Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'tc-taprio.8' command

$ man tc-taprio.8

TAPRIO(8) Linux TAPRIO(8)

NAME

TAPRIO - Time Aware Priority Shaper

SYNOPSIS

tc qdisc ... dev dev parent classid [handle major:] taprio num_tc tcs

map P0 P1 P2 ... queues count1@offset1 count2@offset2 ...

base-time base-time clockid clockid

sched-entry <command 1> <gate mask 1> <interval 1>

sched-entry <command 2> <gate mask 2> <interval 2>

sched-entry <command 3> <gate mask 3> <interval 3>

sched-entry <command N> <gate mask N> <interval N>

DESCRIPTION

The TAPRIO qdisc implements a simplified version of the scheduling state machine defined by IEEE 802.1Q-2018 Section 8.6.9, which allows configuration of a sequence of gate states, where each gate state al? lows outgoing traffic for a subset (potentially empty) of traffic classes.

How traffic is mapped to different hardware queues is similar to mqprio(8) and so the map and queues parameters have the same meaning.

The other parameters specify the schedule, and at what point in time it should start (it can behave as the schedule started in the past).

PARAMETERS

num_tc Number of traffic classes to use. Up to 16 classes supported.

map

The priority to traffic class map. Maps priorities 0..15 to a specified traffic class. See mqprio(8) for more details.

queues

Provide count and offset of queue range for each traffic class. In the format, count@offset. Queue ranges for each traffic classes cannot overlap and must be a contiguous range of queues.

base-time

Specifies the instant in nanoseconds, using the reference of clockid, defining the time when the schedule starts. If 'base-time' is a time in the past, the schedule will start at

base-time + (N * cycle-time)

where N is the smallest integer so the resulting time is greater than "now", and "cycle-time" is the sum of all the intervals of the entries in the schedule;

clockid

Specifies the clock to be used by qdisc's internal timer for measuring time and scheduling events. This argument must be omitted when using the full-offload feature (flags 0x2), since in that case, the clockid is implicitly /dev/ptpN (where N is given by ethtool -T eth0 | grep 'PTP Hardware Clock'), and therefore not necessarily synchronized with the system's CLOCK_TAI.

sched-entry

There may multiple sched-entry parameters in a single schedule. Each one has the

sched-entry <command> <gatemask> <interval>

format. The only supported <command> is "S", which means "Set? GateStates", following the IEEE 802.1Q-2018 definition (Table 8-7). <gate mask> is a bitmask where each bit is a associated with a traffic class, so bit 0 (the least significant bit) being "on" means that traffic class 0 is "active" for that schedule entry. <interval> is a time duration, in nanoseconds, that

specifies for how long that state defined by <command> and <gate mask> should be held before moving to the next entry.

flags

This is a bit mask which specifies different modes for taprio.

0x1 Enables the txtime-assist feature. In this mode, taprio will set the transmit timestamp depending on the interval in which the packet needs to be transmitted. It will then utililize the etf(8) qdisc to sort and transmit the pack? ets at the right time. The second example can be used as a reference to configure this mode.

0x2 Enables the full-offload feature. In this mode, taprio will pass the gate control list to the NIC which will ex? ecute it cyclically in hardware. When using full-off? load, there is no need to specify the clockid argument. The txtime-assist and full-offload features are mutually exclusive, i.e. setting flags to 0x3 is invalid.

txtime-delay

This parameter is specific to the txtime offload mode. It speci? fies the maximum time a packet might take to reach the network card from the taprio qdisc. The value should always be greater than the delta specified in the etf(8) qdisc.

EXAMPLES

The following example shows how an traffic schedule with three traffic classes ("num_tc 3"), which are separated different traffic classes, we are going to call these TC 0, TC 1 and TC 2. We could read the "map" parameter below as: traffic with priority 3 is classified as TC 0, pri? ority 2 is classified as TC 1 and the rest is classified as TC 2.

The schedule will start at instant 1528743495910289987 using the refer? ence CLOCK_TAI. The schedule is composed of three entries each of 300us duration.

```
# tc qdisc replace dev eth0 parent root handle 100 taprio \
      num_tc 3 \
      map 2 2 1 0 2 2 2 2 2 2 2 2 2 2 2 2 \
```

```
        queues 1@0 1@1 2@2 \
        base-time 1528743495910289987 \
        sched-entry S 01 300000 \
        sched-entry S 02 300000 \
        sched-entry S 04 300000 \
        clockid CLOCK_TAI
```

Following is an example to enable the txtime offload mode in taprio. See etf(8) for more information about configuring the ETF qdisc.

```
# tc qdisc replace dev eth0 parent root handle 100 taprio \
        num_tc 3 \
        map 2 2 1 0 2 2 2 2 2 2 2 2 2 2 2 2 \
        queues 1@0 1@0 1@0 \
        base-time 1528743495910289987 \
        sched-entry S 01 300000 \
        sched-entry S 02 300000 \
        sched-entry S 04 400000 \
        flags 0x1 \
        txtime-delay 200000 \
        clockid CLOCK_TAI
# tc qdisc replace dev $IFACE parent 100:1 etf skip_skb_check \
        offload delta 200000 clockid CLOCK_TAI
```

The following is a schedule in full offload mode. The base-time is 200 ns and the cycle-time is implicitly calculated as the sum of all sched-entry durations (i.e. 20 us + 20 us + 60 us = 100 us). Although the base-time is in the past, the hardware will start executing the sched? ule at a PTP time equal to the smallest integer multiple of 100 us, plus 200 ns, that is larger than the NIC's current PTP time.

```
# tc qdisc add dev eth0 parent root taprio \
        num_tc 8 \
        map 0 1 2 3 4 5 6 7 \
        queues 1@0 1@1 1@2 1@3 1@4 1@5 1@6 1@7 \
        base-time 200 \
        sched-entry S 80 20000 \
```

```
        sched-entry S a0 20000 \
        sched-entry S df 60000 \
        flags 0x2
```

AUTHORS

Vinicius Costa Gomes <vinicius.gomes@intel.com>