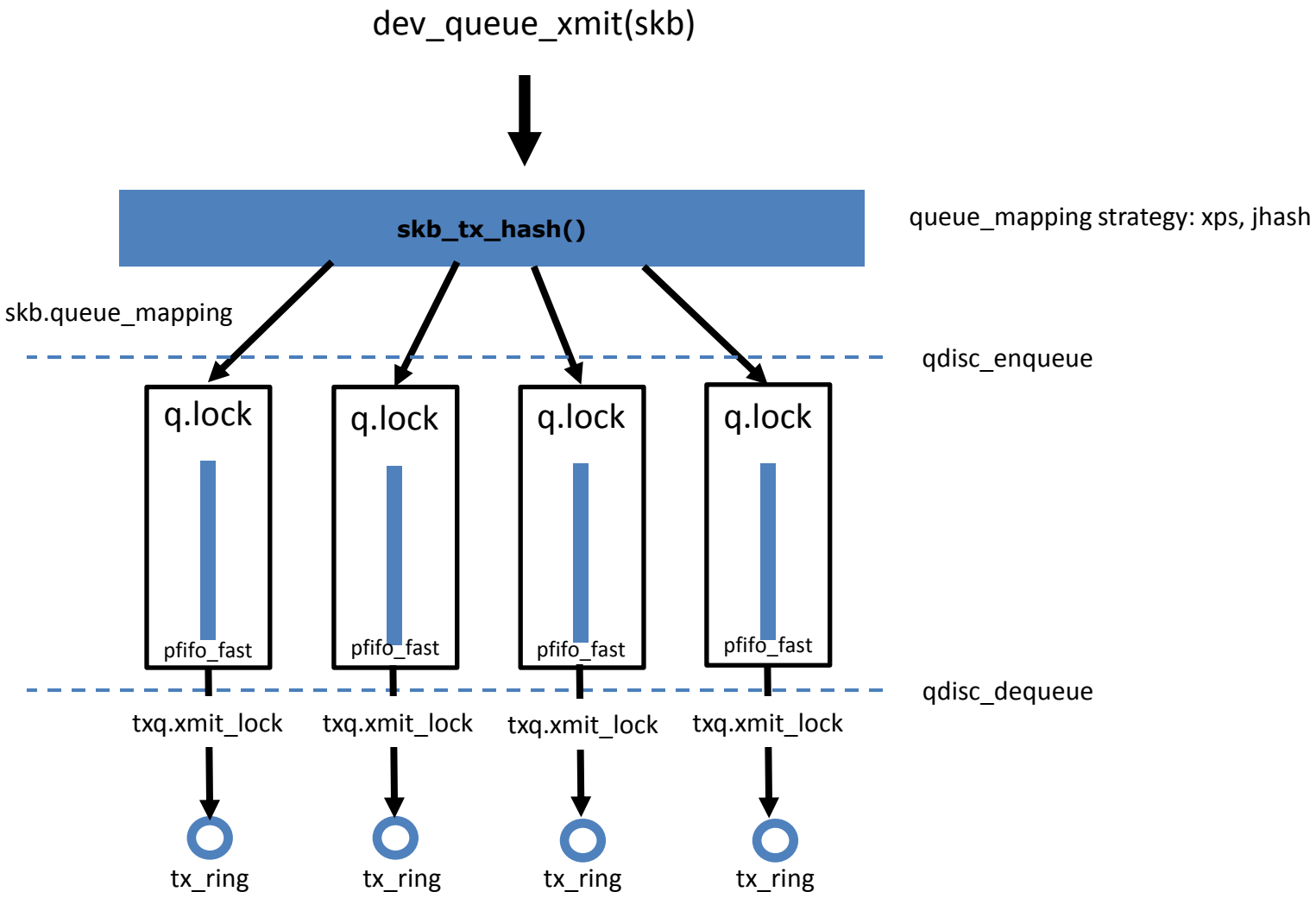
Qdisc Experiments at 10Gbps

John Fastabend
Linux Plumbers Conference 2012

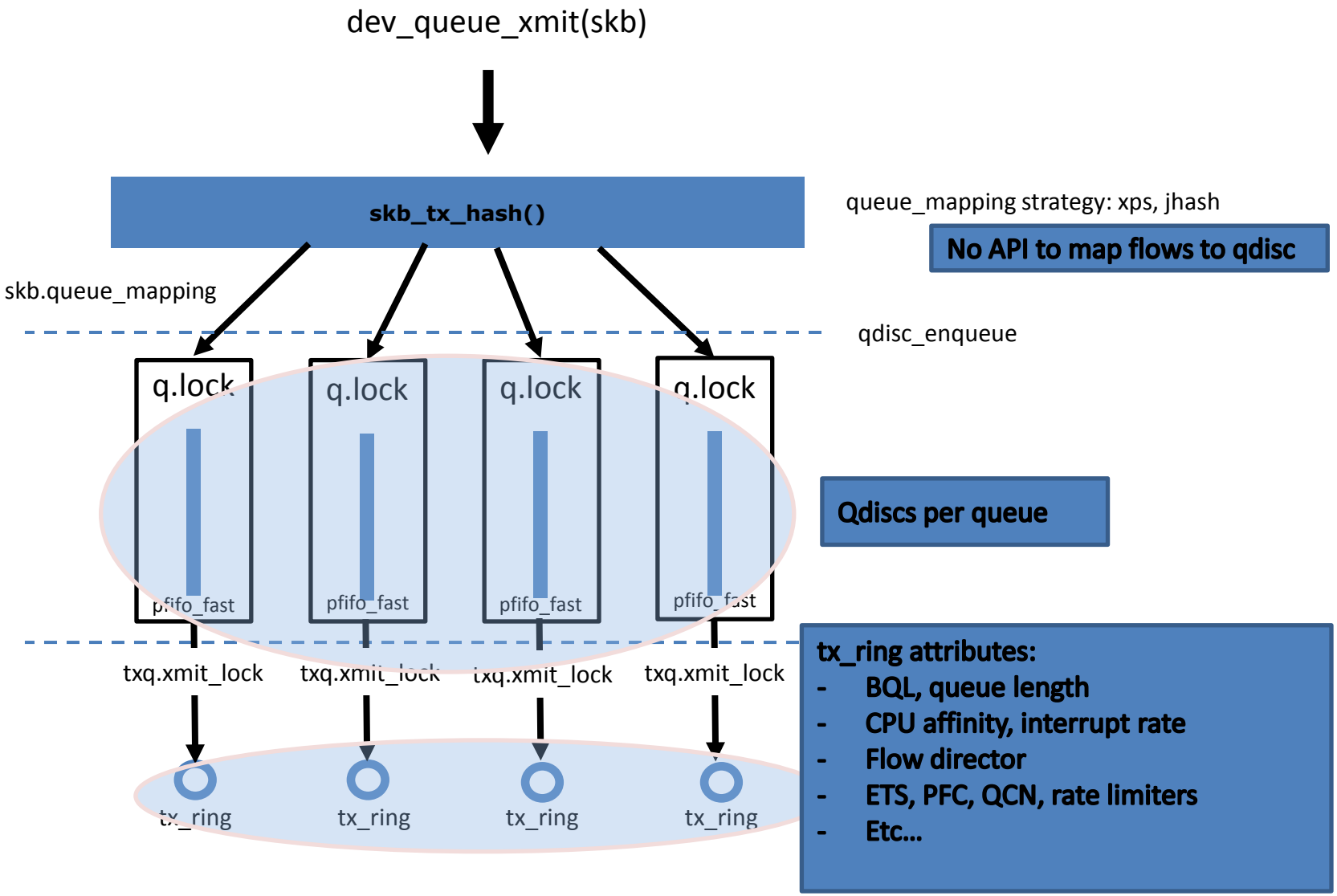
Motivation

- 10Gbps NICs generally available
  - With many queues maybe >100
- 40Gbps and faster is coming soon!
  - With even more queues
- Open-vSwitch
- multiqueue (sch_mq) scheduler is fast… but
  - traffic classification with multiqueue devices?
  - leverage hardware attributes to help QOS?
  - can we do SW QOS?

sch_mq


dev_queue_xmit(skb)
skb_tx_hash()
queue_mapping strategy: xps, jhash
skb.queue_mapping
qdisc_enqueue
q.lock
q.lock
q.lock
q.lock
pfifo_fast
pfifo_fast
pfifo_fast
pfifo_fast
qdisc_dequeue
txq.xmit_lock
txq.xmit_lock
txq.xmit_lock
txq.xmit_lock
tx_ring
tx_ring
tx_ring
tx_ring

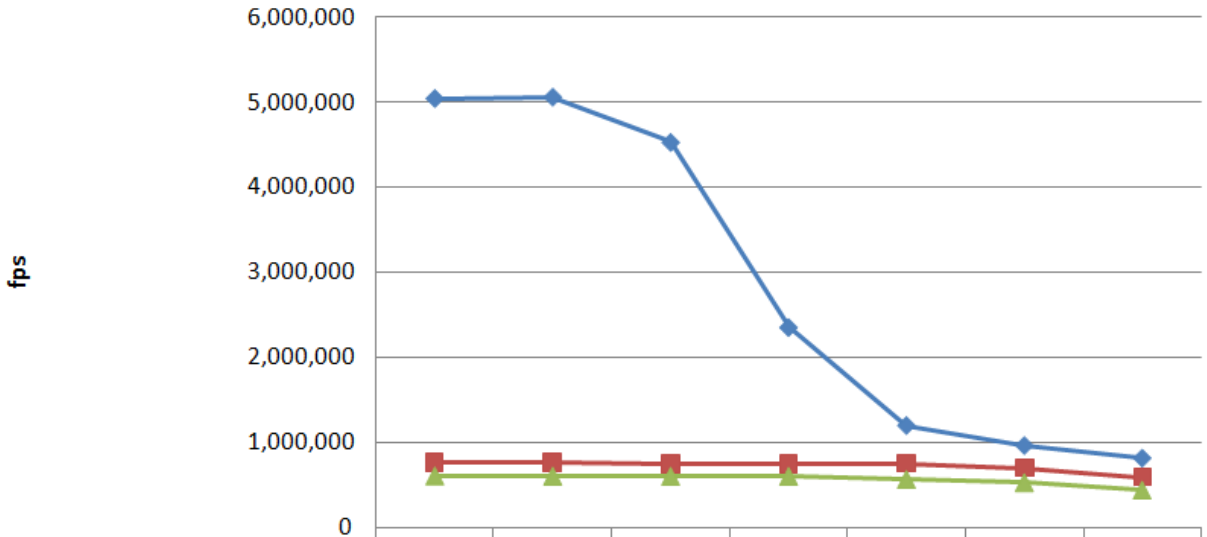
sch_mq


dev_queue_xmit(skb)
skb_tx_hash()
queue_mapping strategy: xps, jhash
No API to map flows to qdisc
skb.queue_mapping
qdisc_enqueue
q.lock
q.lock
q.lock
q.lock
pfifo_fast
pfifo_fast
pfifo_fast
pfifo_fast
Qdiscs per queue
txq.xmit_lock
txq.xmit_lock
txq.xmit_lock
txq.xmit_lock
tx_ring
tx_ring
tx_ring
tx_ring
tx_ring attributes:
- BQL, queue length
- CPU affinity, interrupt rate
- Flow director
- ETS, PFC, QCN, rate limiters
- Etc…

With a mechanism to map skb to queue(s) we could…

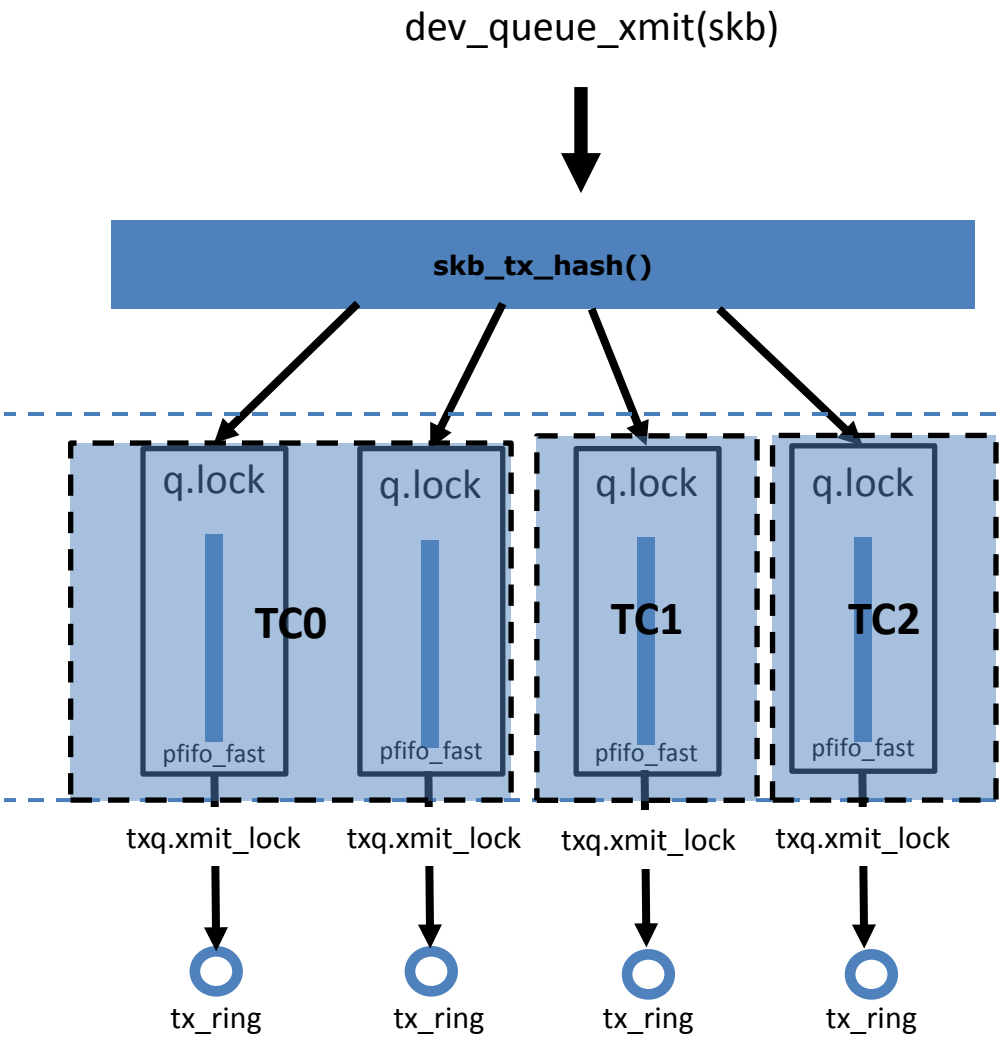
- use correct qdisc for the traffic type
- tune hardware TX rings for type of traffic
- implement hardware QOS per queue(s)
  - ETS (802.1Q), Link strict (802.1Q), rate limits, etc.

HTB qdisc would also work but does not utilize multiqueue HW



	64	128	256	512	1,024	1,280	1,454
Pure IP Fw (fps)	5,045,573	5,060,970	4,528,836	2,349,538	1,197,276	961,505	812,715
RSS Rx, DCB Tx + multiq (fps)	757,335	760,173	739,495	747,097	751,056	697,856	589,868
HTB qdisc (fps)	609,163	607,048	605,625	604,957	567,583	530,156	441,862

sch_mqprio



strategy: map skb priority to queue sets

qdisc_enqueue

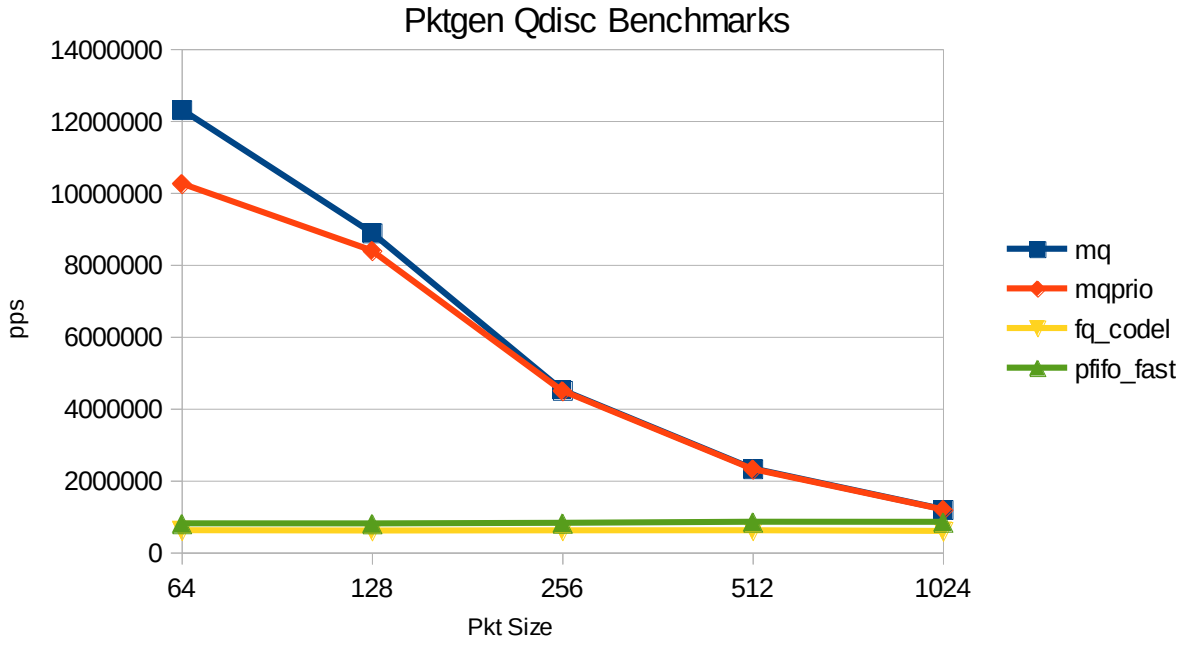
Use correct qdisc for the traffic type.

qdisc_dequeue

Configure tx rings to match traffic type possibly via DCBnl for ETS, low latency or through sysfs for BQL, etc.

Benchmark tests for Qdisc

- Intel Xeon X5570 @ 2.93Ghz -- quad-core 2 sockets + HT
- Pktgen injecting packets into vlan devices -- 16 threads



Cut through Latency

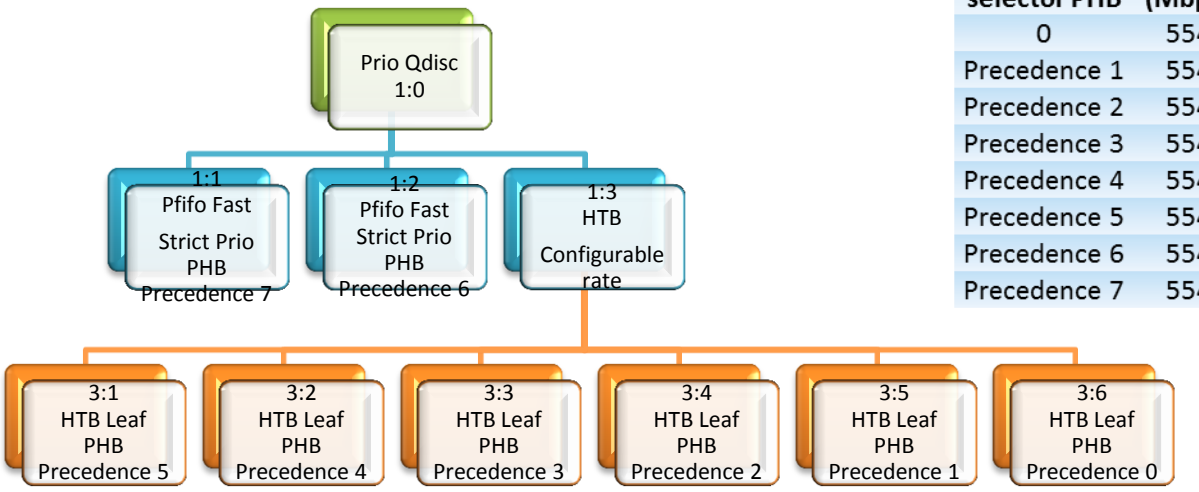
- Prio + HTB latency lots of jitter
  - Avg Latency 6.8ms
- Hardware QOS with mqprio qdisc
  - Avg Latency 0.066ms
- TODO: isolate hardware benefit

Figure1: mqprio HW ETS with link strict

IPv4 :Class selector PHB	Tx Rate (Mbps)	Rx Rate (Mbps)	Avg Latency (ns)	Min Latency (ns)	Max Latency (ns)
0	554	400	24,262,244	8,180	32,271,560
Precedence 1	554	400	24,248,749	8,180	32,345,920
Precedence 2	554	401	24,226,728	8,220	32,159,100
Precedence 3	554	400	24,231,051	8,120	32,150,900
Precedence 4	554	400	24,272,766	8,060	31,926,380
Precedence 5	554	400	24,254,297	8,080	31,930,020
Precedence 6	554	556	66,999	7,980	219,960
Precedence 7	554	556	67,055	7,960	231,720

Figure2: Prio qdisc with HTB

IPv4 :Class selector PHB	Tx Rate (Mbps)	Rx Rate (Mbps)	Avg Latency (ns)	Min Latency (ns)	Max Latency (ns)
0	554	404	32,771,322	8,380	37,955,640
Precedence 1	554	404	25,221,387	8,320	36,395,920
Precedence 2	554	404	34,155,207	8,300	38,338,500
Precedence 3	554	404	24,823,041	8,300	36,603,520
Precedence 4	554	404	32,342,354	8,300	38,334,140
Precedence 5	554	403	27,109,380	8,360	37,964,860
Precedence 6	554	463	2,913,085	7,900	15,697,840
Precedence 7	554	465	6,802,623	7,760	15,770,720



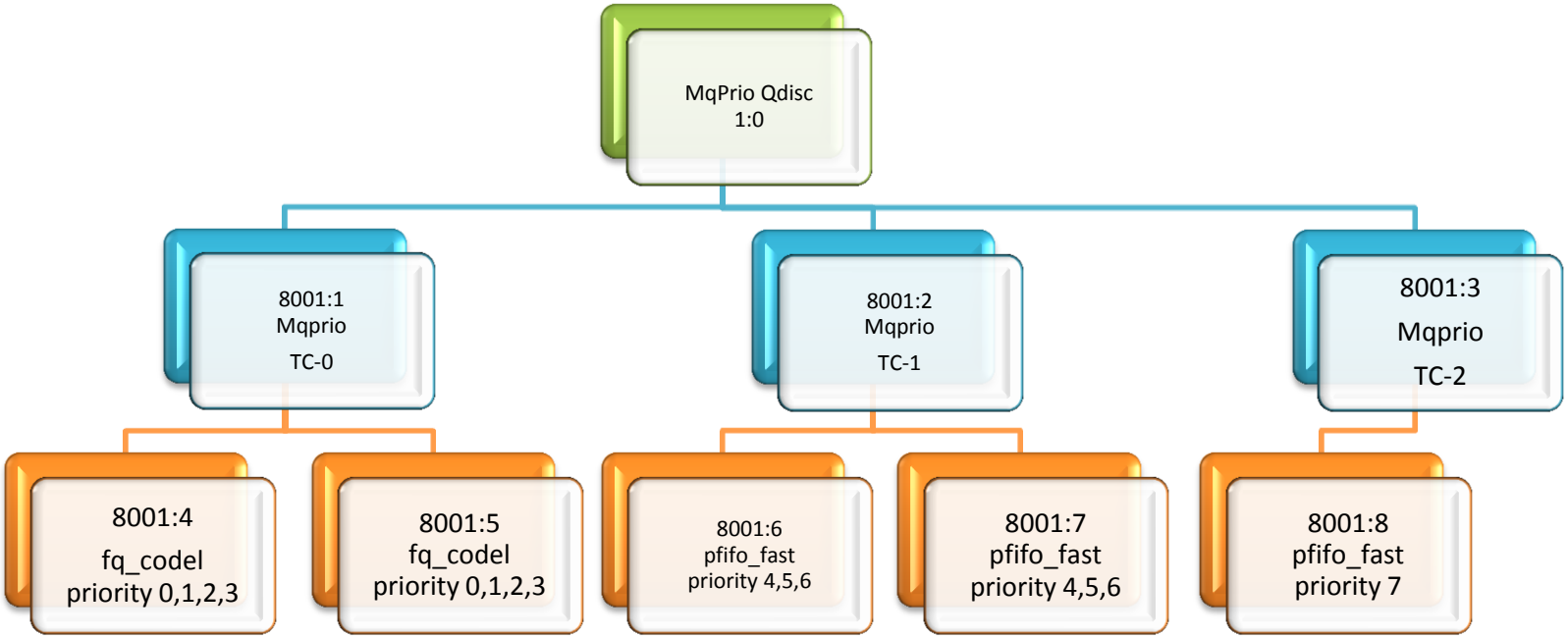
- Problems/Limitations
  - sch_mqprio only allows mapping via priority. Good enough?
  - filters/actions run under qdisc
  - Hard to use qdiscs with gobal state tbf, htb, etc.

- Add pre_enqueue() hook before enqueue()
  - Allows filters/actions to be run before qdisc enqueue
  - Use skb_edit and queue_mapping actions to control mqprio and mq flow to queue maps.

- How?
  - RCU tcf_proto, classifiers, actions (uses call_rcu_bh)
  - Stats per cpu

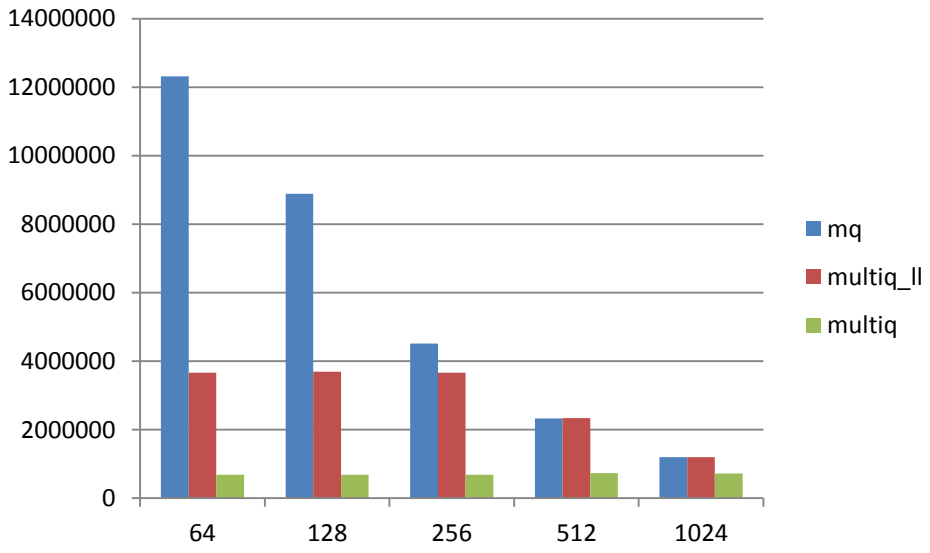
Example

```
#tc qdisc add dev eth3 root mqprio num_tc 3 map 0 0 0 0 1 1 1 2 queues 2@0 2@2 1@4
#tc qdisc add dev eth3 parent 8002:4 fq_codel
#tc qdisc add dev eth3 parent 8002:5 fq_codel
# tc filter add dev eth3 parent 0: prio 1 protocol ip u32  \
        match ip dport 118 0xffff action skbedit queue_mapping 4
```



Lockless Qdisc's

- add qdisc skb list lockless variants
- add flag to allow qdisc to run lockless
- enable lockless qdiscs (ingress, fq_codel, tbf, …)



Or should locking qdisc be attached to mq/mqprio?

Summary

- Problem:
  - Queuing disciplines and traffic classification do not work well with multiqueue devices
- Solution:
  - Write multiqueue aware queuing disciplines e.g. mqprio
  - stack qdiscs under mq aware qdiscs
- Other solutions?
  - use lockless classifiers and actions
  - Hardware offloaded QOS

Questions/Comments?

git://github.com/jrfastab/