;
- sends ND Router Advertisements for the associated IPv6 addresses.

If advertisement packet is received by master node:

- If priority is 0, send advertisement immediately;
- If priority in advertisement packet is greater than nodes priority then transit to backup state
- If priority in advertisement packet is equal to nodes priority and primary IP Address of the sender is greater than the local primary IP Address, then transit to backup state
- Ignore advertisement in other cases

When shutdown event is received, send advertisement packet with priority=0 and transit to Init state.

Configuring VRRP

IPv4

Setting up Virtual Router is quite easy, only two actions are required - create vrrp interface and set Virtual Routers IP address.

For example, add vrrp to ether1 and set VRs address to 192.168.1.1

```
/interface vrrp add name=vrrp1 interface=ether1
/ip address add address=192.168.1.1/32 interface=vrrp1
```

Notice that only 'interface' parameter was specified when adding vrrp. It is the only parameter required to be set manually, other parameters if not specified will be set to their defaults: `vrid=1`, `priority=100` and `authentication=none`.

Note: address on VRRP interface must have /32 netmask.

Before VRRP can operate correctly correct IP address is required on ether1. In this example it is `192.168.1.2/24`

VRRP Examples section contains several configuration examples.

IPv6

To make VRRP work in IPv6 networks, several additional options must be enabled - v3 support is required and protocol type should be set to IPv6:

```
/interface vrrp add name=vrrp1 interface=ether1 version=3 v3-protocol=ipv6
```

Now when VRRP interface is set, we can add global address and enable ND advertisement:

```
/ipv6 address add address=FEC0:0:0:FFFF::1/64 advertise=yes interface=vrrp1
```

No additional address configuration is required as it is in IPv4 case. IPv6 uses link-local addresses to communicate between nodes.

Property reference

Sub-menu: `/interface vrrp`

Property	Description
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; Default: **enabled**)	ARP resolution protocol mode
authentication (*ah* | *none* | *simple*; Default: **none**)	Authentication method to use for VRRP advertisement packets. • `none` - should be used only in low security networks (e.g., two VRRP nodes on LAN). • `ah` - IP Authentication Header. This algorithm provides strong protection against configuration errors, replay attacks and packet corruption/modification. Recommended when there is limited control over the administration of nodes on a LAN. • `simple` - uses clear text password. Protects against accidental misconfiguration of routers on local network.
interface (*string*; Default:)	Interface name on which VRRP instance will be running
interval (*time [10ms..4m15s]*; Default: **1s**)	VRRP update interval in seconds. Defines how often master sends advertisement packets.
mtu (*integer*; Default: **1500**)	Layer3 MTU size
name (*string*; Default:)	VRRP interface name
on-backup (*string*; Default:)	Script to execute when the node is switched to backup state
on-master (*string*; Default:)	Script to execute when the node is switched to master state
password (*string*; Default:)	Password required for authentication. Can be ignored if authentication is not used.
preemption-mode (*yes* | *no*; Default: **yes**)	Whether master node always has the priority. When set to 'no' backup node will not be elected to be a master until the current master fails, even if the backup node has higher priority than the current master. This setting is ignored if Owner router becomes available
priority (*integer: 1..254*; Default: **100**)	Priority of VRRP node used in Master election algorithm. Higher number means higher priority. '255' is reserved to Router that owns VR IP and '0' is reserved for Master router to indicate that it is releasing responsibility.
v3-protocol (*ipv4* | *ipv6*; Default: **ipv4**)	Protocol that will be used by VRRPv3. Valid only if **version** is 3
version (*integer [2, 3]*; Default: **3**)	Which VRRP version to use.
vrid (*integer: 1..255*; Default: **1**)	Virtual Router identifier. Each Virtual router must have unique id number

There are two ways to add scripts to `on-backup` and `on-master`

- specify scripts name added to script repository
- write script directly by putting it in scopes '{ }'.

See more

- VRRP-examples

[Top | Back to Content]

Manual:Virtualization

Applies to RouterOS: 3, v4

RouterOS has three different Virtualization implementations. Choose your topic:

- Metarouter
- Xen
- Kvm

Metarouter

Metarouter is created by MikroTik and currently is supported only on RouterBOARD 4xx series (mips-be) and RB1000 series (powerpc). Currently Metarouter can only create RouterOS virtual machines.

We are planning to add more features to Metarouter, so that it will even exceed Xen in functionality. New hardware support will also be added to Metarouter

Xen

Xen is based on the Linux Xen Virtual machine project, and current RouterOS implementation is supported only on RouterOS X86 systems (PCs). Xen can create Virtual machines of different Operating Systems that supports Xen.

Kvm

Kvm is based on Linux Kvm virtualization software and requires your CPU to support virtualization. Kvm is available only on x86 systems.

Usage Examples

The following are just a few of possible scenarios where virtual machines could be used (some of these currently are possible only in Xen, but Metarouter features will be expanded to allow even more functionality):

In the datacenter

- consolidate a number of routers on one hardware platform
- consolidate routing services and higher levels services such a VOIP switches in the same box
- use a guest machine on top of a router for custom features such as accounting, LDAP or legacy networking
- redundant routers much easier and cheaper to have available in case of crashed systems

In the hosting center

- use RouterOS and extensive networking features as a host with a server (mail, http, ftp...) running as guest or multiple guest virtual machines
- offer virtual routers with VPN solutions that give a network administrator customer his own router on a highspeed backbone to make any kind of tunneled intranet or simply VPN access system

At the wireless ISP client site

- set up two isolated routers and set the wireless control only for the router controlled by the WISP while the Ethernet side router is fully under the clients control

At multiclient sites (such as office buildings)

- in locations serving multiple clients by Ethernet from one backbone connection (wired or wireless), give each customer control over his own isolated virtual router

For network planning and testing

- build a virtual network on one box with the same topography as a planned network and test the configurations so that the fine tuning of the configurations can be done in the lab and not in the field, simulate and monitor the network with advanced scripting and The Dude network monitor utility

In custom applications

- develop your own programs (and even Linux distributions) that can be installed on MikroTik supported platforms with minimum difficulty as software patches and virtual drivers are provided for guest systems
- use low cost RouterBOARD embedded systems easily with your own Linux and the advantage that it will work across all RouterBOARDS with the same CPU

Manual:VPLS Control Word

Summary

Standards: `Pseudowire Fragmentation and Reassembly (RFC 4623)`

VPLS allows remote sites to share an Ethernet broadcast domain by connecting sites through **pseudo-wires(PW)** tunnels over a packet switching network (PSN). Since VPLS encapsulation adds additional overhead, each interface in LSP should be able to transmit large enough packet.

Each ethernet chipset has hardware limitation on maximum packet size that it can transmit. Even now there are Ethernets that supports only one Vlan tag, meaning that maximum packet size without Ethernet header and checksum (L2MTU) is 1504 bytes. Obviously it is not enough to forward VPLS encapsulated Ethernet frame without fragmentation (at least 1524 L2MTU support is required). See MTU article for maximum supported L2MTUs on RouterBOADs.

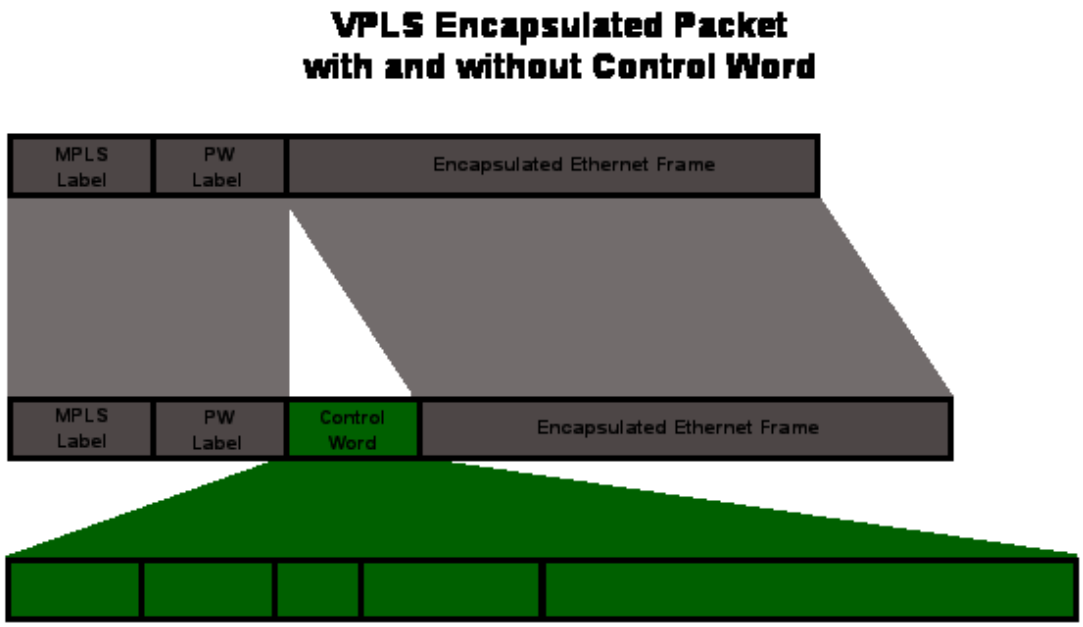
Since not even all RouterBOARDs support enough L2MTU to transmit VPLS encapsulated packet without fragmentation, RouterOS have added Pseudowire Fragmentation and Reassembly (PWE3) support according to RFC 4623 using 4-byte **Control Word (CW)**.

Control Word Usage

In RouterOS Control Word is used for packet fragmentation and reassembly inside VPLS tunnel and is done by utilizing optional **Control Word (CW)**. CW is added between PW label (demultiplexor) and packet payload and adds additional 4-byte overhead.

Until RouterOS v5.5 CW was used always, but, for compatibility with other vendors that do not use CW, feature to turn off Control Word usage was added. CW usage is controlled by one new parameter `use-control-word` in /interface vpls bgp-vpls and /interface vpls cisco-bgp-vpls

VPLS Packet and CW Format



As you can see **Control Word** is divided into 5 fileds:

- 0000 - 4-bits identifies that packet packet is PW (not IP)
- Flags - 4bits
- Frag - 2bits value that indicates payload fragmentation.
- Len - 6bits
- Seq - 16bits sequence number used to detect packet loss / misordering.

According to RFC generation and processing of sequence number is optional.

Example Setup

To show CW usage we will use simple three router setup as illustrated below.

This setup will not explain BGP and LDP configuration since its detailed explanation is found in other articles. `Read here>>`

See Also

- Basic MPLS and LDP based VPLS
- BGP based VPLS
- VPLS with Cisco routers

[Top | Back to Content]

Manual:VRRP-examples

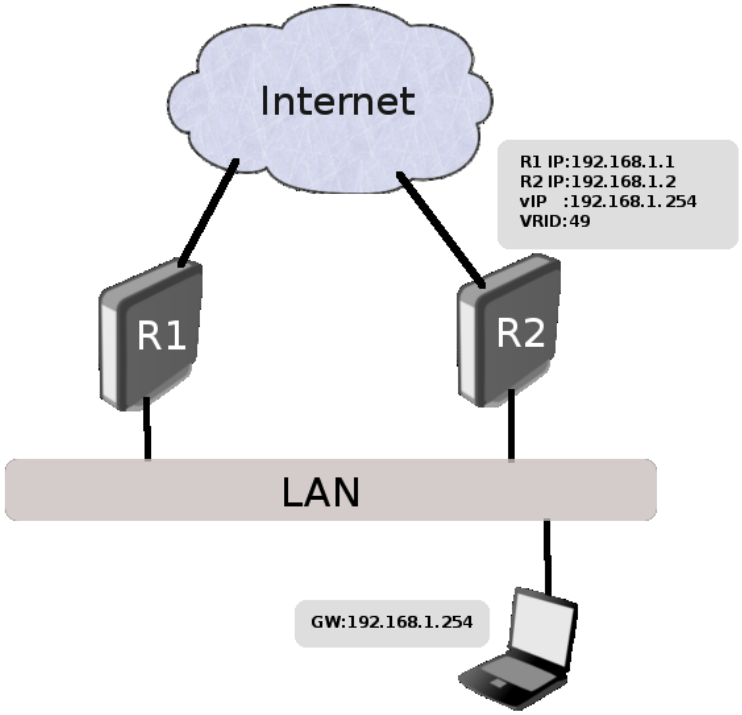
Applies to RouterOS: v3, v4

VRRP Configuration Examples

This section contains several useful VRRP configuration examples

Basic Setup

This is the basic VRRP configuration example.



According to this configuration, as long as the master, R1, is functional, all traffic destined to the external network gets directed to R1. But as soon as R1 fails, R2 takes over as the master and starts handling packets forwarded to the interface associated with IP(R1). In this setup Router R2 is completely idle during Backup period.

Configuration

R1 configuration:

```
/ip address add address=192.168.1.1/24 interface=ether1
/interface vrrp add interface=ether1 vrid=49 priority=254
/ip address add address=192.168.1.254/32 interface=vrrp1
```

R2 configuration:

```
/ip address add address=192.168.1.2/24 interface=ether1
/interface vrrp add interface=ether1 vrid=49
/ip address add address=192.168.1.254/32 interface=vrrp1
```

Testing

First of all check if both routers have correct flags at vrrp interfaces. On router R1 it should look like this

```
/interface vrrp print
 0   RM name="vrrp1" mtu=1500 mac-address=00:00:5E:00:01:31 arp=enabled interface=ether1 vrid=49
       priority=254 interval=1 preemption-mode=yes authentication=none password="" on-backup=""
       on-master=""
```

and on router R2:

```
/interface vrrp print
 0    B name="vrrp1" mtu=1500 mac-address=00:00:5E:00:01:31 arp=enabled interface=ether1 vrid=49
       priority=100 interval=1 preemption-mode=yes authentication=none password=""
       on-backup="" on-master="
```

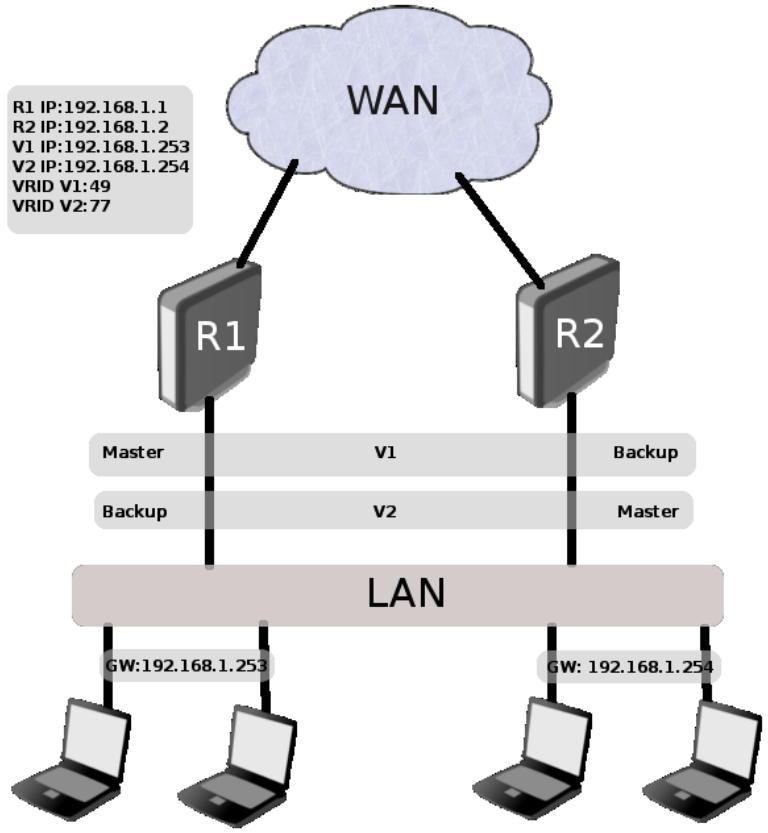
As you can see vrrp interface mac addresses are identical on both routers. Now to check if vrrp is working correctly, try to ping virtual address from client and check arp entries:

```
[admin@client] > /ping 192.168.1.254
192.168.1.254 64 byte ping: ttl=64 time=10 ms
192.168.1.254 64 byte ping: ttl=64 time=8 ms
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 8/9.0/10 ms
[admin@client] /ip arp> print
Flags: X - disabled, I - invalid, H - DHCP, D - dynamic
 #    ADDRESS         MAC-ADDRESS       INTERFACE
 ...
 1 D 192.168.1.254   00:00:5E:00:01:31 bridge1
```

Now unplug ether1 cable on router R1. R2 will become VRRP master, ARP table on client will not change but traffic will start to flow over R2 router.

Load sharing

In basic configuration example R2 is completely idle during Backup state. This behavior may be considered as waste of valuable resources. In such circumstances R2 router can be set as gateway for some clients.
The obvious advantage of this configuration is the establishment of a load-sharing scheme. But by doing so R2 router is not protected by current VRRP setup.
To make this setup work we need two virtual routers.



Configuration for V1 virtual router will be identical to configuration in basic example - R1 is the Master and R2 is Backup router. In V2 Master is R2 and Backup is R1.
With this configuration, we establish a load-sharing between R1 and R2; moreover, we create protection setup by having two routers acting as backups for each other.

Configuration

R1 configuration:

```
/ip address add address=192.168.1.1/24 interface=ether1
/interface vrrp add interface=ether1 vrid=49 priority=254
/interface vrrp add interface=ether1 vrid=77
/ip address add address=192.168.1.253/32 interface=vrrp1
/ip address add address=192.168.1.254/32 interface=vrrp2
```

R2 configuration:

```
/ip address add address=192.168.1.2/24 interface=ether1
/interface vrrp add interface=ether1 vrid=49
/interface vrrp add interface=ether1 vrid=77 priority=254
/ip address add address=192.168.1.253/32 interface=vrrp1
/ip address add address=192.168.1.254/32 interface=vrrp2
```

VRRP without Preemption

Each time when router with higher priority becomes available it becomes Master router. Sometimes it is not desired behavior which can be turned off by setting `preemption-mode=no` in vrrp configuration.

Configuraton

We will be using the same setup as in basic example. Only difference is during configuration set preemption-mode=no. It can be done easily modifying existing configuration:

```
/interface vrrp set [find] preemption-mode=no
```

Testing

Try turning off R1 router, R2 will become Master router because it has highest priority among available routers.

Now turn R1 router on and you will see that R2 router continues to be Master even if R1 has higher priority.

VRRP and scripts

See Also

- VRRP
- Scripting

[Top | Back to Content]

Manual:Virtual Routing and Forwarding

Applies to RouterOS: 3, v4

Packages required: ***routing-test***, ***mpls-test*** *for RouterOS v3;* ***routing***, ***mpls*** *for RouterOS v4+*

Description

RouterOS 3.x allows to create multiple Virtual Routing and Forwarding instances on a single router. This is useful for BGP based MPLS VPNs. Unlike BGP VPLS, which is OSI Layer 2 technology, BGP VRF VPNs work in Layer 3 and as such exchange IP prefixes between routers. VRFs solve the problem of overlapping IP prefixes, and provide the required privacy (via separated routing for different VPNs).

To create a VRF, configure it under **/ip route vrf**. You can now add routes to that VRF - simply specify **routing-mark** attribute. Connected routes from interfaces belonging to a VRF will be installed in the right routing table automatically.

Technically VRFs are based on policy routing. There is exactly one policy route table for each active VRF. The existing policy routing support in MT RouterOS is not changed; but on the other hand, it is not possible to have policy routing *within* a VRF. The main differences between VRF tables and simple policy routing are:

- Routes in VRF tables resolve next-hops in their own route table by default, while policy routes always use the main route table. Read-only route attribute **gateway-table** displays information about which table is used for a particular route (default is **main**).
- Route lookup is different. For policy routing: after route lookup has been done in policy-route table, and no route was found, route lookup proceeds to the main route table. For VRFs: if lookup is done, and no route is found in

VRF route table, the lookup fails with "network unreachable" error. (You can still override this behavior with custom route lookup rules, as they have precedence.)

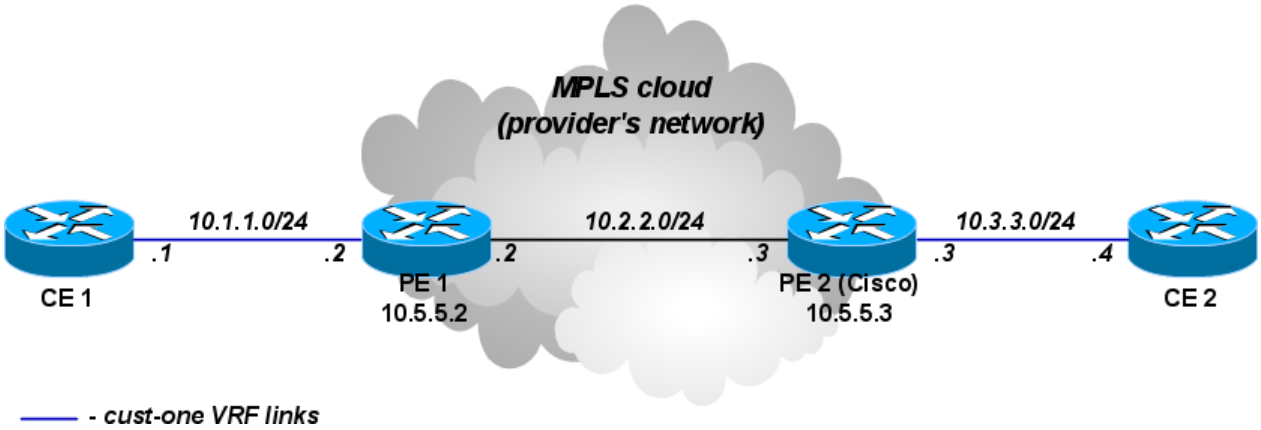
You can use multi-protocol BGP with VPNv4 address family to distribute routes from VRF route tables - not only to other routers, but also to different routing tables in the router itself. First configure the route distinguisher for a VRF. It can be done under **/ip route vrf**. Usually there will be one-to-one correspondence between route distinguishers and VRFs, but that's not a mandatory requirement. Route installation in VRF tables is controlled by BGP extended communities attribute. Configure import and export lists under **/ip route vrf**, **import-route-targets** and **export-route-targets**. Export route target list for a VRF should contained at least the route distinguisher for that VRF. Then configure a list of VRFs for each BGP instance that will participate in VRF routing.

Once list of VRFs for BGP instance, route distinguisher and export route targets has been configured, some active VPNv4 address family routes may be created, depending on BGP redistribution settings. They are installed in a separate route table and, if present, visible under **/routing bgp vpnv4-route**. These so called VPNv4 routes have prefix that consists of a route distinguisher and an IPv4 network prefix. This way you can have overlapping IPv4 prefixes distributed in BGP.

Please note that a VPNv4 route will be distributed only if it has a valid MPLS label. You need to install **mpls-test** package and configure valid label range for this to work. (Default configuration has valid label range.)

Examples

The simplest MPLS VPN setup



In this example rudimentary MPLS backbone (consisting of two Provider Edge (PE) routers PE1 and PE2) is created and configured to forward traffic between Customer Edge (CE) routers CE1 and CE2 routers that belong to *cust-one* VPN.

CE1 Router

```
/ip address add address=10.1.1.1/24 interface=ether1
# use static routing
/ip route add dst-address=10.3.3.0/24 gateway=10.1.1.2
```

CE2 Router

```
/ip address add address=10.3.3.4/24 interface=ether1
/ip route add dst-address=10.1.1.0/24 gateway=10.3.3.3
```

PE1 Router

```
/interface bridge add name=lobridge
/ip address add address=10.1.1.2/24 interface=ether1
/ip address add address=10.2.2.2/24 interface=ether2
/ip address add address=10.5.5.2/32 interface=lobridge
/ip route vrf add disabled=no routing-mark=cust-one route-distinguisher=1.1.1.1:111 \
    export-route-targets=1.1.1.1:111 import-route-targets=1.1.1.1:111 interfaces=ether1
/mpls ldp set enabled=yes transport-address=10.5.5.2
/mpls ldp interface add interface=ether2
/routing bgp instance set default as=65000
/routing bgp instance vrf add instance=default routing-mark=cust-one redistribute-connected=yes
/routing bgp peer add remote-address=10.5.5.3 remote-as=65000 address-families=vpnv4 \
    update-source=lobridge
 # add route to the remote BGP peer's loopback address
/ip route add dst-address=10.5.5.3/32 gateway=10.2.2.3
```

PE2 Router (Cisco)

```
ip vrf cust-one
 rd 1.1.1.1:111
 route-target export 1.1.1.1:111
 route-target import 1.1.1.1:111
 exit

interface Loopback0
 ip address 10.5.5.3 255.255.255.255

mpls ldp router-id Loopback0 force
mpls label protocol ldp

interface FastEthernet0/0
 ip address 10.2.2.3 255.255.255.0
 mpls ip

interface FastEthernet1/0
 ip vrf forwarding cust-one
 ip address 10.3.3.3 255.255.255.0

router bgp 65000
 neighbor 10.5.5.2 remote-as 65000
 neighbor 10.5.5.2 update-source Loopback0
 address-family vpnv4
  neighbor 10.5.5.2 activate
  neighbor 10.5.5.2 send-community both
  exit-address-family
 address-family ipv4 vrf cust-one
  redistribute connected
  exit-address-family
```

```
ip route 10.5.5.2 255.255.255.255 10.2.2.2
```

Results

Check that VPNv4 route redistribution is working:

```
[admin@PE1] > /routing bgp vpnv4-route print detail
Flags: L - label present
 0 L route-distinguisher=1.1.1.1:111 dst-address=10.3.3.0/24 gateway=10.5.5.3
     interface=ether2 in-label=17 out-label=17 bgp-local-pref=100 bgp-med=0
     bgp-origin=incomplete bgp-ext-communities="RT:1.1.1.1:111"

 1 L route-distinguisher=1.1.1.1:111 dst-address=10.1.1.0/24 interface=ether1
     in-label=16 bgp-ext-communities="RT:1.1.1.1:111"
```

Check that the 10.3.3.0 is installed in IP routes, in cust-one route table:

```
[admin@PE1] > /ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 0 ADC  10.1.1.0/24         10.1.1.2         ether1              0
 1 ADb  10.3.3.0/24                         10.5.5.3 recursi... 20
 2 ADC  10.2.2.0/24         10.2.2.2         ether2              0
 3 ADC  10.5.5.2/32         10.5.5.2         lobridge            0
 4 A S  10.5.5.3/32                         10.2.2.3 reachab... 1
```

Let's take closer look at IP routes in cust-one VRF. The 10.1.1.0/24 IP prefix is a connected route that belongs to an interface that was configured to belong to cust-one VRF. The 10.3.3.0/24 IP prefix was advertised via BGP as VPNv4 route from PE2 and is imported in this VRF routing table, because our configured **import-route-targets** matched the BGP extended communities attribute it was advertised with.

```
[admin@PE1] /ip route> print detail where routing-mark=cust-one
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 0 ADC  dst-address=10.1.1.0/24 pref-src=10.1.1.2 gateway=ether1 distance=0 scope=10
        routing-mark=cust-one

 1 ADb  dst-address=10.3.3.0/24 gateway=10.5.5.3 recursive via 10.2.2.3 ether2
        distance=20 scope=40 target-scope=30 routing-mark=cust-one
        bgp-local-pref=100 bgp-origin=incomplete
        bgp-ext-communities="RT:1.1.1.1:111"
```

The same for Cisco:

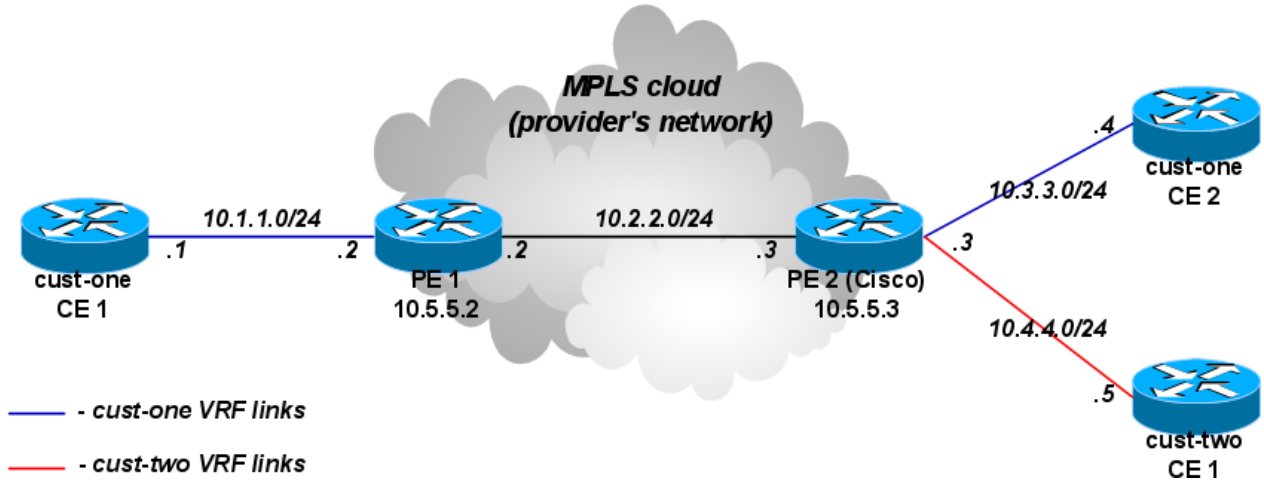
```
PE2#show ip bgp vpnv4 all
BGP table version is 5, local router ID is 10.5.5.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
```

```
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1.1.1.1:111 (default for vrf cust-one)
*>i10.1.1.0/24      10.5.5.2                      100      0 ?
*> 10.3.3.0/24      0.0.0.0                  0         32768 ?

PE2#show ip route vrf cust-one

Routing Table: cust-one
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 1 subnets
B       10.1.1.0 [200/0] via 10.5.5.2, 00:05:33
     10.0.0.0/24 is subnetted, 1 subnets
C       10.3.3.0 is directly connected, FastEthernet1/0
```

You should be able to ping from CE1 to CE2 and vice versa.

```
[admin@CE1] > /ping 10.3.3.4
10.3.3.4 64 byte ping: ttl=62 time=18 ms
10.3.3.4 64 byte ping: ttl=62 time=13 ms
10.3.3.4 64 byte ping: ttl=62 time=13 ms
10.3.3.4 64 byte ping: ttl=62 time=14 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 13/14.5/18 ms
```

A more complicated setup (changes only)



As opposed to the simplest setup, in this example we have two customers: cust-one and cust-two.

We configure two VPNs for then, cust-one and cust-two respectively, and exchange all routes between them. (This is also called "route leaking").

Note that this could be not the most typical setup, because routes are usually not exchanged between different customers. In contrast, by default it should not be possible to gain access from one VRF site to a different VRF site in another VPN. (This is the "Private" aspect of VPNs.) Separate routing is a way to provide privacy; and it is also required to solve the problem of overlapping IP network prefixes. Route exchange is in direct conflict with these two requirement but may sometimes be needed (e.g. temp. solution when two customers are migrating to single network infrastructure).

CE1 Router, *cust-one*

```
/ip route add dst-address=10.4.4.0/24 gateway=10.1.1.2
```

CE2 Router, *cust-one*

```
/ip route add dst-address=10.4.4.0/24 gateway=10.3.3.3
```

CE1 Router, *cust-two*

```
/ip address add address=10.4.4.5 interface=ether1
/ip route add dst-address=10.1.1.0/24 gateway=10.3.3.3
/ip route add dst-address=10.3.3.0/24 gateway=10.3.3.3
```

PE1 Router

```
# replace the old VRF with this:
/ip route vrf add disabled=no routing-mark=cust-one route-distinguisher=1.1.1.1:111 \
    export-route-targets=1.1.1.1:111 import-route-targets=1.1.1.1:111,2.2.2.2:222 interfaces=ether1
```

PE2 Router (Cisco)

```
ip vrf cust-one
 rd 1.1.1.1:111
 route-target export 1.1.1.1:111
 route-target import 1.1.1.1:111
 route-target import 2.2.2.2:222
 exit

ip vrf cust-two
 rd 2.2.2.2:222
 route-target export 2.2.2.2:222
 route-target import 1.1.1.1:111
 route-target import 2.2.2.2:222
 exit

interface FastEthernet2/0
 ip vrf forwarding cust-two
 ip address 10.4.4.3 255.255.255.0

router bgp 65000
```

```
 address-family ipv4 vrf cust-two
  redistribute connected
  exit-address-family
```

Variation: replace the Cisco with another MT

PE2 Mikrotik config

```
/interface bridge add name=lobridge
/ip address
 add address=10.2.2.3/24 interface=ether1
 add address=10.3.3.3/24 interface=ether2
 add address=10.4.4.3/24 interface=ether3
 add address=10.5.5.3/32 interface=lobridge
/ip route vrf
 add disabled=no routing-mark=cust-one route-distinguisher=1.1.1.1:111 \
     export-route-targets=1.1.1.1:111 import-route-targets=1.1.1.1:111,2.2.2.2:222 \
     interfaces=ether2
 add disabled=no routing-mark=cust-two route-distinguisher=2.2.2.2:222 \
    export-route-targets=2.2.2.2:222 import-route-targets=1.1.1.1:111,2.2.2.2:222 \
    interfaces=ether3
/mpls ldp set enabled=yes transport-address=10.5.5.3
/mpls ldp interface add interface=ether1
/routing bgp instance set default as=65000
/routing bgp instance vrf add instance=default routing-mark=cust-one redistribute-connected=yes
/routing bgp instance vrf add instance=default routing-mark=cust-two redistribute-connected=yes
/routing bgp peer add remote-address=10.5.5.2 remote-as=65000 address-families=vpnv4 \
    update-source=lobridge
 # add route to the remote BGP peer's loopback address
/ip route add dst-address=10.5.5.2/32 gateway=10.2.2.2
```

Results

The output of **/ip route print** now is interesting enough to deserve detailed observation.

```
[admin@PE2] /ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 0 ADb  10.1.1.0/24                        10.5.5.2 recurs... 20
 1 ADC  10.3.3.0/24        10.3.3.3        ether2             0
 2 ADb  10.4.4.0/24                                           20
 3 ADb  10.1.1.0/24                        10.5.5.2 recurs... 20
 4 ADb  10.3.3.0/24                                           20
 5 ADC  10.4.4.0/24        10.4.4.3        ether3             0
 6 ADC  10.2.2.0/24        10.2.2.3        ether1             0
 7 A S  10.5.5.2/32                        10.2.2.2 reacha... 1
 8 ADC  10.5.5.3/32        10.5.5.3        lobridge           0
```

The route 10.1.1.0/24 was received from remote BGP peer and is installed in both VRF routing tables.

The routes 10.3.3.0/24 and 10.4.4.0/24 are also installed in both VRF routing tables. Each is as connected route in one table and as BGP route in another table. This has nothing to do with their being advertised via BGP. They are simply being "advertised" to local VPNv4 route table and locally reimported after that. Import and export **route-targets** determine in which tables they will end up.

This can be deduced from its attributes - they don't have the usual BGP properties. (Route 10.4.4.0/24.)

```
[admin@PE2] /ip route> print detail where routing-mark=cust-one
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 0 ADb  dst-address=10.1.1.0/24 gateway=10.5.5.2 recursive via 10.2.2.2 ether1
        distance=20 scope=40 target-scope=30 routing-mark=cust-one
        bgp-local-pref=100 bgp-origin=incomplete
        bgp-ext-communities="RT:1.1.1.1:111"

 1 ADC  dst-address=10.3.3.0/24 pref-src=10.3.3.3 gateway=ether2 distance=0 scope=10
        routing-mark=cust-one

 2 ADb  dst-address=10.4.4.0/24 distance=20 scope=40 target-scope=10
        routing-mark=cust-one bgp-ext-communities="RT:2.2.2.2:222"
```

Static inter-VRF routes

In general it is recommended that all routes between VRF should be exchanged using BGP local import and export functionality. If that is not enough, static routes can be used to achieve this so-called route leaking.

There are two ways to install a route that has gateway in different routing table than the route itself.

The first way is to explicitly specify routing table in gateway field when adding route. This is only possible for the "main" routing table. Example:

```
# add route to 5.5.5.0/24 in 'vrf1' routing table with gateway in the main routing table
add dst-address=5.5.5.0/24 gateway=10.3.0.1@main routing-mark=vrf1
```

The second way is to explicitly specify interface in gateway field. The interface specified can belong to a VRF instance. Example:

```
# add route to 5.5.5.0/24 in the main routing table with gateway at 'ether2' VRF interface
add dst-address=5.5.5.0/24 gateway=10.3.0.1%ether2 routing-mark=main

# add route to 5.5.5.0/24 in the main routing table with 'ptp-link-1' VRF interface as gateway
add dst-address=5.5.5.0/24 gateway=ptp-link-1 routing-mark=main
```

As can be observed, there are two variations possible - to specify gateway as *ip_address%interface* or to simply specify *interface*. The first should be used for broadcast interfaces in most cases. The second should be used for point-to-point interfaces, and also for broadcast interfaces, if the route is a connected route in some VRF. For example, if you have address `1.2.3.4/24` on interface *ether2* that is put in a VRF, there will be connected route to `1.2.3.0/24` in that VRF's routing table. It is acceptable to add static route `1.2.3.0/24` in a different routing table with interface-only gateway, even though *ether2* is a broadcast interface:

```
add dst-address=1.2.3.0/24 gateway=ether2 routing-mark=main
```

References

RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs) [1]

MPLS Fundamentals, chapter 7, *Luc De Ghein*, Cisco Press 2006

References

[1] http://www.ietf.org/rfc/rfc4364.txt

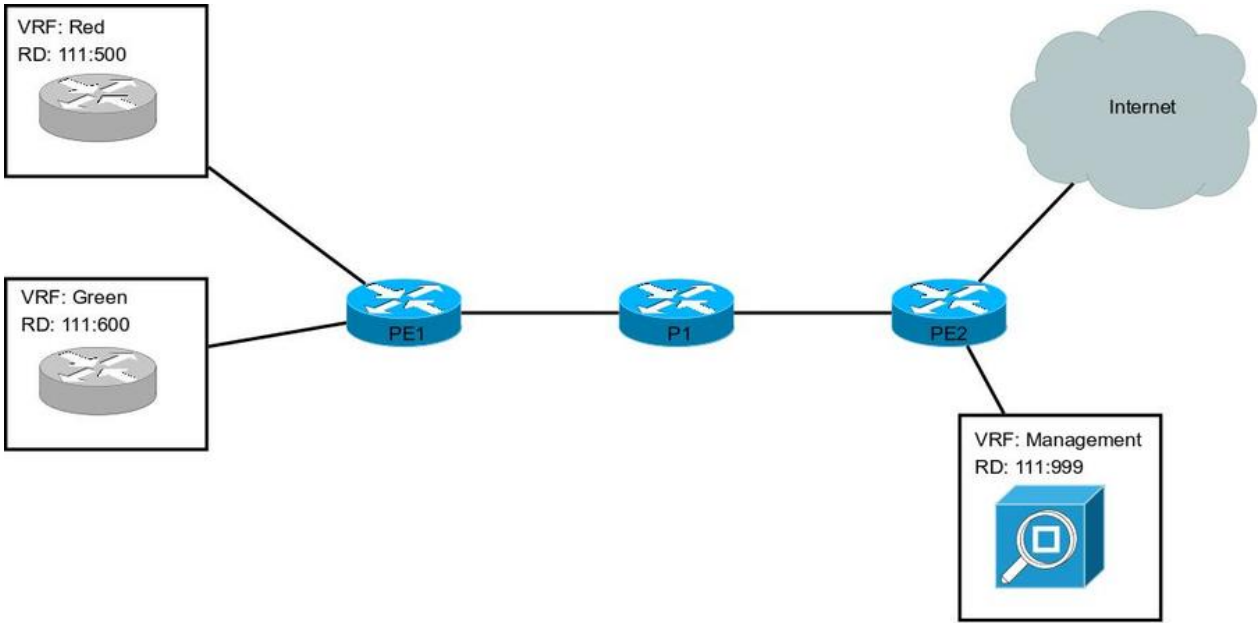
VRF Route Leaking

Packages required: ***routing-test***, ***mpls-test***, RouterOS version 3.25+

Description

When running a multi-tenant MPLS network, it can be useful to leak routes between VRFs. A classic use for this would be to leak your link-nets to a management VRF, or assigning a management address to your CE routers as a /32 address and leaking that. Other uses could be leaking public ip addresses to a separate VRF, to be handled by a different router than the LAN addresses. It is necessary to filter your route leaking to make sure that only non-overlapping addresses are leaked, and it is important to make sure that one VRF doesn't have access to routes of another VRF. This is done by using routing filters and the method of filtering outgoing VRF routes added by the 3.25 software version of RouterOS.

Example Diagram



In this example design, we have two customer VRFs as well as a management VRF. It is assumed that each customer VRF will have potential overlapping IP addresses, however the link addresses are assumed to be non-overlapping. The VRFs are aggregated at the PE1 router and the exit point of the network is at PE2, where the management system is also connected. The management system could be The Dude, or any other NMS software.

VRF Setup

First we take a look at how each VRF is set up at PE1:

```
/ip route vrf
add routing-mark=red route-distinguisher=111:500 import-route-targets=111:500,111:999 \
export-route-targets=111:500 interfaces=ether1.500
/ip route vrf
add routing-mark=green route-distinguisher=111:600 import-route-targets=111:600,111:999 \
export-route-targets=111:600 interfaces=ether1.600
```

This determines that routes learned from the red VRF will on this PE be marked with the BGP extended community 111:500, and exported with same community. The VRF will import routes with the communities 111:500 and 111:999. 111:999 will be imported to ensure that the router can reach the management VRF. Same goes for the green VRF.

```
/routing bgp instance vrf
add instance=default routing-mark=red redistribute-connected=yes redistribute-ospf=yes \
 redistribute-static=yes out-filter=red-out
/routing bgp instance vrf
add instance=default routing-mark=green redistribute-connected=yes redistribute-ospf=yes \
 redistribute-static=yes out-filter=green-out
```

Nothing out of the ordinary happening here, except that it is specified that each vrf will use an out-filter.

The management VRF will be set up on PE2 as follows:

```
/ip route vrf
add routing-mark=management route-distinguisher=111:999 import-route-targets=111:999,111:1000 \
export-route-targets=111:999 interfaces=ether1.999
/routing bgp instance vrf add instance=default routing-mark=management
```

The red and green VRF will be set up here as well with a standard configuration. Notice the additional community that the management VRF imports. It is used the import community for customer routes to keep them separate from the community that the management VRF exports.

Routing filters

On PE1 we set up the red-out and green-out filter:

```
/routing filter
add chain=red-out match-chain=connected-in append-route-targets=111:1000 action=passthrough
add chain=green-out match-chain=connected-in append-route-targets=111:1000 action=passthrough
```

These filters match all routes of the connected type in a VRF (ie. the link address) and appends the extended community 111:1000 to these routes. On PE2 these routes will then be imported into the management VRF.

I recognice that the two filters are essentially the same, but trust me, you would rather have separate chains from the beginning, than reconfigure everyone, when you need extra rules for a certain VRF.

Same setup with loopback management

The previous setup assumed your link addresses were unique. If they are not, you can achieve the same effect by adding a loopback interface (ie. a bridge without members). Distribute a /32 address to all the CE routers you wish to manage this way. It is advisable to use either public ip addresses or addresses that you are sure that none of your customers use - 198.18.0.0/15 is handy. You must then match that network in your routing filters, but it's principally the same thing.

Manual:Interface/Wireless

Overview

Standards:

Package: `wireless`

RouterOS wireless comply with IEEE 802.11 standards, it provides complete support for 802.11a, 802.11b, 802.11g and 802.11n as long as additional features like WPA, WEP, AES encryption, Wireless Distribution System (WDS), Dynamic Frequency selection (DFS), Virtual Access Point, Nstreme and NV2 proprietary protocols and many more. Wireless features compatibility table for different wireless protocols.

Wireless can operate in several modes: client (station), access point, wireless bridge etc. Client/station also can operate in different modes, complete list of supported modes can be found here.

General interface properties

Sub-menu: `/interface wireless`

Property	Description
adaptive-noise-immunity (*ap-and-client-mode* | *client-mode* | *none*; Default: **none**)	This property is only effective for cards based on Atheros chipset.
allow-sharedkey (*yes* | *no*; Default: **no**)	Allow WEP Shared Key cilents to connect. Note that no authentication is done for these clients (WEP Shared keys are not compared to anything) - they are just accepted at once (if access list allows that)
antenna-gain (*integer [0..4294967295]*; Default: **0**)	Antenna gain in dBi, used to calculate maximum transmit power according to **country** regulations.
antenna-mode (*ant-a* | *ant-b* | *rxa-txb* | *txa-rxb*; Default:)	Select antenna to use for transmitting and for receiving • *ant-a* - use only 'a' antenna • *ant-b* - use only 'b' antenna • *txa-rxb* - use antenna 'a' for transmitting, antenna 'b' for receiving • *rxa-txb* - use antenna 'b' for transmitting, antenna 'a' for receiving
area (*string*; Default:)	Identifies group of wireless networks. This value is announced by AP, and can be matched in connect-list by **area-prefix**. This is a proprietary extension.
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; Default: **enabled**)	`Read more >>`
band (*2ghz-b* | *2ghz-b/g* | *2ghz-b/g/n* | *2ghz-onlyg* | *2ghz-onlyn* | *5ghz-a* | *5ghz-a/n* | *5ghz-onlyn*; Default:)	Defines set of used data rates, channel frequencies and widths.
basic-rates-a/g (*12Mbps* | *18Mbps* | *24Mbps* | *36Mbps* | *48Mbps* | *54Mbps* | *6Mbps* | *9Mbps*; Default: **6Mbps**)	Similar to the **basic-rates-b** property, but used for 5ghz, 5ghz-10mhz, 5ghz-5mhz, 5ghz-turbo, 2.4ghz-b/g, 2.4ghz-onlyg, 2ghz-10mhz, 2ghz-5mhz and 2.4ghz-g-turbo bands.

basic-rates-b (*11Mbps* | *1Mbps* | *2Mbps* | *5.5Mbps*; Default: **1Mbps**)
List of basic rates, used for 2.4ghz-b, 2.4ghz-b/g and 2.4ghz-onlyg bands. Client will connect to AP only if it supports all basic rates announced by the AP. AP will establish WDS link only if it supports all basic rates of the other AP.

This property has effect only in AP modes, and when value of **rate-set** is configured.

bridge-mode (*disabled* | *enabled*; Default: **enabled**)
Allows to use station-bridge mode. `Read more >>`

burst-time (*integer* | *disabled*; Default: **disabled**)
Time in microseconds which will be used to send data without stopping. Note that no other wireless cards in that network will be able to transmit data during burst-time microseconds. This setting is available only for AR5000, AR5001X, and AR5001X+ chipset based cards.

channel-width (*10mhz* | *20/40mhz-ht-above* | *20/40mhz-ht-below* | *20mhz* | *40mhz-turbo* | *5mhz*; Default: **20mhz**)
ht above and ht below allows to use additional 20MHz extension channel and if it should be located below or above control (main) channel. Extension channel allows 11n device to use 40MHz of spectrum in total thus increasing max throughput.

comment (*string*; Default:)
Short description of the interface

compression (*yes* | *no*; Default: **no**)
Setting this property to *yes* will allow use of the hardware compression. Wireless interface must have support for hardware compression. Connections with devices that do not use compression will still work.

country (*name of the country* | *no_country_set*; Default: **no_country_set**)
Limits available bands, frequencies and maximum transmit power for each frequency. Also specifies default value of **scan-list**. Value *no_country_set* is an FCC compliant set of channels.

default-ap-tx-limit (*integer [0..4294967295]*; Default: **0**)
This is the value of **ap-tx-limit** for clients that do not match any entry in the access-list. 0 means no limit.

default-authentication (*yes* | *no*; Default: **yes**)
For AP mode, this is the value of **authentication** for clients that do not match any entry in the access-list. For station mode, this is the value of **connect** for APs that do not match any entry in the connect-list

default-client-tx-limit (*integer [0..4294967295]*; Default: **0**)
This is the value of **client-tx-limit** for clients that do not match any entry in the access-list. 0 means no limit

default-forwarding (*yes* | *no*; Default: **yes**)
This is the value of **forwarding** for clients that do not match any entry in the access-list

dfs-mode (*no-radar-detect* | *none* | *radar-detec*; Default: **none**)
Controls DFS (Dynamic Frequency Selection).

- *none* - disables DFS.
- *no-radar-detect* - Select channel from **scan-list** with the lowest number of detected networks. In 'wds-slave' mode this setting has no effect.
- *radar-detect* - Select channel with the lowest number of detected networks and use it if no radar is detected on it for 60 seconds. Otherwise, select different channel. This setting may be required by the country regulations.

This property has effect only in AP mode.

disable-running-check (*yes* | *no*; Default: **no**)
When set to **yes** interface will always have running flag. If value is set to **no**', the router determines whether the card is up and running - for AP one or more clients have to be registered to it, for station, it should be connected to an AP.

disabled (*yes* | *no*; Default: **yes**)
Whether interface is disabled

disconnect-timeout (*time [0s..15s]*; Default: **3s**)

This interval is measured from third sending failure on the lowest data rate. At this point 3 * (**hw-retries** + 1) frame transmits on the lowest data rate had failed.

During **disconnect-timeout** packet transmission will be retried with **on-fail-retry-time** interval. If no frame can be transmitted successfully during **diconnect-timeout**, connection is closed, and this event is logged as "extensive data loss". Successful frame transmission resets this timer.

distance (*integer | dynamic | indoors*; Default: **dynamic**)

How long to wait for confirmation of unicast frames before considering transmission unsuccessful. Value 'dynamic' causes AP to detect and use smallest timeout that works with all connected clients. Acknowledgements are not used in Nstreme protocol.

frame-lifetime (*integer [0..4294967295]*; Default: **0**)

Discard frames that have been queued for sending longer than **frame-lifetime**. By default, when value of this property is *0*, frames are discarded only after connection is closed.

frequency (*integer [0..4294967295]*; Default:)

Channel frequency value in MHz on which AP will operate. Allowed values depend on selected band, and are restricted by **country** setting and wireless card capabilities. This setting has **no effect** if interface is in any of **station** modes, or in *wds-slave* mode, or if DFS is active.

Note: If using mode "superchannel", any frequency supported by the card will be accepted, but on the RouterOS client, any non-standard frequency must be configured in the scan-list, otherwise it will not be scanning in non-standard range. In Winbox, scanlist frequencies are in *bold*, any other frequency means the clients will need scan-list configured.

frequency-mode (*manual-txpower | regulatory-domain | superchannel*; Default: **manual-txpower**)

Three frequency modes are available:

- *regulatory-domain* - Limit available channels and maximum transmit power for each channel according to the value of **country**
- *manual-txpower* - Same as above, but do not limit maximum transmit power.
- *superchannel* - Conformance Testing Mode. Allow all channels supported by the card.

List of available channels for each band can be seen in **/wireless info print**. This mode allows you to test wireless channels outside the default scan-list and/or regulatory domain. This mode should only be used in controlled environments, or if you have a special permission to use it in your region. Before v4.3 this was called Custom Frequency Upgrade, or Superchannel. Since RouterOS v4.3 this mode is available without special key upgrades to all installations.

frequency-offset (*integer [-2147483648..2147483647]*; Default: **0**)

Allows to specify offset if the used wireless card operates at a different frequency than is shown in RouterOS, in case a frequency converter is used in the card. So if your card works at 4000MHz but RouterOS shows 5000MHz, set offset to 1000MHz and it will be displayed correctly. The value is in MHz and can be positive or negative.

hide-ssid (*yes | no*; Default: **no**)

.

- *yes* - AP does not include SSID the beacon frames, and does not reply to probe requests that have broadcast SSID.
- *no* - AP includes SSID in the beacon frames, and replies to probe requests that have broadcast SSID.

This property has effect only in AP mode. Setting it to *yes* can remove this network from the list of wireless networks that are shown by some client software. Changing this setting does not improve security of the wireless network, because SSID is included in other frames sent by the AP.

ht-ampdu-priorities (*list of integer [0..7]*; Default: **0**)
Frame priorities for which AMPDU sending (aggregating frames and sending using block acknowledgement) should get negotiated and used. Using AMPDUs will increase throughput, but may increase latency therefore may not be desirable for real-time traffic (voice, video). Due to this, by default AMPDUs are enabled only for best-effort traffic.

ht-amsdu-limit (*integer [0..8192]*; Default: **8192**)
Max AMSDU that device is allowed to prepare when negotiated. AMSDU aggregation may significantly increase throughput especially for small frames, but may increase latency in case of packet loss due to retransmission of aggregated frame. Sending and receiving AMSDUs will also increase CPU usage.

ht-amsdu-threshold (*integer [0..8192]*; Default: **8192**)
Max frame size to allow including in AMSDU.

ht-basic-mcs (*list of (mcs-0 | mcs-1 | mcs-2 | mcs-3 | mcs-4 | mcs-5 | mcs-6 | mcs-7 | mcs-8 | mcs-9 | mcs-10 | mcs-11 | mcs-12 | mcs-13 | mcs-14 | mcs-15 | mcs-16 | mcs-17 | mcs-18 | mcs-19 | mcs-20 | mcs-21 | mcs-22 | mcs-23)*; Default: **mcs-0; mcs-1; mcs-2; mcs-3; mcs-4; mcs-5; mcs-6; mcs-7**)
Modulation and Coding Schemes [1] that every connecting client must support (refer to 802.11n for MCS specification).

ht-guard-interval (*any | long*; Default: **any**)
Whether to allow use of short guard interval (refer to 802.11n MCS specification to see how this may affect throughput). "any" will use either short or long, depending on data rate, "long" will use long.

ht-rxchains (*list of integer [0..2]*; Default: **0**)
Which antennas to use for receive.

ht-supported-mcs (*list of (mcs-0 | mcs-1 | mcs-2 | mcs-3 | mcs-4 | mcs-5 | mcs-6 | mcs-7 | mcs-8 | mcs-9 | mcs-10 | mcs-11 | mcs-12 | mcs-13 | mcs-14 | mcs-15 | mcs-16 | mcs-17 | mcs-18 | mcs-19 | mcs-20 | mcs-21 | mcs-22 | mcs-23)*; Default: **mcs-0; mcs-1; mcs-2; mcs-3; mcs-4; mcs-5; mcs-6; mcs-7; mcs-8; mcs-9; mcs-10; mcs-11; mcs-12; mcs-13; mcs-14; mcs-15; mcs-16; mcs-17; mcs-18; mcs-19; mcs-20; mcs-21; mcs-22; mcs-23**)
Modulation and Coding Schemes that this device advertises as supported.

ht-txchains (*list of integer [0..2]*; Default: **0**)
Which antetnnas to use for transmit.

hw-fragmentation-threshold (*integer[256..3000] | disabled*; Default: **0**)
Specifies maximum fragment size in bytes when transmitted over wireless medium. 802.11 standard packet (MSDU in 802.11 terminology) fragmentation allows packets to be fragmented before transmiting over wireless medium to increase probability of successful transmission (only fragments that did not transmit correctly are retransmitted). Note that transmission of fragmented packet is less efficient than transmitting unfragmented packet because of protocol overhead and increased resource usage at both - transmitting and receiving party.

hw-protection-mode (*cts-to-self | none | rts-cts*; Default: **none**)
Frame protection support property `read more >>`

hw-protection-threshold (*integer [0..65535]*; Default: **0**)
Frame protection support property `read more >>`

hw-retries (*integer [0..15]*; Default: **7**)
Number of times sending frame is retried without considering it a transmission failure.

Data rate is decreased upon failure and frame is sent again. Three sequential failures on lowest supported rate suspend transmission to this destination for the duration of **on-fail-retry-time**. After that, frame is sent again. The frame is being retransmitted until transmission success, or until client is disconnected after **disconnect-timeout**. Frame can be discarded during this time if **frame-lifetime** is exceeded.

l2mtu (*integer [0..65536]*; Default: **2290**)

mac-address (*MAC*; Default:)

master-interface (*string*; Default:)
Name of wireless interface that has *virtual-ap* capability. Virtual AP interface will only work if master interface is in *ap-bridge*, *bridge* or *wds-slave* mode. This property is only for virtual AP interfaces.

max-station-count (*integer [1..2007]*; Default: **2007**)

Maximum number of associated clients. WDS links also count toward this limit.

mode (*station* | *station-wds* | *ap-bridge* | *bridge* | *alignment-only* | *nstreme-dual-slave* | *wds-slave* | *station-pseudobridge* | *station-pseudobridge-clone* | *station-bridge*; Default: **station**)

Selection between different station and access point (AP) modes. Station modes:

- *station* - Basic station mode. Find and connect to acceptable AP.
- *station-wds* - Same as *station*, but create WDS link with AP, using proprietary extension. AP configuration has to allow WDS links with this device. Note that this mode does not use entries in wds.
- *station-pseudobridge* - Same as *station*, but additionally perform MAC address translation of all traffic. Allows interface to be bridged.
- *station-pseudobridge-clone* - Same as *station-pseudobridge*, but use **station-bridge-clone-mac** address to connect to AP.

AP modes:

- *ap-bridge* - Basic access point mode.
- *bridge* - Same as *ap-bridge*, but limited to one associated client.
- *wds-slave* - Same as *ap-bridge*, but scan for AP with the same **ssid** and establishes WDS link. If this link is lost or cannot be established, then continue scanning. If **dfs-mode** is *radar-detect*, then APs with enabled **hide-ssid** will not be found during scanning.

Special modes:

- *alignment-only* - Put interface in a continuous transmit mode that is used for aiming remote antenna.
- *nstreme-dual-slave* - allow this interface to be used in nstreme-dual setup.

> MAC address translation in **pseudobridge** modes works by inspecting packets and building table of corresponding IP and MAC addresses. All packets are sent to AP with the MAC address used by pseudobridge, and MAC addresses of received packets are restored from the address translation table. There is single entry in address translation table for all non-IP packets, hence more than one host in the bridged network cannot reliably use non-IP protocols. Note: Currently IPv6 doesn't work over Pseudobridge
>
> Virtual AP interfaces do not have this property, they follow the **mode** of their master interface.

mtu (*integer [0..65536]*; Default: **1500**)

multicast-helper (*default* | *disabled* | *full*; Default: **default**)

When set to full multicast packets will be sent with unicast destination MAC address, resolving multicast problem on wireless link. This option should be enabled only on access point, clients should be configured in **station-bridge** mode. Available starting from v5.15.

- disabled - disables the helper and sends multicast packets with multicast destination MAC addresses
- full - all multicast packet mac address are changed to unicast mac addresses prior sending them out
- default - default choice that currently is set to *disabled*. Value can be changed in future releases.

name (*string*; Default:)

name of the interface

noise-floor-threshold (*default* | *integer [-128..127]*; Default: **default**)

This property is only effective for cards based on AR5211 chipset.

nv2-cell-radius (*integer [10..200]*; Default: **30**)

Setting affects the size of contention time slot that AP allocates for clients to initiate connection and also size of time slots used for estimating distance to client. When setting is too small, clients that are farther away may have trouble connecting and/or disconnect with "ranging timeout" error. Although during normal operation the effect of this setting should be negligible, in order to maintain maximum performance, it is advised to not increase this setting if not necessary, so AP is not reserving time that is actually never used, but instead allocates it for actual data transfer.

- on AP: distance to farthest client in km
- on station: no effect

nv2-noise-floor-offset (*default* | *integer [0..20]*; Default: **default**)

nv2-preshared-key (*string*; Default:)

nv2-qos (*default* | *frame-priority*; Default: **default**)

Sets the packet priority mechanism, firstly data from high priority queue is sent, then lower queue priority data until 0 queue priority is reached. When link is full with high priority queue data, lower priority data is not sent. Use it very carefully, setting works on AP

- **frame-priority** - manual setting that can be tuned with Mangle rules.
- **default** - default setting where small packets receive priority for best latency

nv2-queue-count (*integer [2..8]*; Default: **2**)

nv2-security (*disabled* | *enabled*; Default: **disabled**)

on-fail-retry-time (*time [100ms..1s]*; Default: **100ms**)

After third sending failure on the lowest data rate, wait for specified time interval before retrying.

periodic-calibration (*default* | *disabled* | *enabled*; Default: **default**)

Setting *default* enables periodic calibration if info **default-periodic-calibration** property is *enabled*. Value of that property depends on the type of wireless card.

This property is only effective for cards based on Atheros chipset.

periodic-calibration-interval (*integer [1..10000]*; Default: **60**)

This property is only effective for cards based on Atheros chipset.

preamble-mode (*both* | *long* | *short*; Default: **both**)

Short preamble mode is an option of 802.11b standard that reduces per-frame overhead.

- On AP:
  - *long* - Do not use short preamble.
  - *short* - Announce short preamble capability. Do not accept connections from clients that do not have this capability.
  - *both* - Announce short preamble capability.
- On station:
  - *long* - do not use short preamble.
  - *short* - do not connect to AP if it does not support short preamble.
  - *both* - Use short preamble if AP supports it.

prism-cardtype (*100mW* | *200mW* | *30mW*; Default:)

Specify type of the installed Prism wireless card.

proprietary-extension (*post-2.9.25* | *pre-2.9.25*; Default: **post-2.9.25**)

RouterOS includes proprietary information in an information element of management frames. This parameter controls how this information is included.

- *pre-2.9.25* - This is older method. It can interoperate with newer versions of RouterOS. This method is incompatible with some clients, for example, Centrino based ones.
- *post-2.9.25* - This uses standardized way of including vendor specific information, that is compatible with newer wireless clients.

radio-name (*string*; Default: **MAC address of an interface**)

Descriptive name of the device, that is shown in registration table entries on the remote devices.

This is a proprietary extension.

rate-selection (*advanced* | *legacy*; Default: **advanced**)

Starting from v5.9 default value is advanced since legacy mode was inefficient.

rate-set (*configured* | *default*; Default: **default**)

Two options are available:

- *default* - default basic and supported rate sets are used. Values from **basic-rates** and **supported-rates** parameters have no effect.
- *configured* - use values from **basic-rates**, **supported-rates**, **basic-mcs**, **mcs**. Read more >>.

scan-list

(*Comma separated list of frequencies and frequency ranges* | *default*; Default: **default**)

The *default* value is all channels from selected band that are supported by card and allowed by the **country** and **frequency-mode** settings (this list can be seen in info). For default scan list in *5ghz* band channels are taken with 20MHz step, in *5ghz-turbo* band - with 40MHz step, for all other bands - with 5MHz step. If **scan-list** is specified manually, then all matching channels are taken. (Example: **scan-list**=*default,5200-5245,2412-2427* - This will use the default value of scan list for current band, and add to it supported frequencies from 5200-5245 or 2412-2427 range.)

security-profile (*string*; Default: **default**)

Name of profile from security-profiles

ssid (*string (0..32 chars)*; Default: **value of system/identity**)

SSID (service set identifier) is a name that identifies wireless network.

station-bridge-clone-mac (*MAC*; Default:)

This property has effect only in the *station-pseudobridge-clone* mode.

Use this MAC address when connection to AP. If this value is *00:00:00:00:00:00*, station will initially use MAC address of the wireless interface.

As soon as packet with MAC address of another device needs to be transmitted, station will reconnect to AP using that address.

supported-rates-a/g (*list of rates [12Mbps* | *18Mbps* | *24Mbps* | *36Mbps* | *48Mbps* | *54Mbps* | *6Mbps* | *9Mbps]*; Default: **6Mbps; 9Mbps; 12Mbps; 18Mbps; 24Mbps; 36Mbps; 48Mbps; 54Mbps**)

List of supported rates, used for all bands except *2ghz-b*.

supported-rates-b (*list of rates [11Mbps* | *1Mbps* | *2Mbps* | *5.5Mbps]*; Default: **1Mbps; 2Mbps; 5.5Mbps; 11Mbps**)

List of supported rates, used for *2ghz-b*, *2ghz-b/g* and *2ghz-b/g/n* bands. Two devices will communicate only using rates that are supported by both devices. This property has effect only when value of **rate-set** is *configured*.

tdma-debug (*integer [0..4294967295]*; Default: **0**)

tdma-hw-test-mode (*integer [0..4294967295]*; Default:)

tdma-override-rate (*12mbps* | *18mbps* | *24mbps* | *36mbps* | *48mbps* | *54mbps* | *6mbps* | *9mbps* | *disabled* | *ht20-mcs...* | *ht40-mcs...*; Default: **disabled**)

tdma-override-size (*integer [0..4294967295]*; Default:)

tdma-period-size (*integer [1..10]*; Default: **2**)

Specifies TDMA period in milliseconds. It could help on the longer distance links, it could slightly increase bandwidth, while latency is increased too.

tdma-test-mode (*integer [0..4294967295]*; Default: **0**)

tx-power (*integer [-30..30]*; Default:)

tx-power-mode (*default, card-rates, all-rated-fixed, manual-table*; Default: **default**)

sets up tx-power mode for wireless card

- default - use values stored in the card
- card-rates - use transmit power as defined by **tx-power** setting
- all-rated-fixed - use same transmit power for all data rates. Can damage the card if transmit power is set above rated value of the card for used rate
- manual-table - define transmit power for each rate separately. Can damage the card if transmit power is set above rated value of the card for used rate.

update-stats-interval (; Default:)

How often to request update of signals strength and ccq values from clients. Access to registration-table also triggers update of these values.

This is proprietary extension.

wds-cost-range (*start [-end] integer[0..4294967295]*; Default: **50-150**)

Bridge port cost of WDS links are automatically adjusted, depending on measured link throughput. Port cost is recalculated and adjusted every 5 seconds if it has changed by more than 10%, or if more than 20 seconds have passed since the last adjustment.

Setting this property to *0* disables automatic cost adjustment. Automatic adjustment does not work for WDS links that are manually configured as a bridge port.

wds-default-bridge (*string* | *none*; Default: **none**)

When WDS link is established and status of the wds interface becomes *running*, it will be added as a bridge port to the bridge interface specified by this property. When WDS link is lost, wds interface is removed from the bridge. If wds interface is already included in a bridge setup when WDS link becomes active, it will not be added to bridge specified by , and will **(needs editing)**

wds-default-cost (*integer [0..4294967295]*; Default: **100**)

Initial bridge port cost of the WDS links.

wds-ignore-ssid (*yes* | *no*; Default: **no**)

By default, WDS link between two APs can be created only when they work on the same frequency and have the same SSID value. If this property is set to *yes*, then SSID of the remote AP will not be checked. This property has no effect on connections from clients in *station-wds* mode. It also does not work if **wds-mode** is *static-mesh* or *dynamic-mesh*.

wds-mode (*disabled* | *dynamic* | *dynamic-mesh* | *static* | *static-mesh*; Default: **disabled**)

Controls how WDS links with other devices (APs and clients in *station-wds* mode) are established.

- *disabled* does not allow WDS links.
- *static* only allows WDS links that are manually configured in wds
- *dynamic* also allows WDS links with devices that are not configured in wds, by creating required entries dynamically. Such dynamic WDS entries are removed automatically after the connection with the other AP is lost.

  -mesh modes use different (better) method for establishing link between AP, that is not compatible with APs in non-mesh mode. This method avoids one-sided WDS links that are created only by one of the two APs. Such links cannot pass any data.

  When AP or station is establishing WDS connection with another AP, it uses connect-list to check whether this connection is allowed. If station in *station-wds* mode is establishing connection with AP, AP uses access-list to check whether this connection is allowed.

  If **mode** is *station-wds*, then this property has no effect.

wireless-protocol (*802.11* | *any* | *nstreme* | *nv2* | *nv2-nstreme* | *nv2-nstreme-802.11* | *unspecified*; Default: **unspecified**) — Specifies protocol used on wireless interface;

- *unspecified* - protocol mode used on previous RouterOS versions (v3.x, v4.x). Nstreme is enabled by old enable-nstreme setting, Nv2 configuration is not possible.
- *any* : on AP - regular 802.11 Access Point or Nstreme Access Point; on station - selects Access Point without specific sequence, it could be changed by connect-list rules.
- *nstreme* - enables Nstreme protocol (the same as old enable-nstreme setting).
- *nv2* - enables Nv2 protocol.
- *nv2 nstreme* : on AP - uses first wireless-protocol setting, always Nv2; on station - searches for Nv2 Access Point, then for Nstreme Access Point.
- *nv2 nstreme 802.11* - on AP - uses first wireless-protocol setting, always Nv2; on station - searches for Nv2 Access Point, then for Nstreme Access Point, then for regular 802.11 Access Point.

wmm-support (*disabled* | *enabled* | *required*; Default: **disabled**) — Specifies whether to enable WMM.

Basic and MCS Rate table

Default basic and supported rates, depending on selected band

band	basic rates	basic-mcs	mcs	supported rates
2.4ghz-b	1	-	-	1-11
2.4ghz-onlyg	6	-	-	1-11,6-54
2.4ghz-onlyn	6	0-7	0-23	1-11,6-54
2.4ghz-b/g	1-11	-	-	1-11,6-54
2.4ghz-b/g/n	1-11	none	0-23	1-11,6-54
2.4ghz-g-turbo	6	-	-	6-54
5ghz-a	6	-	-	6-54
5ghz-a/n	6	none	0-23	6-54
5ghz-onlyn	6	0-7	0-23	6-54

Used settings when **rate-set=configured**

band	used settings
2.4ghz-b	basic-b, supported-b
2.4ghz-b/g, 2.4ghz-onlyg	basic-b, supported-b, basic-a/g, supported-a/g
2.4ghz-onlyn, 2.4ghz-b/g/n	basic-b, supported-b, basic-a/g, supported-a/g, basic-mcs, supported-mcs
5ghz-a	basic-a/g,supported-a/g
5ghz-a/n, 5ghz-onlyn	basic-a/g,supported-a/g,basic-mcs,supported-mcs

Settings independent from **rate-set**:

1. allowed mcs depending on number of chains:
   - 1 chain: 0-7
   - 2 chains: 0-15
   - 3 chains: 0-23

2. if standard channel width (20Mhz) is not used, then 2ghz modes (except 2.4ghz-b) are not using b rates (1-11)

Frame protection support (RTS/CTS)

802.11 standard provides means to protect transmission against other device transmission by using RTS/CTS protocol. Frame protection helps to fight "hidden node" problem. There are several types of protection:

- RTS/CTS based protection - device willing to send frame at first sends RequestToSend frame and waits for ClearToSend frame from intended destination. By "seeing" RTS or CTS frame 802.11 compliant devices know that somebody is about to transmit and therefore do not initiate transmission themselves
- "CTS to self" based protection - device willing to send frame sends CTS frame "to itself". As in RTS/CTS protocol every 802.11 compliant device receiving this frame know not to transmit. "CTS to self" based protection has less overhead, but it must be taken into account that this only protects against devices receiving CTS frame (e.g. if there are 2 "hidden" stations, there is no use for them to use "CTS to self" protection, because they will not be able to receive CTS sent by other station - in this case stations must use RTS/CTS so that other station knows not to transmit by seeing CTS transmitted by AP).

Protection mode is controlled by **hw-protection-mode** setting of wireless interface. Possible values: **none** - for no protection (default), **rts-cts** for RTS/CTS based protection or **cts-to-self** for "CTS to self" based protection.

Frame size threshold at which protection should be used is controlled by **hw-protection-threshold** setting of wireless interface.

For example, to enable "CTS-to-self" based frame protection on AP for all frames, not depending on size, use command:

```
[admin@MikroTik] /interface wireless> set 0 hw-protection-mode=cts-to-self hw-protection-threshold=0
```

To enable RTS/CTS based protection on client use command:

```
[admin@MikroTik] /interface wireless> set 0 hw-protection-mode=rts-cts hw-protection-threshold=0
```

Nv2

MikroTik has developed a new wireless protocol based on TDMA technology (Time Division Multiple Access) - (Nstreme version 2). See the Nv2 documentation: NV2

TDMA is a channel access method for shared medium networks. It allows several users to share the same frequency channel by dividing the signal into different time slots. The users transmit in rapid succession, one after the other, each using his own time slot. This allows multiple stations to share the same transmission medium (e.g. radio frequency channel) while using only a part of its channel capacity.

The most important benefits of Nv2 are:

- Increased speed
- More client connections in PTM environments
- Lower latency
- No distance limitations
- No penalty for long distances

Starting from RouterOS v5.0beta5 you can configure Nv2 in the Wireless menu. Please take a look at the NV2 protocol implementation status. Nv2 protocol limit is 511 clients.

Nv2 Troubleshooting

Increase throughput on long distance with **tdma-period-size**. In Every "period", the Access Point leaves part of the time unused for data transmission (which is equal to *round trip time* - the time in which the frame can be sent and received from the client), it is used to ensure that client could receive the last frame from Access Point, before sending it's own packets to it. The longer the distance, the longer the period is unused.

For example, the distance between Access Point and client is 30km. Frame is sent in 100us one direction, respectively round-trip-time is ~200us. **tdma-period-size** default value is 2ms, it means 10% of the time is unused. When tdma-period-size is increased to 4ms, only 5% of time is unused. For 60km wireless link, round-trip-time is 400ms, unused time is 20% for default tdma-period-size 2ms, and 10% for 4ms. Bigger tdma-period-size value increases latency on the link.

Access List

Sub-menu: `/interface wireless access-list`

Access list is used by access point to restrict allowed connections from other devices, and to control connection parameters.

Operation:

- Access list rules are checked sequentially.
- Disabled rules are always ignored.
- Only the first matching rule is applied.
- If there are no matching rules for the remote connection, then the default values from the wireless interface configuration are used.
- If remote device is matched by rule that has **authentication=***no* value, the connection from that remote device is rejected.

Warning: If there is no entry in ACL about client which connects to AP (wireless,debug wlan2: A0:0B:BA:D7:4D:B2 not in local ACL, by default accept), then ACL for this client is ignored during all connection time.

For example, if client's signal during connection is -41 and we have ACL rule

```
/interface wireless access-list
add authentication=no forwarding=no interface=wlan2 signal-range=..-55
```

Then connection is not matched to any ACL rule and if signal drops to -70..-80, client will not be disconnected.

To make it work correctly it is required that client is matched by any of ACL rules.

If we modify ACL rules in previous example to:

```
/interface wireless access-list
add interface=wlan2 signal-range=-55
add authentication=no forwarding=no interface=wlan2 signal-range=..-56
```

Then if signal drops to -56, client will be disconnected.

Properties

Property	Description
ap-tx-limit (*integer [0..4294967295]*; Default: **0**)	Limit rate of data transmission to this client. Value *0* means no limit. Value is in bits per second.
authentication (*yes* | *no*; Default: **yes**)	. • *no* - Client association will always fail. • *yes* - Use authentication procedure that is specified in the **security-profile** of the interface.
client-tx-limit (*integer [0..4294967295]*; Default: **0**)	Ask client to limit rate of data transmission. Value *0* means no limit. This is a proprietary extension that is supported by RouterOS clients. Value is in bits per second.
comment (*string*; Default:)	Short description of an entry
disabled (*yes* | *no*; Default: **no**)	
forwarding (*yes* | *no*; Default: **yes**)	. • *no* - Client cannot send frames to other station that are connected to same access point. • *yes* - Client can send frames to other stations on the same access point.
interface (*string* | *all*; Default: **all**)	Rules with **interface**=*all* are used for all wireless interfaces. To make rule that applies only to one wireless interface, specify that interface as a value of this property.
mac-address (*MAC*; Default: **00:00:00:00:00:00**)	Rule matches client with the specified MAC address. Value *00:00:00:00:00:00* matches always.
management-protection-key (*string*; Default: **""**)	
private-algo (*104bit-wep* | *40bit-wep* | *aes-ccm* | *none* | *tkip*; Default: **none**)	Only for WEP modes.
private-key (*string*; Default: **""**)	Only for WEP modes.
private-pre-shared-key (*string*; Default: **""**)	Used in WPA PSK mode.
signal-range (*NUM..NUM - both NUM are numbers in the range -120..120*; Default: **-120..120**)	Rule matches if signal strength of the station is within the range. If signal strength of the station will go out of the range that is specified in the rule, access point will disconnect that station.
time (*TIME-TIME,sun,mon,tue,wed,thu,fri,sat - TIME is time interval 0..86400 seconds; all day names are optional; value can be unset*; Default:)	Rule will match only during specified time. Station will be disconnected after specified time ends. Both start and end time is expressed as time since midnight, 00:00. Rule will match only during specified days of the week.

Align

Sub-menu: `/interface wireless align`

Property	Description
active-mode (*yes* | *no*; Default: **yes**)	If in active mode, station will send out frames for align.
audio-max (*integer [-2147483648..2147483647]*; Default: **-20**)	Maxumum signal strength for beeper
audio-min (*integer [-2147483648..2147483647]*; Default: **-100**)	Minimum signal strength for beeper
audio-monitor (*MAC*; Default: **00:00:00:00:00:00**)	Which MAC address to use for audio monitoring
filter-mac (*MAC*; Default: **00:00:00:00:00:00**)	Filtered out MAC address that will be shown in monitor screen.
frame-size (*integer [200..1500]*; Default: **300**)	Size of the frames used by monitor.
frames-per-second (*integer [1..100]*; Default: **25**)	Frame transmit interval
receive-all (*yes* | *no*; Default: **no**)	If set to "no", monitoring will work only if both wireless stations are in align mode.
ssid-all (*yes* | *no*; Default: **no**)	Whether to show all SSIDs in the monitor or only one configured in wireless settings.

Menu Specific Commands

Property	Description
monitor (*interface name*)	Start align monitoring
test-audio (*integer [-2147483648..2147483647]*)	Test the beeper

Connect List

Sub-menu: `/interface wireless connect-list`

connect-list is used to assign priority and security settings to connections with remote access points, and to restrict allowed connections. connect-list is an ordered list of rules. Each rule in connect-list is attached to specific wireless interface, specified in the `interface` property of that rule (this is unlike access-list, where rules can apply to all interfaces). Rule can match MAC address of remote access point, it's signal strength and many other parameters.

Operation:

- connect-list rules are always checked sequentially, starting from the first.
- disabled rules are always ignored.
- Only the first matching rule is applied.
- If connect-list does not have any rule that matches remote access point, then the default values from the wireless interface configuration are used.
- If access point is matched by rule that has **connect=***no* value, connection with this access point will not be attempted.
- If access point is matched by rule that has **connect=***yes* value, connection with this access point will be attempted.
  - In station mode, if several remote access points are matched by connect list rules with **connect=***yes* value, connection will be attempted with access point that is matched by rule higher in the connect-list.
  - If no remote access points are matched by connect-list rules with **connect=***yes* value, then value of **default-authentication** interface property determines whether station will attempt to connect to any access point. If **default-authentication=***yes*, station will choose access point with best signal and compatible security.
- In access point mode, connect-list is checked before establishing WDS link with remote device. If access point is not matched by any rule in the connect list, then the value of **default-authentication** determines whether WDS

link will be established.

Properties

Property	Description
area-prefix (*string*; Default:)	Rule matches if area value of AP (a proprietary extension) begins with specified value.**area** value is a proprietary extension.
comment (*string*; Default:)	Short description of an entry
connect (*yes* | *no*; Default: **yes**)	Available options: • *yes* - Connect to access point that matches this rule. • *no* - Do not connect to any access point that matches this rule.
disabled (*yes* | *no*; Default: **no**)	
mac-address (*MAC*; Default: **00:00:00:00:00:00**)	Rule matches only AP with the specified MAC address. Value *00:00:00:00:00:00* matches always.
security-profile (*string* | *none*; Default: **none**)	Name of security profile that is used when connecting to matching access points, If value of this property is *none*, then security profile specified in the interface configuration will be used. In station mode, rule will match only access points that can support specified security profile. Value *none* will match access point that support security profile that is specified in the interface configuration. In access point mode value of this property will not be used to match remote devices.
signal-range (*NUM..NUM - both NUM are numbers in the range -120..120*; Default: **-120..120**)	Rule matches if signal strength of the access point is within the range. If station establishes connection to access point that is matched by this rule, it will disconnect from that access point when signal strength goes out of the specified range.
ssid (*string*; Default: "")	Rule matches access points that have this SSID. Empty value matches any SSID. This property has effect only when station mode interface **ssid** is empty, or when access point mode interface has **wds-ignore-ssid**=*yes*
wireless-protocol (*802.11* | *any* | *nstreme* | *tdma*; Default: **any**)	
interface (*string*; Default:)	Each rule in connect list applies only to one wireless interface that is specified by this setting.

Usage

Restrict station connections only to specific access points

- Set value of **default-authentication** interface property to *no*.

```
/interface wireless set station-wlan default-authentication=no
```

- Create rules that matches allowed access points. These rules must have **connect**=*yes* and **interface** equal to the name of station wireless interface.

```
/interface wireless connect-list add interface=station-wlan
connect=yes mac-address=00:11:22:33:00:01
```

/interface wireless connect-list add interface=station-wlan connect=yes mac-address=00:11:22:33:00:02

Disallow connections to specific access points

- Set value of **default-authentication** interface property to *yes*.

```
/interface wireless set station-wlan default-authentication=yes
```

- Create **connect**=*no* rules that match those access points that station should not connect to. These rules must have **connect**=*no* and **interface** equal to the name of station wireless interface.

```
/interface wireless connect-list add interface=station-wlan
connect=no mac-address=00:11:22:33:44:55
```

Select preferred access points

- Create rules that match preferred access points. These rules must have **connect**=*yes* and **interface** equal to the name of station wireless interface.
- Put rules that match preferred access points higher in the connect-list, in the order of preference.

Restrict WDS link establishment

- Place rules that match allowed access points at the top.
- Add deny-all rule at the end of connect list.

Info

Sub-menu: `/interface wireless info`

Property	Description
2ghz-10mhz-power-channels ()	
2ghz-11n-channels ()	
2ghz-5mhz-power-channels ()	
2ghz-b-channels ()	
2ghz-g-channels ()	
2ghz-g-turbo-channels ()	
5ghz-10mhz-power-channels ()	
5ghz-11n-channels ()	
5ghz-5mhz-power-channels ()	
5ghz-channels ()	
5ghz-turbo-channels ()	
capabilities ()	
chip-info ()	
default-periodic-calibration ()	
firmware ()	
ht-chains ()	
interface-type ()	
name ()	
pci-info ()	
supported-bands ()	

Manual TX Power Table

Sub-menu: `/interface wireless manual-tx-power-table`

Property	Description
comment (*string*; Default:)	Short description of an entry
manual-tx-powers (*list of [Rate:TxPower]; Rate ::= 11Mbps | 12Mbps | 18Mbps | 1Mbps | 24Mbps | ... TxPower ::= integer [-30..30]*; Default:)	
name (*string*)	Name of the wireless interface to which tx powers will be applied.

Nstreme

Sub-menu: `/interface wireless nstreme`

This menu allows to switch a wireless card to the nstreme mode. In this case the card will work only with nstreme clients.

Property	Description
comment (*string*; Default:)	Short description of an entry
disable-csma (*yes* | *no*; Default: **no**)	Disable CSMA/CA when polling is used (better performance)
enable-nstreme (*yes* | *no*; Default: **no**)	Whether to switch the card into the nstreme mode
enable-polling (*yes* | *no*; Default: **yes**)	Whether to use polling for clients
framer-limit (*integer [100..4000]*; Default: **3200**)	Maximal frame size
framer-policy (*best-fit* | *dynamic-size* | *exact-size* | *none*; Default: **none**)	The method how to combine frames. A number of frames may be combined into a bigger one to reduce the amount of protocol overhead (and thus increase speed). The card is not waiting for frames, but in case a number of packets are queued for transmitting, they can be combined. There are several methods of framing: • **none** - do nothing special, do not combine packets (framing is disabled) • **best-fit** - put as much packets as possible in one frame, until the framer-limit limit is met, but do not fragment packets • **exact-size** - put as much packets as possible in one frame, until the framer-limit limit is met, even if fragmentation will be needed (best performance) • **dynamic-size** - choose the best frame size dynamically
name (*string*)	Name of an interface, to which setting will be applied. Read only.

Note: The settings here (except for enabling nstreme) are relevant only on Access Point, they are ignored for client devices! The client automatically adapts to the AP settings.
WDS for Nstreme protocol requires using station-wds mode on one of the peers. Configurations with WDS between AP modes (bridge and ap-bridge) will not work.

Nstreme Dual

Sub-menu: `/interface wireless nstreme-dual`

Two radios in nstreme-dual-slave mode can be grouped together to make nstreme2 Point-to-Point connection. To put wireless interfaces into a nstreme2 group, you should set their mode to nstreme-dual-slave. Many parameters from /interface wireless menu are ignored, using the nstreme2, except:

- frequency-mode
- country
- antenna-gain
- tx-power
- tx-power-mode
- antenna-mode

Property	Description
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; Default: **enabled**)	Read more >>
comment (*string*; Default:)	Short description of an entry
disable-csma (*yes* | *no*; Default: **no**)	Disable CSMA/CA (better performance)
disable-running-check (*yes* | *no*; Default: **no**)	Whether the interface should always be treated as running even if there is no connection to a remote peer
disabled (*yes* | *no*; Default: **yes**)	
framer-limit (*integer [64..4000]*; Default: **2560**)	Maximal frame size
framer-policy (*best-fit* | *exact-size* | *none*; Default: **none**)	The method how to combine frames. A number of frames may be combined into one bigger one to reduce the amout of protocol overhead (and thus increase speed). The card are not waiting for frames, but in case a number packets are queued for transmitting, they can be combined. There are several methods of framing: • **none** - do nothing special, do not combine packets • **best-fit** - put as much packets as possible in one frame, until the framer-limit limit is met, but do not fragment packets • **exact-size** - put as much packets as possible in one frame, until the framer-limit limit is met, even if fragmentation will be needed (best performance)
ht-channel-width (*2040mhz* | *20mhz* | *40mhz*; Default: **20mhz**)	
ht-guard-interval (*both* | *long* | *short*; Default: **long**)	
ht-rates (*list of rates [1,2,3,4,5,6,7,8]*; Default: **1,2,3,4,5,6,7,8**)	
ht-streams (*both* | *double* | *single*; Default: **single**)	
l2mtu (*integer [0..65536]*; Default:)	
mtu (*integer [0..65536]*; Default: **1500**)	
name (*string*; Default:)	Name of an entry
rates-a/g (*list of rates [6Mbps,9Mbps, 12Mbps, 18Mbps, 24Mbps, 36Mbps, 48Mbps, 54Mbps]*; Default: **6Mbps,9Mbps,12Mbps, 18Mbps, 24Mbps, 36Mbps, 48Mbps, 54Mbps**)	Rates to be supported in 802.11a or 802.11g standard
rates-b (*list of rates [1Mbps, 2Mbps, 5.5Mbps, 11Mbps]*; Default: **1Mbps, 2Mbps, 5.5Mbps, 11Mbps**)	Rates to be supported in 802.11b standard

remote-mac (*MAC*; Default: **00:00:00:00:00:00**)	Which MAC address to connect to (this would be the remote receiver card's MAC address)
rx-band (*2ghz-b* | *2ghz-g* | *2ghz-n* | *5ghz-a* | *5ghz-n*; Default:)	Operating band of the receiving radio
rx-channel-width (*10mhz*; Default: **20mhz**)	
rx-frequency (*integer [0..4294967295]*; Default:)	RX card operation frequency in Mhz.
rx-radio (*string*; Default:)	Name of the interface used for receive.
tx-band (*2ghz-b* | *2ghz-g* | *2ghz-n* | *5ghz-a* | *5ghz-n*; Default:)	Operating band of the transmitting radio
tx-channel-width (*10mhz*; Default: **20mhz**)	
tx-frequency (*integer [0..4294967295]*; Default:)	TX card operation frequency in Mhz.
tx-radio (*string*; Default:)	Name of the interface used for transmit.

Warning: WDS cannot be used on Nstreme-dual links.

Note: The difference between tx-freq and rx-freq should be about 200MHz (more is recommended) because of the interference that may occur!

Note: You can use different bands for rx and tx links. For example, transmit in 2ghz-g and receive data, using 2ghz-b band.

Registration Table

Sub-menu: `/interface wireless registration-table`

In the registration table you can see various information about currently connected clients. It is used only for Access Points.

All properties are read-only.

Property	Description
802.1x-port-enabled (*yes* | *no*)	whether the data exchange is allowed with the peer (i.e., whether 802.1x authentication is completed, if needed)
ack-timeout (*integer*)	current value of ack-timeout
ap (*yes* | *no*)	Shows whether registered device is configured as access point.
ap-tx-limit (*integer*)	transmit rate limit on the AP, in bits per second
authentication-type ()	authentication method used for the peer
bridge (*yes* | *no*)	
bytes (*integer* , *integer*)	number of sent and received packet bytes
client-tx-limit (*integer*)	transmit rate limit on the AP, in bits per second

comment (*string*)	Description of an entry. comment is taken from appropriate Access List entry if specified.
compression (*yes* | *no*)	whether data compresson is used for this peer
distance (*integer*)	
encryption (*aes-ccm* | *tkip*)	unicast encryption algorithm used
evm-ch0 ()	
evm-ch1 ()	
evm-ch2 ()	
frame-bytes (*integer,integer*)	number of sent and received data bytes excluding header information
frames (*integer,integer*)	Number of frames that need to be sent over wireless link. This value can be compared to **hw-frames** to check wireless retransmits. `Read more >>`
framing-current-size (*integer*)	current size of combined frames
framing-limit (*integer*)	maximal size of combined frames
framing-mode ()	the method how to combine frames
group-encryption ()	group encryption algorithm used
hw-frame-bytes (*integer,integer*)	number of sent and received data bytes including header information
hw-frames (*integer,integer*)	Number of frames sent over wireless link by the driver. Tihs value can be compared to **frames** to check wireless retransmits. `Read more >>`
interface (*string*)	Name of the wireless interface to which wireless client is associated
last-activity (*time*)	last interface data tx/rx activity
last-ip (*IP Address*)	IP address found in the last IP packet received from the registered client
mac-address (*MAC*)	MAC address of the registered client
management-protection (*yes* | *no*)	
nstreme (*yes* | *no*)	Shows whether nstreme is enabled
p-throughput (*integer*)	estimated approximate throughput that is expected to the given peer, taking into account the effective transmit rate and hardware retries. Calculated once in 5 seconds
packed-bytes (*integer, integer*)	number of bytes packed into larger frames for transmitting/receiving (framing)
packed-frames (*integer, integer*)	number of frames packed into larger ones for transmitting/receiving (framing)
packets (*integer.integer*)	number of sent and received network layer packets
radio-name (*string*)	radio name of the peer
routeros-version (*string*)	RouterOS version of the registered client
rx-ccq ()	Client Connection Quality (CCQ) for receive. `Read more >>`
rx-rate (*integer*)	receive data rate
signal-strength (*integer*)	average strength of the client signal recevied by the AP
signal-strength-ch0 ()	
signal-strength-ch1 ()	
signal-strength-ch2 ()	
signal-to-noise ()	

strength-at-rates ()	signal strength level at different rates together with time how long were these rates used
tdma-retx ()	
tdma-rx-size ()	
tdma-timing-offset ()	tdma-timing-offset is proportional to **distance** and is approximately two times the propagation delay. AP measures this so that it can tell clients what offset to use for their transmissions - clients then subtract this offset from their target transmission time such that propagation delay is accounted for and transmission arrives at AP when expected. You may occasionally see small negative value (like few usecs) there for close range clients because of additional unaccounted delay that may be produced in transmitter or receiver hardware that varies from chipset to chipset.
tdma-tx-size (*integer*)	Value in bytes that specifies the size of data unit whose loss can be detected (data unit over which CRC is calculated) sent by device. In general - the bigger the better, because overhead is less. On the other hand, small value in this setting can not always be considered a signal that connection is poor - if device does not have enough pending data that would enable it to use bigger data units (e.g. if you are just pinging over link), this value will not go up.
tdma-windfull ()	
tx-ccq ()	Client Connection Quality (CCQ) for transmit. `Read more >>`
tx-evm-ch0 ()	
tx-evm-ch1 ()	
tx-evm-ch2 ()	
tx-frames-timed-out ()	
tx-rate ()	
tx-signal-strength ()	
tx-signal-strength-ch0 ()	
tx-signal-strength-ch1 ()	
tx-signal-strength-ch2 ()	
uptime (*time*)	time the client is associated with the access point
wds (*yes* | *no*)	whether the connected client is using wds or not
wmm-enabled (*yes* | *no*)	Shows whether WMM is enabled.

Security Profiles

Sub-menu: `/interface wireless security-profiles`

Security profiles are configured under the **/interface wireless security-profiles** path in the console, or in the "Security Profiles" tab of the "Wireless" window in the WinBox. Security profiles are referenced by the wireless interface security-profile parameter and security-profile parameter of the connect lists.

Basic properties

- **mode** (one of *none*, *static-keys-optional*, *static-keys-required* or *dynamic-keys*; default value: *none*) :
  - *none* - Encryption is not used. Encrypted frames are not accepted.
  - *static-keys-required* - WEP mode. Do not accept and do not send unencrypted frames.
    Station in *static-keys-required* mode will not connect to an access point in *static-keys-optional* mode.

- *static-keys-optional* - WEP mode. Support encryption and decryption, but allow also to receive and send unencrypted frames. Device will send unencrypted frames if encryption algorithm is specified as *none*. Station in *static-keys-optional* mode will not connect to an access point in *static-keys-required* mode. See also: **static-sta-private-algo**, **static-transmit-key**
- *dynamic-keys* - WPA mode.

- **name** : see generic properties

WPA properties

These properties have effect only when **mode=***dynamic-keys*.

- **authentication-types** (multiple choice of *wpa-psk*, *wpa2-psk*, *wpa-eap* and *wpa2-eap*; default value is empty) : Set of supported authentication types. Access point will advertise supported authentication types, and client will connect to access point only if supports any of the advertised authentication types.
- **unicast-ciphers** (multiple choice of *tkip*, *aes-ccm*; default value is empty) : Access point advertises that it supports specified ciphers. Client attempts connection only to access points that supports at least one of the specified ciphers.

  One of the ciphers will be used to encrypt unicast frames that are sent between access point and station.
- **group-ciphers** (multiple choice of *tkip*, *aes-ccm*; default value is empty) : Access point advertises one of these ciphers, and uses it to encrypt all broadcast and multicast frames. Client attempts connection only to access points that use one of the specified group ciphers.
  - tkip - Temporal Key Integrity Protocol - encryption protocol, compatible with lagacy WEP equipment, but enhanced to correct some of WEP flaws
  - aes-ccm - more secure WPA encryption protocol, based on the reliable AES (Advanced Encryption Standard). Networks free of WEP legacy should use only this
- **group-key-update** (time interval in the 30s..1h range; default value: *5m*) : Controls how often access point updates group key. This key is used to encrypt all broadcast and multicast frames.

  This property has no effect in station mode.
- **wpa-pre-shared-key**, **wpa2-pre-shared-key** (text) : WPA and WPA2 pre-shared key mode requires all devices in a BSS to have common secret key. Value of this key can be an arbitrary text.

RouterOS also allows to override pre-shared key value for specific clients, using either **private-pre-shared-key** property in the access-list, or the Mikrotik-Wireless-Psk attribute in the RADIUS MAC authentication response. This is an extension.

These properties have effect only when **authentication-types** contains either *wpa-psk* or *wpa2-psk*.

wpa-pre-shared-key is used for *wpa-psk* authentication type. **wpa2-pre-shared-key** is used for *wpa2-psk*.

WPA EAP properties

These properties have effect only when **authentication-types** contains *wpa-eap* or *wpa2-eap*, and **mode=***dynamic-keys*.

- **eap-methods** (array of *eap-tls*, *passthrough*) :
  - *eap-tls* - Use built-in EAP TLS authentication. Both client and server certificates are supported. See description of **tls-mode** and **tls-certificate** properties.
  - *passthrough* - Access point will relay authentication process to the RADIUS server. This value is ignored in station mode.

  Order of values is significant for access point configuration, it is used by access point when offering specified methods to clients.

Example: Access point uses security-profile where **eap-methods=***eap-tls*,*passthrough*:

Access point offers EAP-TLS method to the client.

Client refuses.

Access point starts relaying EAP communication to the radius server.

- **supplicant-identity** (text; default value is same as system/identity of router at the moment of profile creation) : EAP identity that is sent by client at the beginning of EAP authentication. This value is used as a value for User-Name attribute in RADIUS messages sent by RADIUS EAP accounting and RADIUS EAP pass-through authentication.
- **tls-mode** (one of *verify-certificate*, *dont-verify-certificate*, *no-certificates*; default value: *no-certificates*) :
  - *verify-certificate* - Require remote device to have valid certificate. Check that it is signed by known certificate authority. No additional identity verification is done.
    Note: Certificate may include information about time period during which it is valid. If router has incorrect time and date, it may reject valid certificate because router's clock is outside that period.
    See also: certificate configuration.
  - *dont-verify-certificate* - Do not check certificate of the remote device. Access point will not require client to provide certificate.
  - *no-certificates* - Do not use certificates. TLS session is established using 2048 bit anonymous Diffie-Hellman key exchange.

  When using first two modes, remote device has to support one of the "RC4-MD5", "RC4-SHA" or "DES-CBC3-SHA" TLS cipher suites. In the last mode remote device must support "ADH-DES-CBC3-SHA" cipher suite.

  This property has effect only when **eap-methods** contains *eap-tls*.
- **tls-certificate** (*none* or name of certificate; default value: *none*) : Access point always needs certificate when configured with **tls-mode=***verify-certificate*, or **tls-mode=***dont-verify-certificate*. Client needs certificate only if *access point* is configured with **tls-mode=***verify-certificate*. In this case client needs valid certificate that is signed by CA known to the access point.

  This property has effect only if **tls-mode≠***no-certificates*.

  This property has effect only when **eap-methods** contains *eap-tls*.

RADIUS properties

- **radius-mac-authentication** (*yes* or *no*; default value: *no*) : This property affects the way how access point processes clients that are not found in the access-list.
  - *no* - allow or reject client authentication based on the value of **default-authentication** property of the wireless interface.
  - *yes* - Query RADIUS server using MAC address of client as user name. With this setting the value of **default-authentication** has no effect.
- **radius-mac-accounting** (*yes* or *no*; default value: *no*) : **(needs editing)**
- **radius-eap-accounting** (*yes* or *no*; default value: *no*) : **(needs editing)**
- **interim-update** (time interval; default value: *0*) : When RADIUS accounting is used, access point periodically sends accounting information updates to the RADIUS server. This property specifies default update interval that can be overridden by the RADIUS server using Acct-Interim-Interval attribute.
- **radius-mac-format** (one of *XX:XX:XX:XX:XX:XX*, *XXXX:XXXX:XXXX*, *XXXXXX:XXXXXX*, *XX-XX-XX-XX-XX-XX*, *XXXXXX-XXXXXX*, *XXXXXXXXXXXX*, *XX XX XX XX XX XX*; default value: *XX:XX:XX:XX:XX:XX*) : Controls how MAC address of the client is encoded by access point in the User-Name attribute of the MAC authentication and MAC accounting RADIUS requests.

- **radius-mac-mode** (one of *as-username*, *as-username-and-password*; default value: *as-username*) : By default access point uses empty password, when sending Access-Request during MAC authentication. When this property is set to *as-username-and-password*, access point will use the same value for User-Password attribute as for the User-Name attribute.
- **radius-mac-caching** (either *disabled* or time interval; default value: *disabled*) : If this value is set to time interval, the access point will cache RADIUS MAC authentication responses for specified time, and will not contact RADIUS server if matching cache entry already exists. Value *disabled* will disable cache, access point will always contact RADIUS server.

WEP properties

These properties have effect only when **mode** is *static-keys-required* or *static-keys-optional*. See section "Wireless#Statically_configured_WEP_keys".

- **static-key-0**, **static-key-1**, **static-key-2**, **static-key-3** (hexadecimal representation of the key. Length of key must be appropriate for selected algorithm - see section "Statically configured WEP keys; default value is empty) : **(needs editing)**
- **static-algo-0**, **static-algo-1**, **static-algo-2**, **static-algo-3** (one of *none*, *40bit-wep*, *104bit-wep*, *tkip* or *aes-ccm*; default value: *none*) : Encryption algorithm to use with the corresponding key.
- **static-transmit-key** (one of *key-0*, *key-1*, *key-2* or *key-3*; default value: *key-0*) : Access point will use the specified key to encrypt frames for clients that do not use private key. Access point will also use this key to encrypt broadcast and multicast frames.

  Client will use the specified key to encrypt frames if **static-sta-private-algo=***none*.

  If corresponding **static-algo-** property has value *none*, frame will be sent unencrypted (when **mode=***static-keys-optional*) or will not be sent at all (when **mode=***static-keys-required*).
- **static-sta-private-key** (hexadecimal representation of the key. Length of key must be appropriate for selected algorithm - see section "Statically configured WEP keys") : This property is used only in station mode. Access point uses corresponding key either from **private-key** property of access-list, or from Mikrotik-Wireless-Enc-Key attribute in RADIUS Access-Accept MAC authentication response.
- **static-sta-private-algo** (one of *none*, *40bit-wep*, *104bit-wep*, *tkip* or *aes-ccm*) : Encryption algorithm to use with station private key. Value *none* disables use of the private key.

  This property is used only in station mode. Access point has to get corresponding value either from **private-algo** property of access-list, or from Mikrotik-Wireless-Enc-Algo attribute in RADIUS Access-Accept MAC authentication response.

  Station private key replaces key 0 for unicast frames. Station will not use private key to decrypt broadcast frames.

Management frame protection

***Used for**: Deauthentication attack prevention, MAC address cloning issue.*

RouterOS implements proprietary management frame protection algorithm based on shared secret. Management frame protection means that RouterOS wireless device is able to verify source of management frame and confirm that particular frame is not malicious. This feature allows to withstand deauthentication and disassociation attacks on RouterOS based wireless devices.

Management protection mode is configured in security-profile with **management-protection** setting. Possible values are: **disabled** - management protection is disabled (default), **allowed** - use management protection if supported by remote party (for AP - allow both, non-management protection and management protection clients, for client - connect both to APs with and without management protection), **required** - establish association only with

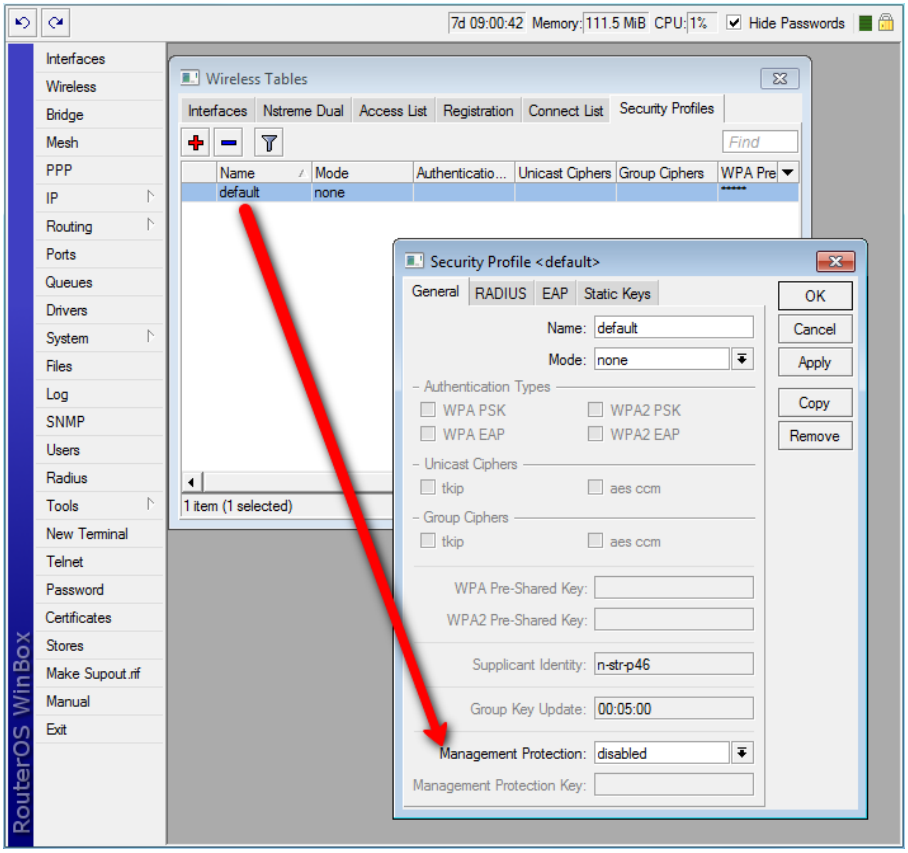
remote devices that support management protection (for AP - accept only clients that support management protection, for client - connect only to APs that support management protection).

Management protection shared secret is configured with security-profile **management-protection-key** setting.

When interface is in AP mode, default management protection key (configured in security-profile) can be overridden by key specified in access-list or RADIUS attribute.

```
[admin@mikrotik] /interface wireless security-profiles> print
 0 name="default" mode=none authentication-types="" unicast-ciphers=""
  group-ciphers="" wpa-pre-shared-key="" wpa2-pre-shared-key=""
  supplicant-identity="n-str-p46" eap-methods=passthrough
  tls-mode=no-certificates tls-certificate=none static-algo-0=none
  static-key-0="" static-algo-1=none static-key-1="" static-algo-2=none
  static-key-2="" static-algo-3=none static-key-3=""
  static-transmit-key=key-0 static-sta-private-algo=none
  static-sta-private-key="" radius-mac-authentication=no
  radius-mac-accounting=no radius-eap-accounting=no interim-update=0s
  radius-mac-format=XX:XX:XX:XX:XX:XX radius-mac-mode=as-username
  radius-mac-caching=disabled group-key-update=5m
  management-protection=disabled management-protection-key=""
```

```
[admin@mikrotik] /interface wireless security-profiles> set default management-protection=
                                                        allowed  disabled  required
```



Operation details

RADIUS MAC authentication

Note: RAIDUS MAC authentication is used by access point for clients that are not found in the access-list, similarly to the **default-authentication** property of the wireless interface. It controls whether client is allowed to proceed with authentication, or is rejected immediately.

When **radius-mac-authentication**=*yes*, access point queries RADIUS server by sending Access-Request with the following attributes:

- User-Name - Client MAC address. This is encoded as specified by the **radius-mac-format** setting. Default encoding is "XX:XX:XX:XX:XX:XX".
- Nas-Port-Id - **name** of wireless interface.
- User-Password - When **radius-mac-mode**=*as-username-and-password* this is set to the same value as User-Name. Otherwise this attribute is empty.
- Calling-Station-Id - Client MAC address, encoded as "XX-XX-XX-XX-XX-XX".
- Called-Station-Id - MAC address and SSID of the access point, encoded as "XX-XX-XX-XX-XX-XX:SSID" (minus separated pairs of MAC address digits, followed by colon, followed by SSID value).
- Acct-Session-Id - Added when **radius-mac-accounting**=*yes*.

When access point receives Access-Accept or Access-Reject response from the RADIUS server, it stores the response and either allows or rejects client. Access point uses following RADIUS attributes from the Access-Accept response:

- Ascend-Data-Rate
- Ascend-Xmit-Rate
- Mikrotik-Wireless-Forward - Same as access-list **forwarding**.
- Mikrotik-Wireless-Enc-Algo - Same as access-list **private-algo**.
- Mikrotik-Wireless-Enc-Key - Same as access-list **private-key**.
- Mikrotik-Wireless-Psk - Same as access-list **private-pre-shared-key**.
- Session-Timeout - Time, after which client will be disconnected.
- Acct-Interim-Interval - Overrides value of **interim-update**.
- Class - If present, value of this attribute is saved and included in Accounting-Request messages.

Caching

Caching of RADIUS MAC authentication was added to support RADIUS authentication for clients that require from the access point very quick response to the association request. Such clients time out before response from RADIUS server is received. Access point caches authentication response for some time and can immediately reply to the repeated association request from the same client.

RADIUS EAP pass-through authentication

When using WPA EAP authentication type, clients that have passed MAC authentication are required to perform EAP authentication before being authorized to pass data on wireless network. With pass-through EAP method the access point will relay authentication to RADIUS server, and use following attributes in the Access-Request RADIUS message:

- User-Name - EAP supplicant identity. This value is configured in the **supplicant-identity** property of the client security profile.
- Nas-Port-Id - **name** of wireless interface.
- Calling-Station-Id - Client MAC address, encoded as "XX-XX-XX-XX-XX-XX".

- Called-Station-Id - MAC address and SSID of the access point, encoded as "XX-XX-XX-XX-XX-XX:SSID" (pairs of MAC address digits separated by minus sign, followed by colon, followed by SSID value).
- Acct-Session-Id - Added when **radius-eap-accounting=***yes*.
- Acct-Multi-Session-Id - MAC address of access point and client, and unique 8 byte value, that is shared for all accounting sessions that share single EAP authentication. Encoded as *AA-AA-AA-AA-AA-AA-CC-CC-CC-CC-CC-CC-XX-XX-XX-XX-XX-XX-XX-XX*.

  Added when **radius-eap-accounting=***yes*.

Access point uses following RADIUS attributes from the Access-Accept server response:

- Class - If present, value of this attribute is saved and included in Accounting-Request messages.
- Session-Timeout - Time, after which client will be disconnected. Additionally, access point will remember authentication result, and if during this time client reconnects, it will be authorized immediately, without repeating EAP authentication.
- Acct-Interim-Interval - Overrides value of **interim-update**.

Statically configured WEP keys

Different algorithms require different length of keys:

- *40bit-wep* - 10 hexadecimal digits (40 bits). If key is longer, only first 40 bits are used.
- *104bit-wep* - 26 hexadecimal digits (104 bits). If key is longer, only first 104 bits are used.
- *tkip* - At least 64 hexadecimal digits (256 bits).
- *aes-ccm* - At least 32 hexadecimal digits (128 bits).

Key must contain even number of hexadecimal digits.

WDS security configuration

WDS links can use all available security features. However, they require careful configuration of security parameters.

It is possible to use one security profile for all clients, and different security profiles for WDS links. Security profile for WDS link is specified in connect-list. Access point always checks connect list before establishing WDS link with another access point, and used security settings from matching connect list entry. WDS link will work when each access point will have connect list entry that matches the other device, has **connect=***yes* and specifies compatible **security-profile**.

WDS and WPA/WPA2

If access point uses security profile with **mode=***dynamic-keys*, then encryption will be used for all WDS links. Since WPA authentication and key exchange is not symmetrical, one of the access points will act as a client for the purpose of establishing secure connection. This is similar to how *static-mesh* and *dynamic-mesh* WDS modes work. Some problems, like single sided WDS link between two incorrectly configured access points that use non-*mesh* mode, is not possible if WPA encryption is enabled. However, non-*mesh* modes with WPA still have other issues (like constant reconnection attempts in case of configuration mismatch) that are solved by use of the -*mesh* WDS modes.

In general, WPA properties on both access points that establish WPA protected WDS link have to match. These properties are **authentication-types**, **unicast-ciphers**, **group-ciphers**. For non-*mesh* WDS mode these properties need to have the same values on both devices. In *mesh* WDS mode each access point has to support the other one as a client.

Theoretically it is possible to use RADIUS MAC authentication and other RADIUS services with WDS links. However, only one access point will interact with the RADIUS server, the other access point will behave as a client.

Implementation of *eap-tls* EAP method in RouterOS is particularly well suited for WDS link encryption. **tls-mode**=*no-certificates* requires no additional configuration, and provides very strong encryption.

WDS and WEP

mode, **static-sta-private-key** and **static-sta-private-algo** parameters in the security profile assigned to the WDS link need to have the same values on both access points that establish WDS link with WPA encryption.

Security profile and access point matching in the connect list

Client uses value of connect-list **security-profile** property to match only those access points that support necessary security.

- **mode**=*static-keys-required* and **mode**=*static-keys-optional* matches only access points with the same **mode** in interface **security-profile**.
- If **mode**=*dynamic-keys*, then connect list entry matches if all of the **authentication-types**, **unicast-ciphers** and **group-ciphers** contain at least one value that is advertised by access point.

Sniffer

Sub-menu: `/interface wireless sniffer`

Wireless sniffer allows to capture frames including Radio header, 802.11 header and other wireless related information.

Property	Description
channel-time (; Default: **200ms**)	
file-limit (*integer [10..4294967295]*; Default: **10**)	Allocated file size in bytes which will be used to store captured data. Applicable if **file-name** is specified.
file-name (*string*; Default:)	Name of the file where to store captured data.
memory-limit (*integer [10..4294967295]*; Default: **10**)	Allocated memory buffer in bytes used to store captured data.
multiple-channels (*yes* | *no*; Default: **no**)	
only-headers (*yes* | *no*; Default: **no**)	If set to yes, then sniffer will capture only information stored in frame headers.
receive-errors (*yes* | *no*; Default: **no**)	
streaming-enabled (*yes* | *no*; Default: **no**)	Whether to stream captured data to specified streaming server
streaming-max-rate (*integer [0..4294967295]*; Default: **0**)	
streaming-server (*IPv4*; Default: **0.0.0.0**)	IP address of the streaming server.

Packets

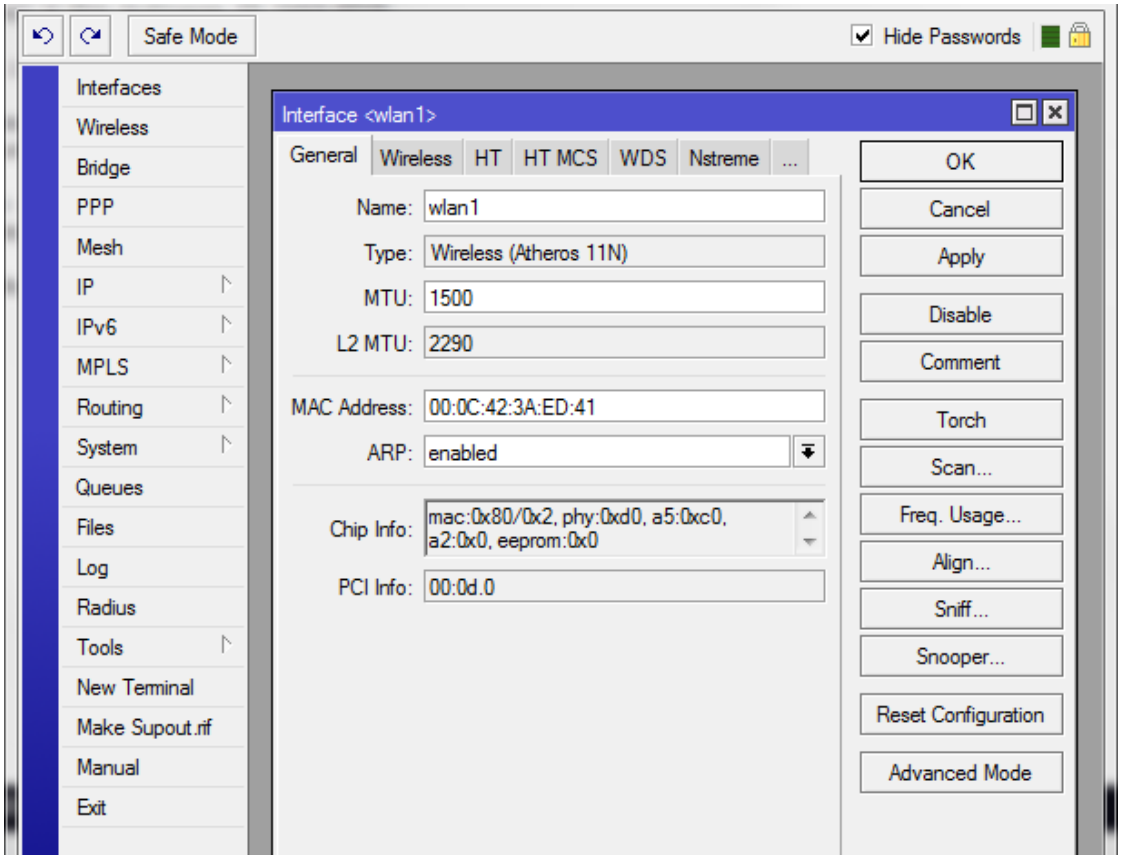
Sub-menu: `/interface wireless sniffer packet`

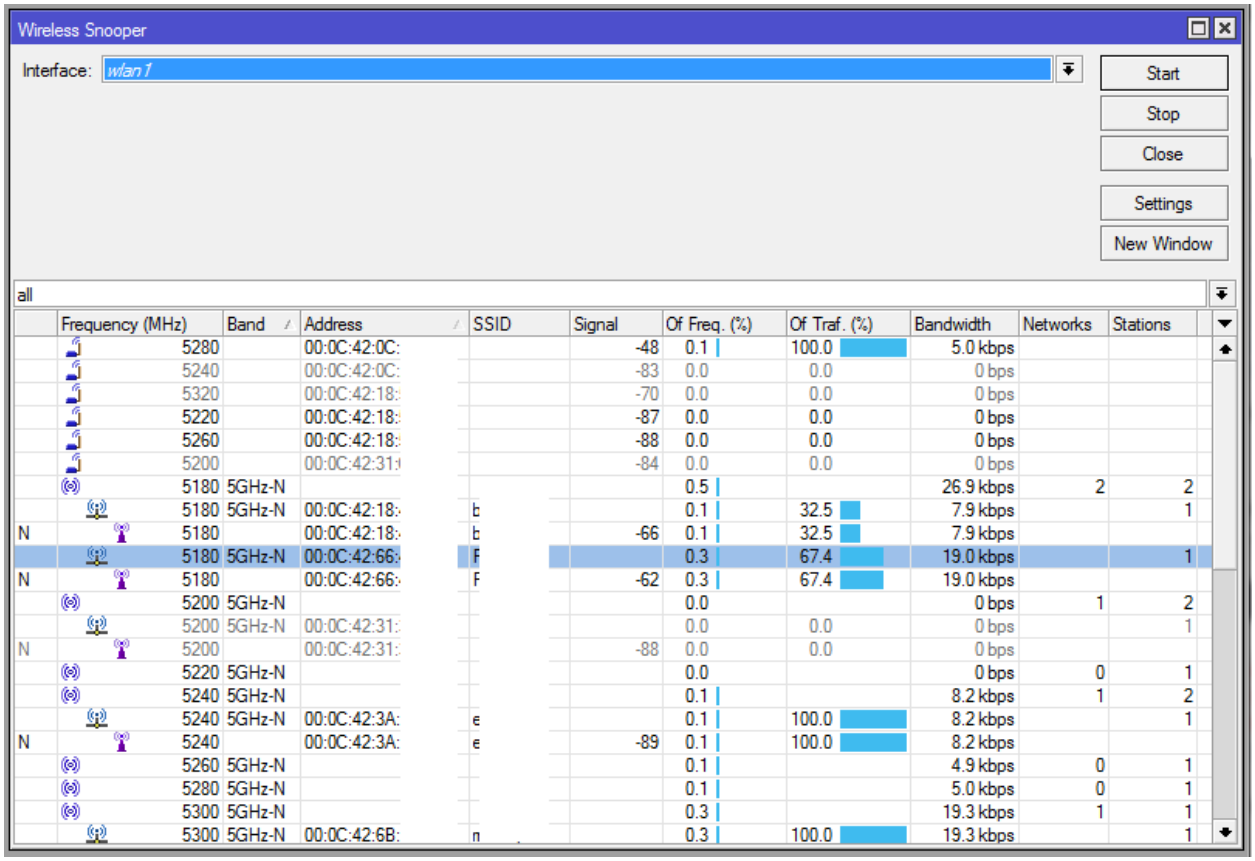
Sub-menu shows captured packets.

Snooper

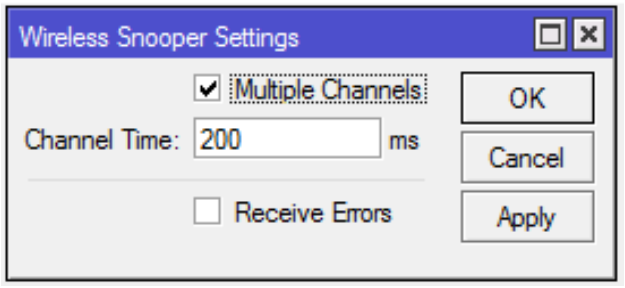
This tool monitors surrounding frequency usage, and displays which devices occupy each frequency. It's available both in console, and also in Winbox.

Sub-menu: `/interface wireless snooper`





Settings



Spectral scan

- See separate document Manual:Spectral_scan

WDS

Sub-menu: `/interface wireless wds`

Properties:

Property	Description
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; Default: **enabled**)	
comment (*string*; Default:)	
disable-running-check (*yes* | *no*; Default: **no**)	
disabled (*yes* | *no*; Default: **yes**)	
l2mtu (*integer [0..65536]*; Default:)	
master-interface (*string*; Default:)	
mtu (*integer [0..65536]*; Default: **1500**)	
name (*string*; Default:)	
wds-address (*MAC*; Default: **00:00:00:00:00:00**)	

Read-only properties:

Property	Description
dynamic (*yes* | *no*)	
mac-address (*MAC*)	
running (*yes* | *no*)	

[Top | Back to Content]

References

[1] http://en.wikipedia.org/wiki/IEEE_802.11n-2009#Data_rates

Manual:System/Watchdog

Applies to RouterOS: v3, v4 +

Summary

This menu allows to configure system to reboot on kernel panic, when an IP address does not respond, or in case the system has locked up. Software watchdog timer is used to provide the last option, so in very rare cases (caused by hardware malfunction) it can lock up by itself. There is a hardware watchdog device available in all RouterBOARD PowerPC and Mipsbe models, which can reboot the system in any case.

Properties

Sub-menu: `/system watchdog`

Property	Description
watch-address (*IP*; Default: **none**)	The system will reboot in case 6 sequental pings to the given IP address (sent once per 10 seconds) will fail. If set to **none** this feature is disabled.
watchdog-timer (*yes* | *no*; Default: **yes**)	Whether to reboot if system is unresponsive for a minute
no-ping-delay (*time*; Default: **5m**)	Specifies how long after reboot not to test and ping watch-address. The default setting means that if watch-address is set and is not reachable, the router will reboot about every 6 minutes.
automatic-supout (*yes* | *no*; Default: **yes**)	When software failure happens, a file named "autosupout.rif" is generated automatically. The previous "autosupout.rif" file is renamed to "autosupout.old.rif"
auto-send-supout (*yes* | *no*; Default: **no**)	After the support output file is automatically generated, it can be sent by email
send-email-from (*string*; Default:)	e-mail address to send the support output file from. If not set, the value set in /tool e-mail is used
send-email-to (*string*; Default:)	e-mail address to send the support output file to.
send-smtp-server (*string*; Default:)	SMTP server address to send the support output file through. If not set, the value set in /tool e-mail is used.

Basic examples

To make system generate a support output file and sent it automatically to support@example.com throught the 192.0.2.1in case of a software crash:

```
[admin@MikroTik] system watchdog> set auto-send-supout=yes \
\... send-to-email=support@example.com send-smtp-server=192.0.2.1
[admin@MikroTik] system watchdog> print
      watch-address: none
     watchdog-timer: yes
      no-ping-delay: 5m
   automatic-supout: yes
```

```
   auto-send-supout: yes
   send-smtp-server: 192.0.2.1
      send-email-to: support@example.com
[admin@MikroTik] system watchdog>
```

[Top | Back to Content]

Manual:Winbox

Summary

Winbox is a small utility that allows administration of Mikrotik RouterOS using a fast and simple GUI. It is a native Win32 binary, but can be run on **Linux** and **Mac OSX** using Wine.

All Winbox interface functions are as close as possible to Console functions, that is why there are no Winbox sections in the manual.

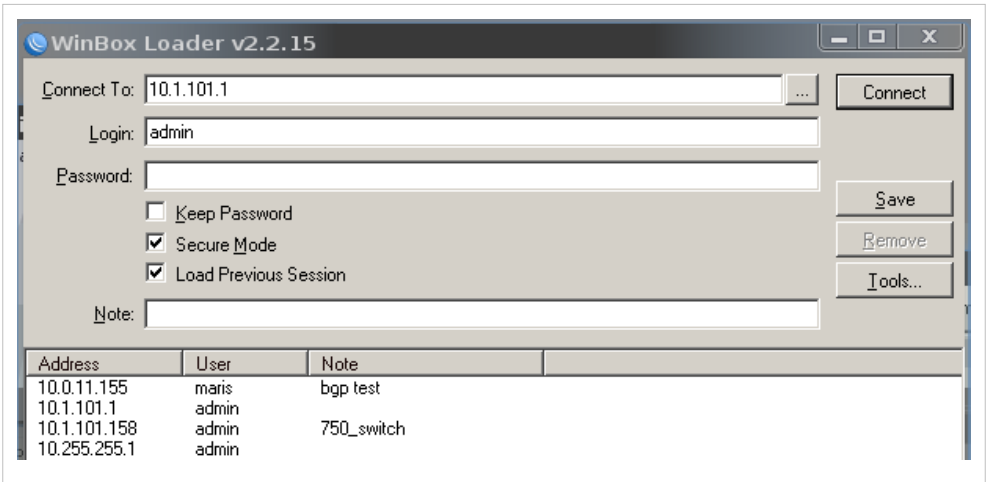
Some of advanced and system critical configurations are not possible from winbox, like MAC address change on an interface.

Starting the Winbox

Winbox loader can be downloaded directly from the router.

Open your browser and enter router's IP address, RouterOS welcome page will be displayed. Click on the link to download **winbox.exe**

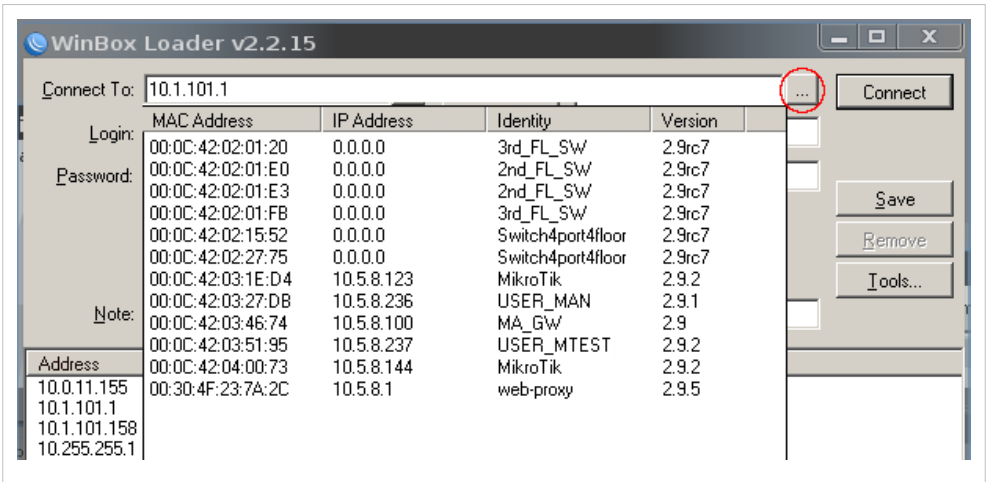
When winbox.exe is downloaded, double click on it and winbox loader window will pop up:



To connect to the router enter IP or MAC address of the router, specify username and password (if any) and click on **Connect** button.

Note: It is recommended to use IP address whenever possible. MAC session uses network broadcasts and is not 100% reliable.

You can also use neighbor discovery, to list available routers by clicking on **[...]** button:



From list of discovered routers you can click on IP or MAC address column to connect to that router. If you click on IP address then IP will be used to connect, but if you click on MAC Address then MAC address will be used to connect to the router.

Note: Neighbor discovery will show also devices which are not compatible with Winbox, like Cisco routers or any other device that uses CDP (Cisco Discovery Protocol)

Description of buttons and fields of loader screen

- **[...]** - discovers and shows MNDP (MikroTik Neighbor Discovery Protocol) or CDP (Cisco Discovery Protocol) devices.

- **Connect** - Connect to the router
- **Save** - Save address, login, password and note. Saved entries are listed at the bottom of loader window.
- **Remove** - Remove selected entry from saved list
- **Tools...** - Allows to run various tools: removes all items from the list, clears cache on the local disk, imports addresses from wbx file or exports them to wbx file.

- **Connect To:** - destination IP or MAC address of the router
- **Login** - username used for authentication
- **Password** - password used for authentication
- **Keep Password** - if unchecked, password is not saved to the list
- **Secure Mode** - if checked, winbox will use TLS encryption to secure session
- **Load Previous Session** - if checked, winbox will try to restore all previously opened windows.
- **Note** - description of the router that will be saved to the list.

Warning: Passwords are saved in plain text. Anyone with access to your file system will be able to retrieve passwords.

It is possible to use command line to pass connect to user and password parameters automatically:

```
winbox.exe [<connect-to> [<login> [<password>]]]
```

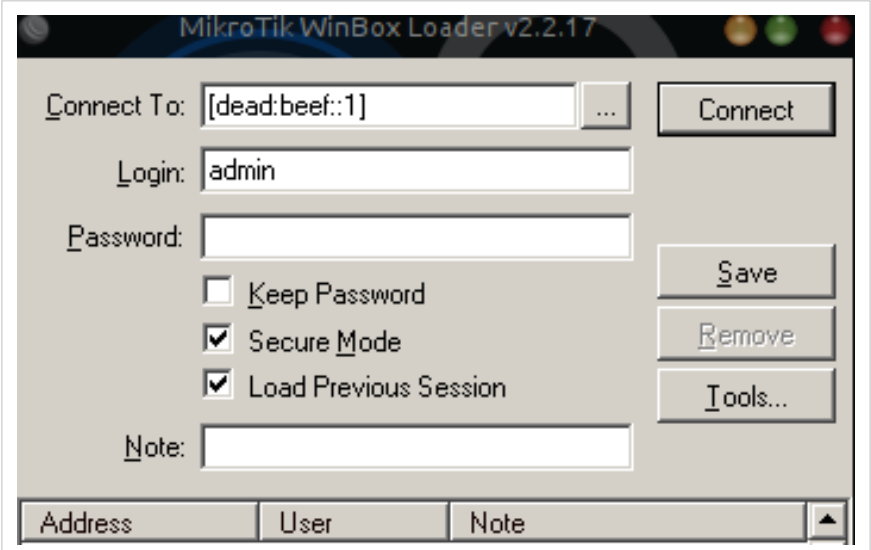
For example:

```
winbox.exe 10.5.101.1 admin
```

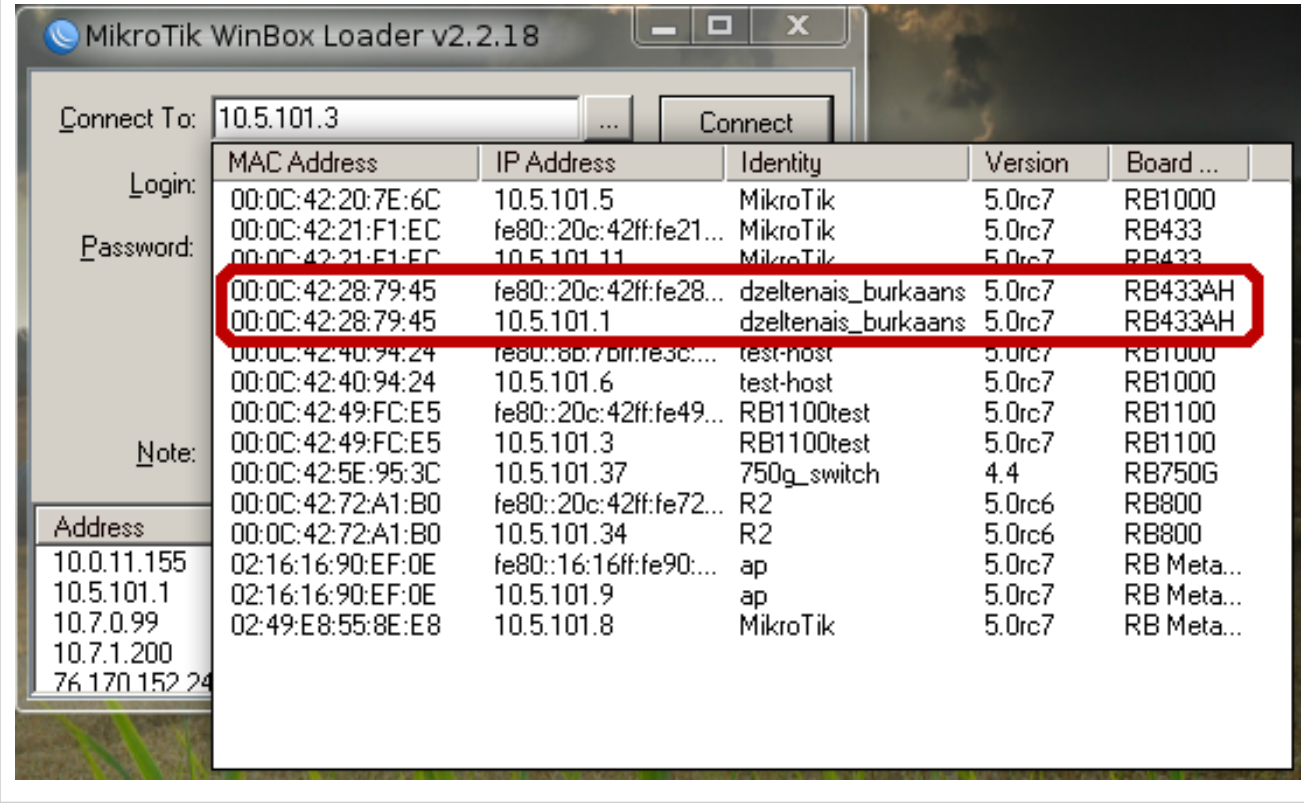
Will connect to router 10.5.101.1 with username "admin"without password.

IPv6 connectivity

Starting from v5RC6 Winbox supports IPv6 connectivity. To connect to the routers IPv6 address, it must be placed in square braces the same as in web browsers when connecting to IPv6 server. Example:



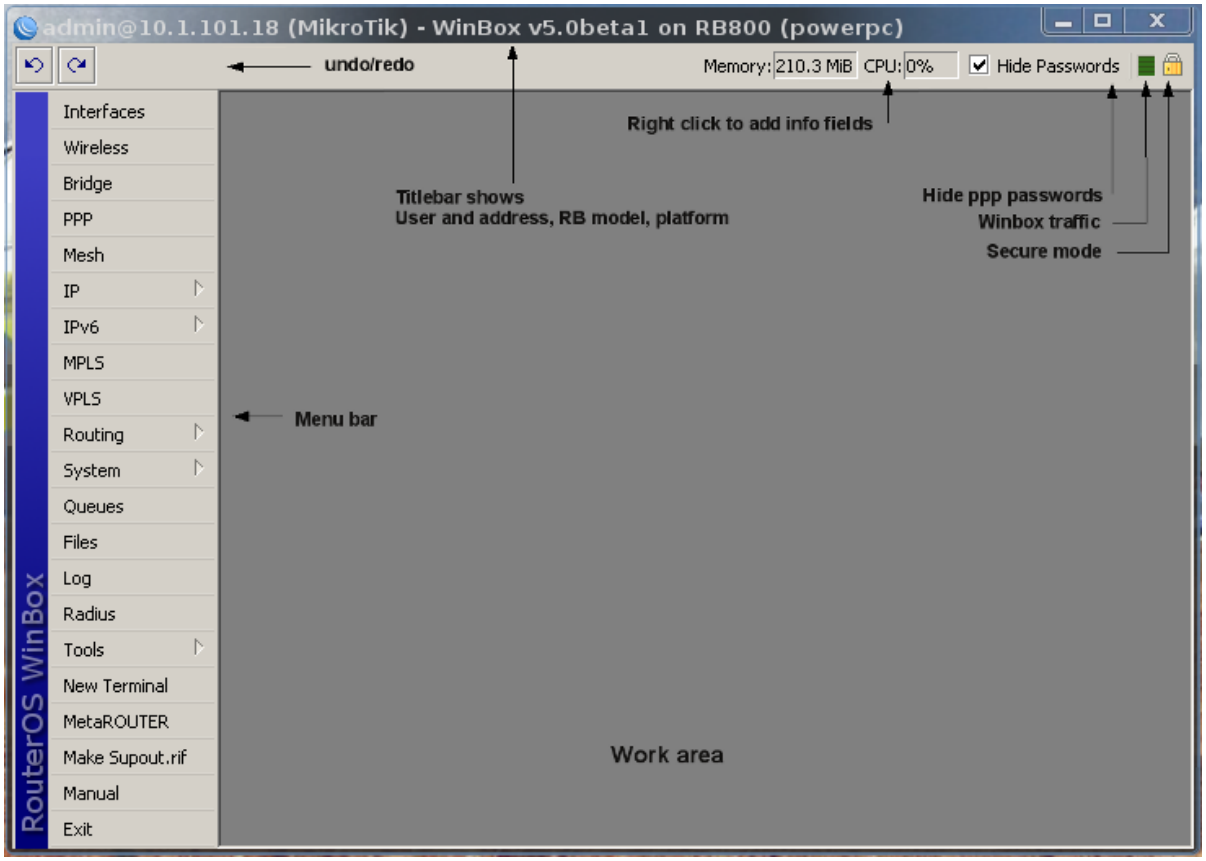
Winbox neighbor discovery is now capable of discovering IPv6 enabled routers. As you can see from the image below, there are two entries for each IPv6 enabled router, one entry is with IPv4 address and another one with IPv6 link-local address. You can easily choose to which one you want to connect:



Interface Overview

Winbox interface has been designed to be intuitive for most of the users. Interface consists of:

- Main toolbar at the top where users ca add various info fields, like CPU and memory usage.
- Menu bar on the left - list of all available menus and sub-menus. This list changes depending on what packages are installed. For example if IPv6 package is disabled, then **IPv6** menu and all it's sub-menus will not be displayed.
- Work area - area where all menu windows are opened.



Title bar shows information to identify with which router Winbox session is opened. Information is displayed in following format:

```
[username]@[Router's IP or MAC] ( [RouterID] ) - Winbox [ROS version] on [RB model] ([platform])
```

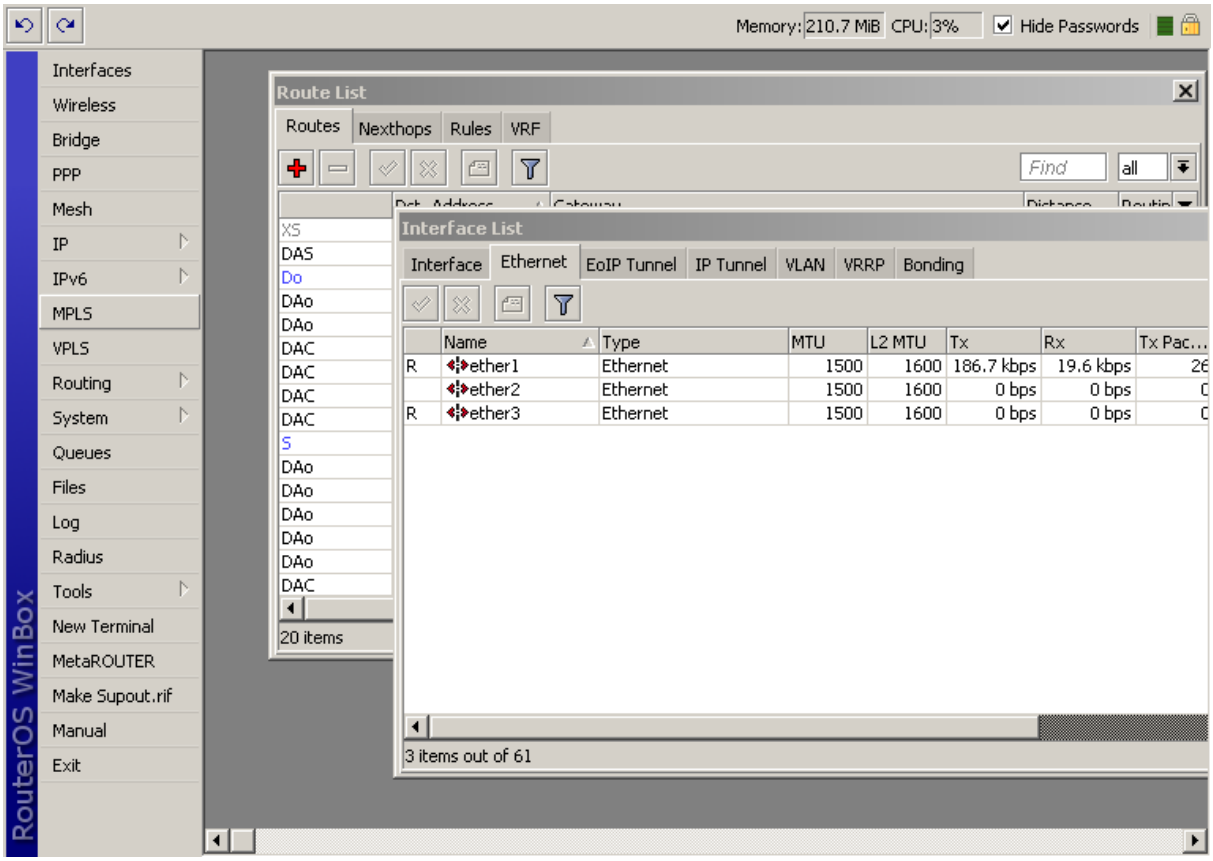
From screenshot above we can see that user **admin** is logged into router with IP address **10.1.101.18**. Router's ID is **MikroTik**, currently installed RouterOS version is **v5.0beta1**, RouterBoard is **RB800** and platform is **PowerPC**.

On the Main toolbar's left side is located **undo** and **redo** buttons to quickly undo any changes made to configuration. On the right side is located:

- winbox traffic indicator displayed as a green bar,
- indicator that shows whether winbox session uses TLS encryption
- checkbox **Hide password**. This checkbox replaces all sensitive information (for example, ppp secret passwords) with '*' asterisk symbols.

Work Area and child windows

Winbox has MDI interface meaning that all menu configuration (child) widows are attached to main (parent) Winbox window and are showed in work area.



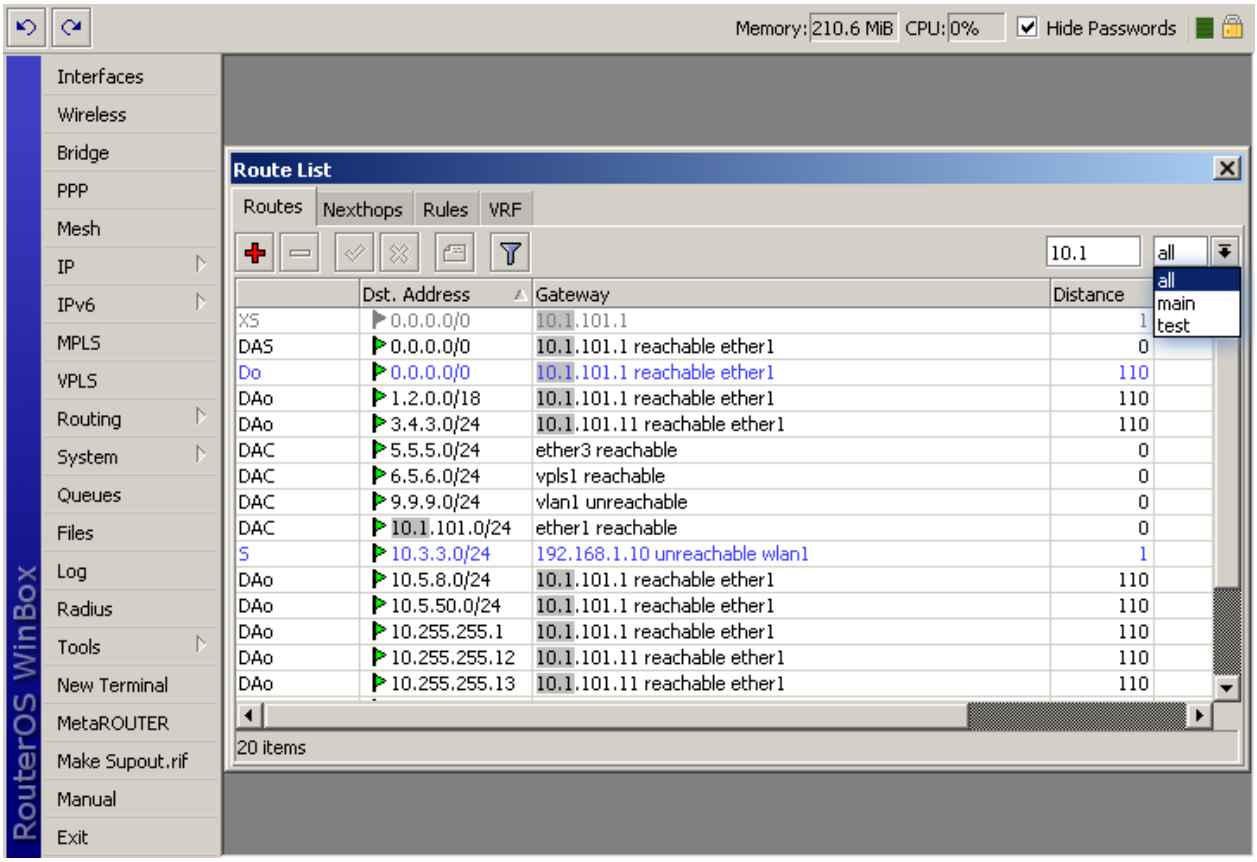
Child windows can not be dragged out of working area. Notice in screenshot above that **Interface** window is dragged out of visible working area and horizontal scroll bar appeared at the bottom. If any window is outside visible work area boundaries the vertical or/and horizontal scrollbars will appear.

Child window menu bar

Each child window has its own toolbar. Most of the windows have the same set of toolbar buttons:

- **Add** - add new item to the list
- **Remove** - remove selected item from the list
- **Enable** - enable selected item (the same as **enable** command from console)
- **Disable** - disable selected item (the same as **disable** command from console)
- **Comment** - add or edit comment
- **Sort** - allows to sort out items depending on various parameters. `Read more >>`

Almost all windows have quick search input field at the right side of the toolbar. Any text entered in this field is searched through all the items and highlighted as illustrated in screenshot below

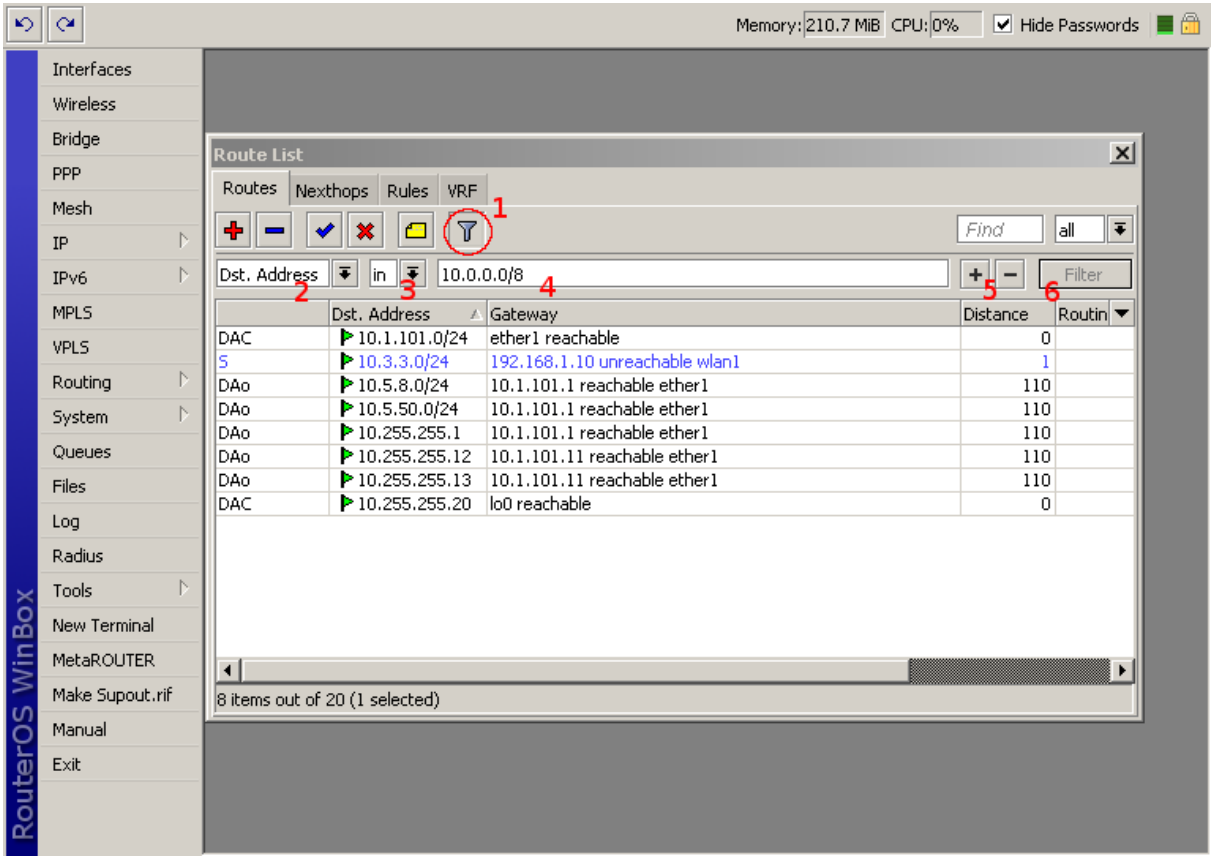


Notice that at the right side next to quick find input filed there is a dropdown box. For currently opened (IP Route) window this dropdown box allows to quickly sort out items by routing tables. For example if **main** is selected, then only routes from main routing table will be listed.

Similar dropdown box is also in all firewall windows to quickly sort out rules by chains.

Sorting out displayed items

Almost every window has a **Sort** button. When clicking on this button several options appear as illustrated in screenshot below



Example shows how to quickly filter out routes that are in 10.0.0.0/8 range

1. Press **Sort** button
2. Chose **Dst.Address** from the first dropdown box.
3. Chose **in** form the second dropdown box. "in" means that filter will check if dst address value is in range of specified network.
4. Enter network against which values will be compared (in our example enter "10.0.0.0/8")
5. These buttons are to add or remove another filter to the stack.
6. Press **Filter** button to apply our filter.

As you can see from screenshot winbox sorted out only routes that are within 10.0.0.0/8 range.

Comparison operators (Number **3** in screenshot) may be different for each window. For example "Ip Route" window has only two **is** and **in**. Other windows may have operators such as "is not", "contains", "contains not".

Winbox allows to build stack of filters. For example if there is a need to filter by destination address and gateway, then

- set first filter as described in example above,
- press **[+]** button to add another filter bar in stack.
- set up seconf filter to filter by gateway
- press **Filter** button to apply filters.

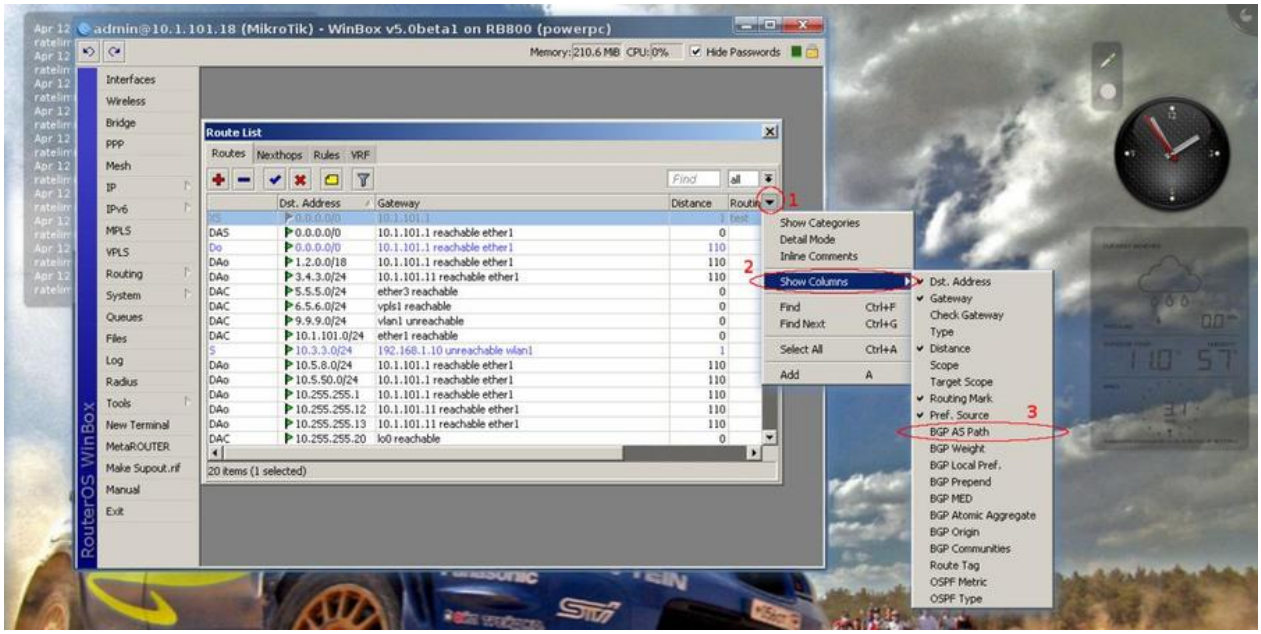
You can also remove unnecessary filter from the stack by pressing **[-]** button.

Customizing list of displayed columns

By default winbox shows most commonly used parameters. However sometimes it is needed to see another parameters, for example "BGP AS Path" or other BGP attributes to monitor if routes are selected properly.

Winbox allows to customize displayed columns for each individual window. For example to add BGP AS path column:

- Click on little arrow button (**1**) on the right side of the column titles or right mouse click on the route list.
- From popped up menu move to **Show Columns** (**2**) and from the sub-menu pick desired column, in our case click on **BGP AS Path** (**3**)

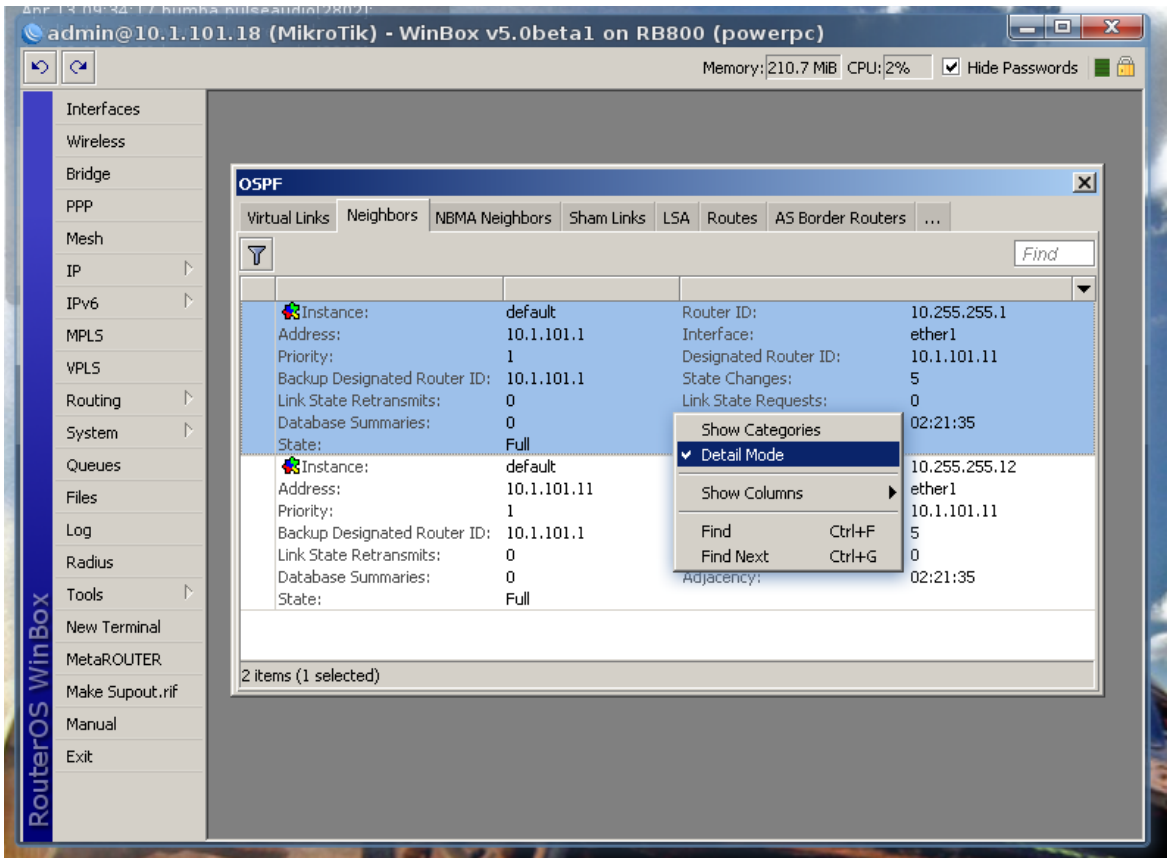


Changes made to window layout are saved and next time when winbox is opened the same column order and size is applied.

Detail mode

It is also possible to enable **Detail mode**. In this mode all parameters are displayed in columns, first column is parameter name, second column is parameter's value.

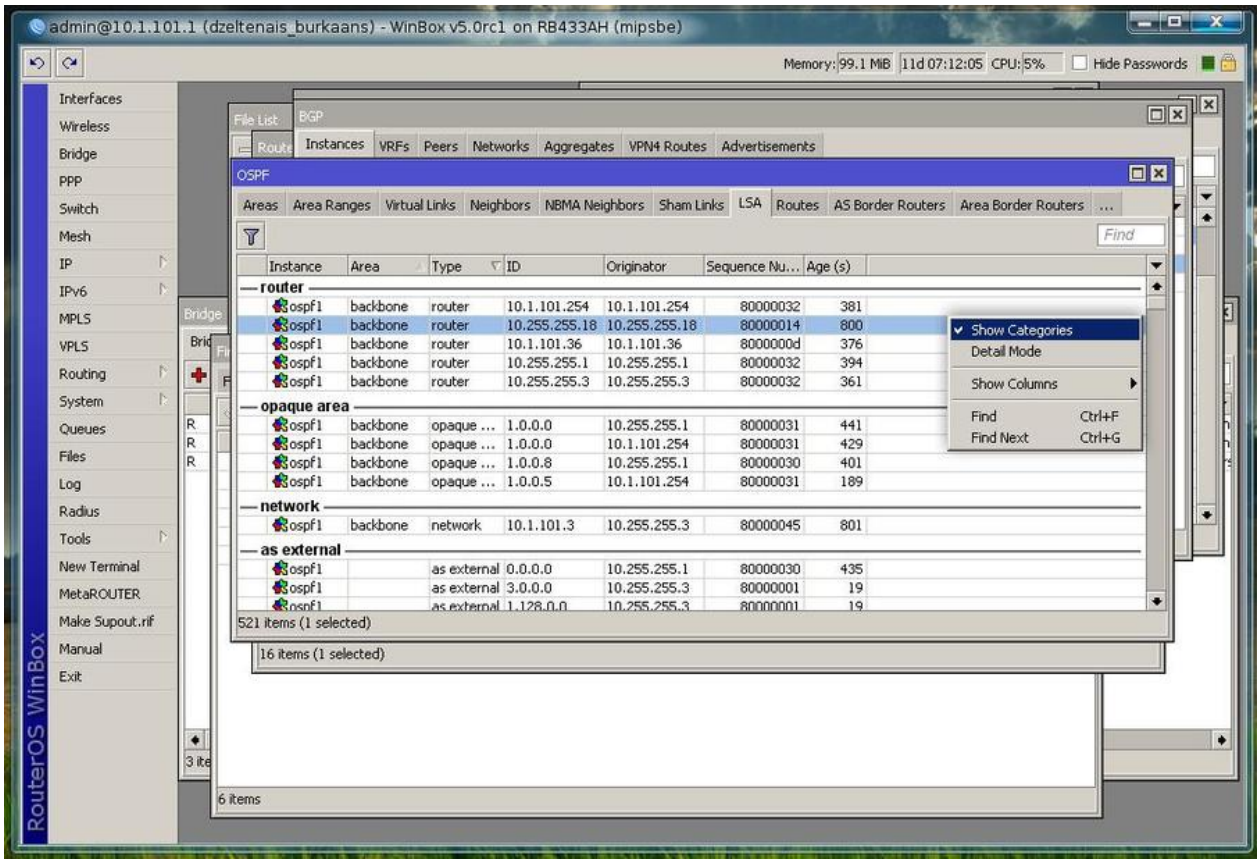
To enable detail mode right mouse click on the item list and from the popupmenu pick **Detail mode**



Category view

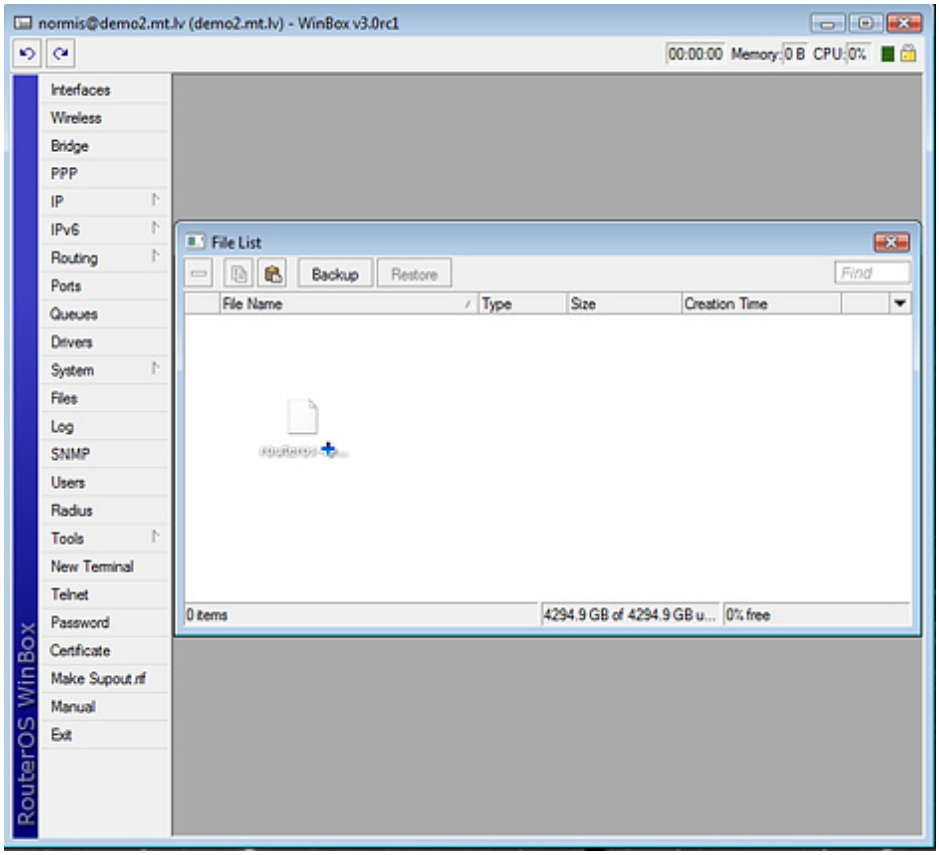
It is possible to list items by categories. In tis mode all items will be grouped alphabetically or by other category. For example items may be categorized alphabetically if sorted by name, items can also be categorized by type like in screenshot below.

To enable Category view, right mouse click on the item list and from the popupmenu pick **Show Categories**



Drag & Drop

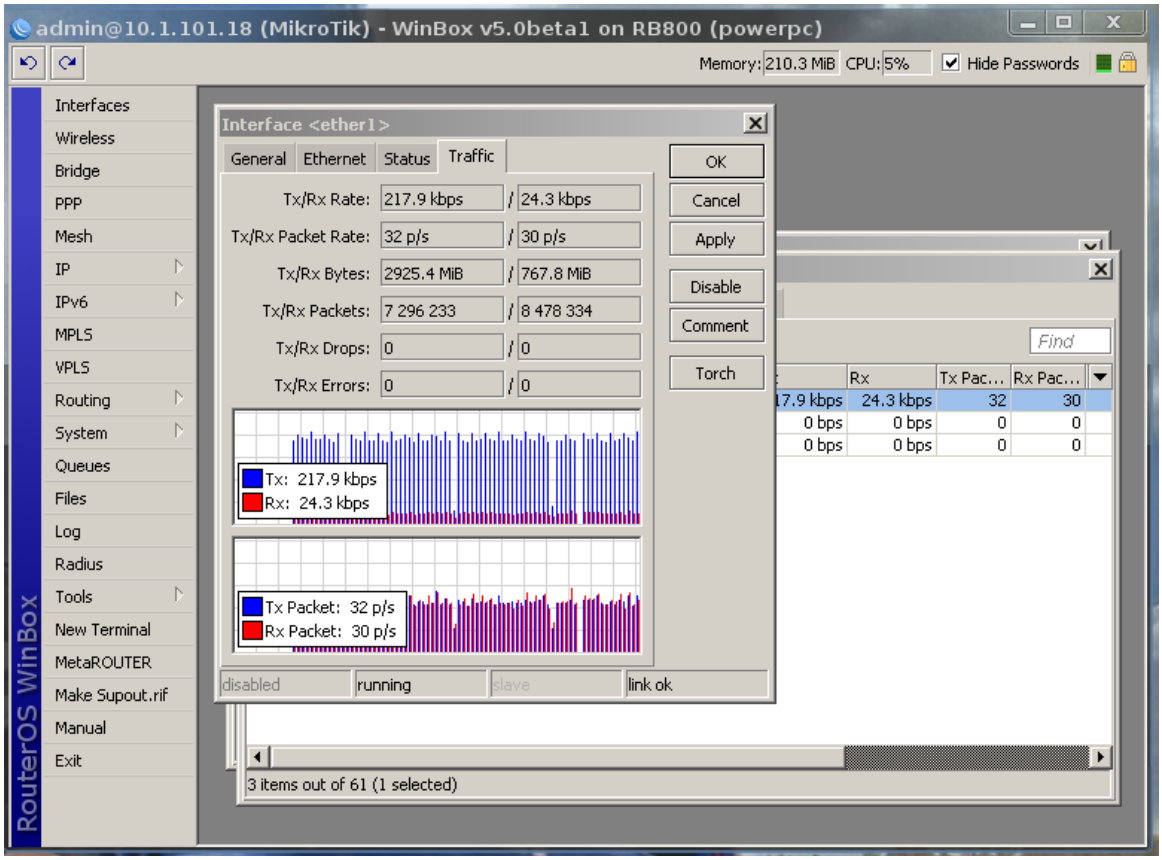
It is possible to upload and download files to/from router using winbox drag & drop functionality.



Note: Drag & Drop does not work if winbox is running on Linux using wine. This is not a winbox problem, wine does not support drag & drop.

Traffic monitoring

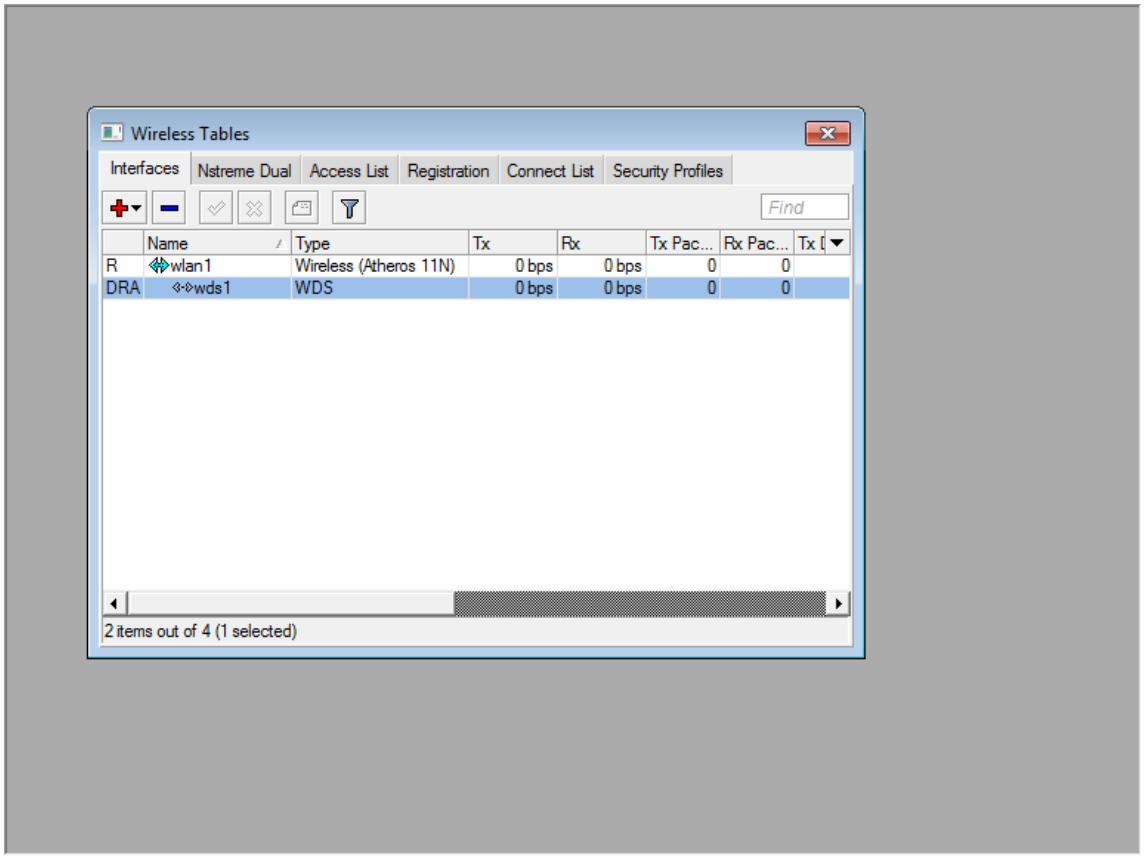
Winbox can be used as a tool to monitor traffic of every interface, queue or firewall rule in real-time. Screenshot below shows ethernet traffic monitoring graphs.


admin@10.1.101.18 (MikroTik) - WinBox v5.0beta1 on RB800 (powerpc)
Memory: 210.3 MiB
CPU: 5%
Hide Passwords
Interfaces
Wireless
Bridge
PPP
Mesh
IP
IPv6
MPLS
VPLS
Routing
System
Queues
Files
Log
Radius
Tools
New Terminal
MetaROUTER
Make Supout.rif
Manual
Exit
RouterOS WinBox
Interface <ether1>
General
Ethernet
Status
Traffic
OK
Cancel
Apply
Disable
Comment
Torch
Tx/Rx Rate: 217.9 kbps / 24.3 kbps
Tx/Rx Packet Rate: 32 p/s / 30 p/s
Tx/Rx Bytes: 2925.4 MiB / 767.8 MiB
Tx/Rx Packets: 7 296 233 / 8 478 334
Tx/Rx Drops: 0 / 0
Tx/Rx Errors: 0 / 0
Tx: 217.9 kbps
Rx: 24.3 kbps
Tx Packet: 32 p/s
Rx Packet: 30 p/s
disabled
running
slave
link ok
Find
Rx
Tx Pac...
Rx Pac...
24.3 kbps
32
30
0 bps
0 bps
0
0
0 bps
0 bps
0
0
3 items out of 61 (1 selected)

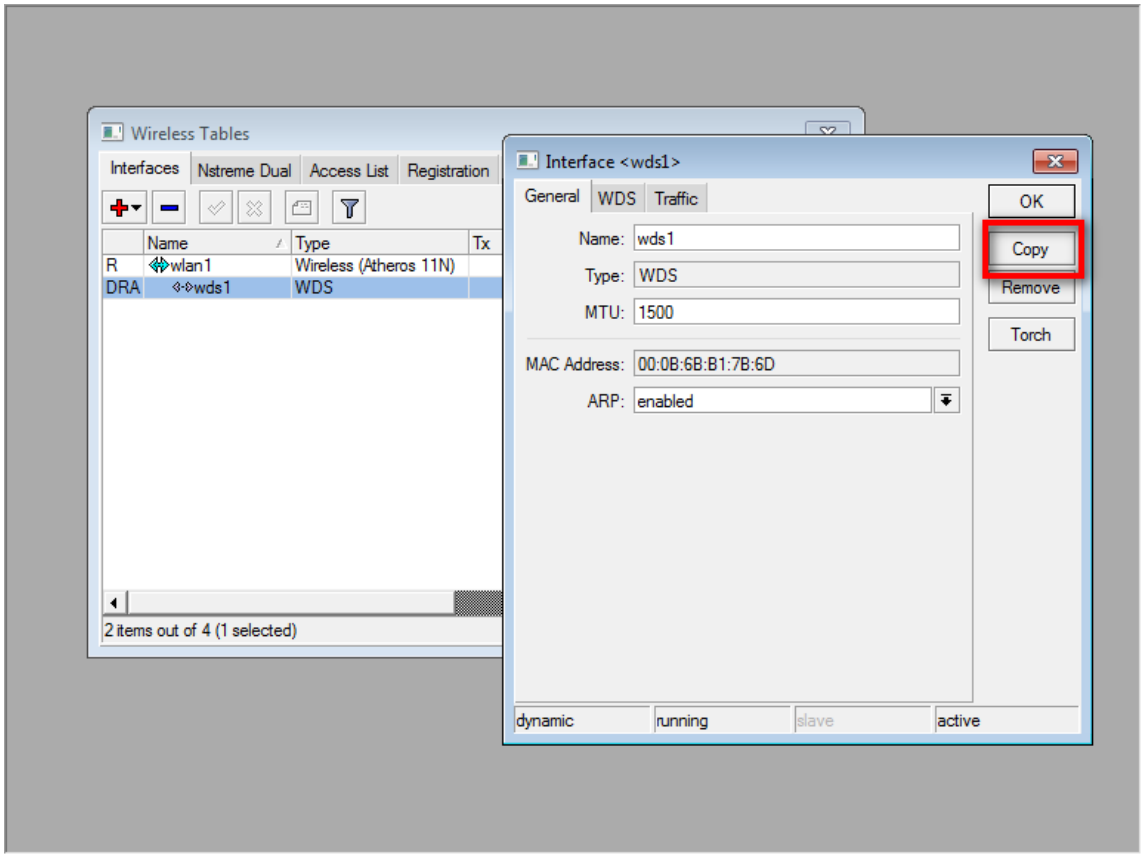
Item copy

This shows how easy it is to copy an item in Winbox. In this example, we will use the COPY button to make a Dynamic WDS interface into a Static interface.

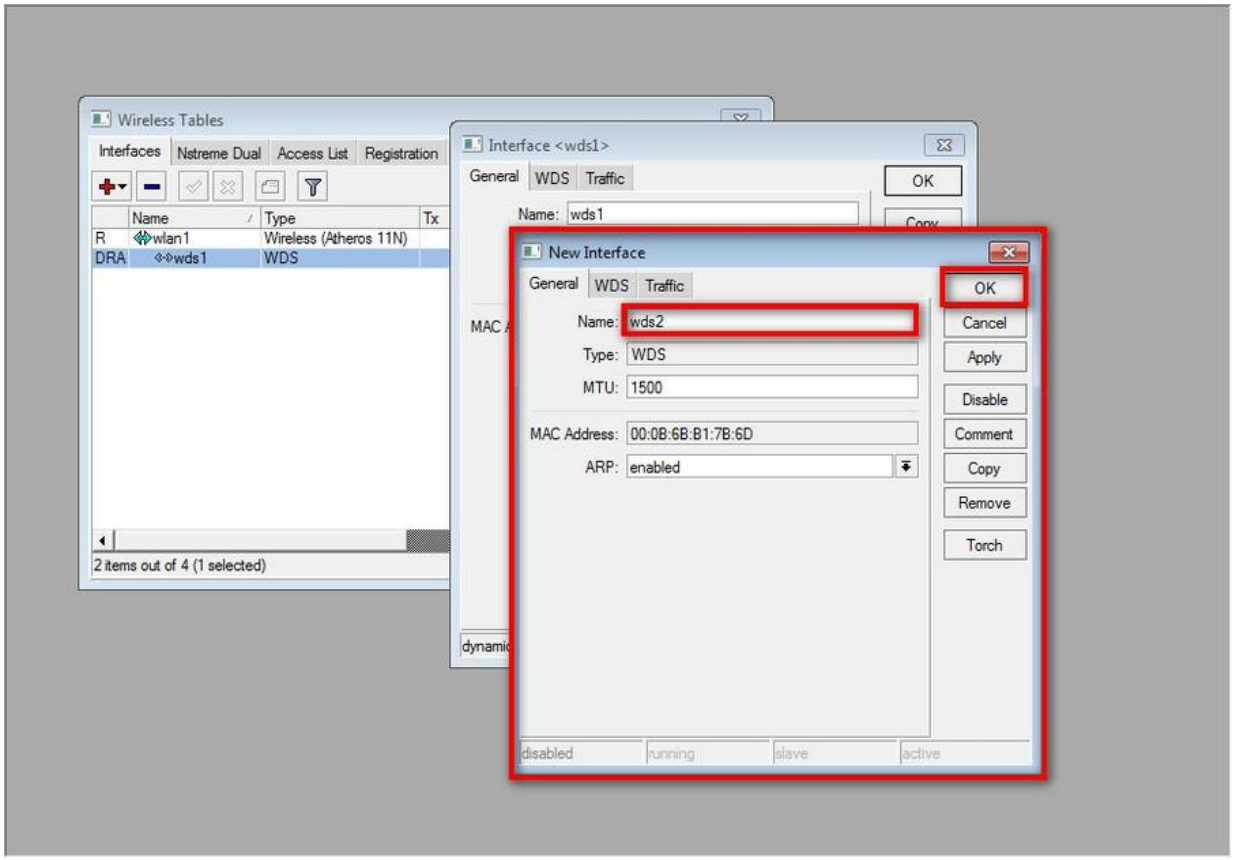
This image shows us the initial state, as you see DRA indicates "D" which means Dynamic:

- 

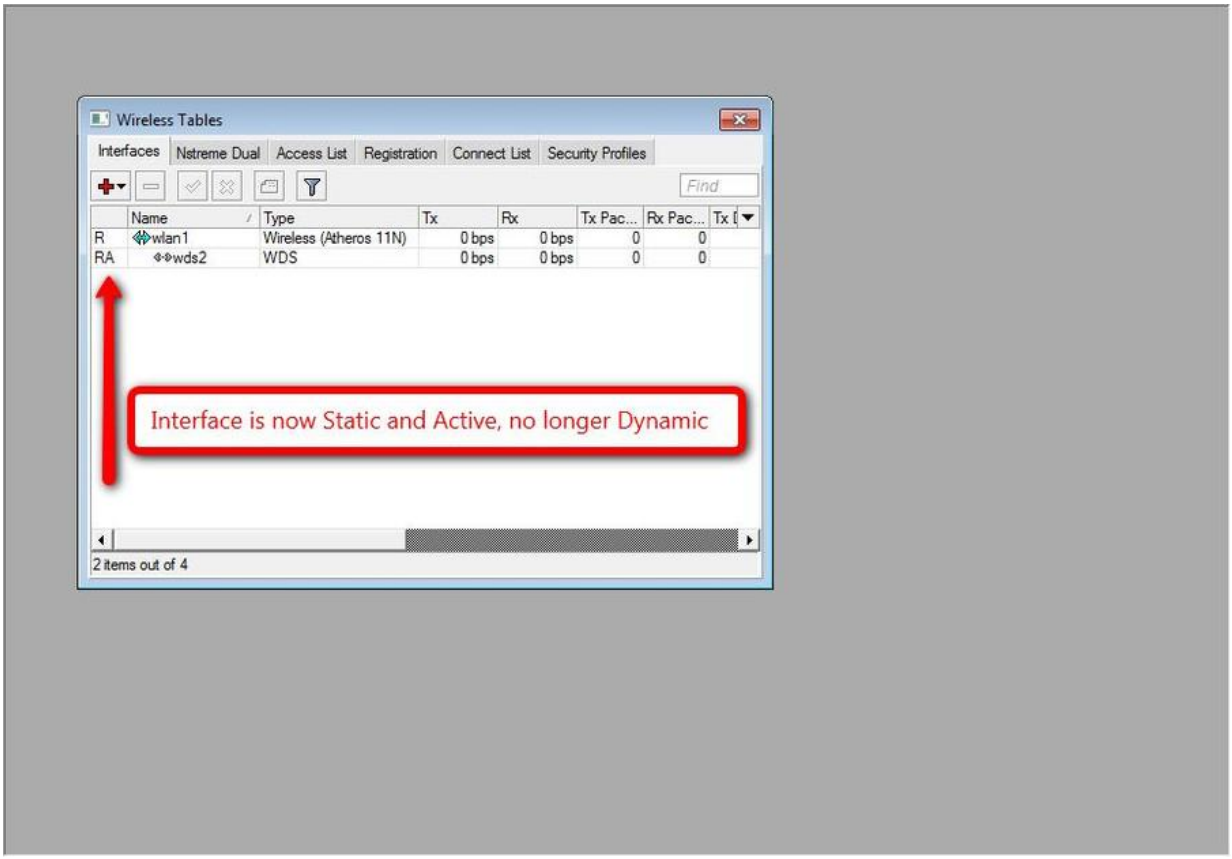
Double-Click on the interface and click on COPY:

- 

A new interface window will appear, a new name will be created automatically (in this case WDS2)

- 

You can see that the new interface status has changed:

- 

Transferring Settings

On Windows Vista/7 Winbox settings are stored in: %USERPROFILE%\AppData\Roaming\Mikrotik\Winbox\winbox.cfg

Simply copy this file to the same location on the new host.

[Top | Back to Content]

Manual:Wireless card diagnostics

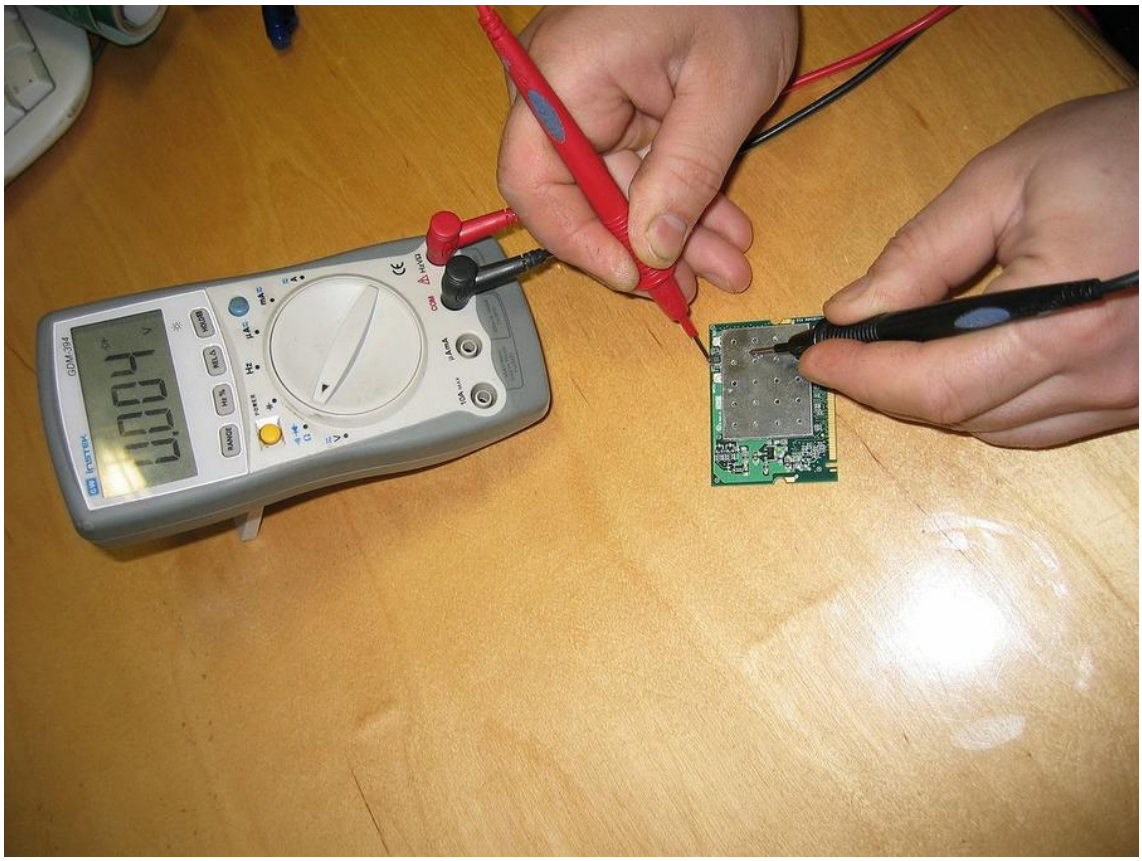
R52, R52Hn and R52H Power Amplifier damage

If the cards are becoming too hot to touch, when inserted in a RouterBOARD, but are disabled - the PA might be damaged. This could be caused by user, or by manufacturing problem. To determine, must return to RMA for close inspection.

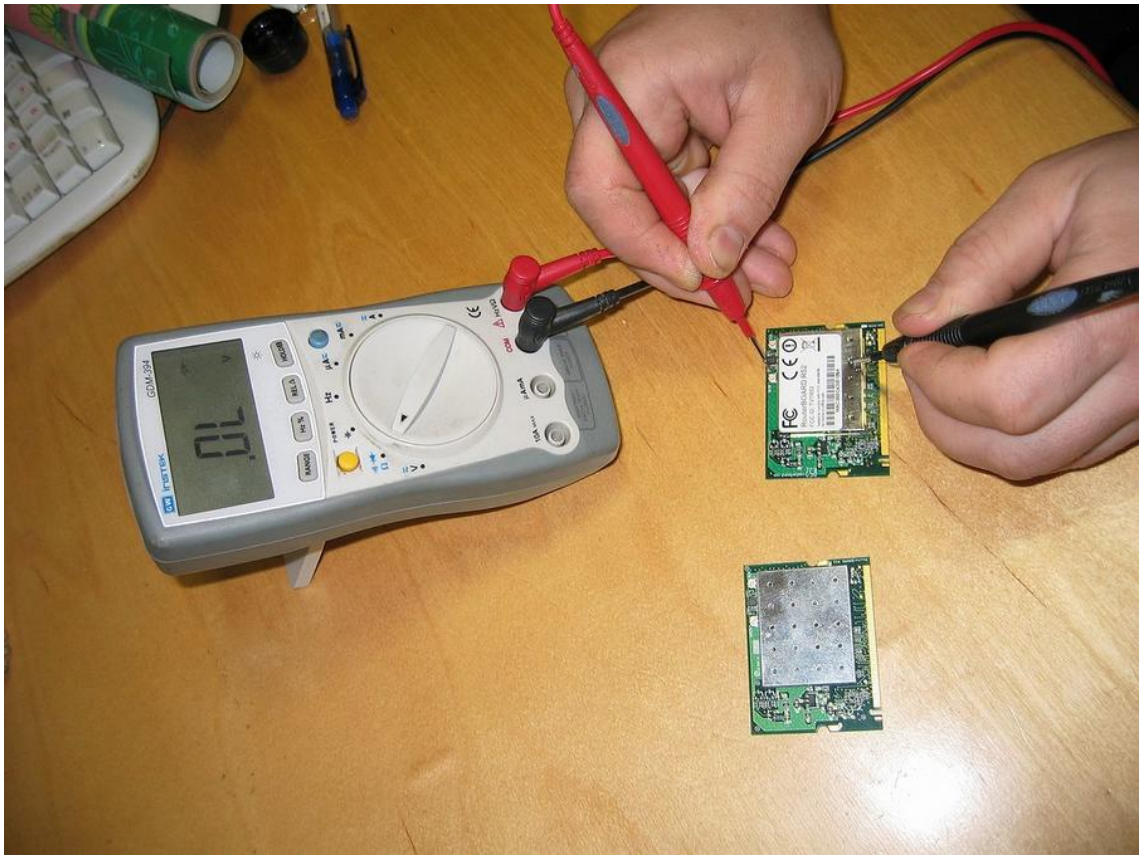
R52, R52Hn and R52H ESD damage

Improper grounding can cause ESD damage to wireless cards during storms or other ESD situations. To test if your R52 or R52H card is malfunctioning due to lightning/storm electrostatic damage, use a multimeter. In case the test fails with this method, the warranty doesn't cover it:

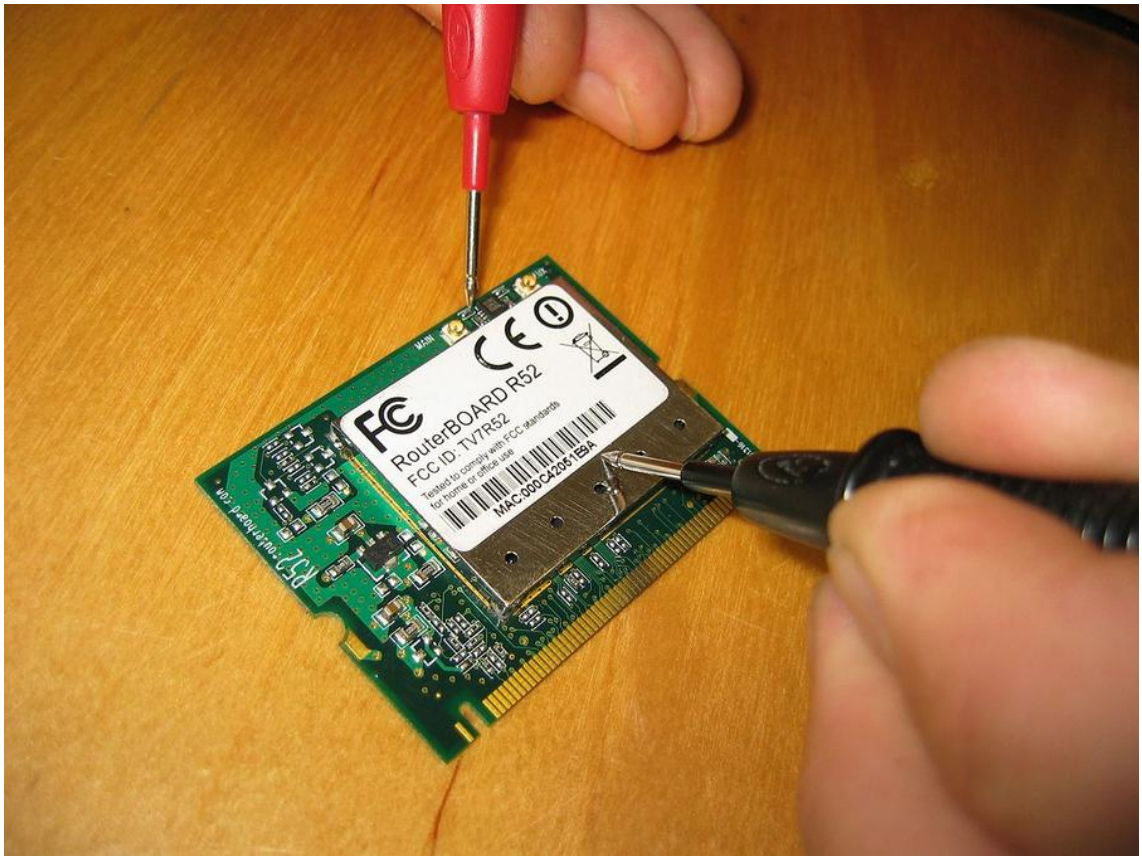
Damaged card:



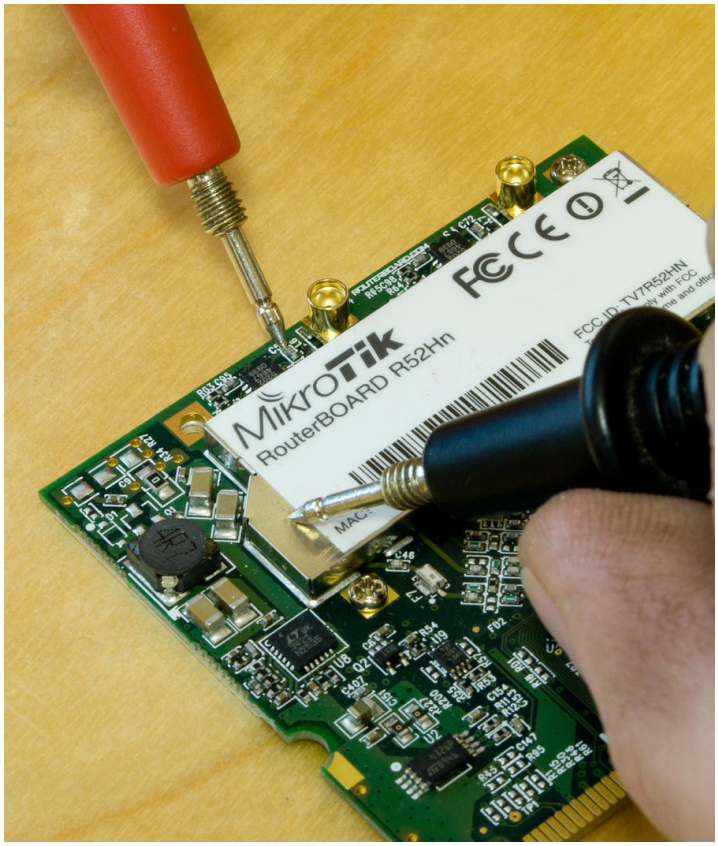
Normal card:



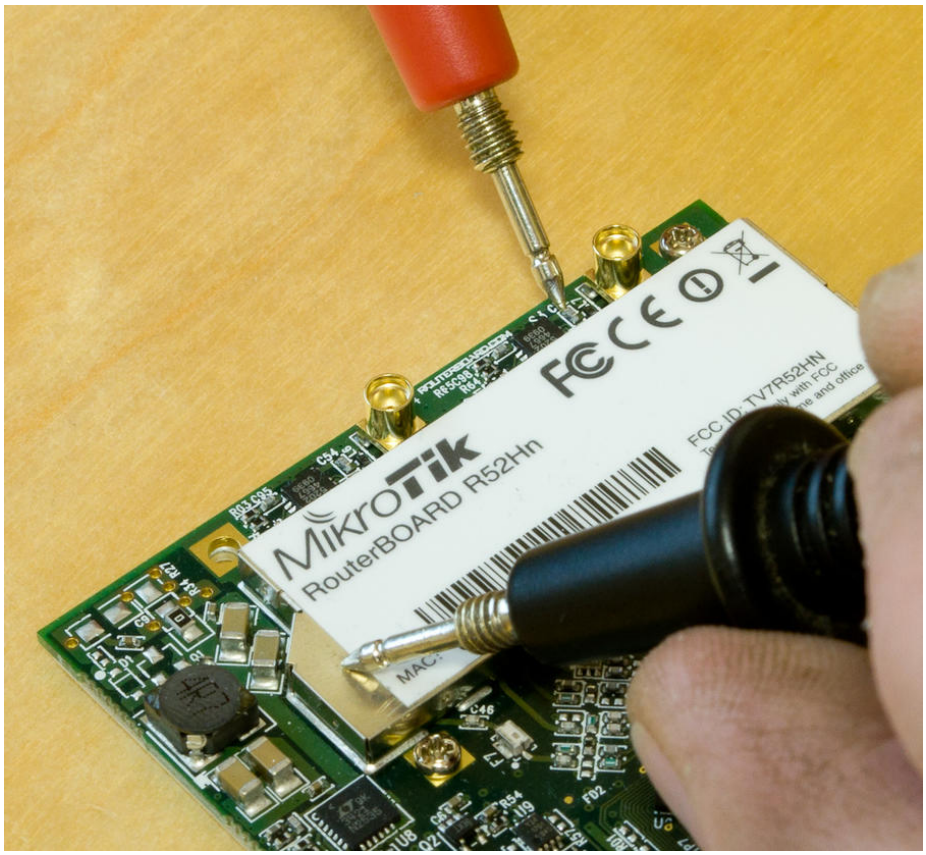
Testing area close-up:



R52Hn card chain 0:



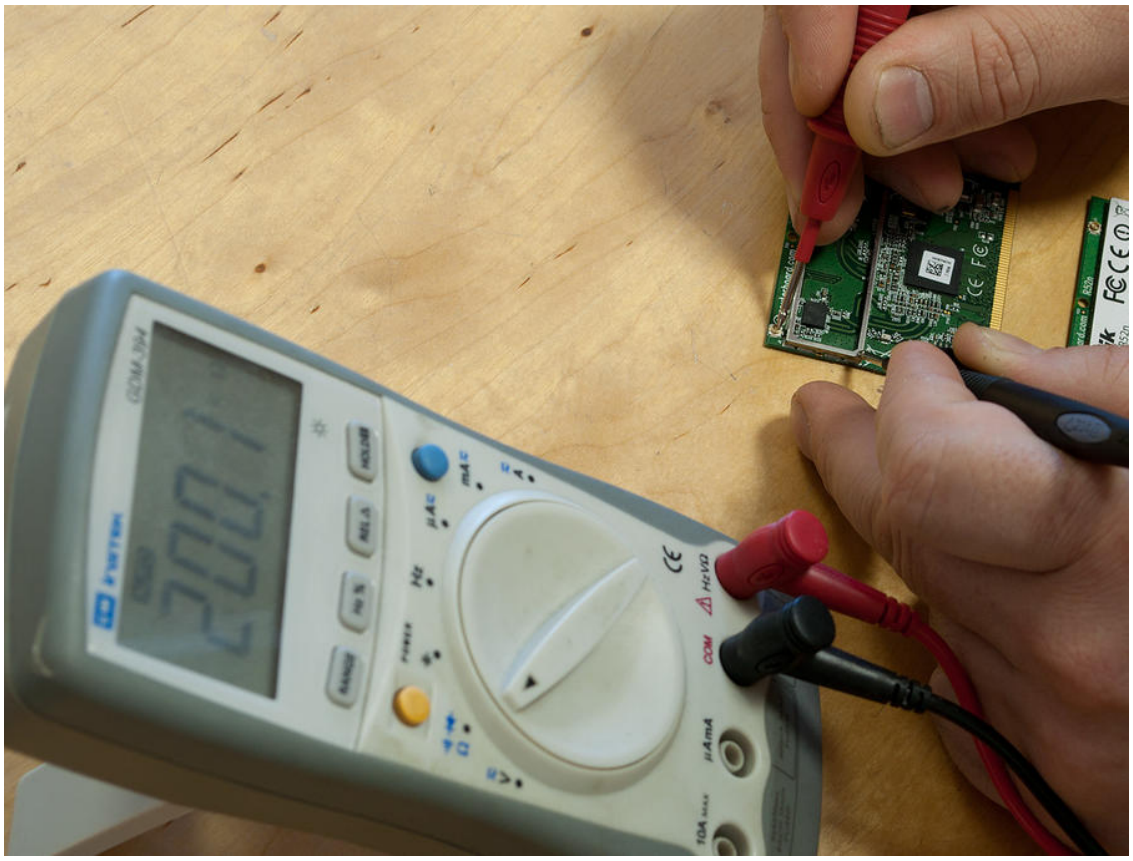
R52Hn chain 1:



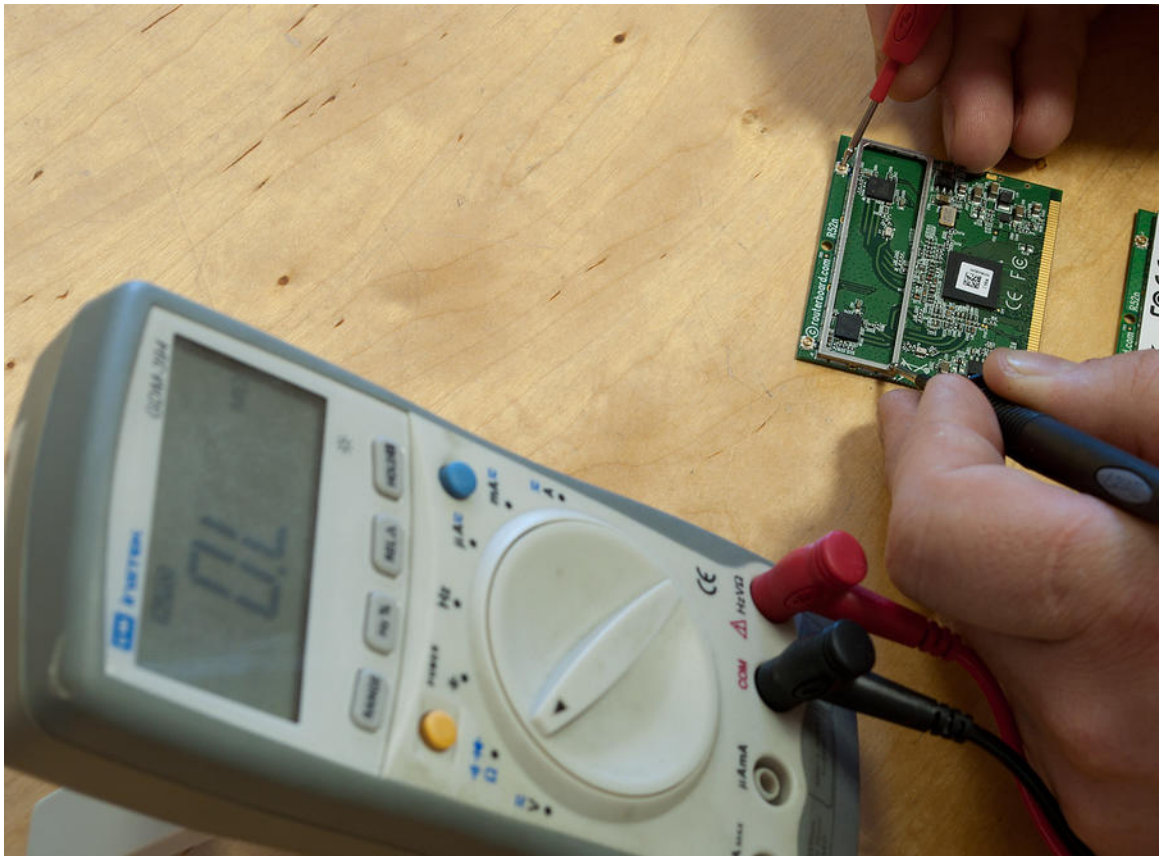
R52n antenna circuit damage test

These images show how to test for antenna circuit damage. If the resistance between shown points is lower than infinity (shown as OL on multimeter), the card is damaged by lightning, and the damage will not be repaired by warranty (don't send to RMA).

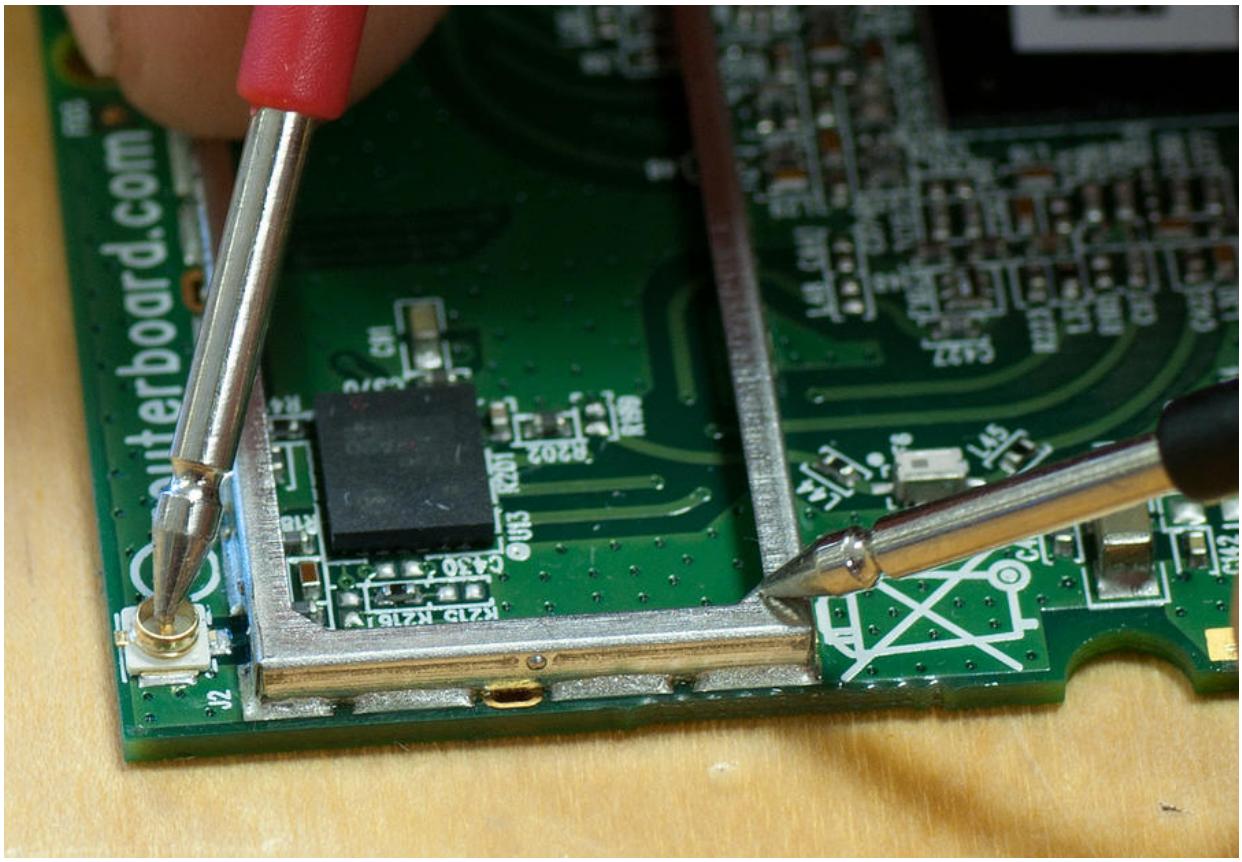
This chain is damaged:



This chain is OK:



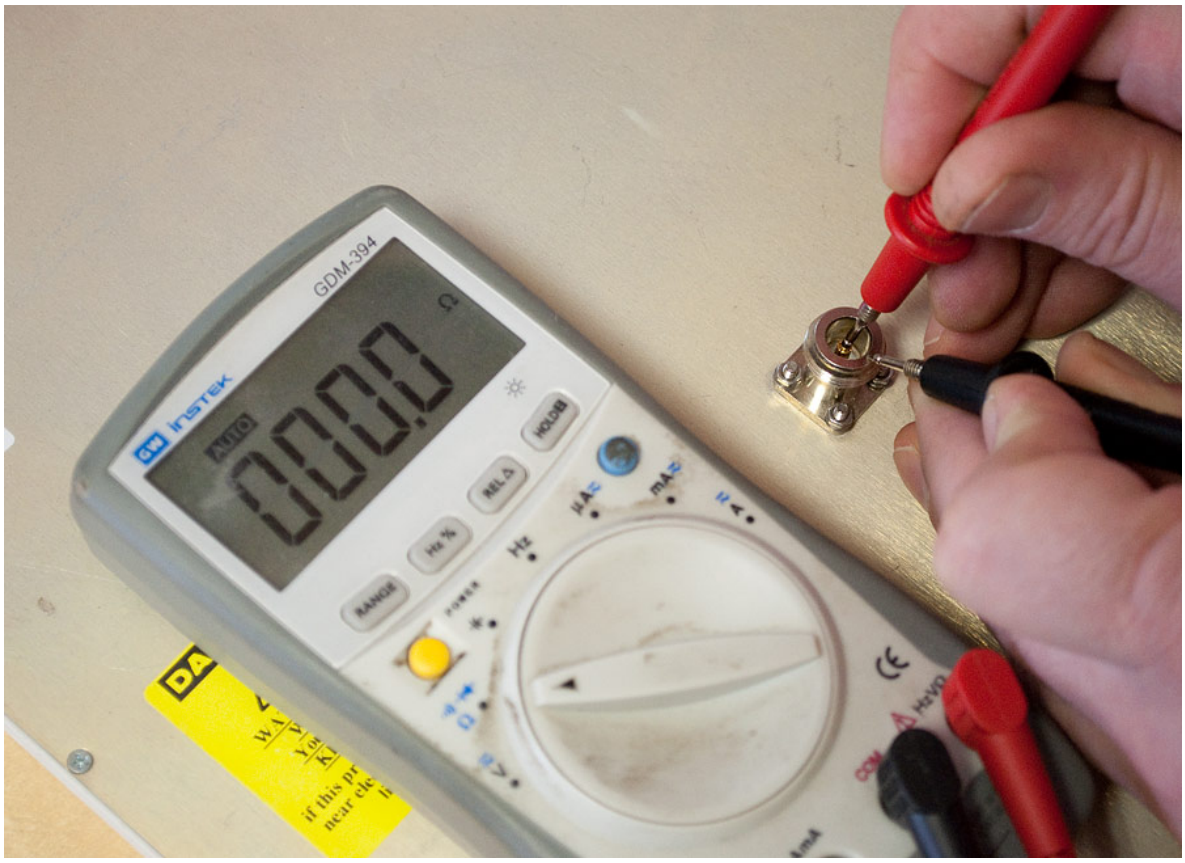
Close-up of testing area:



DC shorted antennas

Also make sure that your antenna is DC shorted:

DC shorted antenna. This antenna doesn't need a Coax lightning arrestor:



NOT DC shorted antenna. This antenna needs a Coax lightning arrestor to avoid sudden wireless card damage. Note the OL (Open Loop) in the multimeter:

Manual:Wireless FAQ

Settings

Why I can't connect to MikroTik 802.11n AP with Apple Mac devices?

This problem is only seen on Mac devices based on Broadcom wireless chipsets. In order to connect with such wireless device to MikroTik 802.11n AP make sure that you don't use 'short' preamble-mode. Use 'long' or 'both' preamble-mode and Mac wireless devices will be able to connect.

By changing some wireless settings the wireless link works unstable

Sometimes when you change some wireless setting for tuning the links you got so far that the link isn't establishing any more or works unstable and you don't remember what settings you had in the beginning. In this case you can use the *reset-configuration* command in the wireless menu - it will reset the all the wireless settings for the specific wireless interface and you will be able to configure the interface from the start. Note that executing this command also disables the interface, so please be careful not to execute this command if you are configuring router remotely using that wireless link that you want to reset the configuration.

What are wireless retransmits and where to check them?

Wireless retransmission is when the card sends out a frame and you don't receive back the acknowledgment (ACK), you send out the frame once more till you get back the acknowledgment. Wireless retransmits can increase the latency and also lower the throughput of the wireless link. To check if the wireless connection has wireless retransmissions you need to compare two fields in the wireless registration table: **frames** and **hw-frames**. If the *hw-frames* value is bigger than *frames* value then it means that the wireless link is making retransmissions. If the

difference is not so big, it can be ignored, but if the *hw-frames* count it two, three or four times or even bigger than the *frames* count then you need to troubleshoot this wireless connection.

Can I compare frames with hw-frames also on Nstreme links?

The **frames** counts only those which contain actual data. In case of Nstreme, only the ACK can be transmitted in a single frame, if there is no other data to send. These ACK frames will not be added to the **frames** count, but they will appear at **hw-frames**. If there is traffic on both directions at maximum speed (eg. there will be no only-ack frames), then you can't compare **frames** to **hw-frames** as in case of regular wireless links.

What TX-power values can I use?

The tx-power default setting is the maximum tx-power that the card can use and is taken from the cards eeprom. If you want to use larger tx-power values, you are able to set them, but **do it at your own risk**, as it will probably damage your card eventually! Usually, one should use this parameter only to reduce the tx-power.

In general tx-power controlling properties should be left at the default settings. Changing the default setting may help with some cards in some situations, but without testing, the most common result is degradation of range and throughput. Some of the problems that may occur are:

- overheating of the power amplifier chip and the card which will cause lower efficiency and more data errors;
- overdriving the amplifier which will cause more data errors;
- excessive power usage for the card and this may overload the 3.3V power supply of the board that the card is located on resulting in voltage drop and reboot or excessive temperatures for the board.

What TX-power-mode is the best?

TX-power-mode tells the wireless card which tx-power values should be used. By default this setting is set to *default*.

- **default** means that the card will use the tx-power values from the cards eeprom and will ignore the setting what is specified by the user in the *tx-power* field.
- **card-rates** means that for different data rates the tx-power is calculated according the cards transmit power algorithm from the cards eeprom and as an argument it takes *tx-power* value specified by the user.
- **all-rates-fixed** means that that the card will use one tx-power value for all data rates which is specified by the user in *tx-power* field.

Note that it is not recommended to use 'all-rates-fixed' mode as the wireless card tx-power for the higher data rates is lower and by forcing to use the fixed tx-power rates also for the higher data rates might results the same problems like in the previous question about tx-power setting. For most of the cases if you want to change the tx-power settings it is recommended to use the *tx-power-mode=card-rates* and it is recommended to lower and not to raise tx-power.

What is CCQ and how are the values determined?

Client Connection Quality (CCQ) is a value in percent that shows how effective the bandwidth is used regarding the theoretically maximum available bandwidth. CCQ is weighted average of values Tmin/Treal, that get calculated for every transmitted frame, where Tmin is time it would take to transmit given frame at highest rate with no retries and Treal is time it took to transmit frame in real life (taking into account necessary retries it took to transmit frame and transmit rate).

What is hw-retries setting?

Number of times sending frame is retried without considering it a transmission failure. Data rate is decreased upon failure and frame is sent again. Three sequential failures on lowest supported rate suspend transmission to this destination for the duration of *on-fail-retry-time*. After that, frame is sent again. The frame is being retransmitted until transmission success, or until client is disconnected after *disconnect-timeout*. Frame can be discarded during this time if *frame-lifetime* is exceeded.

What is disconnect-timeout setting?

This interval is measured from third sending failure on the lowest data rate. At this point 3 * (*hw-retries* + 1) frame transmits on the lowest data rate had failed. During *disconnect-timeout* packet transmission will be retried with *on-fail-retry-time* interval. If no frame can be transmitted successfully during *disconnect-timeout*, connection is closed, and this event is logged as "extensive data loss". Successful frame transmission resets this timer.

What is adaptive-noise-immunity setting?

Adaptive Noise Immunity (ANI) adjusts various receiver parameters dynamically to minimize interference and noise effect on the signal quality [1] This setting is added in the wireless driver for Atheros AR5212 and newer chipset cards

How does wireless device measure signal strength, when access-list or connect-list are used ?

Reported signal level is exponentially weighted moving average with smoothing factor 50%.

What error correction methods are supported in the RouterOS wireless?

ARQ method is supported in nstreme protocols. Regular 802.11 standard does not include ARQ - retrasmission of corrupt frames is based on acknowledgement protocol. RouterOS supports forward error correction coding (convolutional coding) with coding rates: 1/2, 2/3, or 3/4.

Setups

Will an amplifier improve the speed on my link?

It depends on your signal quality and noise. Remember that you can probably get a better link with low output power setting, and a good antenna. Amplifier increases the noise and will only cause problems with the link.

The amplifier gets a boost on both the transmitted **and** received signal. Thus, in "silent" areas, where you are alone or with very few "noise" or "competition", you might get excellent results. On the other side, in crowded areas, with lots of wireless activity, you will also increase signal received from every other competitor or noise source, which may dramatically lower the overall quality of the link. Also, take in account the EIRP to see if your link remains in legal limits.

You could also get better signal on "11b only" radios, which see most of 802.11g as "noise", thus filtering better the usable signal.

How to fine-tune the wireless link with hw-retries?

You should understand that for 802.11 devices there is really limited amount of information (or "feedback" from the environment) that devices can use to tune their behavior:

- signal strength, which could be used to figure out best transmit rate knowing receiver sensitivity. Sill this is not reliable taking into account that sensitivity for different receivers varies (e.g. changes over time), path conditions are not symmetric (and device can only measure signal strength it receives), etc.
- by receiving/not receiving acknowledgment for frame sent.

Taking into account that using signal strength is not reliable, 802.11 device is essentially left with only one "feedback" to tune its operation - success/failure of transmission. When transmission fails (ACK not received in time), there is no way how sender can figure out why it failed - either because of noise, multipath, direct interference (and wether that disturbed actual data frame or the ACK itself) - frame just did not make it and in general it does not matter "why". All that matters is packet error rate.

Therefore RouterOS implements algorithm to try to use medium most efficiently in any environment by using only this limited information, giving users the ability to control how algorithm works and describing the algorithm. And there are only a few usage guidelines, not a set of values you should use in particular situation.

In general - the larger *hw-retries*, the better "feedback" device gets about medium ability to deliver frame at particular rate (e.g. if sending frame with rate 54mbps fails 16 times, it is telling you more than if it fails with 2 retries) and the better it can figure out optimal transmit rate, at the expense of latency it can introduce in network - because during all those failing retries, other devices in this channel can not send. So **bigger** *hw-retries* can be suggested for ptp backbone links - where it is known that link must be always on. **Less** *hw-retries* make rate selection adapt faster at the expense of some accuracy (going below 2 is not reasonable in most cases), this can be suggested for ptmp links, where it is normal for links to connect/disconnect and keeping latency down is important.

on-fail-retry-time and *disconnect-timeout* controls how hard device will try to consider remote party "connected". Larger *disconnect-timeout* will make device not "disconnect" other party, even if there are lots of loss at the smallest possible transmission rate. This again is most useful for "weak" links that are known that they "must" be established (e.g. backbone links). In ptmp networks large *disconnect-timeout* will again increase latency in network during the time e.g. AP will attempt to send data to some client that has just been disabled (AP will try to do this for whole *disconnect-timeout*).

frame-lifetime allows to tune for how long AP is attempting to use frame for transmitting before considering that it is not worth delivering it (for example, if sending frame fails at lowest possible rate, *on-fail-retry-time* timer is enabled, if during this timer *frame-lifetime* expires, particular frame is dropped and next transmission attempt will happen with next frame. Disabled *frame-lifetime* means that wireless will ensure in order delivery of "all" data frames, no matter how long it takes, "or" will drop the connection if everything fails). This allows to optimize for different types of traffic e.g. for real-time traffic - if primary use of wireless network is e.g. voip, then it can be reasonable to limit *frame-lifetime*, because voip tolerates small loss better than high latency.

Is it possible to use the wireless repeater only with one radio interface?

This setup it possible by using WDS on the wireless interface which is running in ap-bridge mode.

Nv2 wireless link disconnects very often

When Nv2 wireless link experiences disconnections and in log section most of the messages are 'control frame timeout', then make sure that the router's RouterOS version is at least v5.14. v5.14 introduced several improvements for the Nv2 wireless protocol. Second suggestion is to lower the transmit output power of the wireless cards if the signal of the link is strong. We suggest to use tx-power-mode=card-rates for lowering the tx-power of the wireless card. If the problem continues try to use different wireless frequency that might have less interference. If that also

didn't help, please contact support@mikrotik.com with a support output file from the effected AP and the Station which are made after those disconnections.

References

[1] http://www.patentstorm.us/patents/7349503.html

Manual:WMM

How WMM works

WMM works by dividing traffic into 4 access categories: background, best effort, video, voice. QoS policy (different handling of access categories) is applied on transmitted packets, therefore it is transmitting device is treating different packets differently - that is - e.g. AP does not have control over how clients are transmitting packets, and clients do not have control over how AP transmits packets.

Mikrotik AP and client classifies packets based on priority assigned to them, according to table (as per WMM spec): 1,2 - background 0,3 - best effort 4,5 - video 6,7 - voice

To be able to use multiple WMM access categories, not just best effort where all packets with default priority 0 go, priority must be set for those packets. By default all packets (incoming and locally generated) inside router have priority 0.

"Better" access category for packet does not necessarily mean that it will be sent over the air before all other packets with "worse" access category. WMM works by executing DCF method for medium access with different settings for each access category (EDCF), which basically means that "better" access category has higher probability of getting access to medium - WMM enabled station can be considered to be 4 stations, one per access category, and the ones with "better" access category use settings that make them more likely to get chance to transmit (by using shorter backoff timeouts) when all are contending for medium. Details can be studied in 802.11e and WMM specification

How to set priority

Priority of packets can be set using "set priority" action of ip firewall mangle rules and/or bridge firewall filter rules. Priority can be set to specific value or to "ingress priority". Ingress priority is priority value that was detected on incoming packet, if available. Currently there are 2 sources of ingress priority - priority in VLAN header and priority from WMM packets received over wireless interface. For all other packets ingress priority is 0.

Note: Starting from v6.x priority can be set from DSCP by setting **from-dscp-high-3-bits**

Note that ingress priority value is not automatically copied to priority value, correct rule needs to be set up to do this!

So there are basically 2 ways to control/set priority (remember, that both require setting up correct rule(s)!): - assign priority with rules with particular matchers (protocol, addresses, etc), - set it from ingress priority.

This essentialy means that if it is not possible or wanted to classify packets by rules, configuration of network must be such that router can extract ingress priority from incoming frames. Remember there are currently 2 sources for this - VLAN tag in packets and received WMM packets.

Do not mix priority of queues with priority assigned to packets. Priorities of queues work separately and specify "importance" of queue and has meaning only within particular queue setup. Think of packet priority as of some kind of mark, that gets attached to packet by rules. Also take into account that this mark currently is only used for

outgoing packets when going over WMM enabled link, and in case VLAN tagged packet is sent out (no matter if that packet is tagged locally or bridged).

Example

For example, in setup

PPPoE server -> WMM AP -> client,

if AP is just forwarding PPPoE traffic (therefore inspecting encapsulated IP packets to match e.g. by protocol is not possible, as packets can be encrypted and compressed), priority must come to AP from PPPoE server in VLAN tag, so you have to use VLAN (between PPPoE server and AP) for this, just to communicate priority information.

Note that you do not have to forward VLAN encapsulated traffic to client - VLAN can be terminated at AP, VLAN tag is needed only when entering AP.

In case AP is PPPoE server itself, there is no need to use VLAN - priority can be set by rules before it is encapsulated in PPPoE.

Priority from DSCP

Another way of setting priority is by using DSCP field in IP header, this can only be done by firewall mange rule "set priority" action. Note that DSCP in IP header can have values 0-63, but priority only 0-7. Effective priority after set from DSCP value will be 3 low bits of DSCP value which is the same as reminder of division by 8. So for example, priority from DSCP values 0,8,16,etc will be 0, from DSCP values 7,15,...,63 - 7.

Remember that DSCP can only be accessed on IP packets!

Note, that to use this feature, DSCP value in IP header should be set somewhere.

It is best to set DSCP value in IP header of packets on some border router (e.g. main router used for connection to internet), based on traffic type. E.g. set DSCP value for packets coming from internet belonging to sip connections to 7, and 0 for the rest. This way packets must be marked only at one place. Then all APs in network set packet priority from DSCP value with just one rule.

In setup:

<internet> - border router - <network> - WMM AP - client

border router sets DSCP value for sip traffic, and WMM AP sets priority from DSCP value. Note that in this setup DSCP is set only for traffic _to_ client. Sometimes it can be useful to set also DSCP on traffic coming _from_ client (e.g. if 2 clients connected to different APs are talking between themselves) - this can be done on APs.

Combining priority setting and handling solutions

Complex networks and different situations can be handled by combining different approaches of carrying priority information to ensure QoS and optimize use of resources, based on "building blocks" described above. Several suggestions:

- the less number of filter rules in whole network, the better (faster) - try to classify packets only when necessary, prefer to do that on fast routers as most probably connection tracking will be required.

- use DSCP to carry priority information in IP packets forwarded in your network, this way you can use it when needed.

- use VLANs where necessary, as they also carry priority information, make sure ethernet bridges and switches in the way, if any, are not clearing priority information in VLAN tag. In MT bridges you have to setup bridge firewall rule to set priority from ingress priority for this!

- remember that QoS does not improve throughput of links, it just treats different packets differently, and also that WMM traffic over wireless link will discriminate regular traffic in the air.

Manual:Tools/Wake on lan

Applies to RouterOS: v3, v4

This tool is introduced in RouterOS since v3.23 and can send the Wake on LAN MagicPacket to any MAC address of your choosing. If the target device supports WOL, it should wake from sleep. Secure WOL is not supported.

```
[admin@MikroTik] > tool wol mac=FF:FF:FF:FF:FF:FF
```

The WOL tool will send a UDP MagicPacket to the Broadcast address with the MAC address embedded in it.

By default, the magic packet will be sent as an IP broadcast out the default gateway interface, but if you want, you can tell the command to use a specific interface:

```
[admin@MikroTik] > tool wol interface=ether1 mac=FF:FF:FF:FF:FF:FF
```

Manual:Webfig

Summary

WebFig is a web based RouterOS utility which allows you to monitor, configure and troubleshoot the router. It is designed as an alternative of WinBox, both have similar layouts and both have access to almost any feature of RouterOS.

WebFig is accessible directly from the router which means that there is no need to install additional software (except web browser with JavaScript support, of course).

As Webfig is platform independent, it can be used to configure router directly from various mobile devices without need of a software developed for specific platform.

Some of the tasks that you can perform with WebFig:

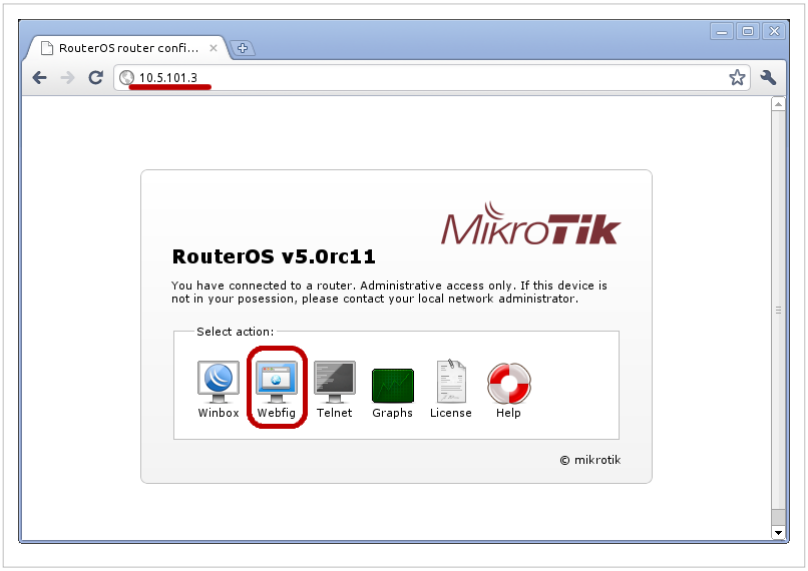
- Configuration - view and edit current configuration;
- Monitoring - display the current status of the router, routing information, interface stats, logs and many more;
- Troubleshooting - RouterOS has built in many troubleshooting tools (like ping, traceroute, packet sniffers, traffic generators and many other) and all of them can be used with WebFig.

Connecting to Router

WebFig can be launched from the routers home page which is accessible by entering routers IP address in the browser. When home page is successfully loaded, choose webfig from the list of available icons as illustrated in screenshot.

After clicking on webfig icon, login prompt will ask you to enter username and password. Enter login information and click connect.

Now you should be able to see webfig in action.

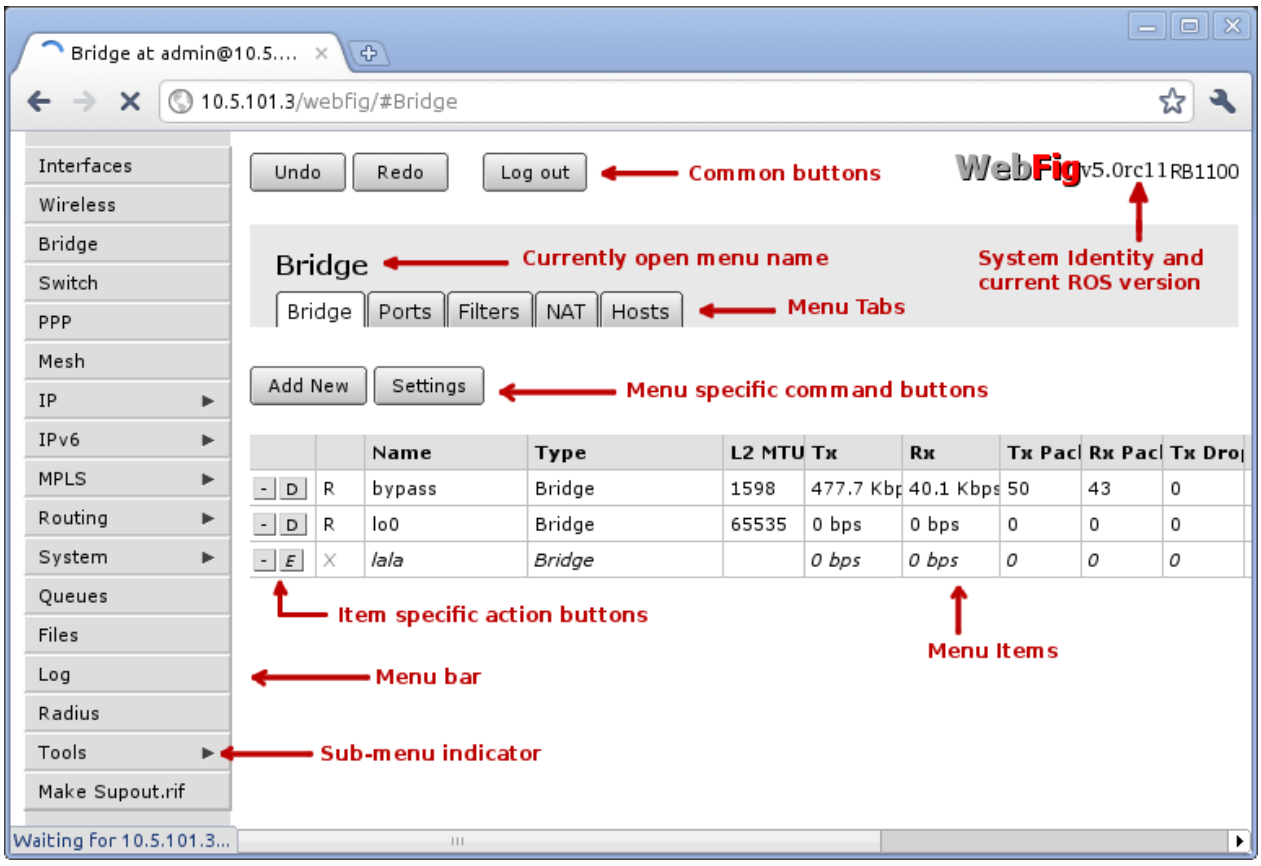


IPv6 Connectivity

RouterOS http service now listens on ipv6 address, too. To connect to IPv6, in your browser enter ipv6 address in square brackets, for example **[2001:db8:1::4]**. If it is required to connect to link local address, don't forget to specify interface name or interface id on windows, for example **[fe80::9f94:9396%ether1]**.

Interface Overview

WebFig interface is designed to be very intuitive especially for WinBox users. It has very similar layout: menu bar on the left side, undo/redo at the top and work are at the rest of available space.



When connected to router, browsers title bar (tab name on Chrome) displays currently opened menu, user name used to authenticate, ip address, system identity, ROS version and RouterBOARD model in following format:

```
[menu] at [username]@[Router's IP] ( [RouterID] ) - Webfig [ROS version] on [RB model] ([platform])
```

Menu bar has almost the same design as WinBox menu bar. Little arrow on the right side of the menu item indicates that this menu has several sub-menus.

When clicking on such menu item, sub-menus will be listed and the arrow will be pointing down, indicating that sub-menus are listed.

At the top you can see three common buttons Undo/Redo buttons similar to winbox and one additional button *Log Out*. In the top right corner, you can see WebFig logo and RouterBOARDS model name.

Work area has tab design, where you can switch between several configuration tabs, for example in screenshot there are listed all tabs available in Bridge menu (Bridge, Ports, Filters, NAT, Rules).

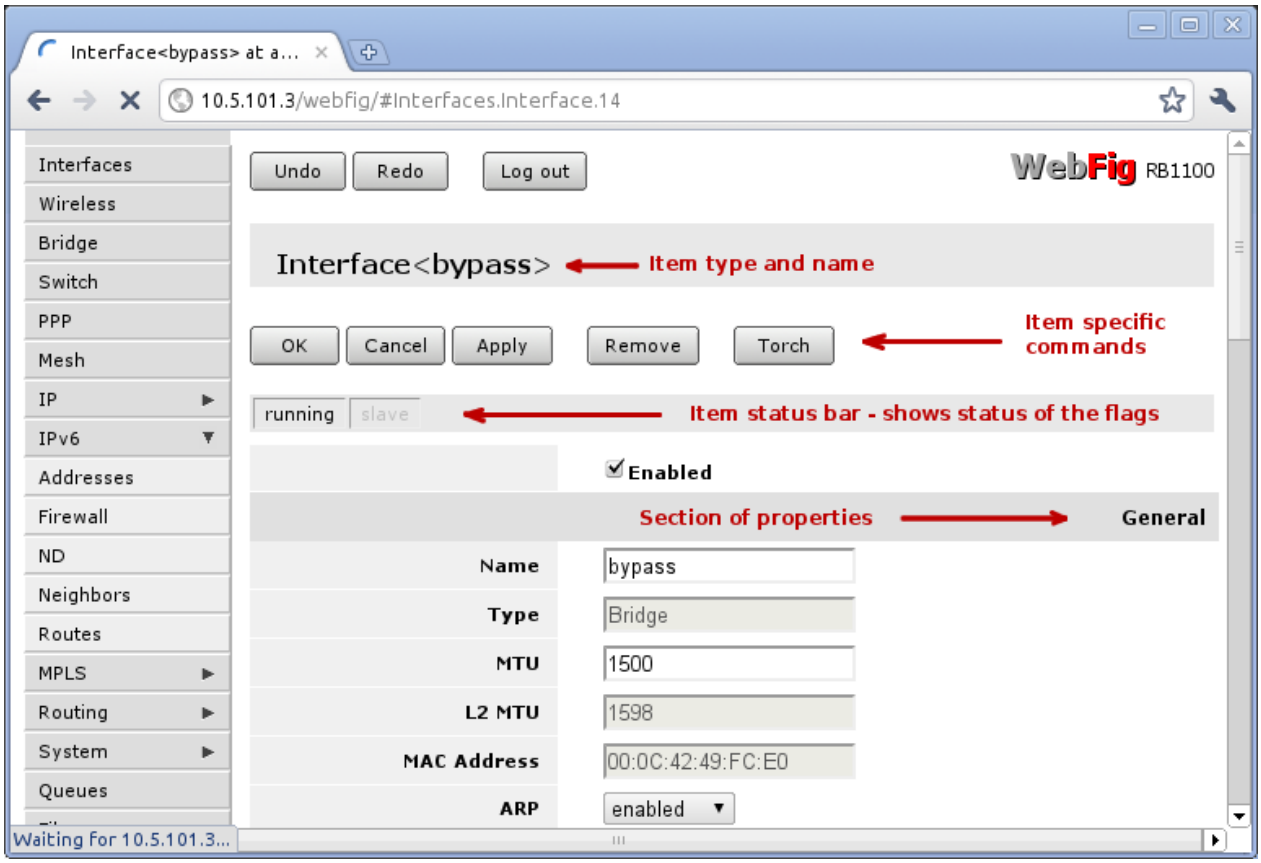
Below the tabs are listed buttons for all menu specific commands, for example *Add New* and *Settings*.

The last part is table of all menu items. First column of an item has item specific command buttons:

- E - enable current item
- D - disable current item
- - - remove current item

Item configuration

When clicking on one of the listed items, webfig will open new page showing all configurable parameters, item specific commands and status.



At the top you can see item type and item name. In example screenshot you can see that item is an *interface* with name **bypass**

There are also item specific command buttons (Ok, Cancel, Apply, Remove and Torch). These can vary between different items. For example *Torch* is available only for interfaces.

Common Item buttons:

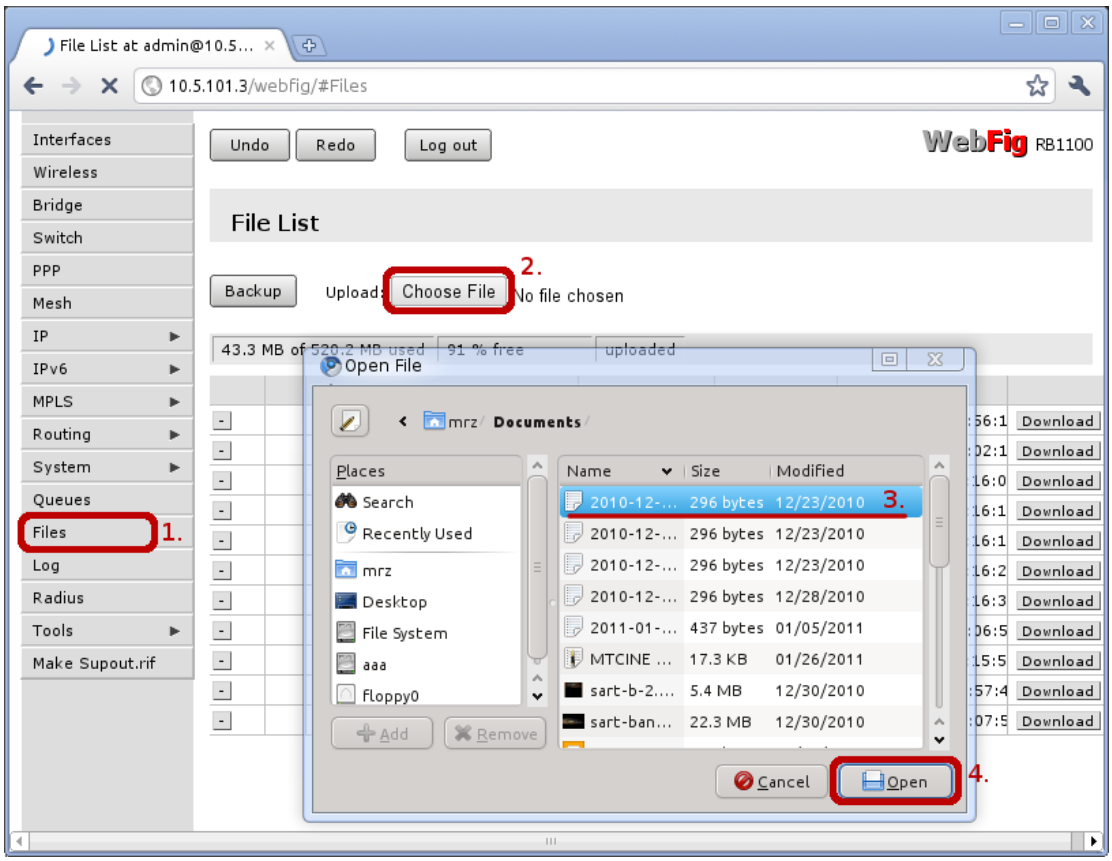
- *Ok* - apply changes to parameters and exit;
- *Cancel* - exit and do not apply changes;
- *Apply* - apply changes and stay on current page;
- *Remove* - remove current item.

Status bar similar to winbox shows current status of item specific flags (e.g running flag). Grey-ed out flag means that it is not active. In example screenshot you can see that *running* is in solid black and *slave* is grey-ed, which means that interface is running and is not a slave interface.

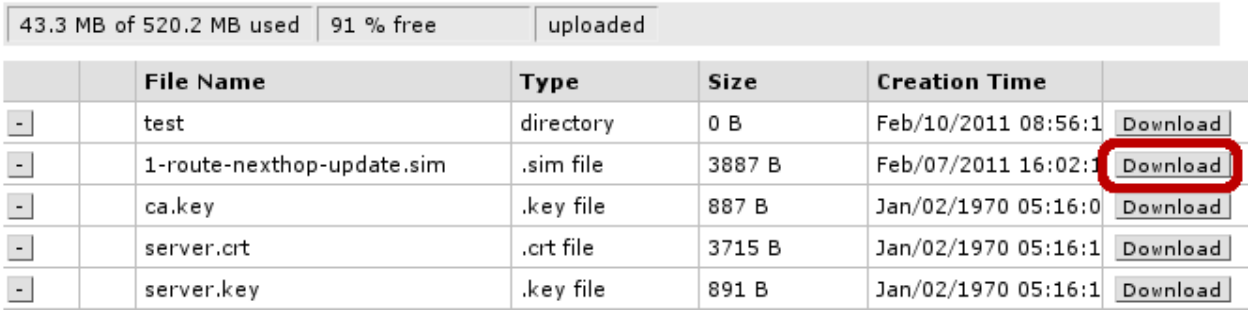
List of properties is divided in several sections, for example "General", "STP", "Status", "Traffic". In winbox these sections are located in separate tabs, but webfig lists them all in one page specifying section name. In screenshotyou can see "General" section. Grey-edout properties mean that they are read-only and configuration is not possible.

Work with Files

Webfig allows to upload files directly to the router, without using FTP services. To upload files, open **Files** menu, click on **Choose File** button, pick file and wait until file is uploaded.



Files also can be easily downloaded from the router, by clicking Download button at the right side of the file entry.



Traffic Monitoring

[Top | Back to Content]

Skins

Webfig skins is handy tool to make interface more user friendly. It is not a security tool. If user has sufficient rights it is possible to access hidden features by other means.

Designing skins

If user has sufficient permissions (group has policy edit permissions) **Design Skin** button becomes available. Pressing that toggle button will open interface editing options. Possible operations are:

- Hide menu - this will hide all items from menu and its submenus;

- Hide submenu - only certain submenu will be hidden
- Hide tabs - if submenu details have several tabs, it is possible to hide them this way;
- Rename menus, items - make some certain features more obvious or translate them into your launguage;
- Add note to to item (in detail view) - to add comments on filed;
- Make item read-only (in detail view) - for user safety very sensitive fields can be made read only
- Hide flags (in detail view) - while it is only possible to hide flag in detail view, this flag will not be visible in list view and in detailed view;
- Add limits for field - (in detail view) where it is list of times that are comma or newline separated list of allowed values:
  - number interval '..' example: 1..10 will allow values from 1 to 10 for fiels with numbers, example, MTU size.
  - field prefix (Text fields, MAC address, set fields, combo-boxes). If it is required to limit prefix length *$* should be added to the end, for example, limiting wireless interface to "station" only will contain
- Add *Tab* - will add grey ribbon with editable label that will separate the fields. Ribbon will be added before field it is added to;
- Add *Separator* - will add low height horizontal separator before the field it is added to.

Note: Number interval cannot be set to extend limitations set by RouterOS for that field

Note: Set fields are argument that consist of set of check-boxes, for example, setting up policies for user groups, RADIUS "Service"

Note: Limitations set for combo-boxes will values selectable from dropdown

Configure wireless interface

To configure

Status page

Note: Starting RouterOS 5.7 webfig interface adds capability for users to create status page where fields from anywhere can be added and arranged.

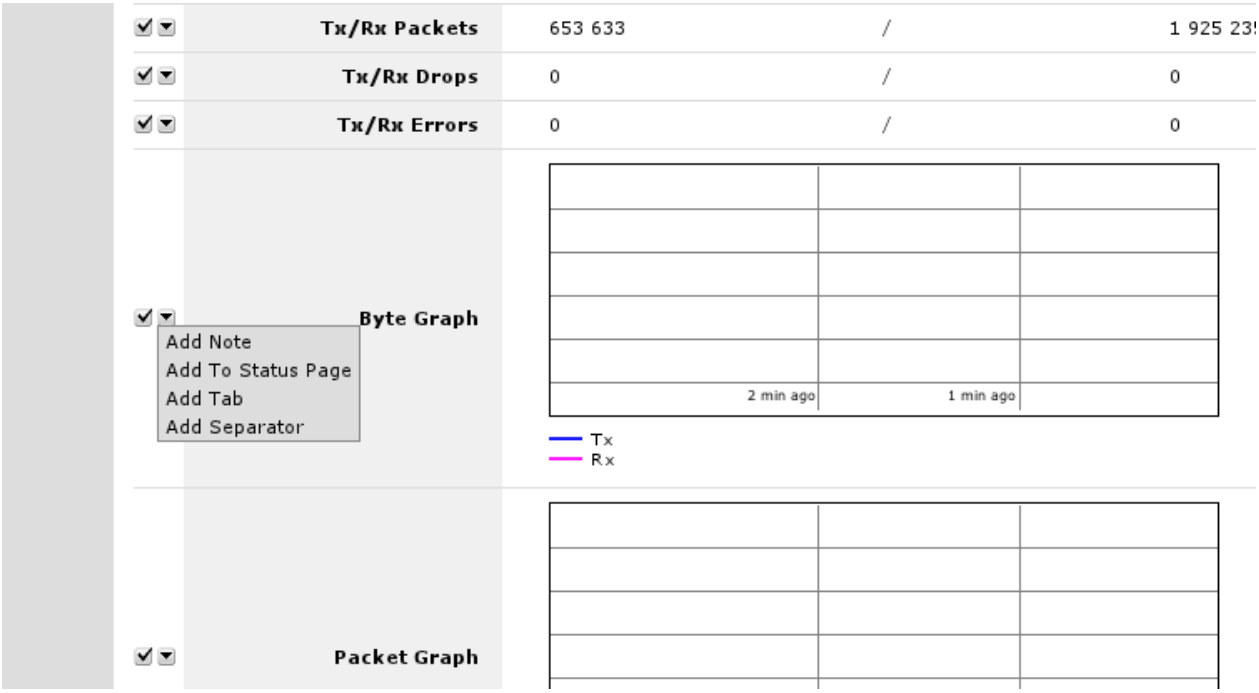
Satus page can be created by users (with sufficient permissions) and fields on the page can be reordered.

When status page is created it is default page that opens when logging in the router through webfig interface.

Addition of fields

To add field to status page user has to enter "Design skin" mode and from drop-down menu at the field choose option - "Add to status page"

As the result of this action desired field in read-only mode will be added to status page. If at the time *Status* page is not present at the time, it will be created for the user automatically.



Two columns

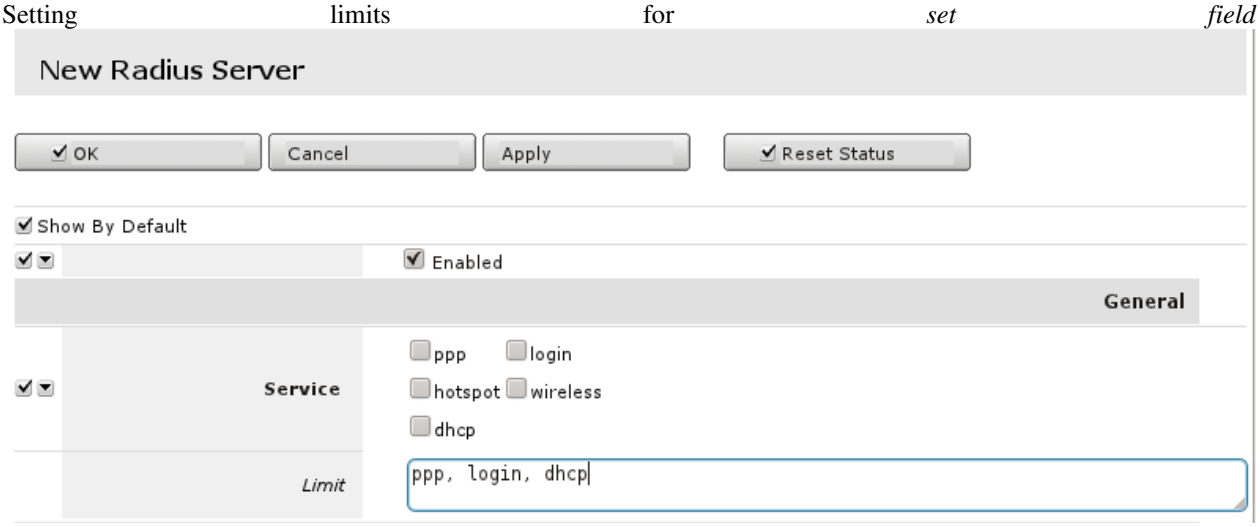
Fields in *Status* page can be arranged in two columns. Columns are filled from top to bottom.

When you have only one column then first item intended for second should be dragged to the top of the first item when black line appear on top of the first item, then drag mouse to the left until shorter black line is displayed as showed in screenshot. Releasing mouse button will create second column. Rest of the fields afterwards can be dragged and dropped same way as with one column design.

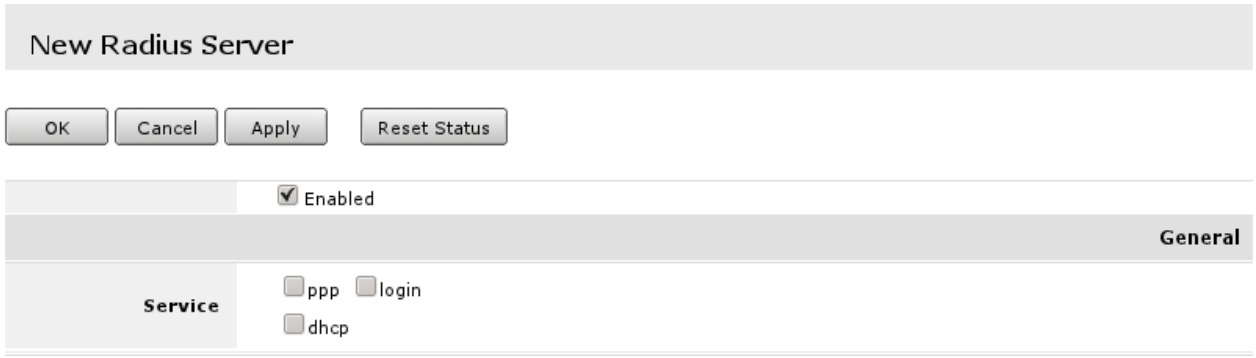
Skin design examples

Set field

Setting limits for *set field*



And the result:



Using skins

To use skins you have to assign skin to group, when that is done users of that group will automatically use selected skin as their default when logging into Webfig.

Note: Webfig is only configuration interface that can use skins

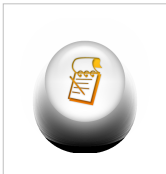
If it is required to use created skin on other router you can copy files to **skins** folder on the other router. On new router it is required to add copied skin to user group to use it.

[Top | Back to Content]

Manual:Wireless Advanced Channels

Applies to RouterOS: v6

Overview



Note: This article describes features not yet available to general public. It's expected to be released in RouterOS v6

Advanced Channels feature provides extended opportunities in wireless interface configuration:

- scan-list that covers multiple bands and channel widths;
- non-standard channel center frequencies (specified with KHz granularity) for hardware that allows that;
- non-standard channel widths (specified with KHz granularity) for hardware that allows that.

Configuring Advanced Channels

Advanced Channels are configured in **interface wireless channels** menu. This menu contains ordered list of user-defined channels that can be grouped by means of **list** property. Channels have the following properties:

- **name** - name by which this channel can be referred to. If **name** is not specified when adding channel, it will be automatically generated from channel frequency and width;
- **list** - name of list this channel is part of. Lists can be used to group channels;
- **frequency** - channel center frequency in MHz, allowing to specify fractional MHz part, e.g. **5181.5**;
- **width** - channel width in MHz, allowing to specify fractional MHz part, e.g. **14.5**;
- **band** - defines default set of data rates when using this channel;
- **extension-channel** - specifies placement of 11n extension channel.

Using Advanced Channels

In order to use Advanced Channels in wireless interface configuration, several interface settings accept channel names or list names as arguments. It is possible to configure interface with channel that interface does not support. In this case interface will not become operational. It is sole responsibility of administrator to configure channels in proper way.

frequency

To use particular Advanced Channel for wireless interface (applies to modes that make use of interface **frequency** setting) specify channel name in interface **frequency** setting. For example, to configure interface to operate with center frequency 5500MHz and channel width 14MHz, use the following commands:

```
[admin@MikroTik] /interface wireless> channels add name=MYCHAN frequency=5500 width=14 band=5ghz-onlyn
 list=MYLIST
[admin@MikroTik] /interface wireless> set wlan1 frequency=MYCHAN
```

scan-list

Interface **scan-list** is used in multiple modes that either gather information for list of channels (like interactive **scan** command) or selects channel to work on (like any of **station** modes or AP modes performing DFS). Interface **scan-list** can be configured with comma-separated list of the following items:

- **default** - default .11 channel list for given country and interface band and channel width;
- numeric frequency ranges in MHz;
- Advanced Channel, referred to by name;
- Advanced Channel list, referred to by list name.

For example, to configure interface to scan 5180MHz, 5200MHz and 5220MHz at first using channel width 20MHz and then using channel width 10MHz, the following commands can be issued:

```
[admin@MikroTik] /interface wireless> channels add frequency=5180 width=20 band=5ghz-a list=20MHz-list
[admin@MikroTik] /interface wireless> channels add frequency=5200 width=20 band=5ghz-a list=20MHz-list
[admin@MikroTik] /interface wireless> channels add frequency=5220 width=20 band=5ghz-a list=20MHz-list
[admin@MikroTik] /interface wireless> channels add frequency=5180 width=10 band=5ghz-a list=10MHz-list
[admin@MikroTik] /interface wireless> channels add frequency=5200 width=10 band=5ghz-a list=10MHz-list
[admin@MikroTik] /interface wireless> channels add frequency=5220 width=10 band=5ghz-a list=10MHz-list
[admin@MikroTik] /interface wireless> set wlan1 scan-list=20MHz-list,10MHz-list
```

Hardware support

Non standard center frequency and width channels can only be used with interfaces that support it.

Currently only Atheros AR92xx based chips support non-standard center frequencies and widths with the following ranges:

- center frequency range: 2200MHz-2500MHz with step 0.5MHz (500KHz), width range: 2.5MHz-30MHz width step 0.5MHz (500KHz);
- center frequency range: 4800MHz-6100MHz with step 0.5MHz (500KHz), width range: 2.5MHz-30MHz width step 0.5MHz (500KHz);

Manual:Wireless AP Client

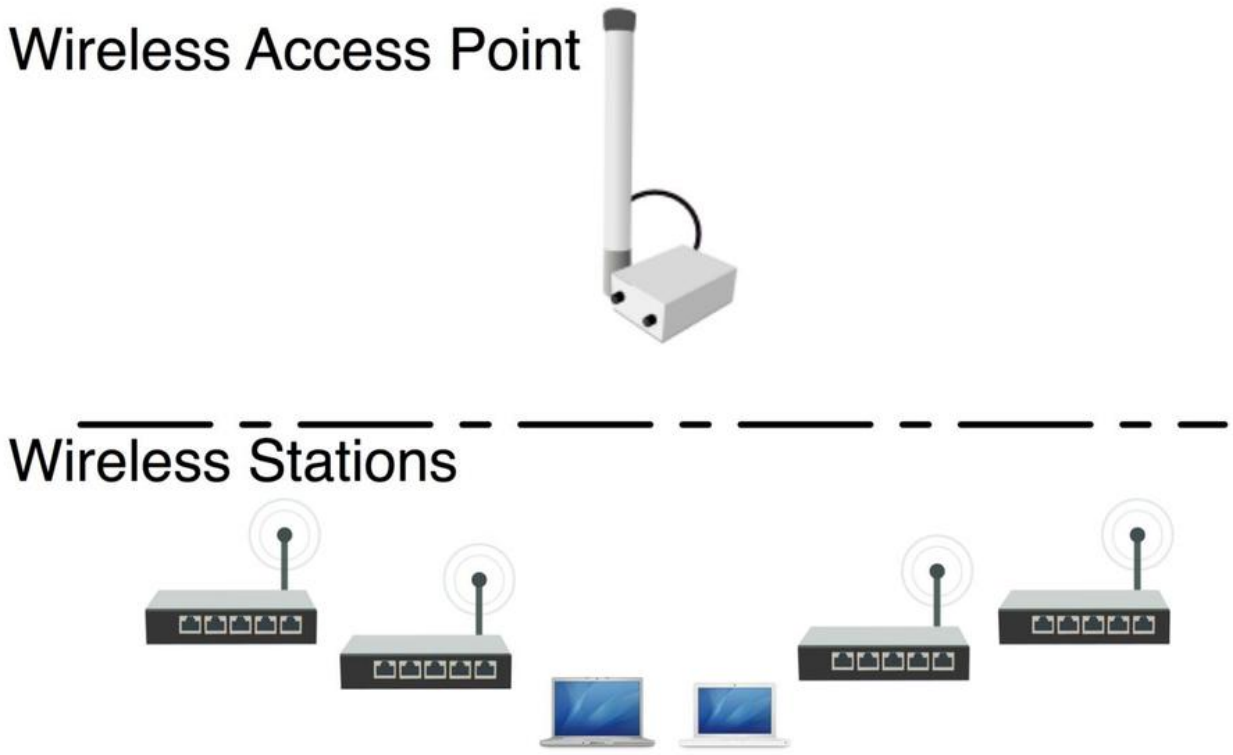
Applies to RouterOS: v3, v4

Summary

Configuration example shows how to establish simple wireless network by using MikroTik RouterOS. MikroTik RouterOS is fully compliant with IEEE802.11a/b/g/n standards, MikroTik RouterOS device [1] can be used as wireless access-point and wireless station (other modes [2] are supported too).

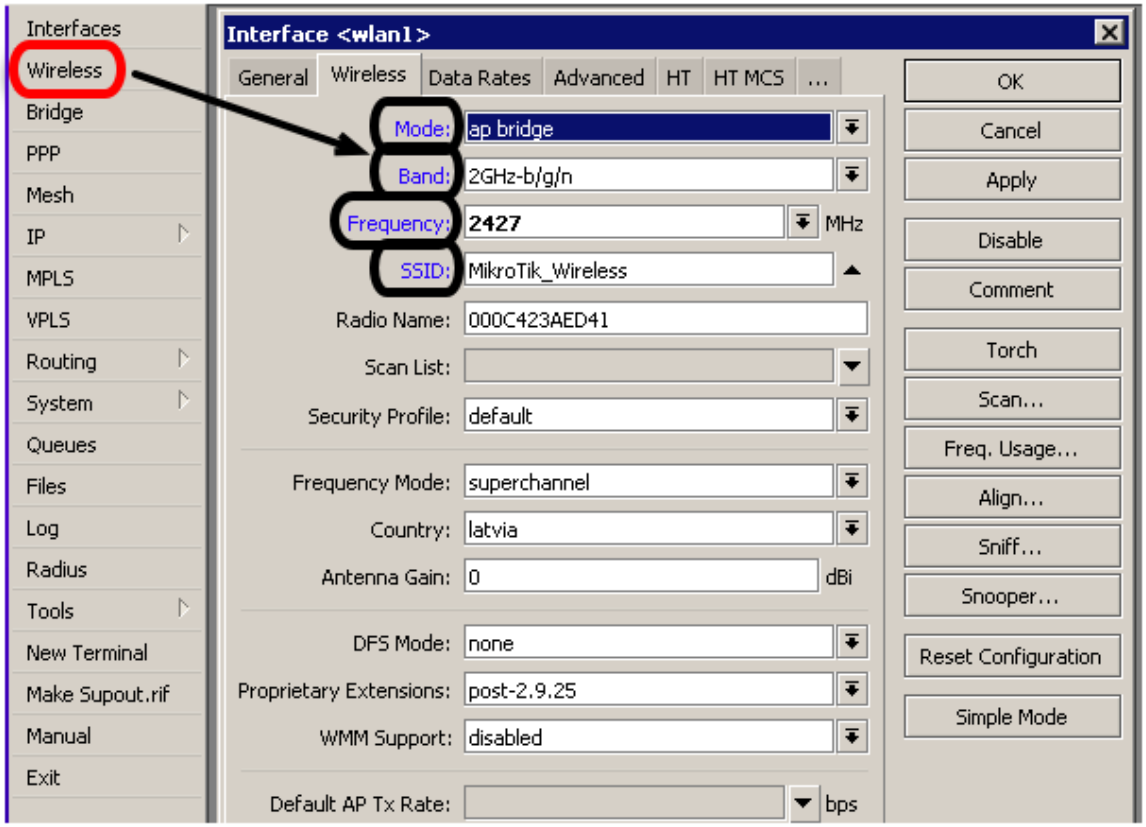
Configuration setup

Our basic configuration setup is



Access Point Configuration

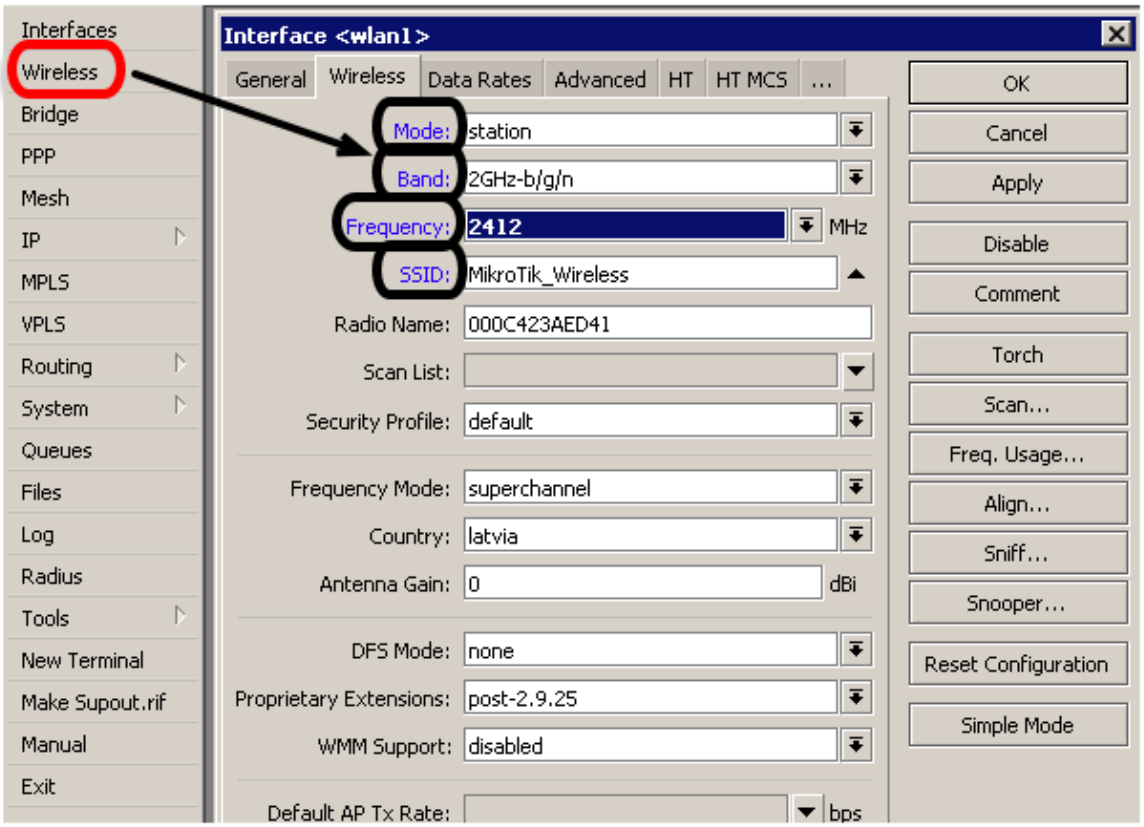
- Connect to the router via Winbox [3]
- Setup **Wireless** interface, necessary configuration options are **mode**=ap-bridge **band**=ap_operated_band **frequency**=ap_operated_frequency **ssid**=network_identification



- These settings are enough to establish wireless connection, additionally you need to add IP address for the wireless interface for IP routing, optionally add security and other settings.

Station Configuration

- Wireless client configuration example is for MikroTik RouterOS, other vendor OS configuration should be looked in the appropriate documentation/forum/mailing list etc.
- Connect to the client router via the same way and proceed to the **Wireless** interface configuration.
- Necessary configuration options are **mode**=station **band**=band_ap_operates_on **ssid=ap_network_ssid**

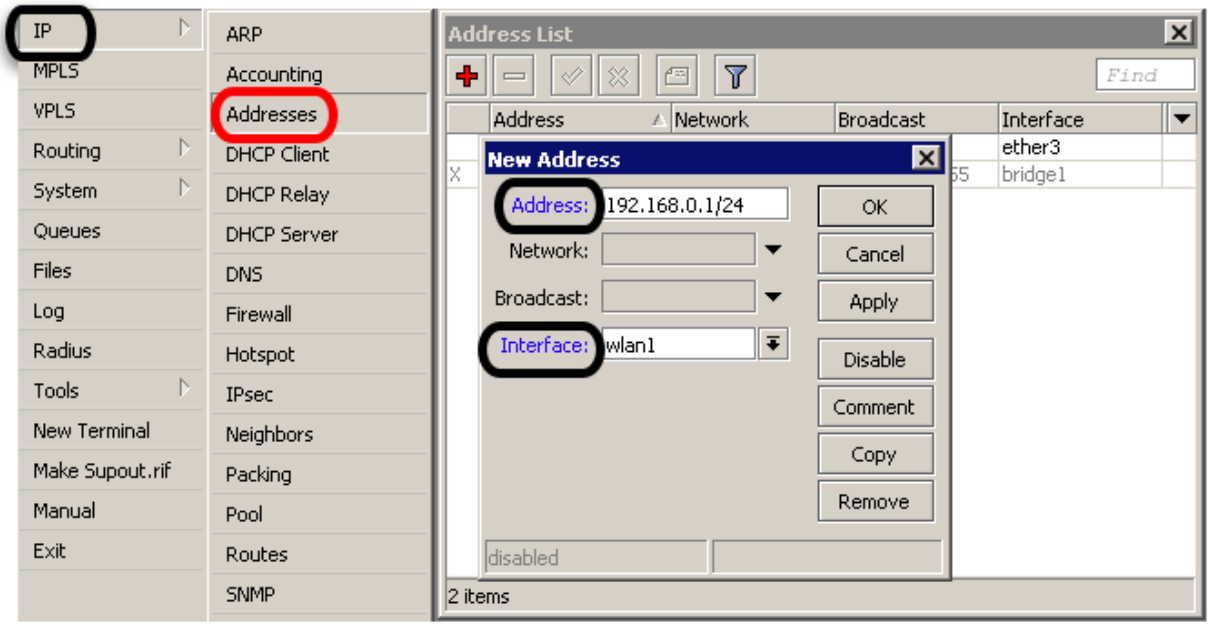


- These settings are enough to establish wireless connection, additionally you need to set IP address for the wireless interface to establish IP routing communication with access point, optionally use security and other settings.

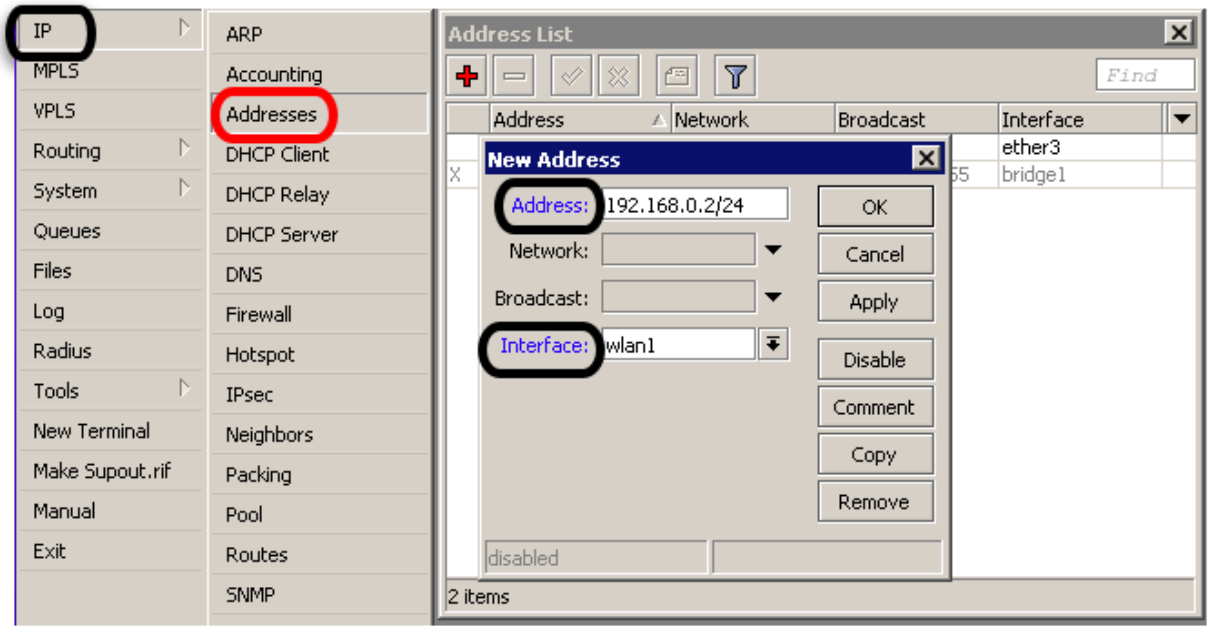
Additional Configuration

IP Configuration

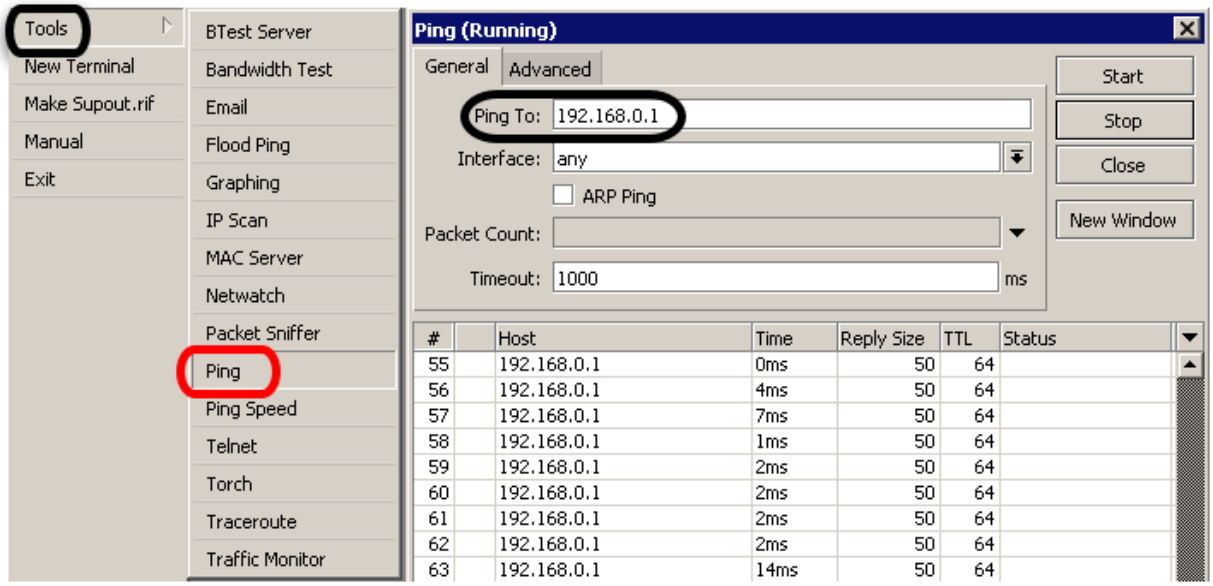
- Add IP address to Access Point router, like 192.168.0.1/24



- Add IP address to Client router, address should be from the same subnet like 192.168.0.2/24



- Check IP communication by ping from station (for example),



Additional Access Point Configuration

- All the necessary settings for the simple Access Point are showed here [4].
- Security profiles are used for WPA/WPA2 protection, configuration options are explained here [5]. Usually all wireless clients share the same security configuration as access point.
- **mode**=ap-bridge allows 2007 clients, max-station-count is used to limit the number of wireless client per Access Point. Wireless **mode**=bridge is used for point-to-point wireless links and allows connection for one station only.
- MikroTik RouterOS license level4 is minimum for mode=ap-bridge
- Other wireless settings are (http://wiki.mikrotik.com/wiki/Category:Wireless explained here)

Additional Station Configuration

- Station adapts to wireless access point frequency, despite of the frequency configuration in Wireless menu. Station uses scan-list to select available Access Point, when superchannel mode is used on wireless Access Point, set custom Access Point frequency to mode=station **scan-list**.

References

[1] http://routerboard.com/
[2] http://wiki.mikrotik.com/wiki/Manual:Interface/Wireless#Wireless_interface_configuration
[3] http://wiki.mikrotik.com/wiki/First_time_startup
[4] http://wiki.mikrotik.com/wiki/Manual:Making_a_simple_wireless_AP
[5] http://wiki.mikrotik.com/wiki/Manual:Interface/Wireless#Security_profiles

Manual:Wireless Debug Logs

Debugging wireless problems using Logs.

By default RouterOS wireless log shows that client connects and disconnects as simple entries:

```
22:32:18 wireless,info 00:80:48:41:AF:2A@wlan1: connected
```

It is enough for regular users to know that the wireless client with MAC address "00:80:48:41:AF:2A" is connected to wireless interface "wlan1". But actually there are more log entries available than are shown in standard logging. They are called 'debug' logs which give more detailed information. In the following Debug Log example you will see the same client connecting to the AP in more detail than found in typical logging:

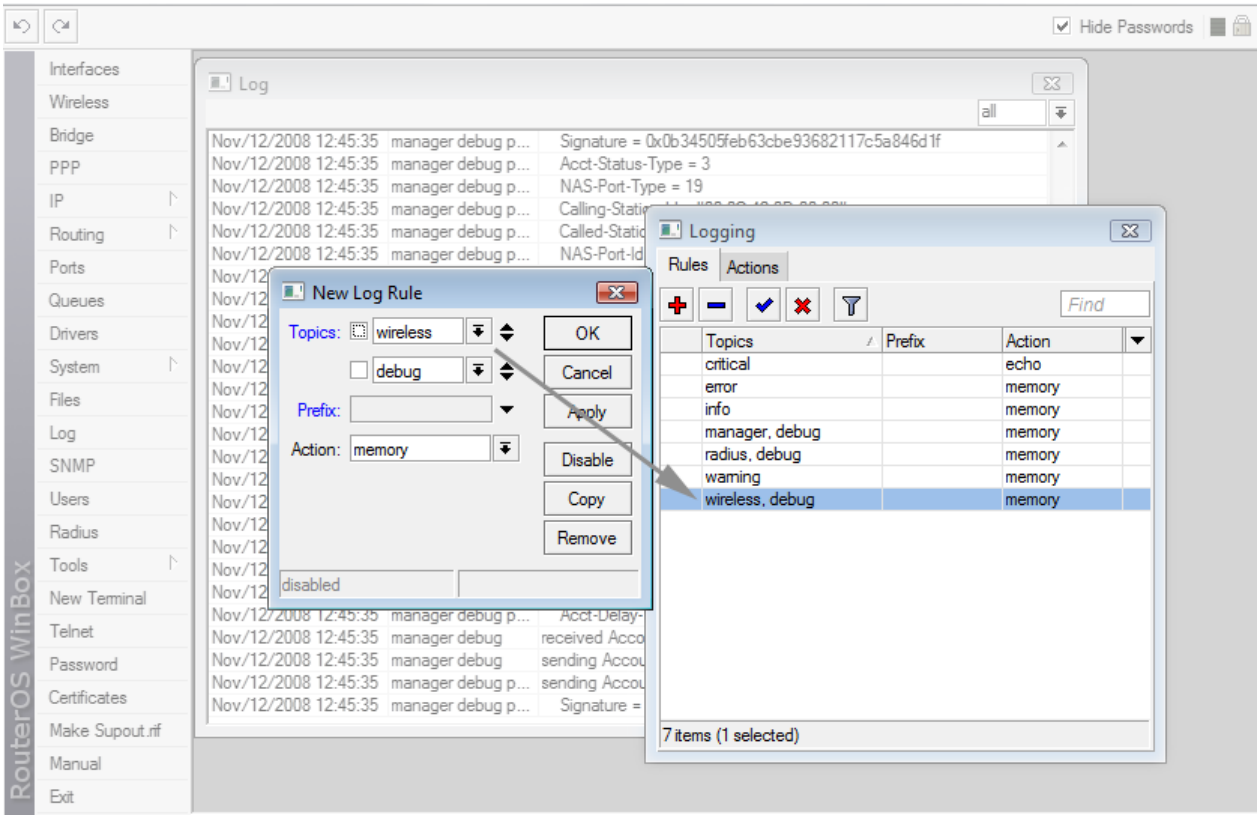
```
22:33:20 wireless,debug wlan1: 00:80:48:41:AF:2A attempts to connect
22:33:20 wireless,debug wlan1: 00:80:48:41:AF:2A not in local ACL, by default accept
22:33:20 wireless,info 00:80:48:41:AF:2A@wlan1: connected
```

Debug Logs will give you more specific informantion on each step of the Client wireless connection and disconnection. The first line shows that the wireless client tried to connect to the AP. On the second line the AP checked to see if that client is allowed to connect to the AP and the resulting action. And only on the third line do you see that the client is connected. This is merely one example of the debug log messages. The description of all debug entries is written below.

To enable the wireless debug logs you should execute such commands:

```
[admin@MikroTik] > /system logging
[admin@MikroTik] system logging> add topics=wireless,debug action=memory
```

Or in Winbox:



This will help you understand and fix wireless problems with ease and with less interaction with the support team.

STATION MODE

<MAC>@<DEV>: lost connection, <REASON>

Station has lost connection to AP because of <REASON>

<MAC>@<DEV>: failed to connect, <REASON>

Station attempted to connect to AP, but failed due to <REASON>

<MAC>@<DEV>: established connection on <FREQ>, SSID <SSID>

Station attempted and succesfully connected to AP with SSID <SSID> on frequency <FREQ>.

<MAC>@<DEV>: MIC failure!!!

TKIP message integrity check failure, somebody must be trying to break into or DOS network, If more than 1 MIC failure is encountered during 60s period, "TKIP countermeasures" state is entered.

<MAC>@<DEV>: enter TKIP countermeasures

Entered TKIP countermeasures state, this means that Station will disconnect from AP and keep silence for 60s.

AP MODE

<DEV>: radar detected on <FREQ>

Radar detected on frequency <FREQ>, AP will look for other channel

<DEV>: data from unknown device <MAC>, sent deauth [(XXX events suppressed, YYY deauths suppressed)]

Data frame from unknown device (read - not registered to this AP) with mac address <MAC> received, AP sent deauthentication frame to it (as per 802.11). XXX is number of events that are not logged so that the log does not become too large (logs are limited to 1 entry per 5s after first 5 entries), YYY is the number of deauthentication frames that should have been sent, but were not sent, so that resources are not wasted sending too many deauthentication frames (only 10 deauth frames per second are allowed).

The likely cause of such a message is that the Station previously connected to the AP, which does not yet know it has been dropped from AP registration table, sending data to AP. Deauthentication message tells the Station that it is no longer connected.

<DEV>: denying assoc to <MAC>, failed to setup compression

Failed to initialize compression on AP, most likely because there are too many clients attempting to connect and use compression.

<DEV>: <MAC> is new WDS master

WDS slave has established connection to WDS master, this means that WDS slave starts accepting clients and acting as AP.

<DEV>: <MAC> was WDS master

This message appears after connection with <MAC> is lost, means that WDS slave will disconnect all clients and start scanning to find new WDS master.

<MAC>@<DEV>: connected [, is AP][, wants WDS]

Station with address <MAC> connected. if "is AP" present - remote device is AP, if "is WDS" presents, remote device wants to establish WDS link.

<MAC>@<DEV>: disconnected, <REASON>

Connection with Station with address <MAC> terminated due to <REASON>

<DEV>: TKIP countermeasures over, resuming

TKIP countermeasures (60s silence period) over, AP resumes acting as AP.

<DEV>: starting TKIP countermeasures

Entering TKIP countermeasures state (60s silence period), all clients will be lost.

<REASON>

"joining failed" - can only happen on Prism cards in station mode, failed to connect to AP due to some reason

"join timeout" - happens on Station, failed to synchronize to AP (receive first beacon frame). Most likely weak signal, remote turned off, strong interference, some other RF related issue that makes communication impossible.

"no beacons" - no beacons received from remote end of WDS link. Most likely weak signal, remote turned off, strong interference, some other RF related issue that makes communication impossible.

"extensive data loss" - local interface decided to drop connection to remote device because of inability to send data to remote after multiple failures at lowest possible rate. Possible causes - too weak signal, remote device turned off, strong interference, some other RF related issue that makes communication impossible.

"decided to deauth, <802.11 reason>" - local interface decided do deauthenticate remote device using 802.11 reason <802.11 reason>.

"inactivity" - remote device was inactive for too long

"device disabled" - local interface got disabled

"got deauth, <802.11 reason>" - received deauthentication frame from remote device, 802.11 reason code is reported in <802.11 reason>

"got disassoc, <802.11 reason>" - received disassociation frame from remote device, 802.11 reason code is reported in <802.11 reason>

"auth frame from AP" - authentication frame from remote device that is known to be AP, most likely mode changes on remote device from AP to Station.

"bad ssid" - bad ssid for WDS link

"beacon from non AP" - received beacon frame from remote device that is known to be non-AP node, most likely mode changes on remote device from Station to AP.

"no WDS support" - does not report WDS support

"failed to confirm SSID" - failed to confirm SSID of other end of WDS link.

"hardware failure" - some hardware failure or unexpected behaviour. Not likely to be seen.

"lost connection" - can only happen on Prism cards in station mode, connection to AP lost due to some reason.

"auth failed <802.11 status>" - happens on Station, AP denies authentication, 802.11 status code is reported in <802.11 status>.

"assoc failed <802.11 status>" - happens on Station, AP denies association, 802.11 status code is reported in <802.11 status>.

"auth timeout" - happens on Station, Station does not receive response to authentication frames, either bad link or AP is ignoring this Station for some reason.

"assoc timeout" - happens on Station, Station does not receive response to association frames, either bad link or AP is ignoring this Station for some reason.

"reassociating" - happens on AP: connection assumed to be lost, because Station that is considered already associated attempts to associate again. All connection related information must be deleted, because during association process connection parameters are negotiated (therefore "disconnected"). The reason why Station reassociates must be looked for on Station (most likely cause is that Station for some reason dropped connection

without telling AP - e.g. data loss, configuration changes).

"compression setup failure" - connection impossible, because not enough resources to do compression (too many stations that want to use compression already connected)

<802.11 reason> and <802.11 status>

These are numeric reason/status codes encoded into 802.11 management messages. Log messages include numeric code and textual description from appropriate standard in 802.11 standards group. Although these are intended to be as descriptive as possible, it must be taken into account that actual reason/status code that appears in management frames depends solely on equipment or software manufacturer - where one device sends 802.11 management frame including proper reason/status code for situation that caused the frame, other may send frame with "unspecified" reason/status code. Therefore reason/status code should only be considered informational.

As 802.11 standards evolve, RouterOS may miss textual descriptions for reason/status codes that some devices use. In such case numeric value should be used to lookup meaning in 802.11 standards.

In order to properly interpret reason/status code, good understanding of 802.11 group standards is necessary. Most of the textual descriptions are self-explaining. Explanation for some of most commonly seen reson/status codes follows.

class 2 frame received (6) - device received "class 2" frame (association/reassociation management frame) before completing 802.11 authentication process;

class 3 frame received (7) - device received "class 3" frame (data frame) before completing association process;

See Also

- Management Frames and Connection/Disconnection messages [1] by GTHill.com

References

[1] http://www.gthill.com/managementframes.pdf

Manual:Wireless Station Modes

Overview

Wireless interface in any of *station* modes will search for acceptable access point (AP) and connect to it. The connection between station and AP will behave in slightly different way depending on type of *station* mode used, so correct mode must be chosen for given application and equipment. This article attempts to describe differences between available *station* modes.

Primary difference between *station* modes is in how L2 addresses are processed and forwarded across wireless link. This directly affects the ability of wireless link to be part of L2 bridged infrastructure.

If L2 bridging over wireless link is not necessary - as in case of routed or MPLS switched network, basic **mode=station** setup is suggested and will provide highest efficiency.

Availability of particular *station* mode depends on **wireless-protocol** that is used in wireless network. Please refer to applicability matrix for information on mode support in protocols. It is possible that connection between station and AP will be established even if particular mode is not supported for given protocol. Beware that such connection will not behave as expected with respect to L2 bridging.

802.11 limitations for L2 bridging

Historically 802.11 AP devices were supposed to be able to bridge frames between wired network segment and wireless, but station device was not supposed to do L2 bridging.

Consider the following network:

```
[X]---[AP]-(      )-[STA]---[Y]
```

where X-to-AP and STA-to-Y are ethernet links, but AP-to-STA are connected wirelessly. According to 802.11, AP can transparently bridge traffic between X and STA, but it is not possible to bridge traffic between AP and Y, or X and Y.

802.11 standard specifies that frames between station and AP device must be transmitted in so called *3 address* frame format, meaning that header of frame contains 3 MAC addresses. Frame transmitted from AP to station has the following addresses:

- destination address - address of station device, also radio receiver address
- radio transmitter address - address of AP
- source address - address of originator of particular frame

Frame transmitted from station to AP has the following addresses:

- radio receiver address - address of AP
- source address - address of station device, also radio transmitter address
- destination address

Considering that every frame must include radio transmitter and receiver address, it is clear that *3 address* frame format is not suitable for transparent L2 bridging over station, because station can not send frame with source address different from its address - e.g. frame from Y, and at the same time AP can not format frame in a way that would include address of Y.

802.11 includes additional frame format, so called *4 address* frame format, intended for "wireless distribution system" (WDS) - a system to interconnect APs wirelessly. In this format additional address is added, producing header that contains the following addresses:

- radio receiver address

- radio transmitter address
- destination address
- source address

This frame format includes all necessary information for transparent L2 bridging over wireless link. Unluckily 802.11 does not specify how WDS connections should be established and managed, therefore any usage of *4 address* frame format (and WDS) is implementation specific.

Different *station* modes attempt to solve shortcomings of standard station mode to provide support for L2 bridging.

Applicability Matrix

The following matrix specifies *station* modes available for each **wireless-protocol**. Note that there are 2 columns for 802.11 protocol: **802.11** specifies availability of mode in "pure" 802.11 network (when connecting to any vendor AP) and **ROS 802.11** specifies availability of mode when connecting to RouterOS AP that implements necessary proprietary extensions for mode to work.

Table applies to RouterOS v5rc11 and above:

	802.11	**ROS 802.11**	**nstreme**	**nv2**
station	V	V	V	V
station-wds		V	V	V
station-pseudobridge	V	V	V	
station-pseudobridge-clone	V	V	V	
station-bridge		V	V	V

Mode *station*

This is standard mode that does not support L2 bridging on station - attempts to put wireless interface in bridge will not produce expected results. On the other hand this mode can be considered the most efficient and therefore should be used if L2 bridging on station is not necessary - as in case of routed or MPLS switched network. This mode is supported for all wireless protocols.

Mode *station-wds*

This mode works only with RouterOS APs. As a result of negotiating connection, separate WDS interface is created on AP for given station. This interface can be thought of point-to-point connection between AP and given station - whatever is sent out WDS interface is delivered to station (and only to particular station) and whatever station sends to AP is received from WDS interface (and not subject to forwarding between AP clients), preserving L2 addresses.

This mode is supported for all wireless protocols except when 802.11 protocol is used in connection to non-RouterOS device. Mode uses *4 address* frame format when used with 802.11 protocol, for other protocols (such as nstreme or nv2), protocol internal means are used.

This mode is safe to use for L2 bridging and gives most administrative control on AP by means of separate WDS interface, for example use of bridge firewall, RSTP for loop detection and avoidance, etc.

Mode *station-pseudobridge*

This mode from wireless connection point of view is the same as standard station mode. It has limited support for L2 bridging by means of some services implemented in station:

- MAC address translation for IPv4 packets - station maintains IPv4-to-MAC mapping table and replaces source MAC address with its own address when sending frame to AP (in order to be able to use *3 address* frame format), and replaces destination MAC address with address from mapping table for frames received from AP. IPv4-to-MAC mappings are built also for VLAN encapsulated frames.
- single MAC address translation for the rest of protocols - station learns source MAC address from first forwarded non-IPv4 frame and uses it as default for reverse translation - this MAC address is used to replace destination MAC address for frames received from AP if IPv4-to-MAC mapping can not be performed (e.g. - non-IPv4 frame or missing mapping).

This mode is limited to complete L2 bridging of data to single device connected to station (by means of single MAC address translation) and some support for IPv4 frame bridging - bridging of non-IP protocols to more than one device will not work. Also MAC address translation limits access to station device from AP side to IPv4 based access - the rest of protocols will be translated by single MAC address translation and will not be received by station itself.

This mode is available for all protocols except nv2 and **should be avoided when possible**. The usage of this node can only be justified if AP does not support better mode for L2 bridging (e.g. when non-RouterOS AP is used) or if only one end-user device must be connected to network by means of station device.

Mode *station-pseudobridge-clone*

This mode is the same as *station-pseudobridge* mode, except that it connects to AP using "cloned" MAC address - that is either address configured in **station-bridge-clone-mac** parameter (if configured) or source address of first forwarded frame. This essentially appears on AP as if end-user device connected to station connected to AP.

Mode *station-bridge*

This mode works only with RouterOS APs and provides support for transparent protocol-independent L2 bridging on station device. RouterOS AP accepts clients in *station-bridge* mode when enabled using **bridge-mode** parameter. In this mode AP maintains forwarding table with information on what MAC addresses are reachable over which station device.

This mode is MikroTik proprietary and can't be used to connect other brand devices.

This mode is safe to use for L2 bridging and should be used whenever there are sufficient reasons to not use *station-wds* mode.

Manual:Xen

Xen Virtualization Overview

XEN is discontinued since version 4.4

Applies to RouterOS: v4.3 and below only

Virtualization techonogies enable single physical device to execute multiple different operating systems. Virtualization support in RouterOS allows to run multiple copies of RouterOS sofware and even other supported operating systems. Note that virtualization support depends on system architecture, not all architectures that RouterOS supports allow virtualization.

Ability to run non-RouterOS sofware allows user to run applications that are not included in RouterOS.

Xen is the RouterOS Virtualization system for X86 machines, Xen is based on Xen Virtual machine of Linux.

x86 Virtualization Support

Virtualization support on x86 architecture systems is implemented using Xen hypervisor (http://www.xen.org). This enables RouterOS to run other operating systems that support Xen paravirtualization in "virtual machines" (guests), controlled by RouterOS software (host).

Support for virtualization for x86 architecture systems is included in RouterOS software versions starting with 3.11. To enable virtualization support "xen" package must be installed.

Host RouterOS software sets up virtual machines such that they use file in RouterOS host file system as disk image(s). Additionally host RouterOS can set up virtual ethernet network interfaces between itself and virtual machine. This enables virtual machines to participate in network under control of host RouterOS software.

In order to execute operating system in virtual machine, you need:

- OS kernel that supports Xen paravirtualization
- OS disk image
- (optionally) initial ram disk to use while booting OS in VM

If RouterOS image is used for booting in VM, OS kernel and initial ram disk are not necessary - specifying RouterOS disk image is sufficient. RouterOS images for use by VMs can be created in 2 ways:

- either by taking image from existing RouterOS x86 installation that supports virtualization (version >= 3.11)
- or by using special RouterOS functions to create RouterOS image to use in VM (note that these functions **do not** produce RouterOS image that can be copied and successfully run from physical media!).

The latter approach is more flexible because allows user to specify disk image size.

To be able to run non-RouterOS operating system in VM, you need Linux kernel, disk image and initial ram disk (if necessary) files.

Note that one disk image at the same time can only be used by one VM.

Creating RouterOS image to use in VM

To create RouterOS image to use in VM use "/xen make-routeros-image" command:

```
[admin@MikroTik] /xen> make-routeros-image file-name=ros1.img file-size=32

[admin@MikroTik] /xen> /file print
 # NAME              TYPE            SIZE       CREATION-TIME
 0 ros1.img          .img file       33554432   jun/06/2008 14:47:23
```

This produces 32MB RouterOS image that is ready to use in VM. New RouterOS image is based on host system sofware and therefore contains all sofware packages that are installed on host system, but does not contain host configuration.

Additionally, "make-routeros-image" has "configuration-script" file parameter that can be used to put on initial configuration script in created image. The script will be run on first boot of image.

VM Configuration

All virtualization for x86 architecture related functions are configured under "/xen" menu.

Memory Available to Host RouterOS

By default all the memory is available to host system, for example for system with 1GB of memory:

```
[admin@MikroTik] > /system resource print
                   uptime: 2m4s
                  version: "3.9"
             free-memory: 934116kB
            total-memory: 963780kB
                     cpu: "Intel(R)"
                cpu-count: 2
            cpu-frequency: 2813MHz
                 cpu-load: 0
           free-hdd-space: 77728884kB
          total-hdd-space: 79134596kB
  write-sect-since-reboot: 989
         write-sect-total: 989
        architecture-name: "x86"
               board-name: "x86"
[admin@MikroTik] > /xen global-settings print
  memory-for-main: unlimited
```

In some cases this may limit ability to allocate necessary memory for running guest VMs, because host system may have used memory for e.g. filesystem caching purposes. Therefore it is advised to configure limit of memory available to host system (exact value for limit depends on what sofware features are used on host system - in general, the same rules as for choosing amount of physical memory for regular RouterOS installation apply):

```
[admin@MikroTik] > /system resource print
                   uptime: 2m4s
                  version: "3.9"
             free-memory: 934116kB
            total-memory: 963780kB
                     cpu: "Intel(R)"
                cpu-count: 2
            cpu-frequency: 2813MHz
                 cpu-load: 0
           free-hdd-space: 77728884kB
          total-hdd-space: 79134596kB
  write-sect-since-reboot: 989
         write-sect-total: 989
```

```
        architecture-name: "x86"
               board-name: "x86"
[admin@MikroTik] > /xen global-settings print
  memory-for-main: unlimited
[admin@MikroTik] > /xen global-settings set memory-for-main=128
[admin@MikroTik] > /system reboot
Reboot, yes? [y/N]:
y
system will reboot shortly

 ....

[admin@MikroTik] > /system resource print
                   uptime: 1m5s
                  version: "3.11"
             free-memory: 114440kB
            total-memory: 131272kB
                     cpu: "Intel(R)"
               cpu-count: 2
            cpu-frequency: 2813MHz
                 cpu-load: 0
           free-hdd-space: 77728884kB
          total-hdd-space: 79134596kB
  write-sect-since-reboot: 794
         write-sect-total: 794
        architecture-name: "x86"
               board-name: "x86"
```

Creating RouterOS VM

Assuming that RouterOS image "ros1.img" is previously made, new VM to run RouterOS can be created:

```
[admin@MikroTik] /xen> add name=ros1 disk-images=hda:ros1.img memory=64 console-telnet-port=64000
[admin@MikroTik] /xen> print detail
Flags: X - disabled, C - configuration-changed
 0 X name="ros1" disk-images="hda:ros1.img" initrd="" kernel="" kernel-cmdline="" cpu-count=1 memory=64 weight=256
     console-telnet-port=64000 state=disabled
```

The following parameters were passed to "add" command:

- disk-images=hda:ros1.img - these parameters specify that file "ros1.img" in host filesystem will be set up as disk "hda" (IDE Primary Master) in guest system;
- memory=64 - this specifies amount of memory for guest VM;
- console-telnet-port=64000 - specifies that host system will listen on port 64000 and once telnetted to, will forward guests console output to telnet client and accept console input from telnet client.

There are few other settings:

- kernel & initrd - VM kernel file to boot and initial ram disk file to use (if specified), as noted before, specifying these is not necessary when booting RouterOS image;
- kernel-cmdline - command line to pass to Linux kernel

- cpu-count - how many CPUs should be made available to VM;
- weight - proportional "importance" of this VM when scheduling multiple VMs for execution. Taking into account that host operating system shares CPUs with all running guest VMs, weight parameter specifies proportional share of CPU(s) that guest operating system will get when multiple operating systems start competing for CPU resource. "Weight" of host operating system is 256. So, for example, if guest VM is also configured with weight 256, if both OSes will be running at 100% CPU usage, both will get equal share of CPU. If guest VM will be configured with weight 128, it will get only 1/3 of CPU.

Starting, Stopping and Connecting to RouterOS VM

To start booting guest VM, enable it:

```
[admin@MikroTik] /xen> enable ros1
[admin@MikroTik] /xen> print
Flags: X - disabled, C - configuration-changed
 #   NAME                                                                MEMORY     WEIGHT STATE
 0   ros1                                                                64         256    running
```

There are 2 (mutually exclusive, because there is just one virtual console provided for guest VM) ways to connect to console of running VM:

- by using "/xen console <VM name>" command, or
- by using telnet program and connecting to port specified in "console-telnet-port" parameter.

There are multiple ways to stop running VM:

- preferred way is to shut down **from** guest VM (e.g. by connecting to guest VM, logging in and issuing "/system shutdown" command).
- force shutdown from host RouterOS by using "/xen shutdown <VM name>" command;
- simply by disabling VM entry in "/xen" menu, note that this is the most dangerous way of stopping running VM, because guest VM can leave its filesystem in corrupt state (disabling VM entry for VM is the same as unplugging power for physical device).

VM shutdown state can be confirmed in "/xen" menu:

```
[admin@MikroTik] /xen> shutdown ros1


[admin@MikroTik] /xen> print
Flags: X - disabled, C - configuration-changed
 #   NAME                                                            MEMORY     WEIGHT STATE
 0   ros1                                                            64         256    shutdown
```

In order to boot VM that is shut down, you must either disable and enable VM entry in "/xen" menu or use "/xen start <VM name>" command.

There is also "/xen reboot <VM name>" command, that can be used to restart running guest VM, but it must be taken into account that using this command is dagerous - although it instructs guest VM to reboot, in most cases it does not cause guest to flush its filesystem and terminate correctly.

If any guest VM related settings are changed for VM entry in "/xen" menu, if guest VM is running, those settings are not applied immediately (because that would involve destroying VM and starting it again). Instead, VM is marked as "configuration-changed" and new settings will be applied on next reboot. For example:

```
[admin@MikroTik] /xen> print
Flags: X - disabled, C - configuration-changed
```

```
 #   NAME                                                                    MEMORY     WEIGHT STATE
 0   ros1                                                                    64         256    running
[admin@MikroTik] /xen> set ros1 memory=32
[admin@MikroTik] /xen> print
Flags: X - disabled, C - configuration-changed
 #   NAME                                                                    MEMORY     WEIGHT STATE
 0 C ros1                                                                    32         256    running
[admin@MikroTik] /xen> shutdown ros1

[admin@MikroTik] /xen> print
Flags: X - disabled, C - configuration-changed
 #   NAME                                                                    MEMORY     WEIGHT STATE
 0   ros1                                                                    32         256    shutdown
[admin@MikroTik] /xen> start ros1

[admin@MikroTik] /xen> print
Flags: X - disabled, C - configuration-changed
 #   NAME                                                                    MEMORY     WEIGHT STATE
 0   ros1                                                                    32         256    running
```

After this command sequence memory of running guest is actually 32Mb.

Reconfiguring RouterOS VM Image

With "/xen reconfigure-routeros-image", RouterOS configuration from existing RouterOS image can be wiped out and new configuration script put on (the script will be executed when VM using this image will next get started):

```
[admin@MikroTik] /xen> reconfigure-routeros-image file-name=ros1.img configuration-script=script.file
```

Configuring VM Networking

In order for guest VM to participate in network, virtual interfaces that connect guest VM with host must be created. Virtual network connection with guest VM can be thought of as point-to-point ethernet network connection, which terminates in guest VM as "/interface ethernet" type interface and in host as "/interface virtual-ethernet" interface. By configuring appropriate data forwarding (either by bridging or routing) to/from virtual-ethernet interface in host system, guest VM can be allowed to participate in real network.

Configuring Network Interfaces for Guest VM

Network interfaces that will appear in guest VM as ethernet interfaces are configured in "/xen interface" menu:

```
[admin@MikroTik] /xen interface> add virtual-machine=ros1 type=dynamic
[admin@MikroTik] /xen interface> print detail
Flags: X - disabled, A - active
 0   virtual-machine=ros1 vm-mac-addr=02:1C:AE:C1:B4:B2 type=dynamic static-interface=none
    dynamic-mac-addr=02:38:19:0C:F3:98 dynamic-bridge=none
```

Above command creates interface for guest VM "ros1" with type "dynamic".

There are 2 types of interfaces:

- dynamic - endpoint of virtual network connection in host ("/interface virtual-ethernet") will be created dynamically when guest VM will be booted. By using this type of interface user avoids manually creating

endpoint of virtual connection in host, at the expense of limited flexibility how this connection can be used (e.g. there is no way how to reliably assign IP address to dynamically created interface). Currently, it can only be automatically added to bridge specified in "dynamic-bridge" parameter. This behaviour is similar to dynamic WDS interfaces for wireless WDS links.

- static - endpoint of virtual network connection in host ("/interface virtual-ethernet") must be manually created. This type of interface allows maximum flexibility because interface that will connect with guest VM is previously known (therefore IP addresses can be added, interface can be used in filter rules, etc.), at the expense of having to create "/interface virtual-ethernet" manually.

VM interfaces have the following parameters:

- virtual-machine - to which VM this interface belongs;
- vm-mac-addr - MAC address of ethernet interface in **guest** system;
- type - interface type as described above
- static-interface - when "type=static", this parameter specifies which "/interface virtual-ethernet" in host system will be connected with guest;
- dynamic-mac-addr - when "type=dynamic", automatically created "/interface virtual-ethernet" in **host** system will have this MAC address;
- dynamic-bridge - when "type=dynamic", dynamically created "/interface virtual-ethernet" will automatically get added as bridge port to this bridge.

Configuring Dynamic Interfaces

To create virtual connection that will have its endpoint in host dynamically made, use the following command:

```
[admin@MikroTik] /xen interface> add virtual-machine=ros1 type=dynamic

[admin@MikroTik] /xen interface> print detail

Flags: X - disabled, A - active

 0   virtual-machine=ros1 vm-mac-addr=02:1C:AE:C1:B4:B2 type=dynamic static-interface=none dynamic-mac-addr=02:38:19:0C:F3:98 dynamic-bridge=none
```

After enabling "ros1" VM, you can confirm that new virtual-ethernet interface is made with given dynamic-mac-addr:

```
[admin@MikroTik] /xen> /interface virtual-ethernet print

Flags: X - disabled, R - running

 #    NAME                                                                                                MTU   ARP        MAC-ADDRESS

 0  R vif1                                                                                                1500  enabled    02:38:19:0C:F3:98
```

And in guest VM ethernet interface is available with given vm-mac-addr:

```
[admin@Guest] > int ethernet print

Flags: X - disabled, R - running, S - slave

 #    NAME                                                                                                MTU   MAC-ADDRESS       ARP

 0 R  ether1                                                                                              1500  02:1C:AE:C1:B4:B2 enabled
```

By configuring "dynamic-bridge" setting, virtual-ethernet interface can be automatically added as bridge port to some bridge in host system. For example, if it is necessary to forward traffic between "ether1" interface on host and VM "ros1" ethernet interface, the following steps must be taken:

Create bridge on host system and add "ether1" as bridge port:

```
[admin@MikroTik] > /interface bridge add name=to-ros1
[admin@MikroTik] > /interface bridge port add bridge=to-ros1 interface=ether1
```

Next, specify that virtual-ethernet should automatically get added as bridge port:

```
[admin@MikroTik] /xen interface> print detail
Flags: X - disabled, A - active
 0 A virtual-machine=ros1 vm-mac-addr=02:1C:AE:C1:B4:B2 type=dynamic static-interface=none dynamic-mac-addr=02:38:19:0C:F3:98 dynamic-bridge=none
[admin@MikroTik] /xen interface> set 0 dynamic-bridge=to-ros1
```

After this virtual-ethernet interface is added as bridge port on host:

```
[admin@MikroTik] /xen interface> /interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE                                  BRIDGE                                     PRIORITY PATH-COST  HORIZON
 0    ether1                                     to-ros1                                    0x80     10         none
 1  D vif1                                       to-ros1                                    0x80     10         none
```

By using similar configuration, user can, for example, "pipe" all traffic through guest VM - if there are 2 physical interfaces in host, user can create 2 bridges and bridge all traffic through guest VM (assuming that operating system in guest is configured in such a way that ensures data forwarding between its interfaces).

Configuring Static Interfaces

To create virtual connection whose endpoint in host system will be static interface, at first create static virtual-ethernet interface:

```
[admin@MikroTik] /interface virtual-ethernet> add name=static-to-ros1 disabled=no
[admin@MikroTik] /interface virtual-ethernet> print
Flags: X - disabled, R - running
 #    NAME                                                                                MTU   ARP        MAC-ADDRESS
 0  R vif1                                                                                1500  enabled    02:38:19:0C:F3:98
 1    static-to-ros1                                                                      1500  enabled    02:3A:1B:DB:FC:CF
```

Next, create interface for guest VM:

```
[admin@MikroTik] /xen interface> add virtual-machine=ros1 type=static static-interface=static-to-ros1
[admin@MikroTik] /xen interface> print
Flags: X - disabled, A - active
 #   VIRTUAL-MACHINE                                                                      TYPE    VM-MAC-ADDR
 0 A ros1                                                                                 dynamic 02:1C:AE:C1:B4:B2
 1 A ros1                                                                                 static  02:DF:66:CD:E9:74
```

Now we can confirm that virtual-ethernet interface is active:

```
[admin@MikroTik] /xen interface> /interface virtual-ethernet print
Flags: X - disabled, R - running
 #    NAME                                                                                MTU   ARP        MAC-ADDRESS
 0  R static-to-ros1                                                                      1500  enabled    02:3A:1B:DB:FC:CF
 1  R vif1                                                                                1500  enabled    02:38:19:0C:F3:98
```

And in guest system:

```
[admin@Guest] > /interface ethernet print
Flags: X - disabled, R - running, S - slave
 #    NAME                                                                                MTU   MAC-ADDRESS       ARP
 0 R  ether1                                                                              1500  02:1C:AE:C1:B4:B2 enabled
```

```
 1 R  ether2                                                                1500  02:DF:66:CD:E9:74 enabled
```

Having static interface in host system allows to use interface in configuration wherever specifying interface is necessary, e.g. adding ip address:

```
[admin@MikroTik] > ip address add interface=static-to-ros1 address=1.1.1.1/24
```

In similar way we add IP address to appropriate interface in guest system and confirm that routing is working:

```
[admin@Guest] > /ip address add interface=ether2 address=1.1.1.2/24
[admin@Guest] > /ping 1.1.1.1
1.1.1.1 64 byte ping: ttl=64 time=5 ms
1.1.1.1 64 byte ping: ttl=64 time<1 ms
1.1.1.1 64 byte ping: ttl=64 time<1 ms
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0/1.6/5 ms
```

Running non-RouterOS Systems as Guests

Xen hypervisor based virtualization for x86 architectures that is used in RouterOS allow to run other operating systems that use Linux kernel that has Xen guest support (DomU support in Xen terminology).

In order to run non-RouterOS system as guest on RouterOS host, you need:

- OS image file
- OS Linux kernel file
- optionally initial ram disk file

There are several ways how to prepare necessary files:

- using already prepared, ready to boot images (the easiest way);
- installing operating system along with necessary virtualization packages and using image from installed system (medium complexity);
- installing operating system, taking image, recompiling Linux kernel, adjusting system to boot under Xen hypervisor (the most complex).

Using Ready to Boot Image

If you have operational Debian GNU/Linux based system with Xen installed, you can use xen-tools scripts (http://www. xen-tools. org) to create/install images and use Xen kernel and initrd from your distribution. Another opportunity is to use already prepared images that are available for download, for example from http://jailtime.org. Note that images **do not** contain the kernel itself, therefore these images can only be used after taking appropriate kernel and initrd file from real distribution.

Additionally, here we provide some sets of files ready to use in guest VMs.

ClarkConnect 4.2 Community Edition SP1 Image

Image is prepared from installation ISO (ftp://ftp.clarkconnect.com/4.2/iso/community-4.2.SP1.iso) and additional Xen kernel package (ftp://ftp.clarkconnect.com/4.2/other/kernel-xen-2.6.18-8.1.14.3.cc.i686.rpm).Minimum software is installed.

Archive contains the following files:

- kernel: vmlinuz-2.6.18-8.1.14.3.ccxen
- initial RAM disk: clark.initrd.rgz, clark.otherinitrd.rgz (either one can be used)
- disk image: clark.img

To use this image in guest VM under RouterOS (remember that you have to upload files from archive to your RouterOS host), use the following command:

```
[admin@MikroTik] /xen> add disk=hda disk-image=clark.img
initrd=clark.otherinitrd.rgz kernel=vmlinuz-2.6.18-8.1.14.3.ccxen
kernel-cmdline="root=/dev/hda1" memory=128 name=clark
```

Password for *root* user is *rootroot*.

Archive can be dowloaded here: http://www.mikrotik.com/download/clark.tar.bz2

CentOS 5.1 Image

Image is prepared using netinstall ISO (ISO file: CentOS-5.1-i386-netinstall.iso, available from mirrors listed at http://isoredirect.centos.org/centos/5/isos/i386/) and network based installation. Minimum software is installed.

Archive contains the following files:

- kernel: vmlinuz-2.6.18-53.el5xen
- inital RAM disk: centos.initrd.rgz
- disk image: centos.img

To use this image in guest VM under RouterOS (remember that you have to upload files from archive to your RouterOS host), use the following command:

```
[admin@MikroTik] /xen> add disk=hda disk-image=centos.img
initrd=centos.initrd.rgz kernel=vmlinuz-2.6.18-53.el5xen
kernel-cmdline="ro root=/dev/hda1" memory=512 name=centos
```

Password for *root* user is *rootroot*.

Archive can be dowloaded here: http://www.mikrotik.com/download/centos.tar.bz2

Installing OS with Virtualization Support

One of ways to simplify OS installation is to install it in image file using some full virtualization software, like VMWare or QEMU. This allows to produce ready to use image file and does not require any additional hardware.

Example: Preparing ClarkConnect Community Edition 4.2 SP1 Image

Below find instructions on how to get ClarkConnect 4.2 Community Edition run as guest VM. Note that ClarkConnect installation does not provide support for virtualization by default, therefore additional tweaks will be necessary.

Installing ClarkConnect

At first, create image where ClarkConnect will be installed:

```
~/xen$ qemu-img create clark.img 1Gb
Formatting 'clark.img', fmt=raw, size=1048576 kB
```

Next, start installation from ClarkConnect installation ISO image:

```
~/xen$ sudo qemu -hda clark.img -cdrom community-4.2.SP1.iso -net nic,vlan=0,macaddr=00:01:02:03:04:aa -net tap,vlan=0,ifname=tap0 -m 128 -boot d
```

Proceed with installation, creating one root partition and (optionally) swap space. Take into account disk size when selecting software packages to install. In this example disk is partitioned with 800MB root partition size and the rest of image for swap. Note that QEMU is instructed to emulate ethernet card, during installation this card is configured with IP address 10.0.0.23/24.

ClarkConnect installation does not provide support for virtualization by default, therefore virtualization support will have to be added manually. ClarkConnect distributes Xen-aware kernel package separately from installation, available at: ftp://ftp.clarkconnect.com/4.2/other/kernel-xen-2.6.18-8.1.14.3.cc.i686.rpm

In order to install this package we have to put it on newly created image. To do this, boot new image:

```
~/xen$ sudo qemu -hda clark.img -net nic,vlan=0,macaddr=00:01:02:03:04:aa -net tap,vlan=0,ifname=tap0 -m 128
```

Assuming that networking with QEMU virtual machine is configured properly, we can use SCP to put on package file:

```
~/xen$ scp ./kernel-xen-2.6.18-8.1.14.3.cc.i686.rpm root@10.0.0.23:/

The authenticity of host '10.0.0.23 (10.0.0.23)' can't be established.

RSA key fingerprint is 70:84:b8:c5:6d:62:37:d1:1e:96:29:d0:77:46:6a:0c.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '10.0.0.23' (RSA) to the list of known hosts.

root@10.0.0.23's password:

kernel-xen-2.6.18-8.1.14.3.cc.i686.rpm                                                    100%   16MB   2.0MB/s   00:08
```

Next, connect to ClarkConnect and install kernel package. Note that this package is not entirely compatible with ClarkConnect 4.2 SP1 system and proper installation fails, but taking into account that the only purpose of installing this package is to get Xen enabled kernel and drivers, forced installation is fine, except that module dependency file must be created manually:

```
~/xen$ ssh root@10.0.0.23
root@10.0.0.23's password:
Last login: Tue Jun 10 07:20:34 2008
[root@server ~]# cd /
[root@server /]# rpm -i kernel-xen-2.6.18-8.1.14.3.cc.i686.rpm --force
```

```
--nodeps
Usage: new-kernel-pkg [-v] [--mkinitrd] [--rminitrd]
       [--initrdfile=<initrd-image>] [--depmod] [--rmmoddep]
       [--kernel-args=<args>] [--banner=<banner>]
       [--make-default] <--install | --remove>
<kernel-version>
       (ex: new-kernel-pkg --mkinitrd --depmod --install 2.4.7-2)
error: %post(kernel-xen-2.6.18-8.1.14.3.cc.i686) scriptlet failed, exit
 status 1
[root@server /]# ls /boot
config-2.6.18-53.1.13.2.cc    initrd-2.6.18-53.1.13.2.cc.img
symvers-2.6.18-8.1.14.3.ccxen.gz  vmlinuz-2.6.18-53.1.13.2.cc
xen-syms-2.6.18-8.1.14.3.cc
config-2.6.18-8.1.14.3.ccxen  memtest86+-1.26
System.map-2.6.18-53.1.13.2.cc    vmlinuz-2.6.18-8.1.14.3.ccxen
grub                          symvers-2.6.18-53.1.13.2.cc.gz
System.map-2.6.18-8.1.14.3.ccxen  xen.gz-2.6.18-8.1.14.3.cc
[root@server /]# depmod -v 2.6.18-8.1.14.3.ccxen -F
/boot/System.map-2.6.18-8.1.14.3.ccxen
 ....
```

Next, copy out some files from installed system:

- Xen enabled kernel (/boot/vmlinuz-2.6.18-8.1.14.3.ccxen)
- initial ram disk (/boot/initrd-2.6.18-53.1.13.2.cc.img)

```
~/xen$ scp root@10.0.0.23:/boot/vmlinuz-2.6.18-8.1.14.3.ccxen ./

root@10.0.0.23's password:

vmlinuz-2.6.18-8.1.14.3.ccxen                                                 100% 2049KB   2.0MB/s   00:01

~/xen$ scp root@10.0.0.23:/boot/initrd-2.6.18-53.1.13.2.cc.img ./

root@10.0.0.23's password:

initrd-2.6.18-53.1.13.2.cc.img                                                100%  434KB 433.8KB/s   00:00
```

Default ClarkConnect installation does not execute login process on Xen virtual console, so in order to have login available on virtual console accessible from RouterOS with "/xen console <VM name>" command, virtual console device should get made inside image (mknod /dev/xvc0 c 204 191).

Default ClarkConnect initial ram disk does not support booting from Xen virtual disk because it does not contain driver for virtual disk. To overcome this problem initial ram disk must be updated.

Updating initrd Manually

One opportunity to make initial ram disk that would support booting from virtual disk is to manually put virtual disk driver in initrd and update it to load this module.

At first we extract contents of initial ram disk that was copied from ClarkConnect image:

```
~/xen$ file initrd-2.6.18-53.1.13.2.cc.img

initrd-2.6.18-53.1.13.2.cc.img: gzip compressed data, from Unix, last modified: Tue Jun 10 14:01:27 2008, max compression

~/xen$ mv initrd-2.6.18-53.1.13.2.cc.img clarkinitrd.gz

~/xen$ gunzip clarkinitrd.gz

~/xen$ file clarkinitrd

clarkinitrd: ASCII cpio archive (SVR4 with no CRC)
```

```
~/xen$ mkdir initrd
~/xen$ cd initrd/
~/xen/initrd$ sudo cpio -idv --no-absolute-filenames < ../clarkinitrd
.
etc
bin
bin/insmod
bin/nash
bin/modprobe
sysroot
sys
lib
lib/sd_mod.ko
lib/libata.ko
lib/scsi_mod.ko
lib/ata_piix.ko
lib/ext3.ko
lib/jbd.ko
sbin
dev
dev/console
dev/systty
dev/tty3
dev/tty2
dev/tty4
dev/ram
dev/tty1
dev/null
init
loopfs
proc
1990 blocks
~/xen/initrd$ cat init
#!/bin/nash

mount -t proc /proc /proc
setquiet
echo Mounted /proc filesystem
echo Mounting sysfs
mount -t sysfs none /sys
echo "Loading scsi_mod.ko module"
insmod /lib/scsi_mod.ko
echo "Loading sd_mod.ko module"
insmod /lib/sd_mod.ko
echo "Loading libata.ko module"
insmod /lib/libata.ko
echo "Loading ata_piix.ko module"
```

```
insmod /lib/ata_piix.ko
echo "Loading jbd.ko module"
insmod /lib/jbd.ko
echo "Loading ext3.ko module"
insmod /lib/ext3.ko
echo Creating block devices
mkdevices /dev
echo Creating root device
mkrootdev /dev/root
umount /sys
echo Mounting root filesystem
mount -o defaults --ro -t ext3 /dev/root /sysroot
echo Switching to new root
switchroot /sysroot
```

From the above we see that init script in initrd image loads drivers for SCSI and ATA disks, as well as EXT3 filesystem modules. In order for ClarkConnect to boot under Xen we have to replace hardware drivers with Xen virtual disk driver and EXT3 filesystem modules with appropriate modules for Xen kernel. Take these modules from installed ClarkConnect system:

```
~/xen/initrd$ cd lib/
~/xen/initrd/lib$ ls
ata_piix.ko  ext3.ko  jbd.ko  libata.ko  scsi_mod.ko  sd_mod.ko
~/xen/initrd/lib$ sudo rm ata_piix.ko libata.ko scsi_mod.ko sd_mod.k
~/xen/initrd/lib$ scp
root@10.0.0.23:/lib/modules/2.6.18-8.1.14.3.ccxen/kernel/fs/jbd/jbd.ko
./
root@10.0.0.23's password:
jbd.ko

100%   70KB  69.8KB/s
00:00
~/xen/initrd/lib$ sudo scp
root@10.0.0.23:/lib/modules/2.6.18-8.1.14.3.ccxen/kernel/fs/ext3/ext3.ko
 ./
root@10.0.0.23's password:
ext3.ko

100%  141KB 141.5KB/s
00:00
~/xen/initrd/lib$ sudo scp
root@10.0.0.23:/lib/modules/2.6.18-8.1.14.3.ccxen/kernel/drivers/xen/blkfront/xenblk.ko
 ./
root@10.0.0.23's password:
xenblk.ko

100%   22KB  22.0KB/s
00:00
```

Next, update init script so that it loads Xen virtual disk driver. Final init script should look like this:

```
~/xen/initrd$ cat init
#!/bin/nash

mount -t proc /proc /proc
setquiet
echo Mounted /proc filesystem
echo Mounting sysfs
mount -t sysfs none /sys
echo "Loading xenblk.ko module"
insmod /lib/xenblk.ko
echo "Loading jbd.ko module"
insmod /lib/jbd.ko
echo "Loading ext3.ko module"
insmod /lib/ext3.ko
echo Creating block devices
mkdevices /dev
echo Creating root device
mkrootdev /dev/root
umount /sys
echo Mounting root filesystem
mount -o defaults --ro -t ext3 /dev/root /sysroot
echo Switching to new root
switchroot /sysroot
```

Create initrd file from directory structure with modifications that have been made:

```
~/xen/initrd$ find * | cpio -o -H newc -O ../clarkinitrd.new
~/xen/initrd$ find . -depth | cpio -ov --format=newc > ../clark.initrd
~/xen/initrd$ cd ../
~/xen$ gzip -c -9 < clark.initrd > clark.initrd.rgz
```

Using mkinitrd Utility

Instead of creating initial ram disk manually as described above is possible to use *mkinitrd* utility available in ClarkConnect distribution. After Xen kernel package is installed as shown above, initial ram disk can be created with command (note that this command must be executed in ClarkConnect, e.g. running in QEMU VM):

```
[root@server /]# mkinitrd clark.otherinitrd.rgz 2.6.18-8.1.14.3.ccxen --omit-scsi-modules --omit-raid-modules --omit-lvm-modules --with=xenblk
```

After this, newly created clark.otherinitrd.rgz must be copied from ClarkConnect image.

Adding ClarkConnect VM in RouterOS

Finally upload files (dont forget to shut down QEMU that executes image) to host RouterOS and create guest VM entry:

```
[admin@MikroTik] /xen> print detail
Flags: X - disabled
 1 X name="clark" disk=hda disk-image="clark.img" initrd="clark.otherinitrd.rgz" kernel="vmlinuz-2.6.18-8.1.14.3.ccxen"
     kernel-cmdline="root=/dev/hda1" cpu-count=1 memory=128 weight=256 console-telnet-port=64000 state=disabled
```

Note that VM is configured with files that were made in previous steps. Also pay attention to "kernel-cmdline" parameter that is supplied. This instructs ClarkConnect where its root file system is - as we are providing ClarkConnect image with "disk=hda", and during installation root filesystem was made as first partition in image, root file system is on device /dev/hda1.

On first boot of ClarkConnect, it will detect changes in hardware and also enable login on virtual console device.

Example: Preparing Centos 5.1 Image

CentOS ir RedHat Linux based Linux distribution. Distribution includes necessary software packages for virtualization support, therefore installing CentOS image that supports virtualization is rather simple.

Installing CentOS 5.1

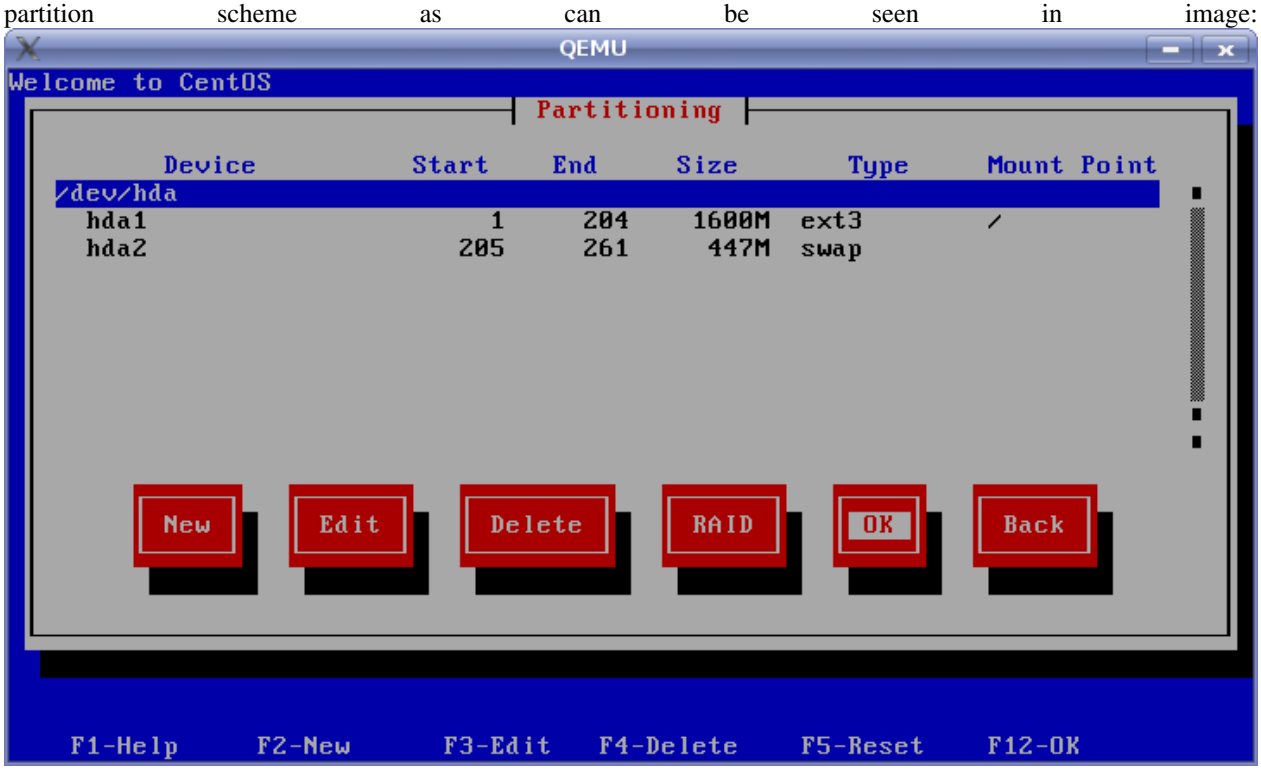
To create example CentOS image, we use QEMU for CentOS installation the same way as in previous ClarkConnect example.

Create image file and start QEMU with CentOS netinstall ISO image:

```
~/xen$ qemu-img create centos.img 2Gb
Formatting 'centos.img', fmt=raw, size=2097152 kB
~/xen$ sudo qemu -hda centos.img -cdrom=CentOS-5.1-i386-netinstall.iso
-net nic,vlan=0,macaddr=00:01:02:03:04:aa -net tap,vlan=0,ifname=tap0
-m 256
```

Note that netinstall ISO image is used - sofware packages will be downloaded from network. This means that network connectivity of QEMU VM must be configured and running.

During installation follow partition scheme as in previous example for ClarkConnect. Example image is created with partition scheme as can be seen in image:



Also during installation select "Virtualization" sofware set:

When installation is complete, CentOS image does not boot under QEMU emulator because it does not support running Xen hypervisor. Nevertheless this does not matter, because all necessary sofware for running as guest is already installed in image. Still this forces to take different approach for extracting necessary files from image (for ClarkConnect this got done by connecting to VM running under QEMU and copying files out).

Preparing Initial Ram Disk

To take Xen kernel from CentOS image and to prepare initrd (that would include driver for virtual disk), use the following steps.

Mount root partition of image using loopback device (note that 1st partition in image starts with sector 63 therefore we use offset in image file to point to beginning of partition:

```
# mount centos.img /mnt -o loop,offset=$[512*63]
```

Next, copy out kernel file:

```
# cp /mnt/boot/vmlinuz-2.6.18-53.el5xen ./
```

To prepear initrd file we use mkinitrd tool. To force it to work on mounted image, use chroot command:

```
# chroot /mnt /bin/sh

sh-3.1# mkinitrd centos.initrd.rgz 2.6.18-53.el5xen --omit-scsi-modules --omit-raid-modules --omit-lvm-modules --with=xenblk

sh-3.1# exit
```

Adding CentOS VM in RouterOS

Now files to be used for running guest VM (kernel, newly made initrd and image) have to be put on RouterOS and appropriate VM entry should be made:

```
[admin@MikroTik] /xen> print detail

Flags: X - disabled

 0 X name="centos" disk=hda disk-image="centos.img" initrd="centos.initrd.rgz" kernel="vmlinuz-2.6.18-53.el5xen"

     kernel-cmdline="ro root=/dev/hda1" cpu-count=1 memory=256 weight=256 console-telnet-port=64000 state=disabled
```

Notice that CentOS kernel is also passed arguments of which partition should be used for root file system, similar to ClarkConnect.

Adding Virtualization Support to Your Favourite Linux Based OS

To be able to run your favourite OS/distribution in guest VM, it must support Xen DomU, therefore enabling Xen support most likely will involve recompiling kernel. Disk and virtual network interface devices have to be accessed by Xen *netfront* and *blockfront* drivers, therefore you should make sure that these drivers are included in your system, either directly in kernel or as modules. Kernel must be compiled with PAE support.

Depending on Linux kernel that your distribution uses, it is possible that kernel source does not have support for Xen. This may mean that patching of kernel is necessary. In such cases you can refer to distributions that use similar kernel version and have vendor patches for Xen support.

... Some time later maybe example will come...

Article Sources and Contributors

Manual:Customizing Hotspot *Source*: http://wiki.mikrotik.com/index.php?oldid=22941 *Contributors*: Marisb, SergejsB

Manual:IP/Hotspot/User *Source*: http://wiki.mikrotik.com/index.php?oldid=20097 *Contributors*: Marisb

Manual:IP/Hotspot/Walled Garden *Source*: http://wiki.mikrotik.com/index.php?oldid=19500 *Contributors*: Marisb

Manual:Simple TE *Source*: http://wiki.mikrotik.com/index.php?oldid=22150 *Contributors*: Marisb, Megis

Manual:API *Source*: http://wiki.mikrotik.com/index.php?oldid=24264 *Contributors*: AlbertStrasheim, Eep, Janisk, Jk, Juris, Karlis, Marisb, Normis, Yangsenyu

User:Boen robot/API *Source*: http://wiki.mikrotik.com/index.php?oldid=22598 *Contributors*: -

Manual:IP/Address *Source*: http://wiki.mikrotik.com/index.php?oldid=20446 *Contributors*: Janisk, Marisb

Manual:IP/ARP *Source*: http://wiki.mikrotik.com/index.php?oldid=20824 *Contributors*: Janisk, Marisb

Manual:IPv6/Address *Source*: http://wiki.mikrotik.com/index.php?oldid=23735 *Contributors*: Marisb

Manual:Router AAA *Source*: http://wiki.mikrotik.com/index.php?oldid=22021 *Contributors*: Janisk, Marisb, Normis

Manual:BGP based VPLS *Source*: http://wiki.mikrotik.com/index.php?oldid=22990 *Contributors*: Eep, Janisk, Marisb, Mplsguy, Normis

Manual:BGP Best Path Selection Algorithm *Source*: http://wiki.mikrotik.com/index.php?oldid=16875 *Contributors*: Atis, Eep, Eugene, Janisk, Marisb, Route

Manual:BGP Case Studies *Source*: http://wiki.mikrotik.com/index.php?oldid=16876 *Contributors*: Atis, Eep, Eugene, Hellbound, Janisk, Marisb, Route, SergejsB

Manual:BGP HowTo & FAQ *Source*: http://wiki.mikrotik.com/index.php?oldid=24179 *Contributors*: Janisk, Marisb, Route

Manual:BGP Load Balancing with two interfaces *Source*: http://wiki.mikrotik.com/index.php?oldid=16878 *Contributors*: Janisk, Marisb, Route

Manual:BGP nexthop selection and validation in RouterOS 3.x *Source*: http://wiki.mikrotik.com/index.php?oldid=17631 *Contributors*: Atis, Eep, Janisk, Marisb, Route, SergejsB

Manual:BGP soft reconfiguration alternatives in RouterOS *Source*: http://wiki.mikrotik.com/index.php?oldid=18350 *Contributors*: Atis, Eep, Janisk, Marisb, SergejsB

Manual:EBGP as PE-CE routing protocol *Source*: http://wiki.mikrotik.com/index.php?oldid=23196 *Contributors*: Marisb, Route

Manual:EoMPLS vs Cisco *Source*: http://wiki.mikrotik.com/index.php?oldid=23687 *Contributors*: Marisb

Manual:Interface/Bonding *Source*: http://wiki.mikrotik.com/index.php?oldid=20456 *Contributors*: Janisk, Marisb, Normis

Manual:Interface/Bridge *Source*: http://wiki.mikrotik.com/index.php?oldid=22068 *Contributors*: Janisk, Kirshteins, Marisb

Manual:MPLS L2VPN vs Juniper *Source*: http://wiki.mikrotik.com/index.php?oldid=23683 *Contributors*: Marisb

Manual:Routing/BFD *Source*: http://wiki.mikrotik.com/index.php?oldid=19825 *Contributors*: Janisk, Marisb, Normis, Route

Manual:Routing/BGP *Source*: http://wiki.mikrotik.com/index.php?oldid=23694 *Contributors*: Janisk, Marisb, Normis, Route

Manual:Simple BGP Multihoming *Source*: http://wiki.mikrotik.com/index.php?oldid=19642 *Contributors*: Marisb

Manual:BCP bridging (PPP tunnel bridging) *Source*: http://wiki.mikrotik.com/index.php?oldid=22208 *Contributors*: Janisk, Marisb, Megis, SergejsB

Manual:Bonding Examples *Source*: http://wiki.mikrotik.com/index.php?oldid=23807 *Contributors*: Eep, Eugene, Marisb, Normis, Peson

Manual:Bootloader upgrade *Source*: http://wiki.mikrotik.com/index.php?oldid=23708 *Contributors*: Cmit, Eep, Girts, Janisk, Marisb, Normis, SergejsB, XlnEax

Manual:Console *Source*: http://wiki.mikrotik.com/index.php?oldid=22857 *Contributors*: Eep, Janisk, Marisb, Normis

Manual:Create Certificates *Source*: http://wiki.mikrotik.com/index.php?oldid=24312 *Contributors*: Becs, Marisb

Manual:System/Certificates *Source*: http://wiki.mikrotik.com/index.php?oldid=24178 *Contributors*: Marisb

Manual:CD Install *Source*: http://wiki.mikrotik.com/index.php?oldid=22698 *Contributors*: Janisk, Marisb, Normis, SergejsB

Manual:Configuration Management *Source*: http://wiki.mikrotik.com/index.php?oldid=24163 *Contributors*: Janisk, Marisb, Normis, SergejsB

Manual:Conformance Testing Mode *Source*: http://wiki.mikrotik.com/index.php?oldid=20473 *Contributors*: Normis

Manual:IP/Firewall/Connection tracking *Source*: http://wiki.mikrotik.com/index.php?oldid=24402 *Contributors*: Janisk, Marisb, Normis

Manual:Connection Rate *Source*: http://wiki.mikrotik.com/index.php?oldid=16964 *Contributors*: Janisk, Marisb, Megis, Normis

Manual:Console login process *Source*: http://wiki.mikrotik.com/index.php?oldid=21955 *Contributors*: Eep, Janisk, Marisb, Normis

Manual:CPU Usage *Source*: http://wiki.mikrotik.com/index.php?oldid=16933 *Contributors*: Agris, Eep, Janisk, Marisb, Maximan, Normis

Manual:Default Configurations *Source*: http://wiki.mikrotik.com/index.php?oldid=23848 *Contributors*: Marisb, Normis

Manual:IP/DHCP Client *Source*: http://wiki.mikrotik.com/index.php?oldid=24221 *Contributors*: Janisk, Marisb

Manual:IP/DHCP Relay *Source*: http://wiki.mikrotik.com/index.php?oldid=23521 *Contributors*: Janisk, Marisb

Manual:IP/DHCP Server *Source*: http://wiki.mikrotik.com/index.php?oldid=24275 *Contributors*: Janisk, Marisb

Manual:IP/DNS *Source*: http://wiki.mikrotik.com/index.php?oldid=20512 *Contributors*: Janisk, Marisb, Normis

Manual:Tools/Dynamic DNS *Source*: http://wiki.mikrotik.com/index.php?oldid=17803 *Contributors*: Janisk, Marisb, Mmv

Manual:IPv6/DHCP Client *Source*: http://wiki.mikrotik.com/index.php?oldid=24325 *Contributors*: Janisk, Marisb

Manual:IPv6/DHCP Server *Source*: http://wiki.mikrotik.com/index.php?oldid=22657 *Contributors*: Marisb

Manual:Interface/EoIP *Source*: http://wiki.mikrotik.com/index.php?oldid=22081 *Contributors*: Eugene, HarvSki, Huri, Janisk, Kirshteins, Marisb

Manual:Interface/Ethernet *Source*: http://wiki.mikrotik.com/index.php?oldid=23358 *Contributors*: Janisk, Kirshteins, Marisb, Normis

Manual:Interface/Gre *Source*: http://wiki.mikrotik.com/index.php?oldid=21702 *Contributors*: Marisb

Manual:Interface/Gre6 *Source*: http://wiki.mikrotik.com/index.php?oldid=22873 *Contributors*: Janisk

Manual:Tools/email *Source*: http://wiki.mikrotik.com/index.php?oldid=23630 *Contributors*: Janisk, Marisb

Manual:Tools/Ping *Source*: http://wiki.mikrotik.com/index.php?oldid=18420 *Contributors*: Marisb

Manual:Routing/Routing filters *Source*: http://wiki.mikrotik.com/index.php?oldid=19790 *Contributors*: Janisk, Marisb, Route

Manual:Tools/Fetch *Source*: http://wiki.mikrotik.com/index.php?oldid=20579 *Contributors*: Cmarzotta, Enk, Janisk, Marisb, Mmv, Nest, Normis, Route, Rus123

Manual:Fast Path *Source*: http://wiki.mikrotik.com/index.php?oldid=24581 *Contributors*: Marisb

Fast Path spanish *Source*: http://wiki.mikrotik.com/index.php?oldid=24576 *Contributors*: Maximan

Manual:IP/Firewall *Source*: http://wiki.mikrotik.com/index.php?oldid=16965 *Contributors*: Janisk, Marisb

Manual:IPv6/Firewall *Source*: http://wiki.mikrotik.com/index.php?oldid=17674 *Contributors*: Marisb

Manual:IP/Firewall/Address list *Source*: http://wiki.mikrotik.com/index.php?oldid=17287 *Contributors*: Janisk, Marisb

Manual:IP/Firewall/Filter *Source*: http://wiki.mikrotik.com/index.php?oldid=22181 *Contributors*: Janisk, Kirshteins, Marisb, Normis, SergejsB

Manual:IPv6/Firewall/Filter *Source*: http://wiki.mikrotik.com/index.php?oldid=23201 *Contributors*: Marisb

Manual:IP/Firewall/L7 *Source*: http://wiki.mikrotik.com/index.php?oldid=21489 *Contributors*: Eep, Hrnous, Janisk, Marisb, Normis, SergejsB

Manual:IP/Firewall/Mangle *Source*: http://wiki.mikrotik.com/index.php?oldid=22182 *Contributors*: Janisk, Marisb, Normis

Manual:IP/Firewall/NAT *Source*: http://wiki.mikrotik.com/index.php?oldid=24582 *Contributors*: Janisk, Marisb, Normis, SergejsB

Manual:First time startup *Source*: http://wiki.mikrotik.com/index.php?oldid=22160 *Contributors*: Jandrade28, Janisk, Kirshteins, Marisb, MarkSorensen, Nest, Normis, Rock on all you f little dudes!, SergejsB

Manual:Flashfig *Source*: http://wiki.mikrotik.com/index.php?oldid=20470 *Contributors*: Janisk, Maximan, Normis, SergejsB

Manual:FTP server *Source*: http://wiki.mikrotik.com/index.php?oldid=20464 *Contributors*: Marisb, Normis

Manual:System/GPS *Source*: http://wiki.mikrotik.com/index.php?oldid=21760 *Contributors*: Marisb, Normis

Manual:Tools/Graphing *Source*: http://wiki.mikrotik.com/index.php?oldid=21173 *Contributors*: Marisb

Manual:Grounding *Source*: http://wiki.mikrotik.com/index.php?oldid=21574 *Contributors*: Marisb, Normis

Manual:Hotspot Introduction *Source*: http://wiki.mikrotik.com/index.php?oldid=19393 *Contributors*: Marisb

Manual:System/Health *Source*: http://wiki.mikrotik.com/index.php?oldid=21436 *Contributors*: Janisk, Normis

Manual:IP/Hotspot *Source*: http://wiki.mikrotik.com/index.php?oldid=19414 *Contributors*: Janisk, Marisb, Normis, SergejsB, Vitell

Manual:HTB *Source*: http://wiki.mikrotik.com/index.php?oldid=22317 *Contributors*: Eep, Janisk, Marisb, Megis, Normis

Manual:Interface/HWMPplus *Source*: http://wiki.mikrotik.com/index.php?oldid=16987 *Contributors*: Janisk, Marisb, Normis, Raivis bucis, Route, SergejsB

Manual:Creating IPv6 loopback address *Source*: http://wiki.mikrotik.com/index.php?oldid=17556 *Contributors*: Janisk, Marisb, Route

Manual:Interface *Source*: http://wiki.mikrotik.com/index.php?oldid=22145 *Contributors*: Janisk, Marisb

Manual:Interface/IPIP *Source*: http://wiki.mikrotik.com/index.php?oldid=21605 *Contributors*: Janisk, Kirshteins, Marisb

Manual:IP *Source*: http://wiki.mikrotik.com/index.php?oldid=16888 *Contributors*: Janisk, Marisb, Normis

Manual:IPv6 *Source*: http://wiki.mikrotik.com/index.php?oldid=17558 *Contributors*: Janisk, Marisb

Manual:IPv6 Overview *Source*: http://wiki.mikrotik.com/index.php?oldid=22697 *Contributors*: Janisk, Marisb, Route

Manual:OSPFv3 with Quagga *Source*: http://wiki.mikrotik.com/index.php?oldid=17612 *Contributors*: Janisk, Marisb, Route

Manual:Routing/IGMP-Proxy *Source*: http://wiki.mikrotik.com/index.php?oldid=16906 *Contributors*: Janisk, Marisb, Normis, Route

Manual:Tools/IP-Scan *Source*: http://wiki.mikrotik.com/index.php?oldid=19645 *Contributors*: Janisk, Marisb

Manual:Initial Configuration *Source*: http://wiki.mikrotik.com/index.php?oldid=22340 *Contributors*: Janisk, Marisb

Manual:Internet access from VRF *Source*: http://wiki.mikrotik.com/index.php?oldid=16989 *Contributors*: Janisk, Marisb, Route

Manual:Internet access from VRF with NAT *Source*: http://wiki.mikrotik.com/index.php?oldid=20471 *Contributors*: Eising, Marisb, Normis

Manual:IP/Hotspot/Profile *Source*: http://wiki.mikrotik.com/index.php?oldid=19273 *Contributors*: Marisb

Manual:IP/IPsec *Source*: http://wiki.mikrotik.com/index.php?oldid=24261 *Contributors*: Eep, Eugene, Janisk, Marisb, Normis, SacXs2, SergejsB

Manual:IPv6/Firewall/Address-list *Source*: http://wiki.mikrotik.com/index.php?oldid=17645 *Contributors*: Marisb

Manual:IPv6/Firewall/Mangle *Source*: http://wiki.mikrotik.com/index.php?oldid=17644 *Contributors*: Marisb

Manual:IPv6/ND *Source*: http://wiki.mikrotik.com/index.php?oldid=20511 *Contributors*: Janisk, Marisb

Manual:IPv6/Neighbors *Source*: http://wiki.mikrotik.com/index.php?oldid=19435 *Contributors*: Marisb

Manual:IPv6/Route *Source*: http://wiki.mikrotik.com/index.php?oldid=23740 *Contributors*: Marisb

Manual:KVM *Source*: http://wiki.mikrotik.com/index.php?oldid=24077 *Contributors*: Janisk, Marisb, Megis, Normis, Route

Manual:Entering a RouterOS License key *Source*: http://wiki.mikrotik.com/index.php?oldid=16869 *Contributors*: Eep, Janisk, Ldvaden, Marisb, Nest, Normis

Manual:Interface/L2TP *Source*: http://wiki.mikrotik.com/index.php?oldid=23777 *Contributors*: Becs, Janisk, Marisb

Manual:License *Source*: http://wiki.mikrotik.com/index.php?oldid=24557 *Contributors*: Becs, Eep, Janisk, Marisb, Maximan, NathanA, Nest, Normis, SergejsB

Manual:Lua *Source*: http://wiki.mikrotik.com/index.php?oldid=20164 *Contributors*: Eep, Janisk, Marisb, Normis

Manual:System/LEDS *Source*: http://wiki.mikrotik.com/index.php?oldid=21969 *Contributors*: Marisb

Manual:System/Log *Source*: http://wiki.mikrotik.com/index.php?oldid=19957 *Contributors*: Janisk, Marisb, Normis

Manual:System/UPS *Source*: http://wiki.mikrotik.com/index.php?oldid=17453 *Contributors*: Marisb

Manual:Layer-3 MPLS VPN example *Source*: http://wiki.mikrotik.com/index.php?oldid=16990 *Contributors*: Janisk, Marisb, Normis, Route

Manual:LCD TouchScreen *Source*: http://wiki.mikrotik.com/index.php?oldid=24465 *Contributors*: Becs, Marisb, Normis

Manual:Limiting maximum number of prefixes accepted *Source*: http://wiki.mikrotik.com/index.php?oldid=17388 *Contributors*: Atis, Eep, Marisb

Manual:Load balancing multiple same subnet links *Source*: http://wiki.mikrotik.com/index.php?oldid=16963 *Contributors*: Janisk, Marisb

MAC access *Source*: http://wiki.mikrotik.com/index.php?oldid=13903 *Contributors*: Marisb, Normis

Manual:Making a simple wireless AP *Source*: http://wiki.mikrotik.com/index.php?oldid=16483 *Contributors*: Marisb, Normis

Manual:Maximum Transmission Unit on RouterBoards *Source*: http://wiki.mikrotik.com/index.php?oldid=23257 *Contributors*: Becs, Janisk, Kirshteins, Marisb, Megis, Mplsguy, Normis, SergejsB

Manual:Metarouter *Source*: http://wiki.mikrotik.com/index.php?oldid=23697 *Contributors*: Janisk, Marisb, Nickshore, Normis, Omega-00, Uldis

Manual:MLPPP over single and multiple links *Source*: http://wiki.mikrotik.com/index.php?oldid=21522 *Contributors*: Marisb, Megis, Normis

Manual:MME wireless routing protocol *Source*: http://wiki.mikrotik.com/index.php?oldid=17441 *Contributors*: Atis, Eep, Marisb, Normis, SergejsB

Manual:MPLS *Source*: http://wiki.mikrotik.com/index.php?oldid=23554 *Contributors*: Eising, Marisb, Mplsguy, Normis, Route, SergejsB

Manual:MPLS over PPPoE *Source*: http://wiki.mikrotik.com/index.php?oldid=16772 *Contributors*: Marisb

Manual:MPLS/EXP bit behaviour *Source*: http://wiki.mikrotik.com/index.php?oldid=16502 *Contributors*: Marisb, Mplsguy, Normis

Manual:MPLS/LDP *Source*: http://wiki.mikrotik.com/index.php?oldid=17240 *Contributors*: Marisb

Manual:MPLS/Overview *Source*: http://wiki.mikrotik.com/index.php?oldid=23235 *Contributors*: Marisb, Mplsguy, Normis, Route

Manual:MPLS/Traffic-eng *Source*: http://wiki.mikrotik.com/index.php?oldid=17239 *Contributors*: Marisb

Manual:MPLSVPLS *Source*: http://wiki.mikrotik.com/index.php?oldid=16498 *Contributors*: Eep, Mag, Marisb, Mplsguy, Normis

Manual:Routing/Multicast *Source*: http://wiki.mikrotik.com/index.php?oldid=22211 *Contributors*: Janisk, Marisb, Normis, Route

Manual:Multicast detailed example *Source*: http://wiki.mikrotik.com/index.php?oldid=23533 *Contributors*: Eep, Janisk, Marisb, Normis, Route

Manual:Multicast SPT Switchover *Source*: http://wiki.mikrotik.com/index.php?oldid=20678 *Contributors*: Eep, Marisb

Manual:My First IPv6 Network *Source*: http://wiki.mikrotik.com/index.php?oldid=23739 *Contributors*: Marisb

Manual:IP/Neighbor discovery *Source*: http://wiki.mikrotik.com/index.php?oldid=22413 *Contributors*: Janisk, Kirshteins, Marisb

Manual:Netinstall *Source*: http://wiki.mikrotik.com/index.php?oldid=24152 *Contributors*: Becs, Janisk, Marisb, MarkSorensen, Normis, SergejsB

Manual:Tools/Netwatch *Source*: http://wiki.mikrotik.com/index.php?oldid=24172 *Contributors*: Janisk, Marisb, Normis

Manual:NTH in RouterOS 3.x *Source*: http://wiki.mikrotik.com/index.php?oldid=17073 *Contributors*: Marisb, Maximan, Normis

Manual:Nv2 *Source*: http://wiki.mikrotik.com/index.php?oldid=23773 *Contributors*: Becs, Janisk, Mplsguy, Normis, SergejsB, Uldis

Manual:Interface/OVPN *Source*: http://wiki.mikrotik.com/index.php?oldid=23916 *Contributors*: Marisb

Manual:OSPF as PE-CE routing protocol *Source*: http://wiki.mikrotik.com/index.php?oldid=16913 *Contributors*: Janisk, Marisb, Route

Manual:OSPF Case Studies *Source*: http://wiki.mikrotik.com/index.php?oldid=23058 *Contributors*: Janisk, Marisb

Manual:OSPF Forwarding Address *Source*: http://wiki.mikrotik.com/index.php?oldid=19471 *Contributors*: Marisb

Manual:OSPF-examples *Source*: http://wiki.mikrotik.com/index.php?oldid=22871 *Contributors*: Janisk, Marisb, Normis, Route

Manual:Routing/OSPF *Source*: http://wiki.mikrotik.com/index.php?oldid=22892 *Contributors*: Janisk, Marisb, Normis, Route

Manual:OSPF and Point-to-Point interfaces *Source*: http://wiki.mikrotik.com/index.php?oldid=17390 *Contributors*: Atis, Eep, Marisb

Manual:Interface/PPPoE *Source*: http://wiki.mikrotik.com/index.php?oldid=23491 *Contributors*: Janisk, Marisb, Normis

Manual:Interface/PPTP *Source*: http://wiki.mikrotik.com/index.php?oldid=22895 *Contributors*: Janisk, Marisb, SergejsB

Manual:IP/Proxy *Source*: http://wiki.mikrotik.com/index.php?oldid=23946 *Contributors*: Janisk, Marisb, Normis

Manual:Packet Flow *Source*: http://wiki.mikrotik.com/index.php?oldid=20478 *Contributors*: Janisk, Marisb, Megis, Normis

Manual:Performance Testing with Traffic Generator *Source*: http://wiki.mikrotik.com/index.php?oldid=24459 *Contributors*: Marisb, Normis

Manual:System/Packages *Source*: http://wiki.mikrotik.com/index.php?oldid=21218 *Contributors*: Enk, Janisk, Marisb, Normis, SergejsB

Manual:Tools/Profiler *Source*: http://wiki.mikrotik.com/index.php?oldid=20847 *Contributors*: Marisb

Manual:IP/Packing *Source*: http://wiki.mikrotik.com/index.php?oldid=18348 *Contributors*: Janisk

Manual:Password reset *Source*: http://wiki.mikrotik.com/index.php?oldid=24165 *Contributors*: Fbsd, Golden, Janisk, Marisb, Normis, Sizwan

Manual:PCC *Source*: http://wiki.mikrotik.com/index.php?oldid=20688 *Contributors*: Janisk, Marisb, Megis, Normis

Manual:PoE-Out *Source*: http://wiki.mikrotik.com/index.php?oldid=24062 *Contributors*: Janisk, Marisb, Normis

Manual:IP/Pools *Source*: http://wiki.mikrotik.com/index.php?oldid=17294 *Contributors*: Janisk, Marisb, Normis

Manual:IPv6/Pool *Source*: http://wiki.mikrotik.com/index.php?oldid=22676 *Contributors*: Marisb

Manual:Port *Source*: http://wiki.mikrotik.com/index.php?oldid=22456 *Contributors*: Marisb

Manual:IPv6 PD over PPP *Source*: http://wiki.mikrotik.com/index.php?oldid=22702 *Contributors*: Marisb

Manual:PPP AAA *Source*: http://wiki.mikrotik.com/index.php?oldid=22902 *Contributors*: Janisk, Marisb

Manual:Routing/Prefix list *Source*: http://wiki.mikrotik.com/index.php?oldid=17242 *Contributors*: Janisk, Marisb

Proxylizer *Source*: http://wiki.mikrotik.com/index.php?oldid=19754 *Contributors*: Girts, Janisd, Normis

Proxylizer/Getting Started *Source*: http://wiki.mikrotik.com/index.php?oldid=21745 *Contributors*: Digicomtech, Edgar abraham, Girts, Janisd, Janisk, Normis

Proxylizer/Introduction *Source*: http://wiki.mikrotik.com/index.php?oldid=10763 *Contributors*: Girts, Janisd, Normis

Manual:Purchasing a License for RouterOS *Source*: http://wiki.mikrotik.com/index.php?oldid=21858 *Contributors*: Eep, Janisk, Marisb, Normis, SergejsB, Sunfire

Manual:Replacement Key *Source*: http://wiki.mikrotik.com/index.php?oldid=17470 *Contributors*: Eep, Marisb, Normis

Manual:Queue *Source*: http://wiki.mikrotik.com/index.php?oldid=24562 *Contributors*: Eep, Janisk, Marisb, Megis, Normis, SergejsB

Manual:Queues - Burst *Source*: http://wiki.mikrotik.com/index.php?oldid=23428 *Contributors*: Eep, Janisk, Marisb, Megis, Normis

Manual:Queues - PCQ *Source*: http://wiki.mikrotik.com/index.php?oldid=21847 *Contributors*: Eep, Janisk, Marisb, Megis, Normis

Manual:Queues - PCQ Examples *Source*: http://wiki.mikrotik.com/index.php?oldid=23527 *Contributors*: Eep, Janisk, Marisb, Megis, Normis, Rieks, SergejsB, Wiki1981

Manual:Queue Size *Source*: http://wiki.mikrotik.com/index.php?oldid=16951 *Contributors*: Janisk, Marisb, Megis

Manual:IP/Route *Source*: http://wiki.mikrotik.com/index.php?oldid=20436 *Contributors*: Eep, Janisk, Marisb

User:Prince500 *Source*: http://wiki.mikrotik.com/index.php?oldid=24134 *Contributors*: Prince500

Manual:RADIUS Client *Source*: http://wiki.mikrotik.com/index.php?oldid=22741 *Contributors*: Agris, Janisk, Marisb, Normis, SergejsB, Uldis

Manual:Routing *Source*: http://wiki.mikrotik.com/index.php?oldid=16923 *Contributors*: Janisk, Marisb, Normis

Manual:Routing/RIP *Source*: http://wiki.mikrotik.com/index.php?oldid=17245 *Contributors*: Janisk, Marisb

Manual:RouterOS FAQ *Source*: http://wiki.mikrotik.com/index.php?oldid=21957 *Contributors*: B.Gates, Dsdee, Eep, Eugene, Grimp, Marisb, Nest, Normis, Rieks

Manual:RouterBOARD bad blocks *Source*: http://wiki.mikrotik.com/index.php?oldid=24124 *Contributors*: Becs, Janisk, Marisb, Normis

Manual:RouterOS features *Source*: http://wiki.mikrotik.com/index.php?oldid=22206 *Contributors*: Janisk, Marisb, Megis, Normis, SergejsB, Uldis

Manual:Route Selection Algorithm in RouterOS *Source*: http://wiki.mikrotik.com/index.php?oldid=16978 *Contributors*: Atis, Eep, Janisk, Marisb, Normis, Route

Manual:Routing Table Matcher *Source*: http://wiki.mikrotik.com/index.php?oldid=16980 *Contributors*: Janisk, Marisb

Manual:Routing/MME *Source*: http://wiki.mikrotik.com/index.php?oldid=17440 *Contributors*: Atis, Eep, Marisb

Manual:Interface/SSTP *Source*: http://wiki.mikrotik.com/index.php?oldid=24565 *Contributors*: Janisk, Marisb, Normis

Manual:IP/Services *Source*: http://wiki.mikrotik.com/index.php?oldid=20677 *Contributors*: Janisk, Marisb, SergejsB

Manual:IP/Settings *Source*: http://wiki.mikrotik.com/index.php?oldid=24585 *Contributors*: Marisb

Manual:Scripting *Source*: http://wiki.mikrotik.com/index.php?oldid=23898 *Contributors*: Burek, Janisk, Marisb, Normis, Proofreader

Manual:Scripting-examples *Source*: http://wiki.mikrotik.com/index.php?oldid=23792 *Contributors*: Janisk, Marisb, Normis, Vitell

Manual:Simple Static IPv6 Routing *Source*: http://wiki.mikrotik.com/index.php?oldid=23743 *Contributors*: Marisb

Manual:Simple Static Routing *Source*: http://wiki.mikrotik.com/index.php?oldid=21612 *Contributors*: Marisb, SergejsB

Manual:Store *Source*: http://wiki.mikrotik.com/index.php?oldid=23424 *Contributors*: Becs, Janisk, Marisb, Nest, Normis, SergejsB

Manual:Support Output File *Source*: http://wiki.mikrotik.com/index.php?oldid=22202 *Contributors*: Janisk, Marisb, Maximan, Normis, SergejsB

Manual:System *Source*: http://wiki.mikrotik.com/index.php?oldid=16850 *Contributors*: Janisk, Marisb

Manual:System/Scheduler *Source*: http://wiki.mikrotik.com/index.php?oldid=23832 *Contributors*: Janisk, Marisb, Normis

Manual:System/SSH client *Source*: http://wiki.mikrotik.com/index.php?oldid=23804 *Contributors*: Janisk, Marisb, Normis

Manual:Tools/Sigwatch *Source*: http://wiki.mikrotik.com/index.php?oldid=17804 *Contributors*: Janisk, Marisb, Normis

Manual:Tools/Sms *Source*: http://wiki.mikrotik.com/index.php?oldid=19384 *Contributors*: Eep, Jurisk, Marisb, Normis

Manual:Using scope and target-scope attributes *Source*: http://wiki.mikrotik.com/index.php?oldid=17248 *Contributors*: Atis, Eep, Janisk, Marisb

Manual:SimTest *Source*: http://wiki.mikrotik.com/index.php?oldid=23858 *Contributors*: Marisb

Manual:SNMP *Source*: http://wiki.mikrotik.com/index.php?oldid=23827 *Contributors*: Janisk, Marisb, Normis, Uldis

Manual:IP/SOCKS *Source*: http://wiki.mikrotik.com/index.php?oldid=16930 *Contributors*: Janisk, Marisb, Normis

Manual:Special Login *Source*: http://wiki.mikrotik.com/index.php?oldid=20485 *Contributors*: Janisk, Marisb, Normis

Manual:Spectral scan *Source*: http://wiki.mikrotik.com/index.php?oldid=23004 *Contributors*: Marisb, Normis

MikroTik Support *Source*: http://wiki.mikrotik.com/index.php?oldid=16931 *Contributors*: Janisk, Marisb, Normis

Manual:Switch Chip Features *Source*: http://wiki.mikrotik.com/index.php?oldid=23761 *Contributors*: Becs, Janisk, Kirshteins, Marisb, Megis, Normis

Manual:System/File *Source*: http://wiki.mikrotik.com/index.php?oldid=19362 *Contributors*: Marisb

Manual:System/LCD *Source*: http://wiki.mikrotik.com/index.php?oldid=23875 *Contributors*: Marisb, Normis

Manual:System/Note *Source*: http://wiki.mikrotik.com/index.php?oldid=21032 *Contributors*: Marisb

Manual:System/Resource *Source*: http://wiki.mikrotik.com/index.php?oldid=19354 *Contributors*: Marisb

Manual:System/Serial Console *Source*: http://wiki.mikrotik.com/index.php?oldid=20488 *Contributors*: Marisb, MarkSorensen, Normis

Manual:IP/SSH *Source*: http://wiki.mikrotik.com/index.php?oldid=23917 *Contributors*: Janisk, Marisb, Normis

Manual:IP/TFTP *Source*: http://wiki.mikrotik.com/index.php?oldid=24125 *Contributors*: Janisk, Normis

Manual:System/Time *Source*: http://wiki.mikrotik.com/index.php?oldid=20110 *Contributors*: Eep, Janisk, Marisb, Normis

Manual:Tools *Source*: http://wiki.mikrotik.com/index.php?oldid=16945 *Contributors*: Janisk, Marisb

Manual:Tools/Traffic Generator *Source*: http://wiki.mikrotik.com/index.php?oldid=23242 *Contributors*: Marisb

Manual:Connection oriented communication (TCP/IP) *Source*: http://wiki.mikrotik.com/index.php?oldid=19069 *Contributors*: Andriss, Marisb

Manual:TE tunnel auto bandwidth *Source*: http://wiki.mikrotik.com/index.php?oldid=16517 *Contributors*: Marisb, Mplsguy

Manual:TE Tunnels *Source*: http://wiki.mikrotik.com/index.php?oldid=16522 *Contributors*: Marisb, Mplsguy, Normis

Manual:TE Tunnels Example *Source*: http://wiki.mikrotik.com/index.php?oldid=19203 *Contributors*: Marisb

Manual:The Dude/First use *Source*: http://wiki.mikrotik.com/index.php?oldid=18446 *Contributors*: Normis

Manual:The Dude/Installation *Source*: http://wiki.mikrotik.com/index.php?oldid=18452 *Contributors*: Normis

Manual:TOC *Source*: http://wiki.mikrotik.com/index.php?oldid=24587 *Contributors*: Becs, Marisb

Manual:Tools/Bandwidth Test *Source*: http://wiki.mikrotik.com/index.php?oldid=19919 *Contributors*: Kirshteins, Marisb

Manual:Tools/Packet Sniffer *Source*: http://wiki.mikrotik.com/index.php?oldid=23105 *Contributors*: Janisk, Kirshteins, Marisb, SergejsB

Manual:Tools/Traffic Monitor *Source*: http://wiki.mikrotik.com/index.php?oldid=17451 *Contributors*: Kirshteins

Manual:Troubleshooting tools *Source*: http://wiki.mikrotik.com/index.php?oldid=22862 *Contributors*: Andriss, Janisk, Marisb, Normis

Manual:Interface/Traffic Engineering *Source*: http://wiki.mikrotik.com/index.php?oldid=22126 *Contributors*: Janisk, Marisb

Manual:IP/Traffic Flow *Source*: http://wiki.mikrotik.com/index.php?oldid=24148 *Contributors*: Janisk, Marisb, Normis

Manual:IP/UPnP *Source*: http://wiki.mikrotik.com/index.php?oldid=17536 *Contributors*: Janisk, Marisb, Normis

Manual:User Manager *Source*: http://wiki.mikrotik.com/index.php?oldid=19155 *Contributors*: Akangage, Bhhenry, Binhtanngo2003, Cmit, Comnetisp, Eep, Girts, Hellbound, Janisk, Levipatick, Marisb, Nest, Normis, Polokus, Rtkrh10, SergejsB, Uldis

Manual:Upgrading RouterOS *Source*: http://wiki.mikrotik.com/index.php?oldid=23803 *Contributors*: Axtell, Eep, Janisk, Marisb, Normis, SergejsB

Manual:Cisco VPLS *Source*: http://wiki.mikrotik.com/index.php?oldid=16871 *Contributors*: Janisk, Marisb, Mplsguy, Normis

Manual:Interface/Virtual-ethernet *Source*: http://wiki.mikrotik.com/index.php?oldid=22659 *Contributors*: Janisk, Marisb, Normis

Manual:Interface/VLAN *Source*: http://wiki.mikrotik.com/index.php?oldid=19562 *Contributors*: Janisk, Kirshteins, Marisb

Manual:Interface/VPLS *Source*: http://wiki.mikrotik.com/index.php?oldid=23886 *Contributors*: Janisk, Marisb

Manual:Interface/VRRP *Source*: http://wiki.mikrotik.com/index.php?oldid=20047 *Contributors*: Burek, Janisk, Marisb, Normis

Manual:Virtualization *Source*: http://wiki.mikrotik.com/index.php?oldid=20250 *Contributors*: Danielillu, Janisk, Marisb, Normis

Manual:VPLS Control Word *Source*: http://wiki.mikrotik.com/index.php?oldid=21732 *Contributors*: Marisb

Manual:VRRP-examples *Source*: http://wiki.mikrotik.com/index.php?oldid=21961 *Contributors*: Janisk, Marisb

Manual:Virtual Routing and Forwarding *Source*: http://wiki.mikrotik.com/index.php?oldid=16975 *Contributors*: Eep, Janisk, Marisb, Normis, Route

VRF Route Leaking *Source*: http://wiki.mikrotik.com/index.php?oldid=19182 *Contributors*: Eising, Marisb

Manual:Interface/Wireless *Source*: http://wiki.mikrotik.com/index.php?oldid=24506 *Contributors*: Eep, Janisk, Marisb, Normis, SergejsB, Uldis

Manual:System/Watchdog *Source*: http://wiki.mikrotik.com/index.php?oldid=23340 *Contributors*: Janisk, Marisb

Manual:Winbox *Source*: http://wiki.mikrotik.com/index.php?oldid=21085 *Contributors*: Janisk, Marisb, Nz monkey

Manual:Wireless card diagnostics *Source*: http://wiki.mikrotik.com/index.php?oldid=21093 *Contributors*: Janisk, Marisb, Normis

Manual:Wireless FAQ *Source*: http://wiki.mikrotik.com/index.php?oldid=23451 *Contributors*: Andreinazc, Janisk, Jorj, Marisb, Normis, SergejsB, Uldis

Manual:WMM *Source*: http://wiki.mikrotik.com/index.php?oldid=23347 *Contributors*: Eep, Janisk, Marisb, Normis

Manual:Tools/Wake on lan *Source*: http://wiki.mikrotik.com/index.php?oldid=17054 *Contributors*: Janisk, Marisb, Normis

Manual:Webfig *Source*: http://wiki.mikrotik.com/index.php?oldid=23656 *Contributors*: Janisk, Marisb, Normis

Manual:Wireless Advanced Channels *Source*: http://wiki.mikrotik.com/index.php?oldid=23771 *Contributors*: Marisb, Mplsguy, Normis, Uldis

Manual:Wireless AP Client *Source*: http://wiki.mikrotik.com/index.php?oldid=20439 *Contributors*: Marisb, SergejsB

Manual:Wireless Debug Logs *Source*: http://wiki.mikrotik.com/index.php?oldid=17342 *Contributors*: Eep, Janisk, Marisb, MarkSorensen, Mplsguy, Normis

Manual:Wireless Station Modes *Source*: http://wiki.mikrotik.com/index.php?oldid=20615 *Contributors*: Mplsguy, Normis

Manual:Xen *Source*: http://wiki.mikrotik.com/index.php?oldid=16862 *Contributors*: Janisk, Marisb, Mplsguy, Normis, SergejsB

Image Sources, Licenses and Contributors

Image:Version.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Version.png *License*: unknown *Contributors*: Normis
Image:Icon-note.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Icon-note.png *License*: unknown *Contributors*: Marisb, Route
File:Simple-te.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Simple-te.png *License*: unknown *Contributors*: Marisb
File: Simple-te-failover.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Simple-te-failover.png *License*: unknown *Contributors*: Marisb
File:Simple-te-extended.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Simple-te-extended.png *License*: unknown *Contributors*: Marisb
Image:Icon-warn.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Icon-warn.png *License*: unknown *Contributors*: Marisb, Route
Image:image10002.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image10002.gif *License*: unknown *Contributors*: Andriss
File:ipv6eui64.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ipv6eui64.png *License*: unknown *Contributors*: Marisb
File:ipv6-simple-address-example.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ipv6-simple-address-example.png *License*: unknown *Contributors*: Marisb
Image:VPLS.png *Source*: http://wiki.mikrotik.com/index.php?title=File:VPLS.png *License*: unknown *Contributors*: Karliskarlis
Image:IBGP eBGP.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:IBGP_eBGP.jpg *License*: unknown *Contributors*: Eugene
Image:BGP redistribution simple.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:BGP_redistribution_simple.jpg *License*: unknown *Contributors*: Eugene
Image:Icon-important.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Icon-important.png *License*: unknown *Contributors*: Marisb, Route
Image:ibgp_load_bal.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ibgp_load_bal.png *License*: unknown *Contributors*: Route
Image:ebgp_load_bal.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ebgp_load_bal.png *License*: unknown *Contributors*: Route
File:bgp_pe_ce.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Bgp_pe_ce.png *License*: unknown *Contributors*: Route
File:EoMPLS-cisco.png *Source*: http://wiki.mikrotik.com/index.php?title=File:EoMPLS-cisco.png *License*: unknown *Contributors*: Marisb
File:bonding-lacp-example.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Bonding-lacp-example.png *License*: unknown *Contributors*: Marisb
Image:bon-tlb.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Bon-tlb.png *License*: unknown *Contributors*: Marisb
Image:bon-alb.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Bon-alb.png *License*: unknown *Contributors*: Marisb
File:l2circuit-juniper.png *Source*: http://wiki.mikrotik.com/index.php?title=File:L2circuit-juniper.png *License*: unknown *Contributors*: Marisb
File:bgp-multihoming.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Bgp-multihoming.png *License*: unknown *Contributors*: Marisb
File:bgp-multihoming-download-sharing.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Bgp-multihoming-download-sharing.png *License*: unknown *Contributors*: Marisb
Image:BCP.png *Source*: http://wiki.mikrotik.com/index.php?title=File:BCP.png *License*: unknown *Contributors*: SergejsB
Image:BCP10.png *Source*: http://wiki.mikrotik.com/index.php?title=File:BCP10.png *License*: unknown *Contributors*: SergejsB
Image:BCP11.png *Source*: http://wiki.mikrotik.com/index.php?title=File:BCP11.png *License*: unknown *Contributors*: SergejsB
Image:BCP12.png *Source*: http://wiki.mikrotik.com/index.php?title=File:BCP12.png *License*: unknown *Contributors*: SergejsB
Image:BCP13.png *Source*: http://wiki.mikrotik.com/index.php?title=File:BCP13.png *License*: unknown *Contributors*: SergejsB
Image:BCP14.png *Source*: http://wiki.mikrotik.com/index.php?title=File:BCP14.png *License*: unknown *Contributors*: SergejsB
Image:BCP15.png *Source*: http://wiki.mikrotik.com/index.php?title=File:BCP15.png *License*: unknown *Contributors*: SergejsB
Image:BCP16.png *Source*: http://wiki.mikrotik.com/index.php?title=File:BCP16.png *License*: unknown *Contributors*: SergejsB
Image:BCP17.png *Source*: http://wiki.mikrotik.com/index.php?title=File:BCP17.png *License*: unknown *Contributors*: SergejsB
Image:Bonding ARP Monitoring Exam.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Bonding_ARP_Monitoring_Exam.jpg *License*: unknown *Contributors*: Eugene
Image:2009-04-06 1317.png *Source*: http://wiki.mikrotik.com/index.php?title=File:2009-04-06_1317.png *License*: unknown *Contributors*: Normis
Image:CD1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:CD1.png *License*: unknown *Contributors*: SergejsB
Image:CD3.png *Source*: http://wiki.mikrotik.com/index.php?title=File:CD3.png *License*: unknown *Contributors*: SergejsB
Image:CD4.png *Source*: http://wiki.mikrotik.com/index.php?title=File:CD4.png *License*: unknown *Contributors*: SergejsB
Image:CD6.png *Source*: http://wiki.mikrotik.com/index.php?title=File:CD6.png *License*: unknown *Contributors*: SergejsB
Image:CD7.png *Source*: http://wiki.mikrotik.com/index.php?title=File:CD7.png *License*: unknown *Contributors*: SergejsB
Image:CD8.png *Source*: http://wiki.mikrotik.com/index.php?title=File:CD8.png *License*: unknown *Contributors*: SergejsB
Image:CD9.png *Source*: http://wiki.mikrotik.com/index.php?title=File:CD9.png *License*: unknown *Contributors*: SergejsB
Image:CD10.png *Source*: http://wiki.mikrotik.com/index.php?title=File:CD10.png *License*: unknown *Contributors*: SergejsB
Image:CD11.png *Source*: http://wiki.mikrotik.com/index.php?title=File:CD11.png *License*: unknown *Contributors*: SergejsB
Image:2009-01-26 1346.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:2009-01-26_1346.jpg *License*: unknown *Contributors*: Normis
Image:dhcp-relay.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Dhcp-relay.png *License*: unknown *Contributors*: Marisb
File:dhcpv6-pd-example.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Dhcpv6-pd-example.png *License*: unknown *Contributors*: Marisb
File:eoip-example.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Eoip-example.png *License*: unknown *Contributors*: Marisb
File:site-to-site-gre-example.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Site-to-site-gre-example.png *License*: unknown *Contributors*: Marisb
File:Winbox-loader2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Winbox-loader2.png *License*: unknown *Contributors*: Marisb
File:Winbox-workarea.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Winbox-workarea.png *License*: unknown *Contributors*: Marisb
File:Webfig-2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Webfig-2.png *License*: unknown *Contributors*: Marisb
File:Flashfigdiagramm.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Flashfigdiagramm.png *License*: unknown *Contributors*: SergejsB
File:Flashfig.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Flashfig.png *License*: unknown *Contributors*: SergejsB
File:Flashfig2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Flashfig2.png *License*: unknown *Contributors*: SergejsB
File:Flashfig3.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Flashfig3.png *License*: unknown *Contributors*: SergejsB
File:Flashfig4.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Flashfig4.png *License*: unknown *Contributors*: SergejsB
File:Flashfig5.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Flashfig5.png *License*: unknown *Contributors*: SergejsB
File:Flashfig6.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Flashfig6.png *License*: unknown *Contributors*: SergejsB
file: graphing-mem.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Graphing-mem.png *License*: unknown *Contributors*: Marisb
file: graphing-mem-winbox.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Graphing-mem-winbox.png *License*: unknown *Contributors*: Marisb
File:WiringT568B.png *Source*: http://wiki.mikrotik.com/index.php?title=File:WiringT568B.png *License*: unknown *Contributors*: Normis
Image:Screw.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Screw.jpg *License*: unknown *Contributors*: Normis
File:DSC1557.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:DSC1557.jpg *License*: unknown *Contributors*: Normis
File:Poe shielded.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Poe_shielded.jpg *License*: unknown *Contributors*: Normis
File:Option-1.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Option-1.jpg *License*: unknown *Contributors*: Normis
File:Option-2.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Option-2.jpg *License*: unknown *Contributors*: Normis
Image:HTB_Example1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:HTB_Example1.png *License*: unknown *Contributors*: Megis
Image:HTB_Example2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:HTB_Example2.png *License*: unknown *Contributors*: Megis

Image:HTB_Example3.png *Source*: http://wiki.mikrotik.com/index.php?title=File:HTB_Example3.png *License*: unknown *Contributors*: Megis
Image:HTB_Example4.png *Source*: http://wiki.mikrotik.com/index.php?title=File:HTB_Example4.png *License*: unknown *Contributors*: Megis
Image:image8008.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image8008.gif *License*: unknown *Contributors*: Andriss
Image:image8009.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image8009.gif *License*: unknown *Contributors*: Andriss
Image:mesh_ex1.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Mesh_ex1.jpg *License*: unknown *Contributors*: Route
Image:hwmp_reactive_a.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Hwmp_reactive_a.png *License*: unknown *Contributors*: Route
Image:hwmp_reactive_b.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Hwmp_reactive_b.png *License*: unknown *Contributors*: Route
Image:hwmp_proactive_a.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Hwmp_proactive_a.png *License*: unknown *Contributors*: Route
Image:hwmp_proactive_b.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Hwmp_proactive_b.png *License*: unknown *Contributors*: Route
Image:hwmp_error_a.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Hwmp_error_a.png *License*: unknown *Contributors*: Route
Image:hwmp_error_b.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Hwmp_error_b.png *License*: unknown *Contributors*: Route
Image:mesh_bad_ex1.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Mesh_bad_ex1.jpg *License*: unknown *Contributors*: Route
Image:mesh_bad_ex2.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Mesh_bad_ex2.jpg *License*: unknown *Contributors*: Route
Image:ipip-sample.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ipip-sample.png *License*: unknown *Contributors*: Marisb
Image:ospfv3_setup.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ospfv3_setup.png *License*: unknown *Contributors*: Route
File:initial_screen_webfig.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Initial_screen_webfig.png *License*: unknown *Contributors*: Janisk
File:webfig_login.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Webfig_login.png *License*: unknown *Contributors*: Janisk
File:goto_system.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Goto_system.png *License*: unknown *Contributors*: Janisk, Marisb
File:users_management.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Users_management.png *License*: unknown *Contributors*: Janisk
File:ediit_create_user.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ediit_create_user.png *License*: unknown *Contributors*: Janisk
File:change_password_user_edit.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Change_password_user_edit.png *License*: unknown *Contributors*: Janisk
File:DHCP_client.png *Source*: http://wiki.mikrotik.com/index.php?title=File:DHCP_client.png *License*: unknown *Contributors*: Janisk
File:add_new_address.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Add_new_address.png *License*: unknown *Contributors*: Janisk
File:adding_new_address.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Adding_new_address.png *License*: unknown *Contributors*: Janisk
File:check_nat_masquerade.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Check_nat_masquerade.png *License*: unknown *Contributors*: Janisk
File:masqurade_rule.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Masqurade_rule.png *License*: unknown *Contributors*: Janisk
File:to_the_routes.png *Source*: http://wiki.mikrotik.com/index.php?title=File:To_the_routes.png *License*: unknown *Contributors*: Janisk
File:add_default_route.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Add_default_route.png *License*: unknown *Contributors*: Janisk
File:route_add_gateway.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Route_add_gateway.png *License*: unknown *Contributors*: Janisk
File:go_to_DNS_settings.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Go_to_DNS_settings.png *License*: unknown *Contributors*: Janisk
File:dns_add_server.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Dns_add_server.png *License*: unknown *Contributors*: Janisk
File:for_2_dns_servers.png *Source*: http://wiki.mikrotik.com/index.php?title=File:For_2_dns_servers.png *License*: unknown *Contributors*: Janisk
File:sntp_client_setup.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Sntp_client_setup.png *License*: unknown *Contributors*: Janisk
File:interface_open_details.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Interface_open_details.png *License*: unknown *Contributors*: Janisk
File:master_port.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Master_port.png *License*: unknown *Contributors*: Janisk
File:remove_bridge_port.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Remove_bridge_port.png *License*: unknown *Contributors*: Janisk
File:secuirtas_profle.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Secuirtas_profle.png *License*: unknown *Contributors*: Janisk
File:creating_security_profile.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Creating_security_profile.png *License*: unknown *Contributors*: Janisk
File:goto_wireless.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Goto_wireless.png *License*: unknown *Contributors*: Janisk
File:wireless_general.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Wireless_general.png *License*: unknown *Contributors*: Janisk
File:wireless_ht.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Wireless_ht.png *License*: unknown *Contributors*: Janisk
File:enable_wireless.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Enable_wireless.png *License*: unknown *Contributors*: Janisk
File:Brtidge_ports_view.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Brtidge_ports_view.png *License*: unknown *Contributors*: Janisk
File:add_bridge_port.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Add_bridge_port.png *License*: unknown *Contributors*: Janisk
File:set_up_bridge.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Set_up_bridge.png *License*: unknown *Contributors*: Janisk
File:correct_address_1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Correct_address_1.png *License*: unknown *Contributors*: Janisk
File:change_passwd_current_user.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Change_passwd_current_user.png *License*: unknown *Contributors*: Janisk
File:wifi_freq_usage1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Wifi_freq_usage1.png *License*: unknown *Contributors*: Janisk
File:wifi_freq_usage.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Wifi_freq_usage.png *License*: unknown *Contributors*: Janisk
File:wifi_adv_mode.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Wifi_adv_mode.png *License*: unknown *Contributors*: Janisk
File:Wifi_select_country.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Wifi_select_country.png *License*: unknown *Contributors*: Janisk
File:dst-nat.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Dst-nat.png *License*: unknown *Contributors*: Janisk
Image:l3vpn-internet.png *Source*: http://wiki.mikrotik.com/index.php?title=File:L3vpn-internet.png *License*: unknown *Contributors*: Route
Image:Mikrotik-per-vrf-snat.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Mikrotik-per-vrf-snat.png *License*: unknown *Contributors*: Eising
file:site-to-site-ipsec-example.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Site-to-site-ipsec-example.png *License*: unknown *Contributors*: Marisb
file:ipsec-l2tp-example.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ipsec-l2tp-example.png *License*: unknown *Contributors*: Marisb
File:ipv6-lifetime.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ipv6-lifetime.png *License*: unknown *Contributors*: Marisb
Image:Key0.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Key0.png *License*: unknown *Contributors*: Normis
Image:Key1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Key1.png *License*: unknown *Contributors*: Normis
Image:Key2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Key2.png *License*: unknown *Contributors*: Normis
Image:Key3.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Key3.png *License*: unknown *Contributors*: Normis
Image:Key4.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Key4.png *License*: unknown *Contributors*: Normis
File:l2tp-rem-office.png *Source*: http://wiki.mikrotik.com/index.php?title=File:L2tp-rem-office.png *License*: unknown *Contributors*: Marisb
File:site-to-site-l2tp-example.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Site-to-site-l2tp-example.png *License*: unknown *Contributors*: Marisb
Image:License menu.png *Source*: http://wiki.mikrotik.com/index.php?title=File:License_menu.png *License*: unknown *Contributors*: Normis
Image:2009-05-21 1608.png *Source*: http://wiki.mikrotik.com/index.php?title=File:2009-05-21_1608.png *License*: unknown *Contributors*: Normis
File:PasteLicense.png *Source*: http://wiki.mikrotik.com/index.php?title=File:PasteLicense.png *License*: unknown *Contributors*: SergejsB
File:ApplyLicenseWinbox.png *Source*: http://wiki.mikrotik.com/index.php?title=File:ApplyLicenseWinbox.png *License*: unknown *Contributors*: SergejsB
Image:Logging2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Logging2.png *License*: unknown *Contributors*: Normis
Image:Logging1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Logging1.png *License*: unknown *Contributors*: Normis
Image:l3vpn.png *Source*: http://wiki.mikrotik.com/index.php?title=File:L3vpn.png *License*: unknown *Contributors*: Route

File:two-link-example.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Two-link-example.png *License*: unknown *Contributors*: Marisb
Image:2009-06-04 1555.png *Source*: http://wiki.mikrotik.com/index.php?title=File:2009-06-04_1555.png *License*: unknown *Contributors*: Normis
Image:2009-06-04 1556.png *Source*: http://wiki.mikrotik.com/index.php?title=File:2009-06-04_1556.png *License*: unknown *Contributors*: Normis
Image:2009-06-04 1557.png *Source*: http://wiki.mikrotik.com/index.php?title=File:2009-06-04_1557.png *License*: unknown *Contributors*: Normis
Image:2009-06-04 1558.png *Source*: http://wiki.mikrotik.com/index.php?title=File:2009-06-04_1558.png *License*: unknown *Contributors*: Normis
Image:2009-06-04 1559.png *Source*: http://wiki.mikrotik.com/index.php?title=File:2009-06-04_1559.png *License*: unknown *Contributors*: Normis
Image:2009-06-04 1560.png *Source*: http://wiki.mikrotik.com/index.php?title=File:2009-06-04_1560.png *License*: unknown *Contributors*: Normis
Image:MTU general explanation.png *Source*: http://wiki.mikrotik.com/index.php?title=File:MTU_general_explanation.png *License*: unknown *Contributors*: Megis
Image:MTUSimpleRouting.png *Source*: http://wiki.mikrotik.com/index.php?title=File:MTUSimpleRouting.png *License*: unknown *Contributors*: SergejsB
Image:MTUVLANENCAP.png *Source*: http://wiki.mikrotik.com/index.php?title=File:MTUVLANENCAP.png *License*: unknown *Contributors*: SergejsB
Image:MTUMPLS2Tags.png *Source*: http://wiki.mikrotik.com/index.php?title=File:MTUMPLS2Tags.png *License*: unknown *Contributors*: SergejsB
Image:MTUVPLS.png *Source*: http://wiki.mikrotik.com/index.php?title=File:MTUVPLS.png *License*: unknown *Contributors*: SergejsB
Image:L2MTU example.png *Source*: http://wiki.mikrotik.com/index.php?title=File:L2MTU_example.png *License*: unknown *Contributors*: Megis
image:snapshot2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Snapshot2.png *License*: unknown *Contributors*: Marisb
File:mlppp.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Mlppp.png *License*: unknown *Contributors*: Marisb
Image:mpls-pppoe-f.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Mpls-pppoe-f.png *License*: unknown *Contributors*: Marisb
Image:mpls-pppoe-vpls.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Mpls-pppoe-vpls.png *License*: unknown *Contributors*: Marisb
Image:Rp-traf.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Rp-traf.jpg *License*: unknown *Contributors*: Marisb
Image:Sg-join.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Sg-join.jpg *License*: unknown *Contributors*: Marisb
Image:Sg-prune.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Sg-prune.jpg *License*: unknown *Contributors*: Marisb
File:First-IPv6-example.png *Source*: http://wiki.mikrotik.com/index.php?title=File:First-IPv6-example.png *License*: unknown *Contributors*: Marisb
File:Tunnel-broker.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Tunnel-broker.png *License*: unknown *Contributors*: Marisb
File:2009-01-27 1224.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:2009-01-27_1224.jpg *License*: unknown *Contributors*: Normis
Image:NetinstallStart.png *Source*: http://wiki.mikrotik.com/index.php?title=File:NetinstallStart.png *License*: unknown *Contributors*: SergejsB
Image:Nconfig.PNG *Source*: http://wiki.mikrotik.com/index.php?title=File:Nconfig.PNG *License*: unknown *Contributors*: SergejsB
Image:NConfig3.png *Source*: http://wiki.mikrotik.com/index.php?title=File:NConfig3.png *License*: unknown *Contributors*: SergejsB
Image:NetinstallC4.png *Source*: http://wiki.mikrotik.com/index.php?title=File:NetinstallC4.png *License*: unknown *Contributors*: SergejsB
Image:NetinstallC5.png *Source*: http://wiki.mikrotik.com/index.php?title=File:NetinstallC5.png *License*: unknown *Contributors*: SergejsB
Image:NetinstallC6.png *Source*: http://wiki.mikrotik.com/index.php?title=File:NetinstallC6.png *License*: unknown *Contributors*: SergejsB
Image:PasswordReset.png *Source*: http://wiki.mikrotik.com/index.php?title=File:PasswordReset.png *License*: unknown *Contributors*: SergejsB
Image:ospf-pe-ce.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ospf-pe-ce.png *License*: unknown *Contributors*: Route
Image:ospf-pe-ce-sham-links.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ospf-pe-ce-sham-links.png *License*: unknown *Contributors*: Route
Image:ospf-header.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ospf-header.png *License*: unknown *Contributors*: Marisb
Image:ospf-hello.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ospf-hello.png *License*: unknown *Contributors*: Marisb
Image:ospf-adjacency.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ospf-adjacency.png *License*: unknown *Contributors*: Marisb
Image:sp-net.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Sp-net.png *License*: unknown *Contributors*: Marisb
Image:sp-tree.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Sp-tree.png *License*: unknown *Contributors*: Marisb
Image:ospf-basic.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ospf-basic.png *License*: unknown *Contributors*: Marisb
Image:backbone-s.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Backbone-s.png *License*: unknown *Contributors*: Marisb
Image:area-br.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Area-br.png *License*: unknown *Contributors*: Marisb
Image:basic-multi-area.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Basic-multi-area.png *License*: unknown *Contributors*: Marisb
Image:vlink-area.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Vlink-area.png *License*: unknown *Contributors*: Marisb
Image:vlink-backbone.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Vlink-backbone.png *License*: unknown *Contributors*: Marisb
Image:stub-example.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Stub-example.png *License*: unknown *Contributors*: Marisb
Image:nssa-example.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Nssa-example.png *License*: unknown *Contributors*: Marisb
File:ospf-forwarding.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ospf-forwarding.png *License*: unknown *Contributors*: Marisb
File:ospf-forwarding-traffic.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ospf-forwarding-traffic.png *License*: unknown *Contributors*: Marisb
Image:image6005.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image6005.gif *License*: unknown *Contributors*: Andriss
Image:image6006.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image6006.gif *License*: unknown *Contributors*: Andriss
Image:ospf-nbma.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ospf-nbma.png *License*: unknown *Contributors*: Route
Image:pppoe-discovery.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Pppoe-discovery.png *License*: unknown *Contributors*: Marisb
File:pppoe-apex.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Pppoe-apex.png *License*: unknown *Contributors*: Marisb
File:pptp-rem-offoce.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Pptp-rem-offoce.png *License*: unknown *Contributors*: Marisb
File:site-to-site-pptp-example.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Site-to-site-pptp-example.png *License*: unknown *Contributors*: Marisb
Image:image10001.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image10001.gif *License*: unknown *Contributors*: Andriss
Image:Bridge_final.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Bridge_final.png *License*: unknown *Contributors*: Megis
Image:IP_final.png *Source*: http://wiki.mikrotik.com/index.php?title=File:IP_final.png *License*: unknown *Contributors*: Megis
Image:Input_interface.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Input_interface.jpg *License*: unknown *Contributors*: Megis
Image:output_interface.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Output_interface.jpg *License*: unknown *Contributors*: Megis
Image:local_process-_in.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Local_process-_in.jpg *License*: unknown *Contributors*: Megis
Image:local_process-_out.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Local_process-_out.jpg *License*: unknown *Contributors*: Megis
Image:connection_tracking.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Connection_tracking.jpg *License*: unknown *Contributors*: Megis
Image:Filter_input.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Filter_input.jpg *License*: unknown *Contributors*: Megis
Image:Filter_forward.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Filter_forward.jpg *License*: unknown *Contributors*: Megis
Image:Filter_output.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Filter_output.jpg *License*: unknown *Contributors*: Megis
Image:src_nat.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Src_nat.jpg *License*: unknown *Contributors*: Megis
Image:dst_nat.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Dst_nat.jpg *License*: unknown *Contributors*: Megis
Image:mangle_prerouting.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Mangle_prerouting.jpg *License*: unknown *Contributors*: Megis
Image:mangle_input.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Mangle_input.jpg *License*: unknown *Contributors*: Megis
Image:mangle_forward.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Mangle_forward.jpg *License*: unknown *Contributors*: Megis
Image:mangle_output.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Mangle_output.jpg *License*: unknown *Contributors*: Megis

Image:mangle_postrouting.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Mangle_postrouting.jpg *License*: unknown *Contributors*: Megis
Image:global_in.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Global_in.jpg *License*: unknown *Contributors*: Megis
Image:global_out.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Global_out.jpg *License*: unknown *Contributors*: Megis
Image:Interface HTB.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Interface_HTB.jpg *License*: unknown *Contributors*: Megis
Image:IPsec_policy.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:IPsec_policy.jpg *License*: unknown *Contributors*: Megis
Image:accounting.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Accounting.jpg *License*: unknown *Contributors*: Megis
Image:use_ip_firewall.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Use_ip_firewall.jpg *License*: unknown *Contributors*: Megis
Image:bridge_input.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Bridge_input.jpg *License*: unknown *Contributors*: Megis
Image:Bridge_forward.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Bridge_forward.jpg *License*: unknown *Contributors*: Megis
Image:Bridge_output.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Bridge_output.jpg *License*: unknown *Contributors*: Megis
Image:Bridge_dst_nat.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Bridge_dst_nat.jpg *License*: unknown *Contributors*: Megis
Image:Bridge_src_nat.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Bridge_src_nat.jpg *License*: unknown *Contributors*: Megis
Image:In-interface-bridge.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:In-interface-bridge.jpg *License*: unknown *Contributors*: Megis
Image:hotspot_in.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Hotspot_in.jpg *License*: unknown *Contributors*: Megis
Image:Bridge Desicion.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Bridge_Desicion.jpg *License*: unknown *Contributors*: Megis
Image:bridge_decision.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Bridge_decision.jpg *License*: unknown *Contributors*: Megis
Image:routing_decision.JPG *Source*: http://wiki.mikrotik.com/index.php?title=File:Routing_decision.JPG *License*: unknown *Contributors*: Megis
Image:routing_adjustment.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Routing_adjustment.jpg *License*: unknown *Contributors*: Megis
Image:TTL=TTL-1.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:TTL=TTL-1.jpg *License*: unknown *Contributors*: Megis
Image:IPSec_Decryption.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:IPSec_Decryption.jpg *License*: unknown *Contributors*: Megis
Image:IPSec_Encryption.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:IPSec_Encryption.jpg *License*: unknown *Contributors*: Megis
Image:out_interface_bridge.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Out_interface_bridge.jpg *License*: unknown *Contributors*: Megis
Image:Hotspot_out.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Hotspot_out.jpg *License*: unknown *Contributors*: Megis
Image:Packet_Flow_Example_1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Packet_Flow_Example_1.png *License*: unknown *Contributors*: Megis
Image:Packet_Flow_Example_2c.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Packet_Flow_Example_2c.png *License*: unknown *Contributors*: Megis
Image:Packet_Flow_Example_3_1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Packet_Flow_Example_3_1.png *License*: unknown *Contributors*: Megis
Image:Packet_Flow_Example_3_2c.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Packet_Flow_Example_3_2c.png *License*: unknown *Contributors*: Megis
Image:Packet_Flow_Example_4c.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Packet_Flow_Example_4c.png *License*: unknown *Contributors*: Megis
Image:Packet_Flow_Example_5c.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Packet_Flow_Example_5c.png *License*: unknown *Contributors*: Megis
File:RB1100AHx2-diagram-streams.png *Source*: http://wiki.mikrotik.com/index.php?title=File:RB1100AHx2-diagram-streams.png *License*: unknown *Contributors*: Marisb
File:1100_outline-wires.png *Source*: http://wiki.mikrotik.com/index.php?title=File:1100_outline-wires.png *License*: unknown *Contributors*: Marisb
File:profiler.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Profiler.png *License*: unknown *Contributors*: Marisb
File:262 hi res.png *Source*: http://wiki.mikrotik.com/index.php?title=File:262_hi_res.png *License*: unknown *Contributors*: Normis
Image:Resethole.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Resethole.jpg *License*: unknown *Contributors*: Normis
File:Passw.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Passw.jpg *License*: unknown *Contributors*: Normis
Image:CRW 5184.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:CRW_5184.jpg *License*: unknown *Contributors*: Normis
Image:LoadBalancing.png *Source*: http://wiki.mikrotik.com/index.php?title=File:LoadBalancing.png *License*: unknown *Contributors*: Normis
Image:setup.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Setup.png *License*: unknown *Contributors*: Janisd
Image:ProxylizerCollab.png *Source*: http://wiki.mikrotik.com/index.php?title=File:ProxylizerCollab.png *License*: unknown *Contributors*: Girts
Image:Purchase1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Purchase1.png *License*: unknown *Contributors*: Normis
Image:Purchase2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Purchase2.png *License*: unknown *Contributors*: Normis
Image:Purchase3.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Purchase3.png *License*: unknown *Contributors*: Normis
Image:Purchase4.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Purchase4.png *License*: unknown *Contributors*: Normis
Image:Purchase5.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Purchase5.png *License*: unknown *Contributors*: Normis
Image:Rep1.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Rep1.jpg *License*: unknown *Contributors*: Normis
Image:Rep2.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Rep2.jpg *License*: unknown *Contributors*: Normis
Image:Rep3.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Rep3.jpg *License*: unknown *Contributors*: Normis
Image:image8001.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image8001.gif *License*: unknown *Contributors*: Andriss
Image:image8006.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image8006.gif *License*: unknown *Contributors*: Andriss
Image:image8002.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image8002.gif *License*: unknown *Contributors*: Andriss
Image:image8003.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image8003.gif *License*: unknown *Contributors*: Andriss
Image:Burst time.16.part1.JPG *Source*: http://wiki.mikrotik.com/index.php?title=File:Burst_time.16.part1.JPG *License*: unknown *Contributors*: Megis
Image:Burst time.16.part2.JPG *Source*: http://wiki.mikrotik.com/index.php?title=File:Burst_time.16.part2.JPG *License*: unknown *Contributors*: Megis
Image:Burst time.8.part1.JPG *Source*: http://wiki.mikrotik.com/index.php?title=File:Burst_time.8.part1.JPG *License*: unknown *Contributors*: Megis
Image:Burst time.8.part2.JPG *Source*: http://wiki.mikrotik.com/index.php?title=File:Burst_time.8.part2.JPG *License*: unknown *Contributors*: Megis
Image:PCQ_Alg.png *Source*: http://wiki.mikrotik.com/index.php?title=File:PCQ_Alg.png *License*: unknown *Contributors*: Megis
Image:PCQ_Example1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:PCQ_Example1.png *License*: unknown *Contributors*: Megis
Image:PCQ_Example2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:PCQ_Example2.png *License*: unknown *Contributors*: Megis
Image:PCQ3.png *Source*: http://wiki.mikrotik.com/index.php?title=File:PCQ3.png *License*: unknown *Contributors*: Megis
Image:PCQ4.png *Source*: http://wiki.mikrotik.com/index.php?title=File:PCQ4.png *License*: unknown *Contributors*: Megis
Image:PCQ.png *Source*: http://wiki.mikrotik.com/index.php?title=File:PCQ.png *License*: unknown *Contributors*: SergejsB
Image:Queue_size_No_Limit.PNG *Source*: http://wiki.mikrotik.com/index.php?title=File:Queue_size_No_Limit.PNG *License*: unknown *Contributors*: Megis
Image:Queue_size_0_packets.PNG *Source*: http://wiki.mikrotik.com/index.php?title=File:Queue_size_0_packets.PNG *License*: unknown *Contributors*: Megis
Image:Queue_size_Unlimited_Packets.PNG *Source*: http://wiki.mikrotik.com/index.php?title=File:Queue_size_Unlimited_Packets.PNG *License*: unknown *Contributors*: Megis
Image:Queue_size_10_packets.PNG *Source*: http://wiki.mikrotik.com/index.php?title=File:Queue_size_10_packets.PNG *License*: unknown *Contributors*: Megis
Image:Queue_size_50_packets.PNG *Source*: http://wiki.mikrotik.com/index.php?title=File:Queue_size_50_packets.PNG *License*: unknown *Contributors*: Megis
Image:rib.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Rib.png *License*: unknown *Contributors*: Eep
Image:conn_route_and_address.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Conn_route_and_address.png *License*: unknown *Contributors*: Eep
Image:scope_and_target_scope.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Scope_and_target_scope.png *License*: unknown *Contributors*: Eep
Image:nh-lookup.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Nh-lookup.png *License*: unknown *Contributors*: Eep
Image:fib.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Fib.png *License*: unknown *Contributors*: Eep

Image:Block.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Block.png *License*: unknown *Contributors*: Normis
File:RTM.png *Source*: http://wiki.mikrotik.com/index.php?title=File:RTM.png *License*: unknown *Contributors*: Marisb
File:sstp-how-works.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Sstp-how-works.png *License*: unknown *Contributors*: Marisb
File:sstp-rem-office.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Sstp-rem-office.png *License*: unknown *Contributors*: Marisb
File:site-to-site-sstp-example.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Site-to-site-sstp-example.png *License*: unknown *Contributors*: Marisb
Image:simple-ipv6-routing.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Simple-ipv6-routing.png *License*: unknown *Contributors*: Marisb
Image:SR1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:SR1.png *License*: unknown *Contributors*: Marisb, SergejsB
Image:Store.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Store.png *License*: unknown *Contributors*: Normis
Image:Supout.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Supout.png *License*: unknown *Contributors*: Normis
Image:Supout2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Supout2.png *License*: unknown *Contributors*: Normis
Image:Supout3.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Supout3.png *License*: unknown *Contributors*: Normis
File:Total-download-cacti.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Total-download-cacti.png *License*: unknown *Contributors*: Normis
Image:special-login-setup.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Special-login-setup.png *License*: unknown *Contributors*: Marisb
File:Spectral-history.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Spectral-history.png *License*: unknown *Contributors*: Marisb, Normis
File:Spectral-scan.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Spectral-scan.png *License*: unknown *Contributors*: Normis
File:Spectral1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Spectral1.png *License*: unknown *Contributors*: Normis
File:Spectral2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Spectral2.png *License*: unknown *Contributors*: Normis
Image:switch1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Switch1.png *License*: unknown *Contributors*: Kirshteins
Image:switch2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Switch2.png *License*: unknown *Contributors*: Kirshteins
Image:switch3.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Switch3.png *License*: unknown *Contributors*: Kirshteins
Image:switch4.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Switch4.png *License*: unknown *Contributors*: Kirshteins
File:ar8316_trunk.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ar8316_trunk.png *License*: unknown *Contributors*: Kirshteins
File:Lcd.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Lcd.png *License*: unknown *Contributors*: Normis
Image:2009-04-03 1314.png *Source*: http://wiki.mikrotik.com/index.php?title=File:2009-04-03_1314.png *License*: unknown *Contributors*: Normis
File:traffic-gen-routing-ipsec.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Traffic-gen-routing-ipsec.png *License*: unknown *Contributors*: Marisb
Image:image2001.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image2001.gif *License*: unknown *Contributors*: Andriss
Image:image2002.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image2002.gif *License*: unknown *Contributors*: Andriss
Image:image2003.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image2003.gif *License*: unknown *Contributors*: Andriss
Image:image2004.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image2004.gif *License*: unknown *Contributors*: Andriss
Image:image2005.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image2005.gif *License*: unknown *Contributors*: Andriss
File:mpls-te-example.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Mpls-te-example.png *License*: unknown *Contributors*: Marisb
File:SS-2010-06-30 11.18.32.png *Source*: http://wiki.mikrotik.com/index.php?title=File:SS-2010-06-30_11.18.32.png *License*: unknown *Contributors*: Normis
File:SS-2010-06-30 11.21.53.png *Source*: http://wiki.mikrotik.com/index.php?title=File:SS-2010-06-30_11.21.53.png *License*: unknown *Contributors*: Normis
File:Dude webpage.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Dude_webpage.png *License*: unknown *Contributors*: Normis
File:The Dude Setup -2010-06-30 11.03.48.png *Source*: http://wiki.mikrotik.com/index.php?title=File:The_Dude_Setup_-2010-06-30_11.03.48.png *License*: unknown *Contributors*: Normis
File:SS-2010-06-30 11.09.15.png *Source*: http://wiki.mikrotik.com/index.php?title=File:SS-2010-06-30_11.09.15.png *License*: unknown *Contributors*: Normis
Image:image11001.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image11001.gif *License*: unknown *Contributors*: Andriss
Image:image11002.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image11002.gif *License*: unknown *Contributors*: Andriss
Image:traffic-flow-1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Traffic-flow-1.png *License*: unknown *Contributors*: Marisb
Image:traffic-flow-2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Traffic-flow-2.png *License*: unknown *Contributors*: Marisb
Image:traffic-flow-4.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Traffic-flow-4.png *License*: unknown *Contributors*: Marisb
Image:Upnp.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Upnp.jpg *License*: unknown *Contributors*: Normis
Image:Downl.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Downl.jpg *License*: unknown *Contributors*: Normis
Image:Winbox1.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Winbox1.jpg *License*: unknown *Contributors*: Normis
Image:Winb2.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Winb2.jpg *License*: unknown *Contributors*: Normis
Image:Up4.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Up4.jpg *License*: unknown *Contributors*: Normis
Image:Dude1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Dude1.png *License*: unknown *Contributors*: SergejsB
Image:Dude2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Dude2.png *License*: unknown *Contributors*: SergejsB
Image:Dude3.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Dude3.png *License*: unknown *Contributors*: SergejsB
Image:Dude5.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Dude5.png *License*: unknown *Contributors*: SergejsB
Image:Dude6.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Dude6.png *License*: unknown *Contributors*: SergejsB
Image:Dude7.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Dude7.png *License*: unknown *Contributors*: SergejsB
Image:Dude8.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Dude8.png *License*: unknown *Contributors*: SergejsB
Image:Dude13.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Dude13.png *License*: unknown *Contributors*: SergejsB
Image:Dude14.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Dude14.png *License*: unknown *Contributors*: SergejsB
Image:image12001.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image12001.gif *License*: unknown *Contributors*: Andriss
Image:image12003.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image12003.gif *License*: unknown *Contributors*: Andriss
Image:image12004.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image12004.gif *License*: unknown *Contributors*: Andriss
Image:image12005.gif *Source*: http://wiki.mikrotik.com/index.php?title=File:Image12005.gif *License*: unknown *Contributors*: Andriss
File:Slash32.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Slash32.png *License*: unknown *Contributors*: Janisk
Image:vrrp-simple.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Vrrp-simple.png *License*: unknown *Contributors*: Marisb
Image:vrrp-no-owner.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Vrrp-no-owner.png *License*: unknown *Contributors*: Marisb
Image:Vrrp-State.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Vrrp-State.png *License*: unknown *Contributors*: Marisb
File:VPLS_CW.png *Source*: http://wiki.mikrotik.com/index.php?title=File:VPLS_CW.png *License*: unknown *Contributors*: Marisb
Image:vrrp-basic.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Vrrp-basic.png *License*: unknown *Contributors*: Marisb
Image:vrrp-load-sharing.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Vrrp-load-sharing.png *License*: unknown *Contributors*: Marisb
Image:l3vpn-simple.png *Source*: http://wiki.mikrotik.com/index.php?title=File:L3vpn-simple.png *License*: unknown *Contributors*: Route
Image:l3vpn-two-customers.png *Source*: http://wiki.mikrotik.com/index.php?title=File:L3vpn-two-customers.png *License*: unknown *Contributors*: Route
Image:Leaking-routes.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Leaking-routes.png *License*: unknown *Contributors*: Eising
Image:2009-02-06 1518.png *Source*: http://wiki.mikrotik.com/index.php?title=File:2009-02-06_1518.png *License*: unknown *Contributors*: Normis
File:Snoop1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Snoop1.png *License*: unknown *Contributors*: Normis

File:Snoop2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Snoop2.png *License*: unknown *Contributors*: Normis
File:Snoop3.PNG *Source*: http://wiki.mikrotik.com/index.php?title=File:Snoop3.PNG *License*: unknown *Contributors*: Normis
File:win-web-snap.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Win-web-snap.png *License*: unknown *Contributors*: Marisb, SergejsB
File:winbox-loader.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Winbox-loader.png *License*: unknown *Contributors*: Marisb
File:winbox-loader2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Winbox-loader2.png *License*: unknown *Contributors*: Marisb
File:winbox-ipv6-loader.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Winbox-ipv6-loader.png *License*: unknown *Contributors*: Marisb
File:winbox-ipv6nd.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Winbox-ipv6nd.png *License*: unknown *Contributors*: Marisb
File:winbox-win-child.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Winbox-win-child.png *License*: unknown *Contributors*: Marisb
File:win-add.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Win-add.png *License*: unknown *Contributors*: Marisb
File:win-remove.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Win-remove.png *License*: unknown *Contributors*: Marisb
File:win-enable.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Win-enable.png *License*: unknown *Contributors*: Marisb
File:win-disable.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Win-disable.png *License*: unknown *Contributors*: Marisb
File:win-comment.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Win-comment.png *License*: unknown *Contributors*: Marisb
File:win-sort.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Win-sort.png *License*: unknown *Contributors*: Marisb
File:winbox-window-search.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Winbox-window-search.png *License*: unknown *Contributors*: Marisb
File:Winbox-window-sort.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Winbox-window-sort.png *License*: unknown *Contributors*: Marisb
File:Winbox-window-field.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Winbox-window-field.png *License*: unknown *Contributors*: Marisb
File:Winbox-window-detail.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Winbox-window-detail.png *License*: unknown *Contributors*: Marisb
File:Winbox-window-category.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Winbox-window-category.png *License*: unknown *Contributors*: Marisb
File:Winbox1.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Winbox1.jpg *License*: unknown *Contributors*: Normis
File:winbox-window-trafmon.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Winbox-window-trafmon.png *License*: unknown *Contributors*: Marisb
Image:2009-04-02_1241.png *Source*: http://wiki.mikrotik.com/index.php?title=File:2009-04-02_1241.png *License*: unknown *Contributors*: Normis
Image:2009-04-02_1241_001.png *Source*: http://wiki.mikrotik.com/index.php?title=File:2009-04-02_1241_001.png *License*: unknown *Contributors*: Normis
Image:2009-04-02_1242.png *Source*: http://wiki.mikrotik.com/index.php?title=File:2009-04-02_1242.png *License*: unknown *Contributors*: Normis
Image:2009-04-02_1242_001.png *Source*: http://wiki.mikrotik.com/index.php?title=File:2009-04-02_1242_001.png *License*: unknown *Contributors*: Normis
File:Storm1.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Storm1.jpg *License*: unknown *Contributors*: Normis
File:Storm2.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Storm2.jpg *License*: unknown *Contributors*: Normis
Image:Storm3.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Storm3.jpg *License*: unknown *Contributors*: Normis
File:DSC0634.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:DSC0634.jpg *License*: unknown *Contributors*: Normis
File:DSC0633.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:DSC0633.jpg *License*: unknown *Contributors*: Normis
File:Dz1.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Dz1.jpg *License*: unknown *Contributors*: Normis
File:Dz2.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Dz2.jpg *License*: unknown *Contributors*: Normis
File:Dz3.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Dz3.jpg *License*: unknown *Contributors*: Normis
File:Dc grounded.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Dc_grounded.jpg *License*: unknown *Contributors*: Normis
File:Dc notgrounded.jpg *Source*: http://wiki.mikrotik.com/index.php?title=File:Dc_notgrounded.jpg *License*: unknown *Contributors*: Normis
File:Webfig-1.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Webfig-1.png *License*: unknown *Contributors*: Marisb
File:Webfig-submenu.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Webfig-submenu.png *License*: unknown *Contributors*: Marisb
File:webfig-enable.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Webfig-enable.png *License*: unknown *Contributors*: Marisb
File:webfig-disable.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Webfig-disable.png *License*: unknown *Contributors*: Marisb
File:webfig-remove.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Webfig-remove.png *License*: unknown *Contributors*: Marisb
File:webfig-3.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Webfig-3.png *License*: unknown *Contributors*: Marisb
File:Webfig-upload.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Webfig-upload.png *License*: unknown *Contributors*: Marisb
File:Webfig-download.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Webfig-download.png *License*: unknown *Contributors*: Marisb
File:webfig-add-to-stsatus-page.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Webfig-add-to-stsatus-page.png *License*: unknown *Contributors*: Janisk
File:webfig-two-columns.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Webfig-two-columns.png *License*: unknown *Contributors*: Janisk
File:webfig-set-field-limits-design.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Webfig-set-field-limits-design.png *License*: unknown *Contributors*: Janisk
File:webfig-set-field-limits-done.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Webfig-set-field-limits-done.png *License*: unknown *Contributors*: Janisk
Image:AP_CLIENT.png *Source*: http://wiki.mikrotik.com/index.php?title=File:AP_CLIENT.png *License*: unknown *Contributors*: SergejsB
Image:ap_client2.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ap_client2.png *License*: unknown *Contributors*: SergejsB
Image:ap_client3.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ap_client3.png *License*: unknown *Contributors*: SergejsB
Image:ap_client4.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ap_client4.png *License*: unknown *Contributors*: SergejsB
Image:ap_client5.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ap_client5.png *License*: unknown *Contributors*: SergejsB
Image:ap_client6.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Ap_client6.png *License*: unknown *Contributors*: SergejsB
Image:Debuglogs.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Debuglogs.png *License*: unknown *Contributors*: Normis
Image:Centos_partitioning.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Centos_partitioning.png *License*: unknown *Contributors*: Mplsguy
Image:Centos_packages.png *Source*: http://wiki.mikrotik.com/index.php?title=File:Centos_packages.png *License*: unknown *Contributors*: Mplsguy