MikroTik RouterOS Workshop

Let's talk about QoS

Las Vegas

MUM USA 2011
About Me

Jānis Meģis, MikroTik

Jānis (Technical, Trainer, NOT Sales)
- Support & Training Engineer for almost 7 years
- Specialization: QoS, PPP, Firewall, Routing
- Teaching MikroTik RouterOS classes since 2005
Workshop Plan

- Packet Flow Diagram
- HTB
- Queue Types (PCQ, multi-queue-fifo)
- Burst
- Queue Size
- Queue tree and Simple queues
Packet Flow Diagram
Packet Flow Diagram

Packet flow diagram is “The Big Picture” of RouterOS.

It is impossible to properly manage and maintain complex configurations without the knowledge - what happens when and why?

Packet flow Diagram consist of 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
Bridging or Layer-2 (MAC)
Routing or Layer-3 (IP)
Diagram Abbreviations
Simple Routing
Bridging with IP firewall
Bridge-to-Bridge Routing (part1)
Bridge-to-Bridge Routing (part 2)
IPSec Encryption
IPSec Decryption
Hierarchical Token Bucket
Lets Simplify!

QoS Packet Flow Diagram
Global-Out or Interface HTB?

There are two fundamental differences

- In case of SRC-NAT (masquerade) Global-Out will be aware of private client addresses, but Interface HTB will not – Interface HTB is after SRC-NAT

- Each Interface HTB only receives traffic that will be leaving through a particular interface – there is no need for to separate upload and download in mangle
Mangle

The mangle facility allows you to mark IP packets with special marks.

These marks are used by other router facilities like routing and bandwidth management to identify the packets.

Additionally, the mangle facility is used to modify some fields in the IP header, like TOS (DSCP) and TTL fields.
Hierarchical Token Bucket

All bandwidth management implementation in RouterOS is based on Hierarchical Token Bucket (HTB)

HTB allows you to create hierarchical queue structure and determine relations between queues

RouterOS supports 3 virtual HTBs (global-in, global-total, global-out) and one more just before every output interface
HTB (cont.)

- When packet travels **through** the router, it passes all 4 HTB trees.
- When packet travels **to** the router, it passes only global-in and global-total HTB.
- When packet travels **from** the router, it passes global-out, global-total and interface HTB.
HTB Features - Structure

As soon as queue have at least one child it become parent queue

All child queues (don't matter how many levels of parents they have) are on the same bottom level of HTB

Child queues make actual traffic consumption, parent queues are responsible only for traffic distribution

Child queues will get limit-at first and then rest of the traffic will distributed by parents
HTB Features – Dual Limitation

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

At first HTB will try to satisfy every child queue's limit-at – only then it will try to reach max-limit
Dual Limitation

Maximal rate of the parent should be equal or bigger than sum of committed rates of the children

\[ \text{MIR (parent)} \geq \text{CIR(child1)} + \ldots + \text{CIR(childN)} \]

Maximal rate of any child should be less or equal to maximal rate of the parent

\[ \text{MIR (parent)} \geq \text{MIR(child1)} \]
\[ \text{MIR (parent)} \geq \text{MIR(child2)} \]
\[ \text{MIR (parent)} \geq \text{MIR(childN)} \]
HTB - limit-at

Max-limit=10M

"Queue01"
parent=Local-interface

Limit-at=4M
Max-limit=10M

"Queue02"
parent=Queue01

Limit-at=6M
Max-limit=10M
Priority=1

"Queue03"
parent=Queue01

Limit-at=2M
Max-limit=10M
Priority=3

"Queue04"
parent=Queue02

Limit-at=2M
Max-limit=10M
Priority=5

"Queue05"
parent=Queue02
HTB Features - Priority

- Work only for child queues to arrange them
- 8 is the lowest priority, 1 is the highest
- Queue with higher priority will get chance to satisfy its max-limit before other queues
- Actual traffic prioritization will work only if limits are specified. Queue without limits will not prioritize anything
QoS Myth buster

- HTB priority doesn't rearrange packet sequence – it doesn't put some packets before others
  - In HTB “Priority” is an option that helps to decide what packets will pass and what packets will be dropped
  - This drop decision is based on limitations, so if there are no limits there are no need to drop anything, so priority have no effect
  - Priority doesn't affect CIR traffic – it just passes through QoS (even if parent's don't have such amount of traffic)
HTB – limit-at of the Parent

Max-limit=10M

"Queue01"
parent=Local-interface

Limit-at=8M
Max-limit=10M

"Queue02"
pARENT=Queue01

Limit-at=2M
Max-limit=10M
Priority=1

"Queue03"
pARENT=Queue01

Limit-at=2M
Max-limit=10M
Priority=3

"Queue04"
pARENT=Queue02

Limit-at=2M
Max-limit=10M
Priority=5

"Queue05"
pARENT=Queue02
HTB – limit-at > parent's max-limit

Max-limit=10M

"Queue01"
parent=Local-interface

Limit-at=4M
Max-limit=10M

"Queue02"
parent=Queue01

Limit-at=6M
Max-limit=10M
Priority=1

"Queue03"
parent=Queue01

Limit-at=2M
Max-limit=10M
Priority=3

"Queue04"
parent=Queue02

Limit-at=12M
Max-limit=15M
Priority=5

"Queue05"
parent=Queue02
QoS Myth Buster

QoS can't control the amount of received traffic that you see on your interfaces.

- In Packet Flow diagram global-in is way after Input interface where statistic is registered
- Effect of traffic slowing down most probably is effect of TCP protocol behaviour
- If clients PC was able to send out traffic it have to arrive somewhere it can't just disappear

Only way to see QoS in action is to monitor TX of opposite interface.
QoS Myth Buster

- QoS doesn't know how much actual bandwidth is available

  - In Packet Flow diagram all HTB are before output interface and output interfaces driver is the first one that **might** know how much actual bandwidth you have.

  - Interface driver knows the maximal hardware limitation of your interface, IF actual limitation is smaller, the only way to provide QoS with limitation information is to specify all limits yourself
Queue Types
Default Queue Types
FIFO

Behaviour:
What comes in first is handled first, what comes in next waits until the first is finished. Number of waiting units (Packets or Bytes) is limited by “queue size” option. If queue “is full” next units are dropped.
BFIFO

IN

Check Queue Size

Queue Size < BFIFO Limit

Enqueue Byte

Queue Size ≥ BFIFO Limit

Drop Byte

OUT
MQ PFIFO

- Multi queue packet FIFO queue was designed for multi-core router solutions (RB1100AHx2).
- MQ PFIFO should be used as default interface queue for any Ethernets that have several RX/TX queues (you can check that in /system resources IRQ menu).
- MQ FIFO is alternative to RPS (receive Packet Steering) – so do not use both on same interface it will result in performance loss.
RED

Behaviour:
Same as FIFO with an additional feature – additional drop probability even if queue is not full.

This probability is based on comparison of average queue length over some period of time to minimal and maximal threshold – closer to maximal threshold the bigger the chance of a drop.
The RED algorithm involves the following steps:

1. Compute the average queue length.
2. Check if the average (Avr) is less than the minimum threshold (Min Thres).
3. If Avr < Min Thres, enqueue the packet.
4. Check if the minimum threshold (Min Thres) is less than the average (Avr) and if Avr is less than the minimum threshold (Min Thres).
5. If Min Thres < Avr & Avr < Min Thres, calculate the drop probability.
6. If the drop probability is high, drop the packet.
7. If the drop probability is low, include the packet and proceed.
8. Check if the average (Avr) is greater than the maximum threshold (Max Thres).
9. If Avr > Max Thres, drop the packet.

The process repeats for each incoming packet.
SFQ

Behaviour:
Based on a hash value from the source and destination address SFQ divides the traffic into 1024 sub-streams
Then the Round Robin algorithm will distribute an equal amount of traffic to each sub-stream
SFQ Example

- SFQ should be used for equalizing similar connections
- Usually used to manage information flow to or from the servers, so it can offer services to every customer
- Ideal for p2p limitation, it is possible to place strict limitation without dropping connections,
PCQ

PCQ was introduced to optimize massive QoS systems, where most of the queues are exactly the same for different sub-streams.

Starting from version 5.0rc5 PCQ have burst support and IPv6 support.
PCQ Classification (1)
PCQ Classification (2)
PCQ Rate (1)

pcq-rate=128000

2 users

queue=pcq-down
max-limit=512k

4 users

7 users

- 128k
- 128k
- 128k
- 73k
- 73k
- 73k
- 73k
- 73k
- 73k
- 73k
- 73k
PCQ Rate (2)
Burst
QoS Feature “Burst”

- Burst is one of the best ways to increase HTTP performance.
- Bursts are used to allow higher data rates for a short period of time.
- If an average data rate is less than burst-threshold, burst could be used (actual data rate can reach burst-limit).
- Average data rate is calculated from the last burst-time seconds.
Burst - Average Data Rate

Average data rate is calculated as follows:

- **burst-time** is being divided into 16 periods
- router calculates the **average data rate** of each class over these small periods

Note, that the **actual burst period** is not equal to the burst-time. It can be several times shorter than the burst-time depending on the max-limit, burst-limit, burst-threshold, and actual data rate history (see the graph example on the next slide)
Burst
Burst (Part 2)
Queue Size
Queue Size

Queue size has a direct impact on the performance of the queue – it is a choice between packet loss and higher latency.

In RouterOS queue sizes are common between the queue types.

To understand Queue size's impact on the traffic we will look at simplified example.

- We will ignore packet retransmits.
- We will assume that process that run continuously can be divided into steps.
There are 25 steps and there are total of 1610 incoming packets over this time frame.
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.
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)
There were 320 (19.9%) packets dropped and 80 (5.0%) packets had 1 step delay.
There were 190 (11.8%) packets dropped and 400 (24.8%) packets had 1 step delay.
Simple Queues

Simple queues are ordered - similar to firewall rules

In order to get to 999\textsuperscript{th} queue packet will have to be checked for match to all 998 previous queues

Each simple queue \textbf{might} stand for 3 separate queues:

- One in Global-in ("direct" part)
- One in Global-out ("reverse" part)
- One in Global-total ("total" part)
Simple Queues and Mangle

INPUT INTERFACE

Mangle Prerouting

Global-In
Global-Total

Mangle Forward

Router

Mangle Input

Mangle Output

Mangle Postrouting

OUTPUT INTERFACE

Out-Interface
HTB

Global-Out
Global-Total

© MikroTik 2011
Queue Tree

Tree queue is one directional only and can be placed in any of the available HTBs

Queue Tree queues don't have any order – all traffic is processed simultaneously

All child queues must have packet marks from “/ip firewall mangle” facility assigned to them

If placed in the same HTB, Simple queue will take all the traffic away from the Queue Tree queue
Queue Tree – Winbox View

<table>
<thead>
<tr>
<th>Name</th>
<th>Parent</th>
<th>Packet Mark</th>
<th>Limit At</th>
<th>Max Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total_download</td>
<td>local_ether1</td>
<td>basic_client_traffic</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>basic_client_download</td>
<td>Total_download</td>
<td>business_client_traffic</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>business_client_download</td>
<td>Total_download</td>
<td>business_client_traffic</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>standard_client_download</td>
<td>Total_download</td>
<td>standard_client_traffic</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total_upload</td>
<td>public_ether3</td>
<td>basic_client_traffic</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>basic_client_upload</td>
<td>Total_upload</td>
<td>basic_client_traffic</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>business_client_upload</td>
<td>Total_upload</td>
<td>business_client_traffic</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>standard_client_upload</td>
<td>Total_upload</td>
<td>standard_client_traffic</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

0 B queued 0 packets queued
Simple Queue and Queue Tree (Vegas Style) demonstration
Good luck!