RouterOS How To List

Document revision 1.5 (Sun Dec 05 10:37:12 GMT 2004)
This document applies to V2.8

Table of Contents

How to Protect your MikroTik RouterOS™?

Description

To protect your MikroTik RouterOS™, you should not only change admin's password but also set up packet filtering. All packets with destination to the router are processed against the ip firewall **input** chain. Note, that the input chain does not affect packets which are being transferred through the router.

You can add following rules under **/ip firewall rule input** (just 'copy and paste' to the router using Terminal Console or configure the relevant arguments in WinBox):

```
/ip firewall rule input add connection-state=invalid action=drop \
        comment="Drop invalid connections"
/ip firewall rule input add connection-state=established \
        comment="Allow established connections"
/ip firewall rule input add connection-state=related \
        comment="Allow related connections"
/ip firewall rule input add protocol=udp comment="Allow UDP"
/ip firewall rule input add protocol=icmp comment="Allow ICMP Ping"
/ip firewall rule input add src-address=10.0.0.0/24 \
        comment="Allow access from our local network. Edit this!"
/ip firewall rule input add src-address=192.168.0.0/24 protocol=tcp dst-port=8080 \
        comment="This is web proxy service for our customers. Edit this!"
/ip firewall rule input add action=drop log=yes \
        comment="Log and drop everything else"
```

Use **/ip firewall rule input print packets** command to see how many packets have been processed against these rules. Use **reset-counters** command to reset the counters. Examine the system log file **/log print** to see the packets which have been dropped.

You may need to include additional rules to allow access from certain hosts, etc. Remember that firewall rules are processed in the order they appear on the list. After a rule matches the packet, no more rules are processed for it. After adding new rules, move them up using the **move** command.

How to Protect your MikroTik RouterOS™ from being used as Spam Relay?

Description

To protect your MikroTik RouterOS™ from being used as spam relay you have to:

1. Protect your router using firewall rules. See the How To section about it!
2. Configure web proxy access list.

The web proxy access list is configured under **/ip web-proxy access**. For example, add following rules to it to allow access from certain hosts (just 'copy and paste' to the router using Terminal Console or configure the relevant arguments in WinBox):

```
/ip web-proxy access add src-address=192.168.0.0/24 \
        comment="Our customers"
/ip web-proxy access add dst-port=23-25 action=deny \
        comment="Deny using us as telnet and SMTP relay"
/ip web-proxy access add action=deny \
        comment="Deny everything else"
```

Note, that first you should have rules that allow certain services, and the last rule should always be one that denies access for everything else. Rules are processed in the order they appear on the list. After a rule matches the request, no more rules are processed for it. After adding new rules, move them up using the **move** command.

How to Connect your Home Network to xDSL Line?

Description

You have your Home DSL modem installed, and want to have a secure connection to the Internet for your home network. For that, you have to install MikroTik router between the DSL modem and your home network:

Follow the steps below to connect your home network to xDSL line:

1. Make your MikroTik router with two Ethernet NICs, one for the Home DSL modem, one for your home network. See instructions in the Basic Setup Guide.
2. When installing, make sure you install the dhcp software package.
3. Enable both interfaces, for example:

```
/interface enable ether1,ether2
```

4. Configure DHCP client on the external (xDSL) interface to receive IP configuration form your service provider:

```
/ip dhcp-client set enabled=yes interface=ether1
```

5. Check, if you have received IP configuration using **lease print**, for example:

```
[admin@MikroTik] ip dhcp-client> lease print
          address: 81.198.16.4/21
          expires: may/10/2001 04:41:49
          gateway: 81.198.16.1
      primary-dns: 195.13.160.52
    secondary-dns: 195.122.1.59
[admin@MikroTik] ip dhcp-client>
```

6. Add your private network address to **ether2** interface, for example:

```
/ip address add address=192.168.0.1/24 interface=ether2
```

7. Configure masquerading for your local network:

```
/ip firewall src-nat add out-interface=ether1 action=masquerade \
    comment="Masquerades everything leaving the external interface"
```

8. Configure firewall to protect your router:

```
/ip firewall rule input add connection-state=invalid action=drop \
        comment="Drop invalid connection packets"
/ip firewall rule input add connection-state=established \
        comment="Allow established connections"
/ip firewall rule input add connection-state=related \
        comment="Allow related connections"
/ip firewall rule input add protocol=udp comment="Allow UDP"
/ip firewall rule input add protocol=icmp comment="Allow ICMP Ping"
/ip firewall rule input add src-address=192.168.0.0/24 \
        comment="From my home network"
/ip firewall rule input add action=drop log=yes \
        comment="Log and drop everything else"
```

9. (Optional) Configure DHCP server to hand out IP configuration on your home network:

```
/ip pool add name=private ranges=192.168.0.2-192.168.0.254
/ip dhcp-server network add gateway=192.168.0.1 address=192.168.0.0/24 \
    dns-server=195.13.160.52,195.122.1.59 domain="mail.com"
/ip dhcp-server add name=home interface=ether2 lease-time=3h \
    address-pool=private
/ip dhcp-server enable home
```

That's it! You can access the Internet from your home network!

How To Keep My Router Up-To-Date

Description

To keep your router up to date, you should:

- Upgrade to the latest RouterOS software version
- If you have a RouterBoard, upgrade the BIOS firmware

In this How-To section we will show you how to upgrade your RouterBoard's BIOS firmware version. See also how to upgrade RouterOS version.

1. At first, check if you have a routerboard package installed:

```
[admin@MikroTik] system package> print
Flags: I - invalid
 #   NAME                VERSION             BUILD-TIME           UNINSTALL
 0   routerboard         2.8.14              aug/06/2004 15:30:32 no
 1   security            2.8.14              aug/06/2004 14:08:54 no
 2   system              2.8.14              aug/06/2004 14:03:02 no
 3   advanced-tools      2.8.14              aug/06/2004 14:04:55 no
 4   wireless            2.8.14              aug/06/2004 14:42:17 no
[admin@MikroTik] system package>
```

2. Check your RouterBoard BIOS version:

```
[admin@MikroTik] system routerboard> print
         routerboard: yes
               model: 230
       serial-number: 8387617
    current-firmware: 1.3.1 (Aug/06/2004 15:30:19)
    upgrade-firmware: 1.3.1 (Aug/06/2004 15:30:19)
[admin@MikroTik] system routerboard>
```

   Check for the newest BIOS update in the **all packages** archive available from our download page. BIOS update file is called **wlb-bios-[version_number].fwf** where **version_number** is the version of BIOS firmware.

3. If this file contains a newer version than your router has, copy it to router via FTP using the **binary** file transfer mode. When done, you should see the file, containing BIOS firmware in the **file** list:

```
[admin@MikroTik] system routerboard> /file print
 # NAME                          TYPE         SIZE       CREATION-TIME
 0 wlb-bios-1.3.2.fwf            routerbios   73079      sep/07/2004 00:12:05
[admin@MikroTik] system routerboard>
```

4. Check the RouterBoard's BIOS firmware version and you should see that you can upgrade it to a newer version (1.3.2):

```
[admin@MikroTik] system routerboard> print
         routerboard: yes
               model: 230
       serial-number: 8387617
    current-firmware: 1.3.1 (Aug/06/2004 15:30:19)
    upgrade-firmware: 1.3.2 (Aug/22/2004 12:13:56)
[admin@MikroTik] system routerboard>
```

5. Now you can upgrade the BIOS version, using the **upgrade** command:

```
[admin@MikroTik] system routerboard> upgrade
Firmware upgrade requires reboot of the router. Continue? [y/n]
```

Choose **y** and the software will upgrade BIOS, don't restart the router manually - it will be done automatically! When router has rebooted, check the new BIOS version:

```
[admin@MikroTik] system routerboard> print
         routerboard: yes
               model: 230
       serial-number: 8387617
    current-firmware: 1.3.2 (Aug/22/2004 12:13:56)
    upgrade-firmware: 1.3.2 (Aug/22/2004 12:13:56)
[admin@MikroTik] system routerboard>
```

How to Transparently Bridge two Networks?

Description

Remote networks can be easily bridged using Ethernet over IP (EoIP) or WDS feature of MikroTik RouterOS™. We will show it for the case when the networks are connected through Atheros wireless interface. Using EoIP, the can be extended to any other type of interfaces, like PPTP, CISCO/Aironet, Prism. WDS works only on Prism and Atheros based cards.

Let us assume the following network setup:

Transparent Bridge, using EoIP tunnel

Follow the steps below to create transparent bridge using EoIP interfaces:

1. Make sure you have communication between MikroTik routers, i.e., one router is configured as server (AP), the other one as client (station):

```
[admin@AP] > interface wireless set wlan1 mode=bridge ssid=mikrotik \
\... disabled=no

[admin@Station] interface wireless> print
[admin@Station] interface wireless> set wlan1 mode=station ssid=mikrotik \
\... disabled=no
```

2. Make sure the IP configuration is correct, and you can access one router from the other one:

```
[admin@AP] > ip address add address=10.1.0.1/24 interface=wlan1

[admin@Station] > ip address add address=10.1.0.2/24 interface=wlan1

[admin@Station] > ping 10.1.0.1
10.1.0.1 64 byte pong: ttl=64 time=1 ms
10.1.0.1 64 byte pong: ttl=64 time=1 ms
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 1/1.0/1 ms
[admin@Station] >
```

3. Add EoIP tunnel interfaces:

```
[admin@AP] > interface eoip add remote-address=10.1.0.2 tunnel-id=1 disabled=no

[admin@Station] > interface eoip add remote-address=10.1.0.1 tunnel-id=1 \
\... disabled=no
```

4. Add bridge interfaces and put :

```
[admin@AP] > interface bridge add forward-protocols=ip,arp,other disabled=no
[admin@AP] > interface bridge port set eoip-tunnel1,ether1 bridge=bridge1

[admin@Station] > interface bridge add forward-protocols=ip,arp,other \
\... disabled=no
[admin@Station] > interface bridge port set eoip-tunnel1,ether1 bridge=bridge1
```

Note, that connection is lost to the router if you are connected through 'ether1'.

5. Move the IP addresses from Ethernet to bridge interfaces:

```
[admin@AP] ip address> set [find interface=ether1 ] interface=bridge1
[admin@AP] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #   ADDRESS            NETWORK         BROADCAST       INTERFACE
  0   10.0.0.215/24      10.0.0.0        10.0.0.255      bridge1
  1   10.1.0.1/24        10.1.0.0        10.1.0.255      wlan1
[admin@AP] ip address>

[admin@Station] ip address> set [find interface=ether1 ] interface=bridge1
[admin@Station] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #   ADDRESS            NETWORK         BROADCAST       INTERFACE
  0   10.0.0.216/24      10.0.0.0        10.0.0.255      bridge1
  1   10.1.0.2/24        10.1.0.0        10.1.0.255      wlan1
[admin@Station] ip address>
```

Now you can connect again to the router through **ether1** which belongs to **bridge1**

Test the bridge by pinging from 10.0.0.215 to 10.0.0.216. Note, that the bridge needs 10...30s to learn addresses and start passing through traffic.

Similarly you can create transparent bridge if you have prism or CISCO/Aironet interfaces, or encrypted PPTP tunnel. However, the EoIP tunnel can be established between two MikroTik routers only.

Transparent Bridge, using WDS

You can also use WDS to bridge 2 networks transparently.

1. Configure wireless interface **wlan1** on **AP**:

```
[admin@AP]> interface wireless set wlan1 ssid=mikrotik frequency=5805 \
\... mode=bridge wds-mode=dynamic disabled=no
```

Do the same configuration on **Client** wireless interface (**wlan1**):

```
[admin@Client]> interface wireless set wlan1 ssid=mikrotik frequency=5805 \
\... mode=bridge wds-mode=dynamic disabled=no
```

2. Check whether the WDS link is established:

```
[admin@AP] interface wireless wds> print
Flags: X - disabled, R - running, D - dynamic
 0  RD name="wds1" mtu=1500 mac-address=00:0B:6B:31:02:4B arp=enabled
       disable-running-check=yes master-interface=wlan1
       wds-address=00:0B:6B:31:08:22
[admin@AP] interface wireless wds>
```

3. Create a bridge interface on **AP**, and add **wlan1** and **ether1** interfaces to the bridge. The WDS interface will be added automatically to the bridge if you specify **wds-default-bridge** parameter:

```
[admin@AP]> interface bridge add name=wds-bridge
[admin@AP]> interface bridge port set wlan1,ether1 bridge=wds-bridge
[admin@AP]> interface wireless set wlan1 wds-default-bridge=wds-bridge
```

Do the same on **Client**:

```
[admin@Client]> interface bridge add name=wds-bridge
[admin@Client]> interface bridge port set wlan1,ether1 bridge=wds-bridge
[admin@Client]> interface wireless set wlan1 wds-default-bridge=wds-bridge
```

4. Add IP address on **AP**:

```
[admin@AP]> ip address add address=10.1.0.1/24 interface=wds-bridge
```

And on **Client**:

```
[admin@Client]> ip address add address=10.1.0.2/24 interface=wds-bridge
```

5. Test the link:

```
[admin@AP]> ping 10.1.0.2
10.1.0.2 64 byte ping: ttl=64 time=1 ms
10.1.0.2 64 byte ping: ttl=64 time=1 ms
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 1/1.0/1 ms
[admin@AP]>
```

How to Link Public Addresses to the Local Ones?

Description

The current topic shows how to configure 'full NAT', i.e. when a computer having it's own address in the local network gets it translated when talking to outer (public) networks.

Let us assume two addresses (10.0.0.216 and 10.0.0.217) are assigned to the router. In this example we will 'full NAT' the internal address 192.168.0.4 to the external 10.0.0.216 one while keeping 10.0.0.217 for the router itself as well as for masquerading the internal network.

To add 10.0.0.216/24 and 10.0.0.217/24 addresses to the router's Public interface and 192.168.0.254/24 to the router's Local interface:

```
/ip address
    add address=10.0.0.216/24 interface=Public
    add address=10.0.0.217/24 interface=Public
    add address=192.168.0.254/24 interface=Local
    print
Flags: X - disabled, I - invalid, D - dynamic
  #   ADDRESS            NETWORK         BROADCAST       INTERFACE
  0   10.0.0.216/24      10.0.0.0        10.0.0.255      Public
  1   10.0.0.217/24      10.0.0.0        10.0.0.255      Public
  2   192.168.0.254/24   192.168.0.0     192.168.0.255   Local
```

While adding the default route to the router you should be aware of having two addresses. You should specify the address that the router will be using while talking to the outer networks:

```
/ip route
    add gateway=10.0.0.1 preferred-source=10.0.0.217
    print
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, r - rip, o - ospf, b - bgp
    #    DST-ADDRESS        G GATEWAY         DISTANCE INTERFACE
    0  S 0.0.0.0/0          r 10.0.0.1        1        Public
    1 DC 10.0.0.0/24        r 0.0.0.0         0        Public
    2 DC 192.168.0.0/24     r 0.0.0.0         0        Local
```

Add DST-NAT rule allowing access to the internal server from external networks:

```
/ip firewall dst-nat
```

```
    add dst-address=10.0.0.216/32 action=nat to-dst-address=192.168.0.4
    print
Flags: X - disabled, I - invalid, D - dynamic
 0   dst-address=10.0.0.216/32 action=nat to-dst-address=192.168.0.4
```

To add SRC-NAT rules allowing the internal server to talk to the outer networks having its source address translated to 10.0.0.216, while translating other internal hosts' source addresses to 10.0.0.217:

```
/ip firewall src-nat
    add src-address=192.168.0.4/32 action=nat to-src-address=10.0.0.216
    add src-address=192.168.0.0/24 action=nat to-src-address=10.0.0.217
    print
Flags: X - disabled, I - invalid, D - dynamic
 0   src-address=192.168.0.4/32 action=nat to-src-address=10.0.0.216

 1   src-address=192.168.0.0/24 action=nat to-src-address=10.0.0.217
```

How to Apply Different Treatment for Overseas Traffic

Description

You want to deny, slow down or proxy oversea traffic. To distinguish oversea traffic from the local country traffic, 'mangle mark' function can be used. It will 'mark' the packets to / from the networks that reside in your country and the oversea traffic with different marks, so that you may apply different treatment for these flows.

To prepare mangle list, you need to get a list of local networks. List of network numbers belonging to ISPs in Latvia can be extracted from file http://www.nic.lv/local.net Generate router script file (.rsc) (for example, using spreadsheet program, such as Microsoft Excel), upload it to the router via FTP and import it (with '/import' command). Here is a condensed example of a such a script for Latvian networks:

```
/ip firewall mangle
add in-interface=ether1 dst-address=159.148.0.0/16 action=passthrough \
\.. mark-connection=mark-con-latvia comment="mark all latvian traffic"
add dst-address=193.41.195.0/24 action=passthrough \
\.. mark-connection=mark-con-latvia comment="mark all latvian traffic"
add dst-address=193.41.33.0/24 action=passthrough \
\.. mark-connection=mark-con-latvia comment="mark all latvian traffic"
add dst-address=193.41.45.0/24 action=passthrough \
\.. mark-connection=mark-con-latvia comment="mark all latvian traffic"
add dst-address=193.68.64.0/19 action=passthrough \
\.. mark-connection=mark-con-latvia comment="mark all latvian traffic"
...
add connection=mark-con-latvia action=passthrough mark-flow=latvia comment="mark
latvia"
add flow=!latvia action=passthrough mark-flow=overseas comment="mark all oversea
traffic"
```

Next, you should define, what to do with the marked packets. The basic usages are:

1. Limit access to/from oversea sites from some or all hosts of your network. For example, to deny oversea traffic for 10.0.0.0/24 network:

```
/ip firewall rule forward
    add action=drop flow=overseas src-address=10.0.0.0/24
    add action=drop flow=overseas dst-address=10.0.0.0/24
```

   The same way you may deny oversea traffic for some computers (for example, to specify a particular host, you should use the host's address with the mask of 32).

2. Slow down speed of oversea connections. For example, to limit overall speed of downloading from overseas to 128Kbit/s and of uploading - to 64Kbit/s:

```
/queue tree
    add flow=overseas parent=local max-limit=131072
    add flow=overseas parent=public max-limit=65536
```

3. To use local transparent web proxy server for oversea HTTP traffic:

```
/ip web-proxy set enabled=yes transparent-proxy=yes address=:3128
/ip firewall dst-nat add flow=overseas in-interface=local protocol=tcp \
dst-port=80 action=redirect to-dst-port=3128
```

Notes

The example above assumes that your local PCs have IP addresses either from 159.148.0.0/16 network or outside any network from within Latvian IP space.

How to Build a Transparent Traffic Shaper?

Description

You want to use MikroTik RouterOS™ as a transparent Traffic Shaper on an existing Ethernet network. You can simply plug it between the network and the existing router. To achieve this, RouterOS™ should be configured as follows (this is written assuming that there is no other configuration on the shaper and there are two Ethernet cards in it):

1. Enable and name the ethernet interfaces. The interface which is connecter to the network will be called **int**, and the one connected to the existing router - **ext**:

```
/interface set ether1,ether2 disabled=no
/interface set ether1 name=int
/interface set ether2 name=ext
```

2. Then let's assume the gateway has the IP address of 10.0.0.1/24. Adding the address of, for example, **10.0.0.2/24** (you will need this address later to be able configure the shaper remotely) to this interface, you should be able to ping the gateway. If it is not happening, then you should swap cables (i.e. plug the one connected to the **ext** interface to the **int** and vice versa). **Note:** if neither works, there might be a firewall set up on the gateway, remove it temporarily and try again.

```
/ip address add interface=ext address=10.0.0.2/24
/ping 10.0.0.1
```

3. Create a bridge interface which bridges both **int** and **ext** physical interfaces:

```
/interface bridge add name=bridge
/interface bridge port set ext,int bridge=bridge
```

Note the IP address should be transferred to the **bridge** interface:

```
/ip address set [/ip address find] interface=bridge
```

Now you can simply add the desirable queues. **Note** that you may use the names of the real interfaces for these queues. For example, to limit all download to 256Kbit/s and all upload to 128Kbit/s, it is enough to add two queues:

```
/queue simple add limit-at=131072 interface=ext
/queue simple add limit-at=262144 interface=int
```

For more information and examples of how to limit data rate and provide quality of service, see **Bandwidth Control** Manual.

How to Configure Router to Send E-mails When Power Failure has Occured?

Description

Suppose you want to receive e-mail (most GSM operators provide service to forward e-mail as SMS) from your router every time power has failed and UPS is running on bateries.

To do so you should set up Scheduler to check the state of UPS and write a script to send e-mail every time UPS switches to **on-battery** mode. As you do not want e-mail to come every time the router checks the state of the UPS, this script should be run only once. To achieve this, two scripts should be created. One of them should work while the UPS is on utility power, and send e-mail, should it fail. It should also change the Scheduler task so that the other one is executed next time the task runs. The second script should run while the UPS is on battery, and change the Scheduler task back to execute the first script, once the UPS has switched back to utility power.

First of all scripts should be written. The first script will send an e-mail to the mail@company.com mailbox via the 11.22.33.44 server (you should change these values appropriately):

```
/system script add name=wait-on-battery source={
 /system ups monitor once do {
  :if ($on-battery) do {
   /system scheduler set ups-monitor script=wait-off-battery
   /tool e-mail send subject="on-battery" to="mail@company.com" \
   from="router@company.com" server=11.22.33.44
  }
 }
}
/system script add name=wait-off-battery source={
 /system ups monitor once do {
  :if (!$on-battery) do {
   /system scheduler set ups-monitor script=wait-on-battery
  }
 }
}
```

And now the initial Scheduler task should be created. It will run every 5 seconds:

```
/system scheduler add name="ups-monitor" interval=5s \
 script="wait-on-battery"
```

How to bind HotSpot usernames with their IP addresses?

Description

If dhcp-pool login method is used, it is possible to specify, what address a user will get after he/she logs in using **address** property in **/ip hotspot user** submenu. With enabled-address login method this property is ignored, so there is no direct method of doing this.

But it is possible to bind HotSpot usernames with both IP and MAC addresses. To do this, the **arp** mode of the interface clients are connected to should be switched to **reply-only**. In order not to disconnect all the clients, you should first convert all the dynamic entries in /**ip arp** table to static ones using the following command:

```
:foreach i in=[/ip arp find dynamic=yes ] do={/ip arp add copy-from=$i}
```

Now only the clients who have their IP and MAC addresses added to the /**ip arp** table manually will be able to use Internet. You can add HotSpot users specifying their MAC addresses, and they will not be able to connect from different IP addresses except those added to the ARP table.

How to use Peer-to-Peer filtering?

Description

This chapter shows some examples on how to use Peer-to-Peer traffic matching feature introducted in RouterOS™ version 2.8.

Logging

To log all P2P traffic the following rule should be added:

If the firewall logging is enabled in the router then in the log file you will see P2P packet information like this:

Drop

To drop all P2P traffic the following rule should be added:

You can enable the logging for the dropped packets by adding the **log=yes** to the previous command. Then in the log file you will see such similar entries:

If you want to allow some of your users to use P2P then you need to add 2 (one for download, one for upload) accept rules before the drop rule:

One Way P2P

In case of DC++ you can't just add dst-address of the user in the forward chain and then drop all other P2P traffic - DC++ send out some P2P info to the other P2P user, from which you are downloading. If the upload P2P traffic is blocked then you will not be able to download too. To make one way P2P you should decrease the speed of the other way to a small speed limit, for example, P2P upload traffic limit to 10000bps (10Kbps). Then users will be able to download the P2P traffic, but their upload traffic will be maximum 10Kbps.

To do that, mark all P2P traffic using Firewall mangle:

And then add queues to limit upload traffic to 10Kbps:

Individual IP P2P limit

This section will help you to make P2P limitation to individual IPs and with different speed limit for each IP. Suppose we have 2 clients and we would like to limit one client s P2P traffic to 256Kbps(download)/64Kbps(upload) and the other client s P2P traffic to 384Kbps(download)/128Kbps(upload). First client s IP address is 10.1.5.49 and the second client s IP is 10.1.5.50. To do this, mark all P2P traffic using Firewall mangle:

Then mark P2P traffic of the first client (upload/download):

Next, mark P2P traffic of the second client (upload/download):

Add queue rules for the first client (upload/download):

And finally, add queue rules for the second client (upload/download):

If we have masquerade enabled then we can't limit the download stream. Mangle is the first firewall module that gets packets, when they are received. Next DST-NAT is done, which not only execute DST-NAT rules, but also performs un-SRC-NATting. That is why mangle do not 'see' the real addresses of the clients. As SRC-NAT is not allowing to establish connections to the NATted clients, it is possible to match all responses in already existing connections established by the clients using connection marks. To do this, first of all, connection-mark all packets from the IP of each client with different marks for each client using **action=passthrough**:

Then we can remark these connections with a different flow mark and also mark the p2p traffic:

Finally, add a queue rule:

Burst

We have already configured mangle rules and queues for download:

We want to allow bursting up to 400000bps for 5min (in case when the download speed is maximum all the time), after 5min, the speed limit will be back to 256000. To do that we need to modify the queue rule:

We specified burst time 600 seconds (10min). This time is needed for the calculation of the specific moment when the router will drop the queue speed limit from **burst-time** to **max-limit**. Router is calculating the average value: sum of the speed in each second in the **burst-time**, divided with the **burst-time**. Now there are two cases:

1. If this value is lower that the **burst-threshold** then the queue speed limit will be raised to the **burst-limit**
2. If this value is higher that the **burst-threshold** then the queue speed limit will be dropped down to **max-limit**

In our case the user is downloading at the maximum speed. This means he could download at the **burst-limit** speed 5min - average value is still equal to **burst-threshold** which is 200000 (400000*300/600=200000). In the next second the speed limit will be greater than the **burst-threshold** and the speed limit will be dropped to the **max-limit**.

Using PCQ

Suppose we have a network and we want to limit Peer-to-Peer traffic for each client in this network to

64Kbit/s upload and 128Kbit/s for download. This queue type is called PCQ. You can also use it in the previous examples instead of the default queue type.

First of all, create a PCQ queues - one for upload (this should classify by **src-address**), and one for download (this should classify by **dst-address**):

Then we should 'catch' the P2P traffic using mangle rule:

Now we can create queues:

How to make a Wireless Bridge, using WDS?

Description

Let us consider that we have a **MainAP** wireless Access Point which serves several wireless clients (from network **192.168.0.0/24**) and is connected to **ClientAP** which also serves clients from the same network. This example illustrates how to use WDS between **MainAP** and **ClientAP** so that they also act as APs for wireless clients.

At first, let us setup the wireless interface for **MainAP**:

```
/interface wireless set wlan1 ssid=ap2ap band=2.4GHz-B \
    frequency=2442 mode=ap-bridge wds-mode=static disabled=no
```

Now add a WDS interface and note that the **wds-address** is MAC address from remote AP (in this case ClientAP):

```
/interface wireless wds add master-interface=wlan1 \
    wds-address=00:02:6F:01:CE:31 disabled=no
```

Put the **wds1** and **wlan1** interfaces into a bridge:

```
/interface bridge add disabled=no
/interface bridge port set 1,2 bridge=bridge1
[admin@MikroTik] interface bridge port> print
 # INTERFACE BRIDGE PRIORITY PATH-COST
 0 ether1    none    128      10
 1 wlan1     none    128      10
 2 wds1      none    128      10
[admin@MikroTik] interface bridge port>
```

Finally, add an IP address to the **bridge1** interface:

```
/ip address add address=192.168.0.254/24 interface=bridge1
[admin@MikroTik] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         BROADCAST       INTERFACE
 0   10.0.0.2/24        10.0.0.0        10.0.0.255      ether1
 1   192.168.0.254/24   192.168.0.0     192.168.0.255   bridge1
[admin@MikroTik] ip address>
```

Now let us configure **ClientAP**. The configuration is similar to **MainAP** configuration. At first, configure the wlan1 interface:

```
/interface wireless set wlan1 ssid=ap2ap frequency=2442 \
    band=2.4GHz-B mode=ap-bridge wds-mode=static disabled=no
```

Then add a WDS interface:

```
/interface wireless wds add master-interface=wlan1 \
    wds-address=00:0B:6B:31:01:6A disabled=no
```

Add the **wds1** and **wlan1** interfaces to a bridge:

```
/interface bridge add disabled=no
/interface bridge port set wlan1,wds1 bridge=bridge1
```

If you want, you can assign an IP address to the bridge interface, but there is no need for it (as this AP works transparently). Now you are able to ping the clients from **MainAP**:

```
/ping 192.168.0.10
192.168.0.10 64 byte ping: ttl=64 time=8 ms
192.168.0.10 64 byte ping: ttl=64 time=7 ms
192.168.0.10 64 byte ping: ttl=64 time=7 ms
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 7/7.3/8 ms
```

How to limit p2p traffic, using masquerading and PCQ

Example

Let us consider a situation where the limited network is 192.168.0.0/24. We will limit the p2p download traffic to 256kbit/s and upload to 128kbit/s

The 192.168.0.0/24 network has to be masquaraded in order to get public access (it will use the address 10.0.0.217). To do so, we will masquerade this network.

```
[admin@MikroTik] ip firewall src-nat> add src-address=192.168.0.0/24 \
\... action=masquerade
[admin@MikroTik] ip firewall src-nat> print
Flags: X - disabled, I - invalid, D - dynamic
 0   src-address=192.168.0.0/24 action=masquerade
[admin@MikroTik] ip firewall src-nat>
```

Then we have to mark download and upload traffic. To do so with masqueraded traffic, let's add 2 mangle rules - the first one stands for marking the p2p connection with the mark **p2p_con** which is comming from the local network (**192.168.0.0/24**) , the second one will mark all packets whithin this connection with mark **p2p_limit**, which will be used for limiting the upload and download traffic.

```
[admin@MikroTik] ip firewall mangle> add src-address=192.168.0.0/24 p2p=all-p2p \
\... mark-connection=p2p_con action=passthrough
[admin@MikroTik] ip firewall mangle> add connection=p2p_con action=accept
mark-flow=p2p_limit
[admin@MikroTik] ip firewall mangle>
```

Next, we will make two PCQ types - one for download (pcq-download), and one for upload (pcq-upload).

```
[admin@MikroTik] queue type> add kind=pcq name=pcq-download \
\... pcq-rate=256000 pcq-classifier=dst-address
[admin@MikroTik] queue type> add kind=pcq name=pcq-upload \
\... pcq-rate=128000 pcq-classifier=src-address
[admin@MikroTik] queue type> print
 0 name="default" kind=pfifo bfifo-limit=15000 pfifo-limit=50 red-limit=60
   red-min-threshold=10 red-max-threshold=50 red-burst=20 sfq-perturb=5
   sfq-allot=1514 pcq-rate=0 pcq-limit=50 pcq-classifier

 1 name="ethernet-default" kind=pfifo bfifo-limit=15000 pfifo-limit=50
   red-limit=60 red-min-threshold=10 red-max-threshold=50 red-burst=20
   sfq-perturb=5 sfq-allot=1514 pcq-rate=0 pcq-limit=50 pcq-classifier

 2 name="wireless-default" kind=sfq bfifo-limit=15000 pfifo-limit=50
```

```
   red-limit=60 red-min-threshold=10 red-max-threshold=50 red-burst=20
   sfq-perturb=5 sfq-allot=1514 pcq-rate=0 pcq-limit=50 pcq-classifier

 3 name="synchronous-default" kind=red bfifo-limit=15000 pfifo-limit=50
   red-limit=60 red-min-threshold=10 red-max-threshold=50 red-burst=20
   sfq-perturb=5 sfq-allot=1514 pcq-rate=0 pcq-limit=50 pcq-classifier

 4 name="pcq-download" kind=pcq bfifo-limit=15000 pfifo-limit=50 red-limit=60
   red-min-threshold=10 red-max-threshold=50 red-burst=20 sfq-perturb=5
   sfq-allot=1514 pcq-rate=256000 pcq-limit=50 pcq-classifier=dst-address

 5 name="pcq-upload" kind=pcq bfifo-limit=15000 pfifo-limit=50 red-limit=60
   red-min-threshold=10 red-max-threshold=50 red-burst=20 sfq-perturb=5
   sfq-allot=1514 pcq-rate=128000 pcq-limit=50 pcq-classifier=src-address
[admin@MikroTik] queue type>
```

And finally, add the queue rules.

```
[admin@MikroTik] queue tree> add name=down parent=Local \
\... flow=p2p_limit queue=pcq-download
[admin@MikroTik] queue tree> add name=up parent=Public \
\... flow=p2p_limit queue=pcq-upload
[admin@MikroTik] queue tree> print
Flags: X - disabled, I - invalid, D - dynamic
 0    name="down" parent=Local flow=p2p_limit limit-at=0
      queue=pcq-download priority=8 max-limit=0 burst-limit=0
      burst-threshold=0 burst-time=0

 1    name="up" parent=Public flow=p2p_limit limit-at=0 queue=pcq-upload
      priority=8 max-limit=0 burst-limit=0 burst-threshold=0 burst-time=0
[admin@MikroTik] queue tree>
```

How to guarantee and prioritize traffic?

Description

Queue trees can be used for more sophisticated applications where you need to limit traffic for specific users, protocols, ports etc.

In this example we will show you:

- how to guarantee bandwidth to certain services and use the 'idle' bandwidth
- how to prioritize a service (**POP3**) among others (**HTTP** and **FTP**)

You can see how we will share the traffic in the picture (the network **192.168.0.0/24** is masqueraded):

1. At first, mangle the **HTTP**, **FTP** and **POP3** download traffic. As our network **192.168.0.0/24** is masqueraded, we need to mark the outgoing connection with **mark-connection** parameter:

```
/ip firewall mangle
add in-interface=Local dst-address=:80 protocol=tcp action=passthrough \
    mark-connection=http-con comment="" disabled=no
add in-interface=Local dst-address=:110 protocol=tcp action=passthrough \
    mark-connection=pop3-con comment="" disabled=no
add in-interface=Local dst-address=:21 protocol=tcp action=passthrough \
    mark-connection=ftp-con comment="" disabled=no
```

   and only then we can mark the downstream traffic with a flow mark:

```
/ip firewall mangle
add protocol=tcp connection=http-con action=passthrough mark-flow=HTTP \
    comment="" disabled=no
add protocol=tcp connection=pop3-con action=passthrough mark-flow=POP3 \
    comment="" disabled=no
```

```
add protocol=tcp connection=ftp-con action=passthrough mark-flow=FTP \
    comment="" disabled=no
```

2. When we have marked the packets with a **flow-mark**, we can use them to build a queue tree.
   Add a queue that will guarantee 80% of all available (512kbps) bandwidth, which is 409,6kbps, for HTTP service and if there is more bandwidth available (some services are idle), use it:

```
/queue tree
add name="http-queue" parent=Local flow=HTTP limit-at=409600 max-limit=512000
```

   Add a queue that will guarantee 15% (76,8kbps) for FTP:

```
/queue tree
add name="ftp-queue" parent=Local flow=FTP limit-at=76800 max-limit=512000
```

   Now add a queue that will guarantee 5% (25,6kbps) of all available bandwidth to **POP3** service. Set the priority for this service to **7**. It means that this queue will have a higher priority than the previous ones (by default the priority is **8**) so it will be processed before **http-queue** and **ftp-queue**:

```
/queue tree
add name="pop3-queue" parent=Local flow=POP3 limit-at=25600 max-limit=512000 \
priority=7
```

   The benefit from higher priority is that **POP3** traffic, as it will be processed first, will have the smallest delay going through the router from all services.
   Using **limit-at** and **max-limit** parameters, you can control the minimum guaranteed and maximum allowed bandwidth for a service. At first, the **limit-at** data rate is achieved, then, if more bandwidth is available, it is used by this service (up to 512kbps in this example).

Note: for a correct queue tree setup the amount of **limit-at** values for queue tree leaves (queues which have no child-queues) must be equal (or lower) to available bandwidth. In this case 25,6kbps + 76,8kbps + 409,6kbps = 512kbps.