CPRI “FrontHaul” requirements continuing discussion with TSN

Peter.AshwoodSmith@Huawei.com (presenter)
Tao.Wan@Huawei.com (simulations)

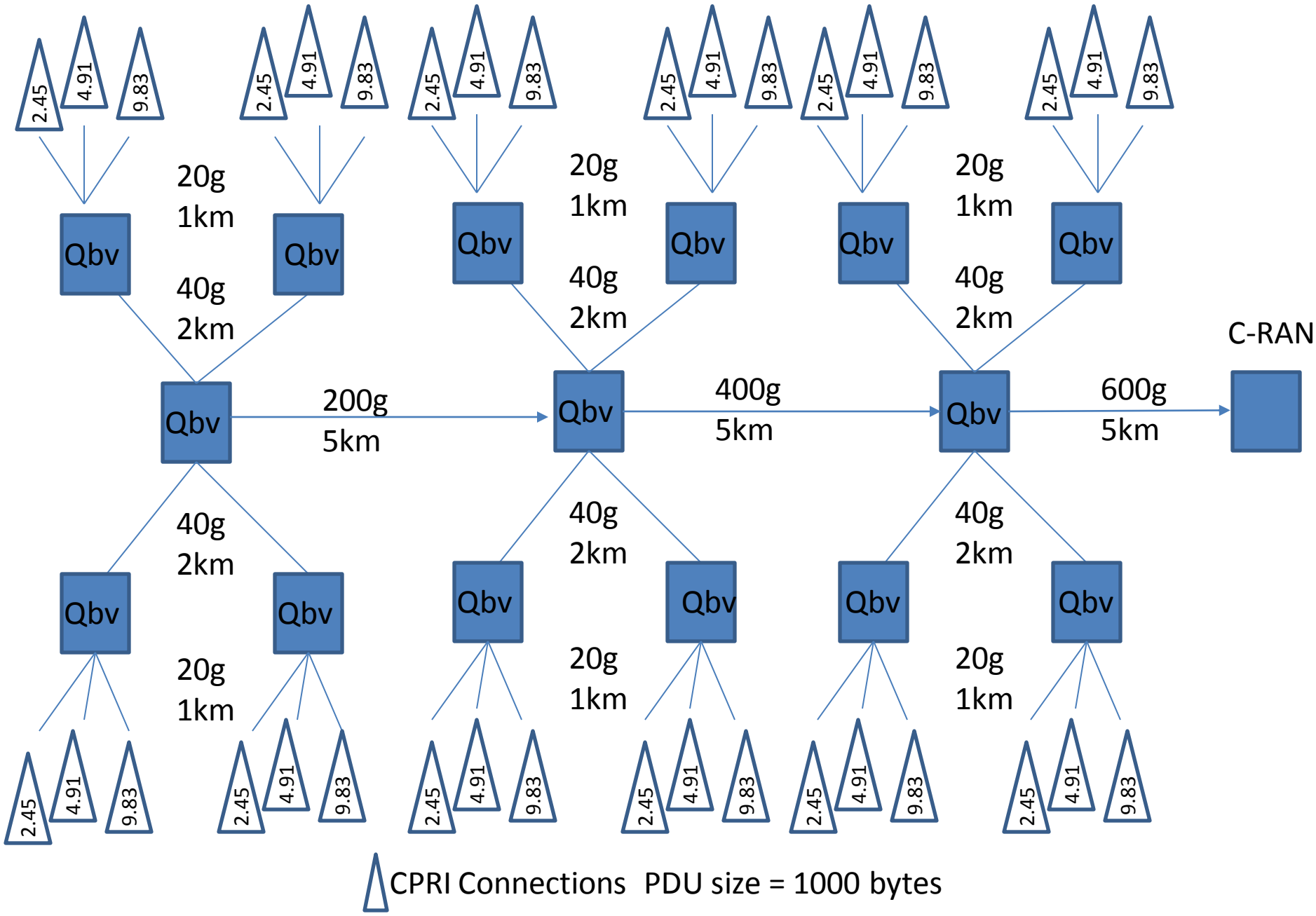
Nov 2014 San Antonio, Texas

Experimental Goal - ongoing

- **To run the simplest possible simulation of a CPRI front haul network.**
- **Make use of the TSN concepts.**
- **Look at the before/after results for jitter/delay.**
- **Understand the pros/cons involved.**
- **Make any recommendations on required changes to TSN.**
- **Initial focus on Scheduled traffic (802.1Qbv) simplified assumptions. I.e. perfect frequency sync, ns level scheduling, 1 Q per CPRI connection & no background traffic.**
- **Ongoing – will add more and more reality (eg card2card delay/jitter, clock errors, encoding delays).**

Network Simulator (NS-3)

- **A discrete-event network simulator, targeted primarily for research and educational use.**
- **Free software, licensed under the GNU GPLv2 license, and publicly available at http://www.nsnam.org/**
- **Clean slate design (not based on NS-2), aiming for easy use and extension. Written entirely in c++ with Python wrappers.**
- **NS-3 models used in our simulation include, but not limited to: core, network, Internet, UDP Echo Application, and Flow Monitor.**


SIMULATED FRONTHAUL NETWORK
2.45
4.91
9.83
Qbv
20g
1km
40g
2km
200g
5km
400g
5km
600g
5km
C-RAN
CPRI Connections PDU size = 1000 bytes

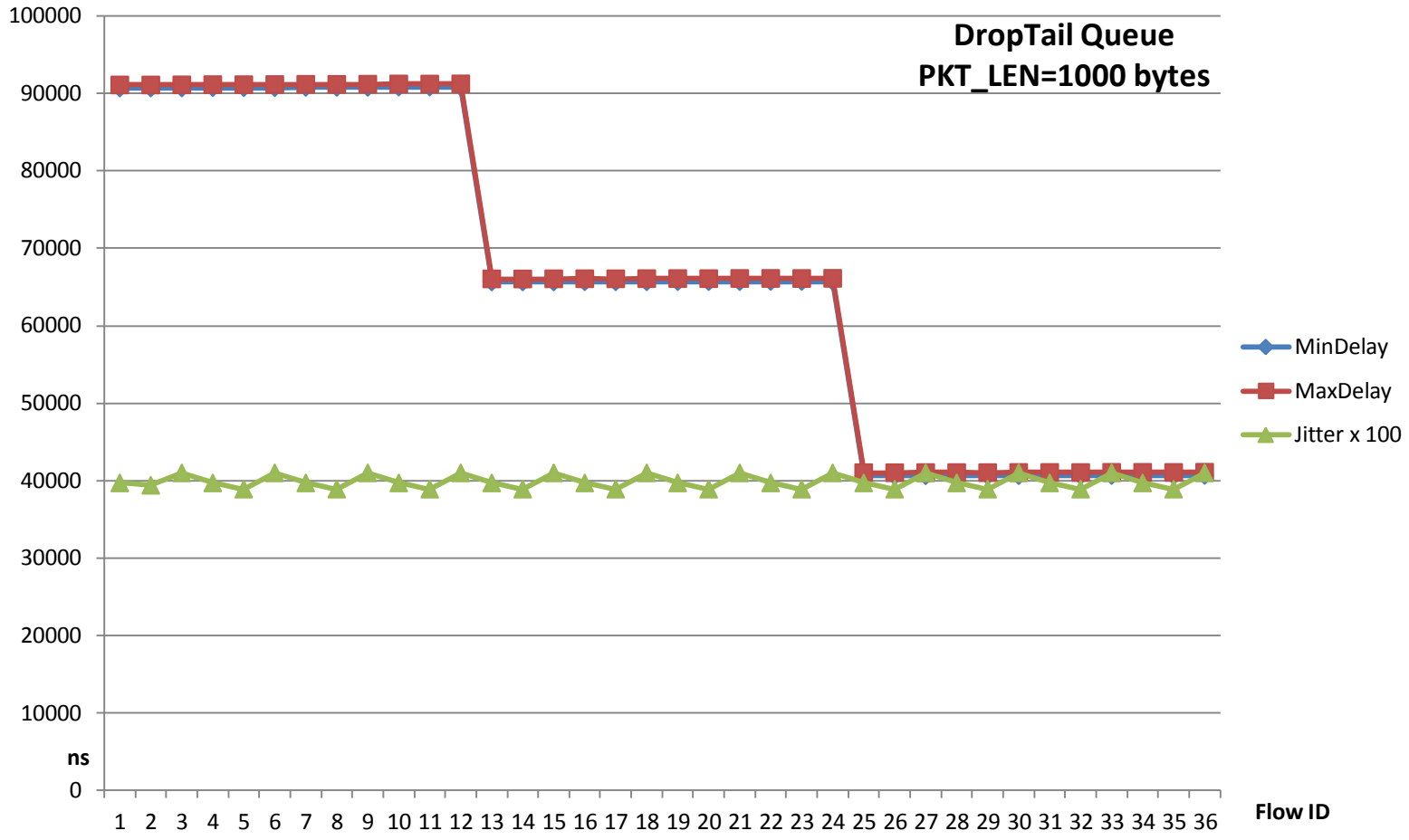
Queue models in our simulation

- **We use two queue models, namely *DropTail queue* and *Scheduled Queue*, to study the jitter behavior of existing Ethernet dropTail Q's and a first crude model of the proposed Qbv scheduled Q's.**

- **DropTail queue comes with NS-3. It is a first-in-first-out queue that drops packets at the tail. Note a couple of other queue models are also available in NS-3, but not tested since they primarily aim at improving TCP performance. Our testing uses a UDP application with perfect rates to simulate three CPRI speeds into a single DropTail queue at each egress link.**

- **Scheduled queues in our implementation was totally local, no attempt to phase synch (+ delay)between neighbors, just frequency locked (i.e. jitter reduction attempt, not delay reduction).**

Delay & Jitter for DropTail Queue



Stable delays and jitters for all flows. Jitter is around 400ns.
Note date rates are 2.45, 4.91, and 9.83Gbps.

First Simple Local Scheduling Algorithm

- **One queue per data flow (i.e., no queue sharing among flows), and each is processed according to a schedule (per port)**

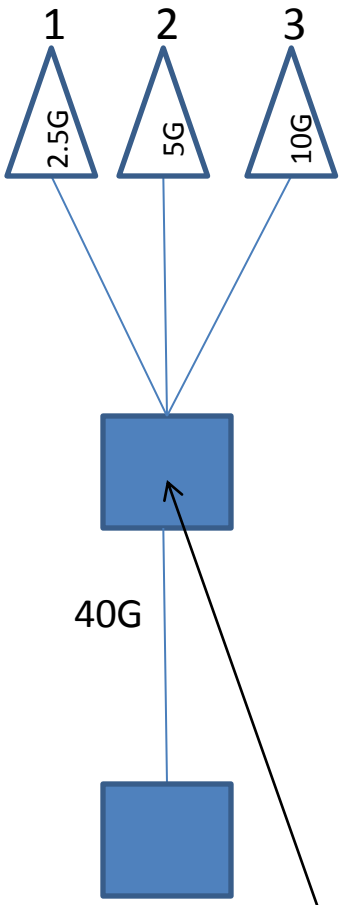
- **A schedule has a fixed cycle of constant duration, equivalent to the least common multiplier of packet transmission intervals of all flows, and repeats infinitely (until a new schedule is calculated and activated).**

- **For example, if a packet length is of 1000 bytes, a data flows of 5Gbps would have a packet transmission interval of 1600ns, and 8Gbps flow would have an interval of 1000ns. In this case, the schedule cycle is of 8000ns. Our simulation uses three data rates of 2.5Gbps, 5Gbps, and 10Gbps. Thus, the schedule cycle is of 3200ns for a packet length of 1K bytes.**

- **A schedule cycle is divided into a number of time slots of equal size. Each slot is at least equal to the packet transmission time of the outgoing link. (see next page for example).**

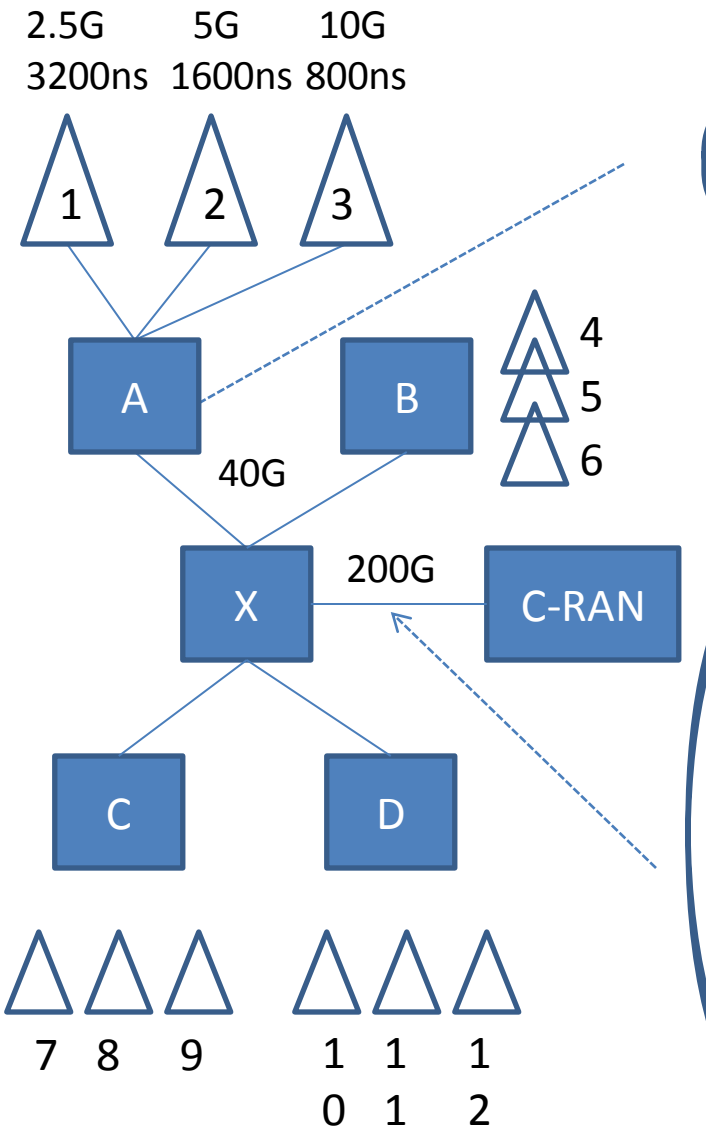
- **Each schedule is calculated independently for each port without any global optimization. The scheduling algorithm further assumes that the first packets from all flows arrive all at time 0 (i.e., when the schedule starts).**

A schedule example



- **Packet length = 1000 bytes**
- **Packet transmission intervals of 2.5Gbps, 5Gbps, and 10Gbps are of 3200ns, 1600ns, and 800ns respective (= packet_length*8/data rate).**
- **Schedule cycle = lcm(3200, 1600, 800) = 3200ns.**
- **Packet transmission time (ptt) for 40Gbps link is 200ns = (packet_length*8/link_speed).**
- **Slot size = k*ptt. For convenience, we choose k = 1.2 to make sure a slot is always enough for transmitting a packet . So the slot size = 240ns. Note for any k>1, (k-1)/k percent of bandwidth is lost.**
- **Total number of slots = schedule_cycle / slot size = 3200/240 = 13. We further add a non-used trailing slot of 80ns (= 3200 % 240). This results in further bandwidth loss.**
- **Below is a schedule for this 40G link:**

A schedule example (Cycle = 3200ns, packet=1000Bytes)



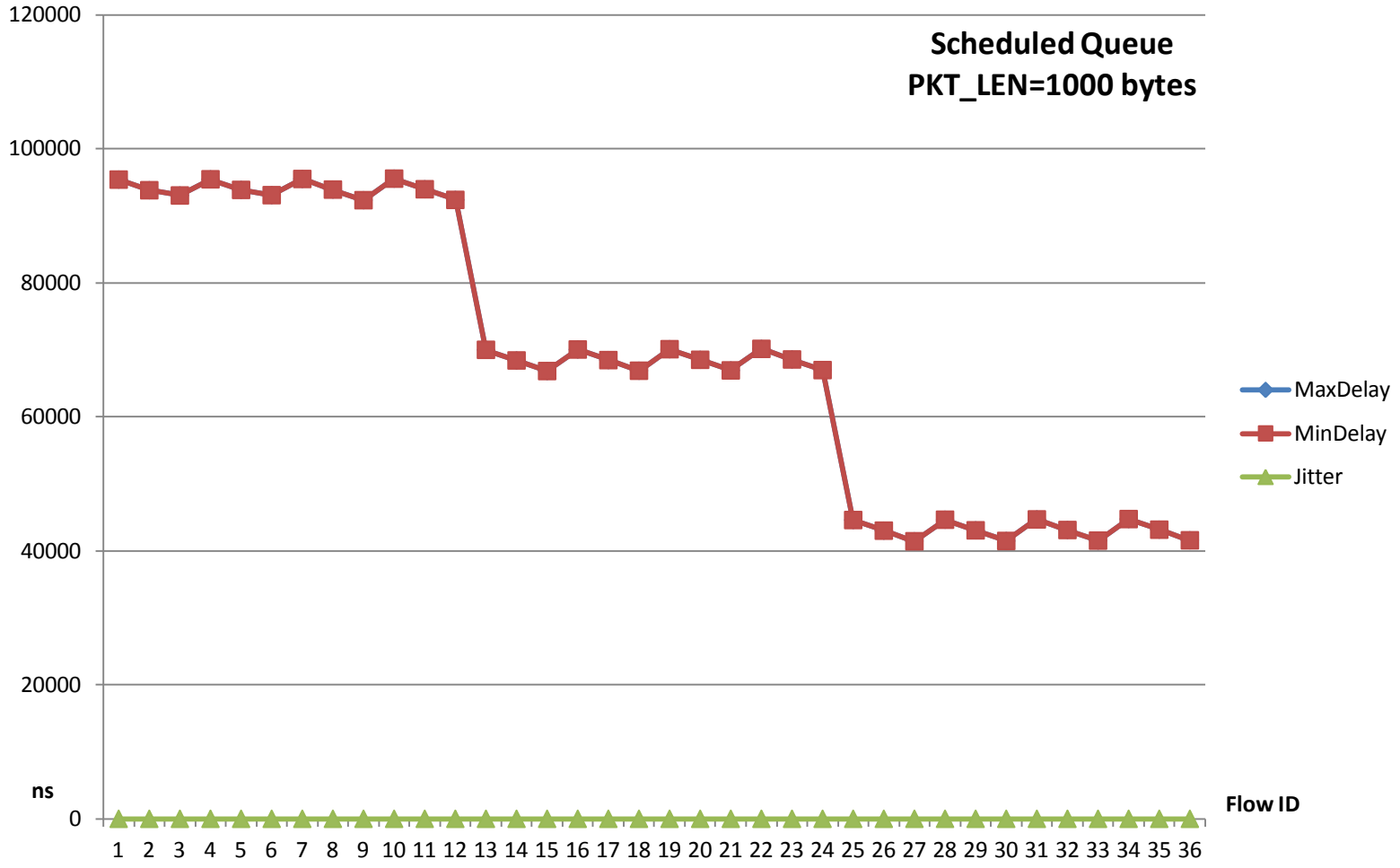
1	2	3	0	0	3	0	2	3	0	0	3	0								
1	2	3	0	0	3	0	2	3	0	0	3	0								

TotalSlots=13, slotSize = 240ns, trail=80ns

1	2	3	4	5	6	7	8	9	10	11	12	0	0	0	0	0	0	3	0	0	6
0	0	9	0	0	12	0	0	0	0	0	0	2	3	0	5	6	0	8	9	0	11
12	0	0	0	0	0	0	3	0	0	6	0	0	9	0	0	12	0	0	0	0	0
1	2	3	4	5	6	7	8	9	10	11	12	0	0	0	0	0	0	3	0	0	6
0	0	9	0	0	12	0	0	0	0	0	0	2	3	0	5	6	0	8	9	0	11
12	0	0	0	0	0	0	3	0	0	6	0	0	9	0	0	12	0	0	0	0	0

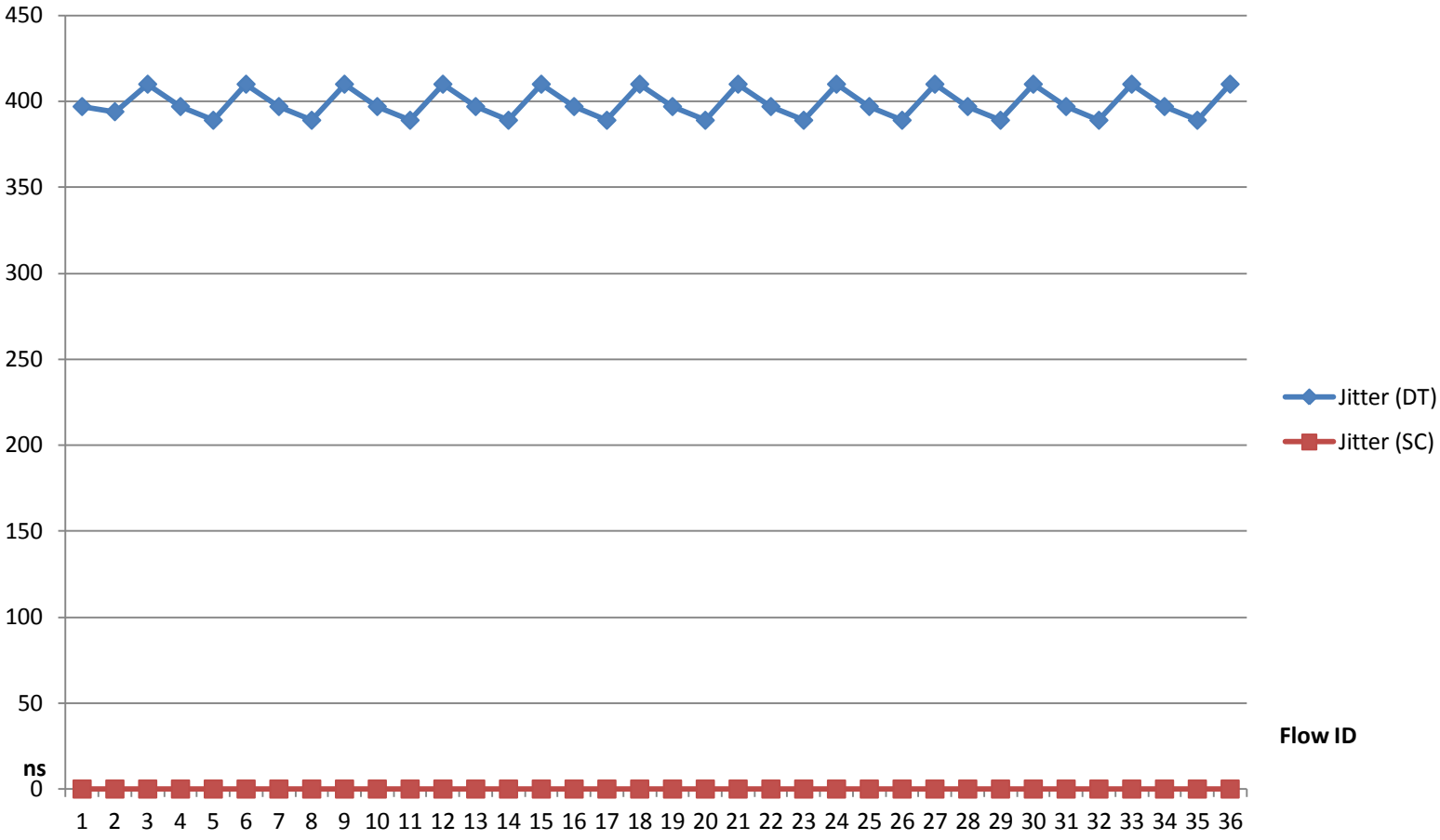
TotalSlots=66, slotSize = 48ns, trail=32ns

Delay & Jitter for Scheduled Queue



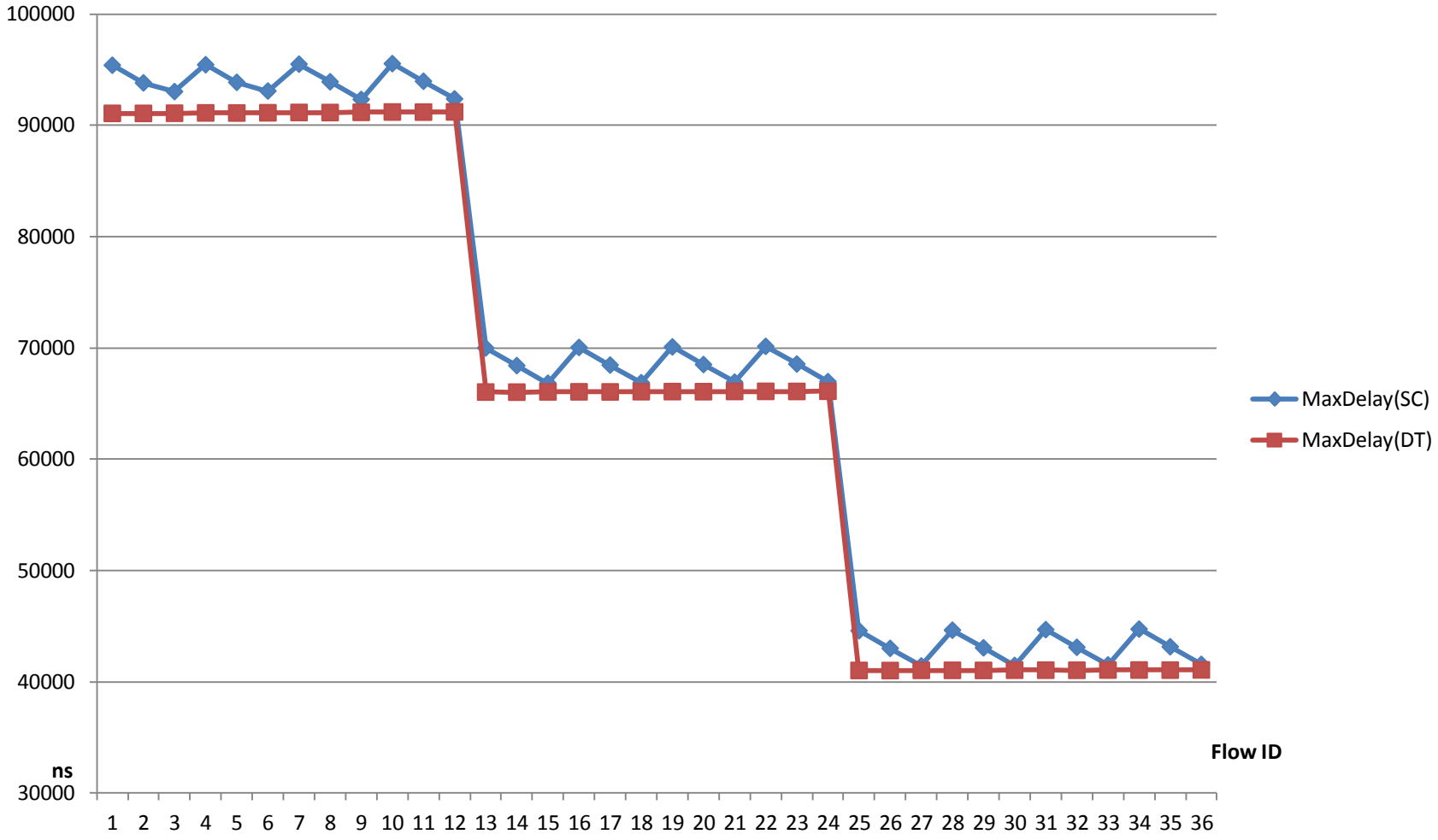
All jitters are reduced to ~zero (k=1.2).
Note date rates are 2.5, 5.0, and 10.0Gbps.

Jitter Comparison



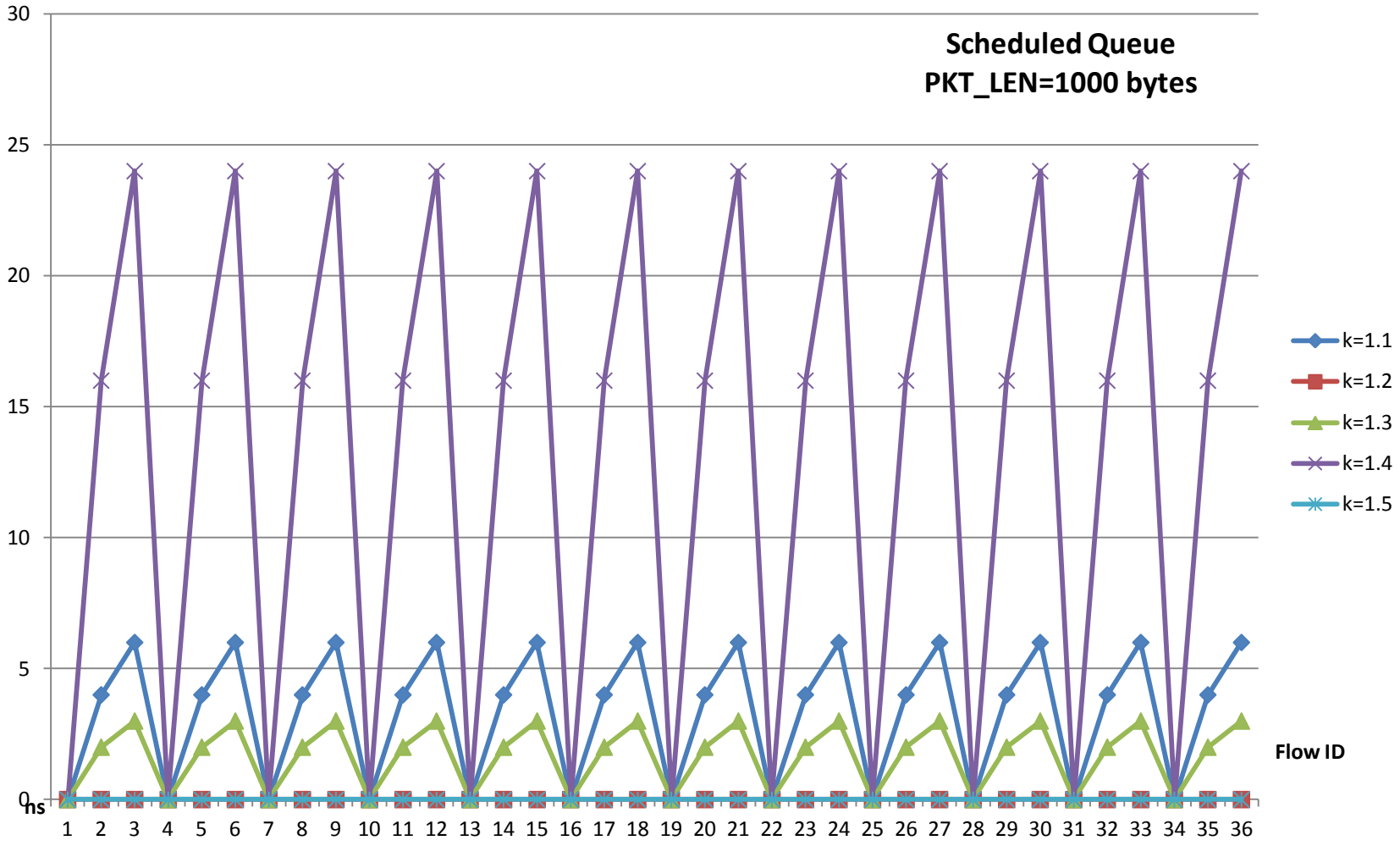
Scheduled queue reduces all jitters to ~zero (the best case).

Maximum Delay Comparison



Scheduled queue increases delay by 4347ns in the worst case.

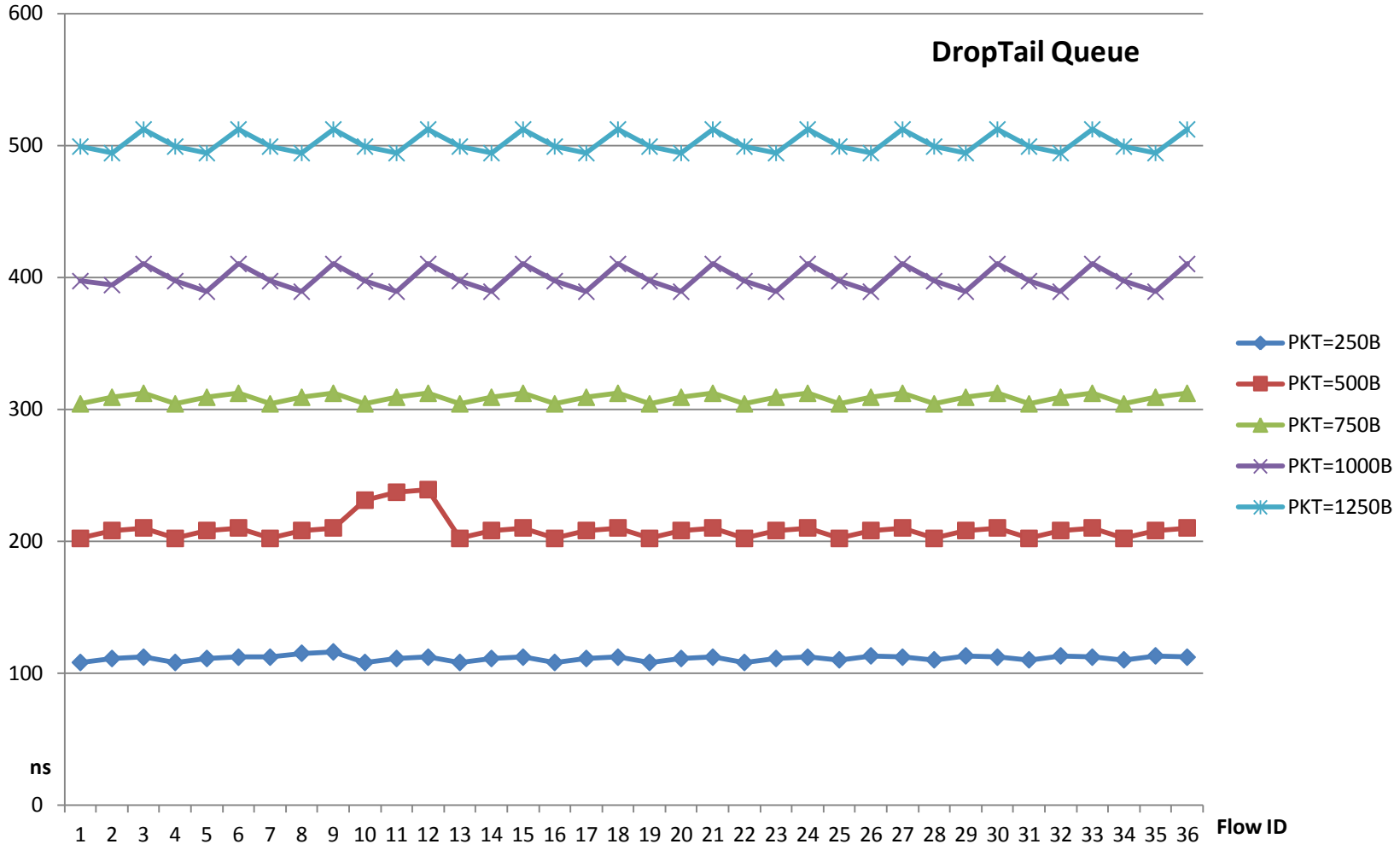
Jitter for Scheduled Queue



All jitters are zero when k=1.2, or 1.5.
When k=1.1, 1.3, or 1.4, small amount of jitters appear.

Jitters for different packet sizes

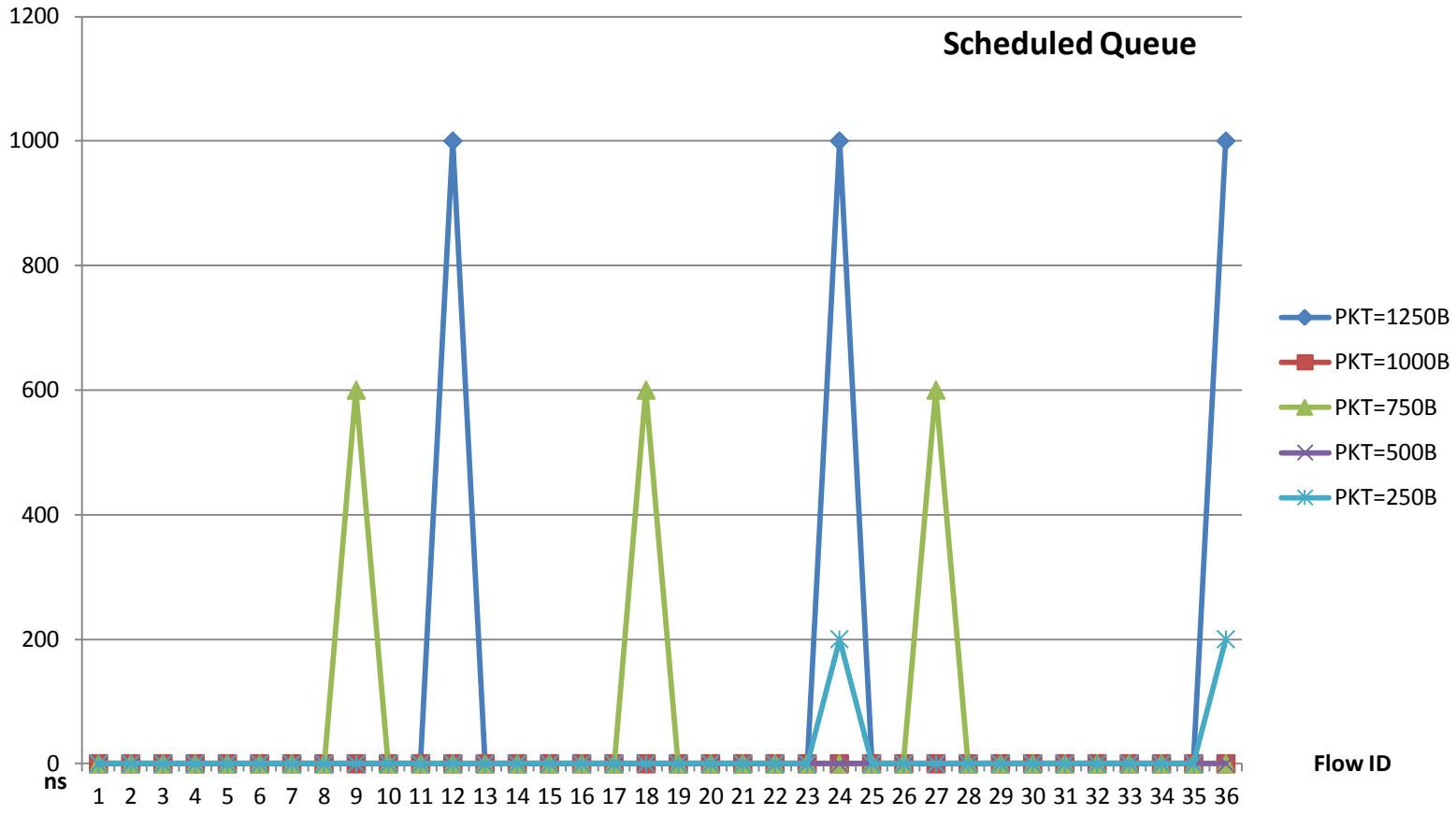
(DropTail Queue)



Jitters increase when packets get larger.

Jitters for different packet sizes

(Scheduled Queue)



All jitters are zero when packets are of 500Bytes and 1000Bytes.
When packets are of other sizes, some flows do have nonzero jitter.

Some Thoughts

Single DropTail queue seem to incur small amounts of jitter, and perform consistently with different packet sizes. If background traffic is pre-emptable then worst case would be an additional 64-128 bits of jitter (about 1.5-3ns at 40G).

Scheduled queues have potential to nearly eliminate jitter, but require more effort in developing advanced scheduling algorithms. Especially difficult globally.

More than 8 scheduled queues will be required if we are to use this method with CPRI however we still have to explore the gates concept…

Some bandwidth will be wasted matching the input speeds to the scheduled time slots and intervals.

Thank You