Egress and Ingress Example

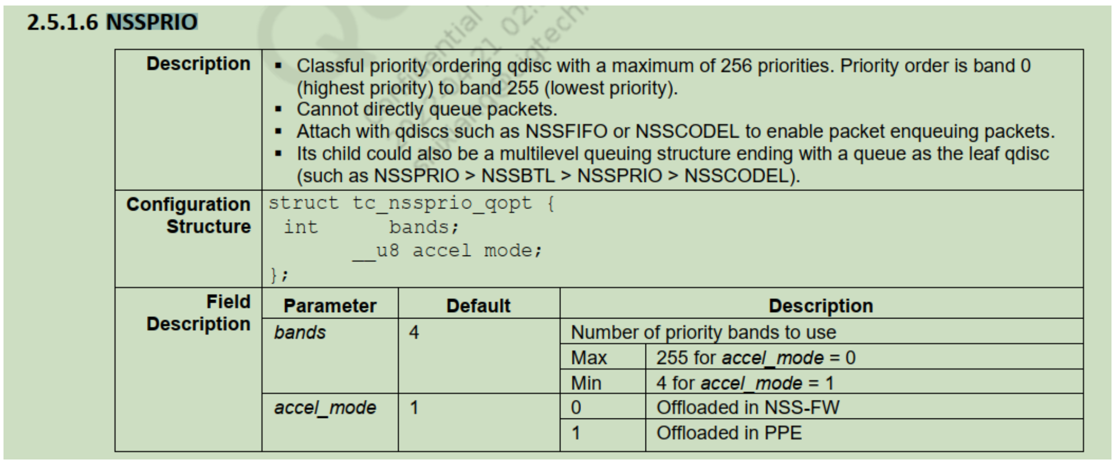
We need some iptables lib to do this test. (libxt_CLASSIFY.so libxt_dscp.so libxt_tos.so)
The lib is not contained in current software. If you want to test, we can send you these lib and you should put them into /usr/lib/iptables. These libs will be contained in the future release.

The first test, egress test.

```
egress qos test
    iptables -t mangle -I OUTPUT -m tos --tos 0x0 -o eth0 -j CLASSIFY --set-class 100:0
    tc qdisc add dev eth0 root handle 1: nssprio bands 2
    tc qdisc add dev eth0 parent 1:1 handle 10: nsstbl rate 10Mbit burst 30K
    tc qdisc add dev eth0 parent 1:2 handle 20: nsstbl rate 1000Mbit burst 30K
    tc qdisc add dev eth0 parent 10: handle 100: nsspfifo limit 100
    tc qdisc add dev eth0 parent 20: handle 200: nsspfifo limit 100 set_default
```

The iptables will check the packet tos. if it is 0, it will set the mark 100:0 on that packet.

These tc commands set up 2 queues, 100: and 200:, with 200: as the default queue where all nonclassified packets enqueue. In this configuration, the 100: queue attaches at higher priority compared to queue 200:. It ensures that traffic directed to queue 100: (that is, flows with tos=0) are scheduled with higher priority compared to other flows in the system. Use nssprio to create two queues and use nsstbl to set the speed limitation of the queue.
From this example, the packet that tos is 0 will go to 100:0 that speed is set to 10Mbit



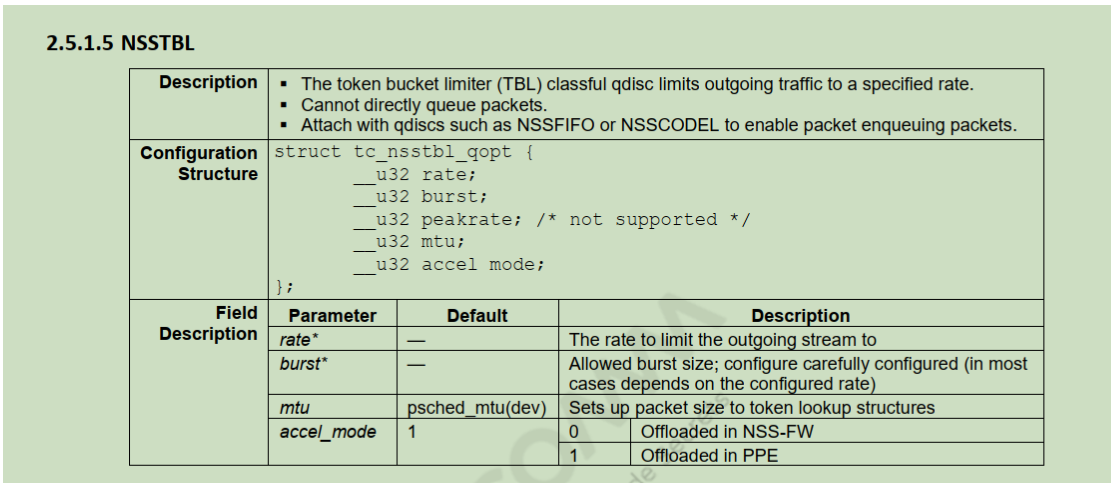
2.5.1.6 NSSPRIO

Description	▪ Classful priority ordering qdisc with a maximum of 256 priorities. Priority order is band 0 (highest priority) to band 255 (lowest priority). ▪ Cannot directly queue packets. ▪ Attach with qdiscs such as NSSFIFO or NSSCODEL to enable packet enqueuing packets. ▪ Its child could also be a multilevel queuing structure ending with a queue as the leaf qdisc (such as NSSPRIO > NSSBTL > NSSPRIO > NSSCODEL).

Configuration Structure	`struct tc_nssprio_qopt {`
	`int bands;`
	`__u8 accel mode;`
	`};`

Field Description

Parameter	**Default**	**Description**	
bands	4	Number of priority bands to use	
		Max	255 for *accel_mode* = 0
		Min	4 for *accel_mode* = 1
accel_mode	1	0	Offloaded in NSS-FW
		1	Offloaded in PPE



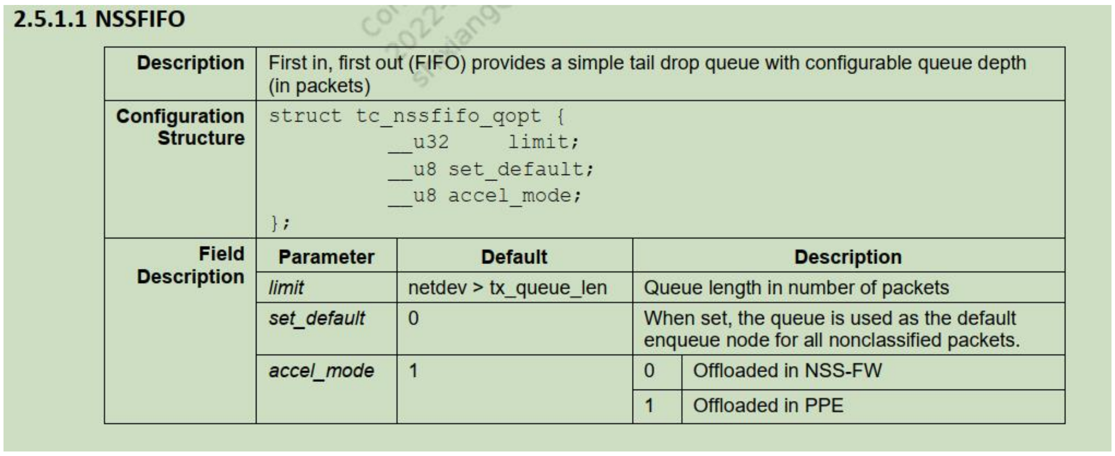
2.5.1.5 NSSTBL

Description	▪ The token bucket limiter (TBL) classful qdisc limits outgoing traffic to a specified rate. ▪ Cannot directly queue packets. ▪ Attach with qdiscs such as NSSFIFO or NSSCODEL to enable packet enqueuing packets.
Configuration Structure	see below

```
struct tc_nsstbl_qopt {
        __u32 rate;
        __u32 burst;
        __u32 peakrate; /* not supported */
        __u32 mtu;
        __u32 accel mode;
};
```

Field Description

Parameter	Default	Description	
*rate**	—	The rate to limit the outgoing stream to	
*burst**	—	Allowed burst size; configure carefully configured (in most cases depends on the configured rate)	
mtu	psched_mtu(dev)	Sets up packet size to token lookup structures	
accel_mode	1	0	Offloaded in NSS-FW
		1	Offloaded in PPE



2.5.1.1 NSSFIFO

Description	First in, first out (FIFO) provides a simple tail drop queue with configurable queue depth (in packets)
Configuration Structure	see below

```
struct tc_nssfifo_qopt {
          __u32     limit;
          __u8 set_default;
          __u8 accel_mode;
};
```

Field Description

Parameter	Default	Description	
limit	netdev > tx_queue_len	Queue length in number of packets	
set_default	0	When set, the queue is used as the default enqueue node for all nonclassified packets.	
accel_mode	1	0	Offloaded in NSS-FW
		1	Offloaded in PPE

The second test, ingress test.

```
    tc qdisc add dev eth0 handle ffff: ingress
    /etc/init.d/qca-nss-mirred start
    tc filter add dev eth0 parent ffff: u32 match u32 0 0 action nssmirred redirect dev ifb1
fromdev eth0
    tc qdisc add dev ifb1 root handle 1: nssprio bands 3
    tc qdisc add dev ifb1 parent 1:1 handle 11: nsspfifo limit 100
    tc qdisc add dev ifb1 parent 1:2 handle 12: nsspfifo limit 100
    tc qdisc add dev ifb1 parent 1:3 handle 13: nsspfifo limit 100 set_default
    tc filter add dev ifb1 parent 1: protocol ip u32 match ip dst 192.168.1.90/4 flowid 11:0
    ifconfig ifb1 up
```

The Linux traffic control utility (**tc**) supports ingress Qdisc. For traffic received on an interface, this ingress Qdisc is traversed. Ingress Qdisc supports only policing through **tc** filters and shapers cannot be attached to it. Ingress Qdisc is used together with Intermediate Functional Block interface for attaching shapers and shaping the ingress traffic.

The first command: Attach the ingress Qdisc to the given interface where ingress shaping is needed

The second command: It will create the virtual interface, ifb1.

The third command: Attach the filter rule to redirect ingress traffic to an IFB interface. Use the custom action object **nssmirred** to redirect traffic. This action triggers the virtual interface creation and redirection through **nssmirred.ko** in NSS.

The command then creates three queues. The 13: is the default queue for packet. 12: and 11: are the high priority queues. The last command sets the ip 192.168.1.90/4 to go to the high priority queue 11:0.

Some command to check the tc status.

Method	example
check the interface rules	tc -s qdisc show dev eth0
delete the rules on interface	tc qdisc del dev eth0 root
change the rules when the tc is running	tc qdisc change dev eth0 root handle 1: nsstbl rate 10Mbit burst 30K