Quality of Service (QoS)

Table of contents

Quality of Service (QoS) is a mechanism of assigning a priority to a network flow (socket, rdma_cm connection) and manage its guarantees, limitations and its priority over other flows. This is accomplished by mapping the user's priority to a hardware TC (traffic class) through a 2/3 stage process. The TC is assigned with the QoS attributes and the different flows behave accordingly.

Mapping Traffic to Traffic Classes

Mapping traffic to TCs consists of several actions which are user controllable, some controlled by the application itself and others by the system/network administrators. The following is the general mapping traffic to Traffic Classes flow:

1. The application sets the required Type of Service (ToS).
2. The ToS is translated into a Socket Priority (sk_prio).
3. The sk_prio is mapped to a User Priority (UP) by the system administrator (some applications set sk_prio directly).
4. The UP is mapped to TC by the network/system administrator.
5. TCs hold the actual QoS parameters

QoS can be applied on the following types of traffic. However, the general QoS flow may vary among them:

- **Plain Ethernet** - Applications use regular inet sockets and the traffic passes via the kernel Ethernet driver
- **RoCE** - Applications use the RDMA API to transmit using Queue Pairs (QPs)
- **Raw Ethernet QP** - Application use VERBs API to transmit using a Raw Ethernet QP

Plain Ethernet Quality of Service Mapping

Applications use regular inet sockets and the traffic passes via the kernel Ethernet driver. The following is the Plain Ethernet QoS mapping flow:

1. The application sets the ToS of the socket using setsockopt (IP_TOS, value).

2. ToS is translated into the sk_prio using a fixed translation:

```
TOS 0 <=> sk_prio 0
TOS 8 <=> sk_prio 2
TOS 24 <=> sk_prio 4
TOS 16 <=> sk_prio 6
```

3. The Socket Priority is mapped to the UP in the following conditions:

    1. If the underlying device is a VLAN device, egress_map is used controlled by the vconfig command. This is per VLAN mapping.

    2. If the underlying device is not a VLAN device, the mapping is done in the driver.

4. The UP is mapped to the TC as configured by the mlnx_qos tool or by the lldpad daemon if DCBX is used.

> ⓘ **Note**
>
> Socket applications can use setsockopt (SK_PRIO, value) to directly set the sk_prio of the socket. In this case, the ToS to sk_prio fixed mapping is not needed. This allows the application and the administrator to utilize more than the 4 values possible via ToS.

> ⓘ **Note**
>
> In the case of a VLAN interface, the UP obtained according to the above mapping is also used in the VLAN tag of the traffic.

RoCE Quality of Service Mapping

Applications use RDMA-CM API to create and use QPs. The following is the RoCE QoS mapping flow:

1. The application sets the ToS of the QP using the rdma_set_option option(RDMA_OPTION_ID_TOS, value).

2. ToS is translated into the Socket Priority (sk_prio) using a fixed translation:

```
TOS 0 <=> sk_prio 0
TOS 8 <=> sk_prio 2
TOS 24 <=> sk_prio 4
TOS 16 <=> sk_prio 6
```

3. The Socket Priority is mapped to the User Priority (UP) using the tc command.

- In the case of a VLAN device where the parent real device is used for the purpose of this mapping

- If the underlying device is a VLAN device, and the parent real device was not used for the mapping, the VLAN device's egress_map is used

4. UP is mapped to the TC as configured by the mlnx_qos tool or by the lldpad daemon if DCBX is used.

> **Note**
>
> With RoCE, there can only be 4 predefined ToS values for the purpose of QoS mapping.

Map Priorities with set_egress_map

For RoCE old kernels that do not support set_egress_map, use the tc_wrap script to map between sk_prio and UP. Use tc_wrap with option -u. For example:

```
tc_wrap -i <ethX> -u <skprio2up mapping>
```

Quality of Service Properties

The different QoS properties that can be assigned to a TC are:

- Strict Priority
- Enhanced Transmission Selection (ETS)
- Rate Limit
- Trust State
- Receive Buffer
- DCBX Control Mode

Strict Priority

When setting a TC's transmission algorithm to be 'strict', then this TC has absolute (strict) priority over other TC strict priorities coming before it (as determined by the TC number: TC 7 is the highest priority, TC 0 is lowest). It also has an absolute priority over nonstrict TCs (ETS).
This property needs to be used with care, as it may easily cause starvation of other TCs. A higher strict priority TC is always given the first chance to transmit. Only if the highest strict priority TC has nothing more to transmit, will the next highest TC be considered. Nonstrict priority TCs will be considered last to transmit.
This property is extremely useful for low latency low bandwidth traffic that needs to get immediate service when it exists, but is not of high volume to starve other transmitters in the system.

Enhanced Transmission Selection (ETS)

Enhanced Transmission Selection standard (ETS) exploits the time periods in which the offered load of a particular Traffic Class (TC) is less than its minimum allocated bandwidth by allowing the difference to be available to other traffic classes.

After servicing the strict priority TCs, the amount of bandwidth (BW) left on the wire may be split among other TCs according to a minimal guarantee policy.
If, for instance, TC0 is set to 80% guarantee and TC1 to 20% (the TCs sum must be 100), then the BW left after servicing all strict priority TCs will be split according to this ratio. Since this is a minimum guarantee, there is no maximum enforcement. This means, in the same example, that if TC1 did not use its share of 20%, the reminder will be used by TC0.
ETS is configured using the mlnx_qos tool (mlnx_qos) which allows you to:

- Assign a transmission algorithm to each TC (strict or ETS)
- Set minimal BW guarantee to ETS TCs
  Usage:

```
mlnx_qos -i \[options\]
```

Rate Limit

Rate limit defines a maximum bandwidth allowed for a TC. Please note that 10% deviation from the requested values is considered acceptable.

Trust State

Trust state enables prioritizing sent/received packets based on packet fields.
The default trust state is PCP. Ethernet packets are prioritized based on the value of the field (PCP/DSCP).
For further information on how to configure Trust mode, please refer to HowTo Configure Trust State on NVIDIA Adapters community post.

Note

Setting the Trust State mode shall be done before enabling SR-IOV in order to propagate the Trust State to the VFs.

Receive Buffer

By default, the receive buffer configuration is controlled automatically. Users can override the receive buffer size and receive buffer's xon and xoff thresholds using mlnx_qos tool.
For further information, please refer to HowTo Tune the Receive buffers on NVIDIA Adapters community post.

DCBX Control Mode

DCBX settings, such as "ETS" and "strict priority" can be controlled by firmware or software. When DCBX is controlled by firmware, changes of QoS settings cannot be done by the software. The DCBX control mode is configured using the mlnx_qos -d os/fw command.
For further information on how to configure the DCBX control mode, please refer to mlnx_qos community post.

Quality of Service Tools

mlnx_qos

mlnx_qos is a centralized tool used to configure QoS features of the local host. It communicates directly with the driver thus does not require setting up a DCBX daemon on the system.
The mlnx_qos tool enables the administrator of the system to:

- Inspect the current QoS mappings and configuration
  The tool will also display maps configured by TC and vconfig set_egress_map tools,

in order to give a centralized view of all QoS mappings.

- Set UP to TC mapping
- Assign a transmission algorithm to each TC (strict or ETS)
- Set minimal BW guarantee to ETS TCs
- Set rate limit to TCs
- Set DCBX control mode
- Set cable length
- Set trust state

> **Note**
>
> For an unlimited ratelimit, set the ratelimit to 0.

Usage

```
mlnx_qos -i <interface> \[options\]
```

Options

--version	Show the program's version number and exit
-h, --help	Show this help message and exit
-f LIST, --	Set priority flow control for each priority. LIST is a comma separated value for each priority starting from 0 to 7. Example: 0,0,0,0,1,1,1,1 enable PFC on TC4-7

pfc=LIST	
-p LIST, --prio_tc=LIST	Maps UPs to TCs. LIST is 8 comma-separated TC numbers. Example: 0,0,0,0,1,1,1,1 maps UPs 0-3 to TC0, and UPs 4-7 to TC1
-s LIST, --tsa=LIST	Transmission algorithm for each TC. LIST is comma separated algorithm names for each TC. Possible algorithms: strict, ets and vendor. Example: vendor,strict,ets,ets,ets,ets,ets,ets sets TC0 to vendor, TC1 to strict, TC2-7 to ets
-t LIST, --tcbw=LIST	Set the minimally guaranteed %BW for ETS TCs. LIST is comma-separated percents for each TC. Values set to TCs that are not configured to ETS algorithm are ignored but must be present. Example: if TC0,TC2 are set to ETS, then 10,0,90,0,0,0,0,0will set TC0 to 10% and TC2 to 90%. Percents must sum to 100
-r LIST, --ratelimit=LIST	Rate limit for TCs (in Gbps). LIST is a comma-separated Gbps limit for each TC. Example: 1,8,8 will limit TC0 to 1Gbps, and TC1,TC2 to 8 Gbps each
-d DCBX, --dcbx=DCBX	Set dcbx mode to firmware controlled(fw) or OS controlled(os). Note, when in OS mode, mlnx_qos should not be used in parallel with other dcbx tools, such as lldptool
--trust=TRUST	set priority trust state to pcp or dscp
--dscp2prio=DSCP2PRIO	Set/del a (dscp,prio) mapping. Example 'set,30,2' maps dscp 30 to priority 2. 'del,30,2' resets the dscp 30 mapping back to the default setting priority 0
--cable_len=CA	Set cable_len for buffer's xoff and xon thresholds

BLE_LEN	
-i INTF, --interface=INTF	Interface name
-a	Show all interface's TCs

Get Current Configuration

```
ofed_scripts/utils/mlnx_qos -i ens1f0
DCBX mode: OS controlled
Priority trust state: dscp
dscp2prio mapping:
prio:0 dscp:07,06,05,04,03,02,01,00,
prio:1 dscp:15,14,13,12,11,10,09,08,
prio:2 dscp:23,22,21,20,19,18,17,16,
prio:3 dscp:31,30,29,28,27,26,25,24,
prio:4 dscp:39,38,37,36,35,34,33,32,
prio:5 dscp:47,46,45,44,43,42,41,40,
prio:6 dscp:55,54,53,52,51,50,49,48,
prio:7 dscp:63,62,61,60,59,58,57,56,
Cable len: 7
PFC configuration:
priority 0 1 2 3 4 5 6 7
enabled 0 0 0 0 0 0 0 0
tc: 0 ratelimit: unlimited, tsa: vendor
priority: 1
tc: 1 ratelimit: unlimited, tsa: vendor
priority: 0
tc: 2 ratelimit: unlimited, tsa: vendor
priority: 2
tc: 3 ratelimit: unlimited, tsa: vendor
```

```
priority: 3
tc: 4 ratelimit: unlimited, tsa: vendor
priority: 4
tc: 5 ratelimit: unlimited, tsa: vendor
priority: 5
tc: 6 ratelimit: unlimited, tsa: vendor
priority: 6
tc: 7 ratelimit: unlimited, tsa: vendor
priority: 7
```

Set ratelimit. 3Gbps for tc0 4Gbps for tc1 and 2Gbps for tc2

```
# mlnx_qos -i <interface> -p 0,1,2 -r 3,4,2
tc: 0 ratelimit: 3 Gbps, tsa: strict
up: 0
skprio: 0
skprio: 1
skprio: 2 (tos: 8)
skprio: 3
skprio: 4 (tos: 24)
skprio: 5
skprio: 6 (tos: 16)
skprio: 7
skprio: 8
skprio: 9
skprio: 10
skprio: 11
skprio: 12
skprio: 13
skprio: 14
skprio: 15
up: 3
up: 4
up: 5
```

```
up: 6
up: 7
tc: 1 ratelimit: 4 Gbps, tsa: strict
up: 1
tc: 2 ratelimit: 2 Gbps, tsa: strict
up: 2
```

ConfigureQoS. Map UP0,7 to tc0,1,2,3 to tc1 and 4,5,6 to tc2. Set tc0,tc1 as ets and tc2 as strict. Divide ets 30% for tc0 and 70% for tc1

```
# mlnx_qos -i <interface> -s ets,ets,strict -p 0,1,1,1,2,2,2 -t 30,70
tc: 0 ratelimit: 3 Gbps, tsa: ets, bw: 30%
up: 0
skprio: 0
skprio: 1
skprio: 2 (tos: 8)
skprio: 3
skprio: 4 (tos: 24)
skprio: 5
skprio: 6 (tos: 16)
skprio: 7
skprio: 8
skprio: 9
skprio: 10
skprio: 11
skprio: 12
skprio: 13
skprio: 14
skprio: 15
up: 7
tc: 1 ratelimit: 4 Gbps, tsa: ets, bw: 70%
up: 1
up: 2
up: 3
```

```
tc: 2 ratelimit: 2 Gbps, tsa: strict
up: 4
up: 5
up: 6
```

tc and tc_wrap.py

The tc tool is used to create 8 Traffic Classes (TCs).
The tool will either use the sysfs (/sys/class/net/<ethX>/qos/tc_num) or the tc tool to create the TCs.

Usage

```
tc_wrap.py -i <interface> \[options\]
```

Options

--version	show program's version number and exit
-h, --help	show this help message and exit
-u SKPRIO_UP, --skprio_up=SKPRIO_UP	maps sk_prio to priority for RoCE. LIST is <=16 comma separated priority. index of element is sk_prio
-i INTF, --interface=INTF	Interface name

Example

Run:

```
tc_wrap.py -i enp139s0
```

Output:

```
Tarrfic classes are set to 8

UP 0
```

```
skprio: 0 (vlan 5)
UP 1
skprio: 1 (vlan 5)
UP 2
skprio: 2 (vlan 5 tos: 8)
UP 3
skprio: 3 (vlan 5)
UP 4
skprio: 4 (vlan 5 tos: 24)
UP 5
skprio: 5 (vlan 5)
UP 6
skprio: 6 (vlan 5 tos: 16)
UP 7
skprio: 7 (vlan 5)
```

Additional Tools

tc tool compiled with the sch_mqprio module is required to support kernel v2.6.32 or higher. This is a part of iproute2 package v2.6.32-19 or higher. Otherwise, an alternative custom sysfs interface is available.

- mlnx_qos tool (package: ofed-scripts) requires python version 2.5 < = X
- tc_wrap.py (package: ofed-scripts) requires python version 2.5 < = X