QoS with Switch Chip

Introduction

Queues in RouterOS are processed using CPU resources, so limiting traffic with queues on devices with relatively weak CPUs is not an effective configuration. In other words, switch-based units will be overloaded very fast, because they are meant to process layer 2 traffic by using a switch-chip, not CPU. To avoid such inefficiency, RouterOS allows limiting traffic using switch chips.

CRS3xx, CRS5xx series, and CCR2116, CCR2216 devices

> This paragraph applies to CCR2116, CCR2216 devices, and CRS3xx, CRS5xx series switches. It doesn't apply to CRS1xx/CRS2xx series switches.

For CRS3xx series switches, it is possible to limit ingress traffic that matches certain parameters with ACL rules and it is possible to limit ingress/egress traffic per port basis. The policer is used for ingress traffic, the shaper is used for egress traffic. The ingress policer controls the received traffic with packet drops. Everything that exceeds the defined limit will get dropped. This can affect the TCP congestion control mechanism on end hosts and the achieved bandwidth can be actually less than defined. The egress shaper tries to queue packets that exceed the limit instead of dropping them. Eventually, it will also drop packets when the output queue gets full, however, it should allow utilizing the defined throughput better.

Port-based traffic police (ingress) and shaper (egress):

```
/interface ethernet switch port
set ether1 ingress-rate=10M egress-rate=5M
```

MAC-based traffic policer:

```
/interface ethernet switch rule
add ports=ether1 switch=switch1 src-mac-address=64:D1:54:D9:27:E6/FF:FF:FF:FF:FF:FF rate=10M
```

VLAN-based traffic policer:

```
/interface bridge
set bridge1 vlan-filtering=yes
/interface ethernet switch rule
add ports=ether1 switch=switch1 vlan-id=11 rate=10M
```

Protocol-based traffic policer:

```
/interface ethernet switch rule
add ports=ether1 switch=switch1 mac-protocol=ipx rate=10M
```

CRS1xx/CRS2xxSeries devices

> This subsection does not apply to CRS3xx series devices!

Configuration schemes

MAC based traffic scheduling and shaping: [MAC address in UFDB] -> [QoS Group] -> [Priority] -> [Queue] -> [Shaper]

VLAN based traffic scheduling and shaping: [VLAN id in VLAN table] -> [QoS Group] -> [Priority] -> [Queue] -> [Shaper]

Protocol based traffic scheduling and shaping: [Protocol in Protocol VLAN table] -> [QoS Group] -> [Priority] -> [Queue] -> [Shaper]

PCP/DEI based traffic scheduling and shaping: [Switch port PCP/DEI mapping] -> [Priority] -> [Queue] -> [Shaper]

DSCP based traffic scheduling and shaping: [QoS DSCP mapping] -> [Priority] -> [Queue] -> [Shaper]

MAC based traffic scheduling using internal Priority

In Strict Priority scheduling mode, the highest priority queue is served first. The queue number represents the priority and the queue with the highest queue number has the highest priority. Traffic is transmitted from the highest priority queue until the queue is empty, and then moves to the next highest priority queue, and so on. If no congestion is present on the egress port, the packet is transmitted as soon as it is received. If congestion occurs at the port where high-priority traffic keeps coming, the lower-priority queues starve.

On all CRS switches the scheme where MAC-based egress traffic scheduling is done according to internal Priority would be the following: [MAC address] -> [QoS Group] -> [Priority] -> [Queue];
In this example, host1 (E7:16:34:00:00:01) and host2 (E7:16:34:00:00:02) will have higher priority 1 and the rest of the hosts will have lower priority 0 for transmitted traffic on port ether7. Note that CRS has a maximum of 8 queues per port.

```
/interface bridge
add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether6 hw=yes
add bridge=bridge1 interface=ether7 hw=yes
add bridge=bridge1 interface=ether8 hw=yes
```

Create a QoS group for use in UFDB:

```
/interface ethernet switch qos-group
add name=group1 priority=1
```

Add UFDB entries to match specific MACs on ether7 and apply QoS group1:

```
/interface ethernet switch unicast-fdb
add mac-address=E7:16:34:00:00:01 port=ether7 qos-group=group1 svl=yes
add mac-address=E7:16:34:00:00:02 port=ether7 qos-group=group1 svl=yes
```

Configure ether7 port queues to work according to Strict Priority and QoS scheme only for destination address:

```
/interface ethernet switch port
set ether7 per-queue-scheduling="strict-priority:0,strict-priority:0,strict-priority:0,strict-priority:0,strict-
prior\
    ity:0,strict-priority:0,strict-priority:0,strict-priority:0" priority-to-queue=0:0,1:1 \
    qos-scheme-precedence=da-based
```

MAC based traffic shaping using internal Priority

The scheme where MAC based traffic shaping is done according to internal Priority would be following: [MAC address] -> [QoS Group] -> [Priority] -> [Queue] -> [Shaper];
In this example, unlimited traffic will have priority 0 and limited traffic will have priority 1 with a bandwidth limit of 10Mbit. Note that CRS has a maximum of 8 queues per port.

Create a group of ports for switching:

```
/interface bridge
add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether6 hw=yes
add bridge=bridge1 interface=ether7 hw=yes
add bridge=bridge1 interface=ether8 hw=yes
```

Create a QoS group for use in UFDB:

```
/interface ethernet switch qos-group
add name=group1 priority=1
```

Add UFDB entry to match specific MAC on ether8 and apply QoS group1:

```
/interface ethernet switch unicast-fdb
add mac-address=E7:16:34:A1:CD:18 port=ether8 qos-group=group1 svl=yes
```

Configure ether8 port queues to work according to Strict Priority and QoS scheme only for destination address:

```
/interface ethernet switch port
set ether8 per-queue-scheduling="strict-priority:0,strict-priority:0,strict-priority:0,strict-priority:0,strict-
prior\
    ity:0,strict-priority:0,strict-priority:0,strict-priority:0" priority-to-queue=0:0,1:1 \
    qos-scheme-precedence=da-based
```

Apply bandwidth limit for queue1 on ether8:

```
/interface ethernet switch shaper
add port=ether8 rate=10M target=queue1
```

If the CRS switch supports Access Control List, this configuration is simpler:

```
/interface ethernet switch acl policer
add name=policer1 yellow-burst=100k yellow-rate=10M

/interface ethernet switch acl
add mac-dst-address=E7:16:34:A1:CD:18 policer=policer1
```

VLAN-based traffic scheduling + shaping using internal Priorities

A best practice is to assign lower internal QoS Priority for traffic limited by shaper to make it also less important in the Strict Priority scheduler. (higher priority should be more important and unlimited)

In this example:
Switch port ether6 is using a shaper to limit the traffic that comes from ether7 and ether8.
When a link has reached its capacity, the traffic with the highest priority will be sent out first.
VLAN10 -> QoS group0 = lowest priority
VLAN20 -> QoS group1 = normal priority
VLAN30 -> QoS group2 = highest priority

```
/interface bridge
add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether6 hw=yes
add bridge=bridge1 interface=ether7 hw=yes
add bridge=bridge1 interface=ether8 hw=yes
```

Create QoS groups for use in the VLAN table:

```
/interface ethernet switch qos-group
add name=group0 priority=0
add name=group1 priority=1
add name=group2 priority=2
```

Add VLAN entries to apply QoS groups for certain VLANs:

```
/interface ethernet switch vlan
add ports=ether6,ether7,ether8 qos-group=group0 vlan-id=10
add ports=ether6,ether7,ether8 qos-group=group1 vlan-id=20
add ports=ether6,ether7,ether8 qos-group=group2 vlan-id=30
```

Configure ether6, ether7, ether8 port queues to work according to Strict Priority and QoS scheme only for VLAN-based QoS:

```
/interface ethernet switch port
set ether6 per-queue-scheduling="strict-priority:0,strict-priority:0,strict-priority:0,strict-priority:0,strict-
prior\
    ity:0,strict-priority:0,strict-priority:0,strict-priority:0" priority-to-queue=0:0,1:1,2:2 \
    qos-scheme-precedence=vlan-based
set ether7 per-queue-scheduling="strict-priority:0,strict-priority:0,strict-priority:0,strict-priority:0,strict-
prior\
    ity:0,strict-priority:0,strict-priority:0,strict-priority:0" priority-to-queue=0:0,1:1,2:2 \
    qos-scheme-precedence=vlan-based
set ether8 per-queue-scheduling="strict-priority:0,strict-priority:0,strict-priority:0,strict-priority:0,strict-
prior\
    ity:0,strict-priority:0,strict-priority:0,strict-priority:0" priority-to-queue=0:0,1:1,2:2 \
    qos-scheme-precedence=vlan-based
```

Apply bandwidth limit on ether6:

```
/interface ethernet switch shaper
add port=ether6 rate=10M
```

PCP based traffic scheduling

By default, CRS1xx/CRS2xx series devices will ignore the PCP/CoS/802.1p value and forward packets based on FIFO (First-In-First-Out) manner. When the device's internal queue is not full, then packets are in a FIFO manner, but as soon as a queue is filled, then higher-priority traffic will be sent out first. Let's consider a scenario when **ether1** and **ether2** are forwarding data to **ether3**, but when **ether3** is congested, then packets are going to be scheduled, we can configure the switch to hold lowest priority packets until all higher priority packets are sent out, this is a very common scenario for VoIP type setups, where some traffic needs to be prioritized.

To achieve such a behavior, switch together **ether1**, **ether2**, and **ether3** ports:

```
/interface bridge
add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether1 hw=yes
add bridge=bridge1 interface=ether2 hw=yes
add bridge=bridge1 interface=ether3 hw=yes
```

Enable **Strict Policy** for each internal queue on each port:

```
/interface ethernet switch port
set ether1,ether2,ether3 per-queue-scheduling="strict-priority:0,strict-priority:0,strict-priority:0,strict-
priority:0,strict-priority:0,strict-priority:0,strict-priority:0,strict-priority:0"
```

Map each PCP value to an internal priority value, for convenience reasons simply map PCP to an internal priority 1-to-1:

```
/interface ethernet switch port
set ether1,ether2,ether3 pcp-based-qos-priority-mapping=0:0,1:1,2:2,3:3,4:4,5:5,6:6,7:7
```

Since the switch will empty the largest queue first and you need the highest priority to be served first, then you can assign this internal priority to a queue 1-to-1:

```
/interface ethernet switch port
set ether1,ether2,ether3 priority-to-queue=0:0,1:1,2:2,3:3,4:4,5:5,6:6,7:7
```

Finally, set each switch port to schedule packets based on the PCP value:

```
/interface ethernet switch port
set ether1,ether2,ether3 qos-scheme-precedence=pcp-based
```

Bandwidth Limiting

Both Ingress Port policer and Shaper provide bandwidth limiting features for CRS switches:

Ingress Port Policer sets RX limit on port:

```
/interface ethernet switch ingress-port-policer
add port=ether5 meter-unit=bit rate=10M
```

Shaper sets TX limit on port:

```
/interface ethernet switch shaper
add port=ether5 meter-unit=bit rate=10M
```

Traffic Storm Control

The same Ingress Port policer also can be used for traffic storm control to prevent disruptions on Layer 2 ports caused by broadcast, multicast, or unicast traffic storms.

Broadcast storm control example on ether5 port with 500 packet limit per second:

```
/interface ethernet switch ingress-port-policer
add port=ether5 rate=500 meter-unit=packet packet-types=broadcast
```

Example with multiple packet types that include ARP and ND protocols and unregistered multicast traffic. Unregistered multicast is the traffic which is not defined in the Multicast Forwarding database:

```
/interface ethernet switch ingress-port-policer
add port=ether5 rate=5k meter-unit=packet packet-types=broadcast,arp-or-nd,unregistered-multicast
```