Switch Testing with IETF draft for Some Congestion Experienced –SCE TCP stack

July 31th 2019

Contacts

Giuseppe.scaglione@hpe.com

Chuck.Tuffli@hpe.com

Table of Contents

Introduction

This paper describes testing results using a real hardware based Ethernet switch connected directly to Linux servers with TCP Kernel drivers supporting the ietf draft called Some Congestion Experience (SCE) ECN Code point.

The draft can be found at: *https://tools.ietf.org/html/draft-morton-taht-tsvwg-sce-00*.

The source code for the Kernel drivers can be found at: https://github.com/chromi/sce

Tests were done emulating the switch marking "SCE" on the IP header if congestion was experienced, and also with traditional "CE" marking per RFC 3168, and the "dctcp-sce" or "reno-sce" TCP flavors configured on the Linux servers.

The authors look forward to ongoing work in this area on related and/or alternative technologies. Tests were only repeated a few times. The authors seek feedback to the methods and results contained herein.

Test setup

Four HPE ProLiant-DL380-Gen10 servers with 25G Ethernet NICs connected to an Aruba HPE switch. Those are powerful multi cores servers that can easily produce 25Gbps of traffic.

The switch was bridging across the four 25G Ethernet ports, and it was configured to remark all IP traffic ECN capable to ECN= "11" (Congestion Experienced CE) with a linear remark probability proportional to the queue depth (the probability of remark increments linearly with the percentage of queue in use) and a "start remarking" point set to "anything greater than zero queueing". The start-remarking-point was changed in some CE tests to different values, while the remarking probability was kept linear for all tests. The following figure shows the switch queue congestion management.

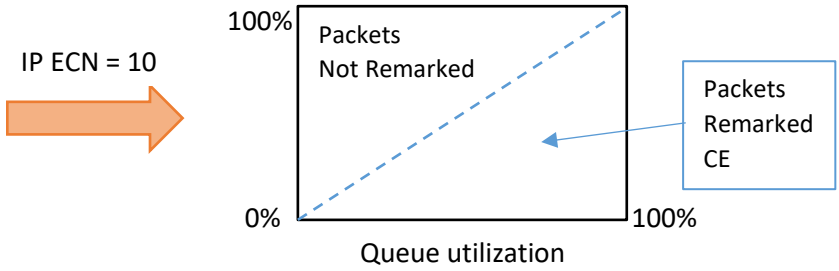


Fig 1. Linear Remark probability starting at greater than 0% queue utilization

On other tests the start-remarking point was set at higher values giving the following representation:

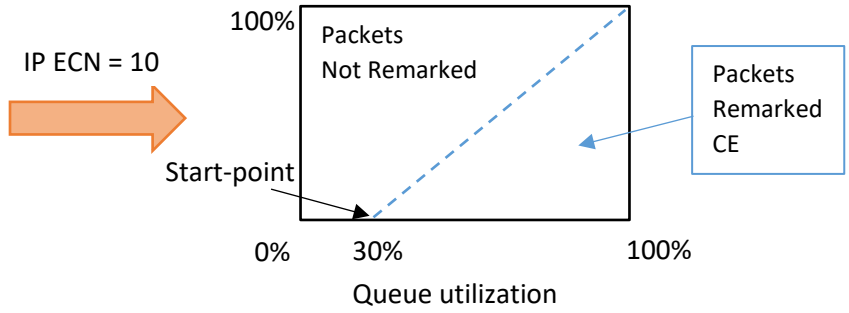


Fig 2. Linear Remark probability starting at 30% of queue

The remark probability is given by the following equation:

*remark_probability = probability_rise * (current_mem_utilization - mem_start_point)*

During testing the *probability_rise* was fixed at 1% for all tests, and the *mem_start_point* was 0 for SCE testing, and moved at 10% and 30% for some CE testing.

Server setup

Servers loaded with TCP-SCE capable Kernel running Ubuntu 18.04.2 LTS.
System settings:
On senders: net.ipv4.tcp_ecn = 1
On receivers: net.ipv4.tcp_sce = 1
Available TCP congestion control: reno, cubic-sce, dctcp-sce, reno-sce, dctcp, cubic.

In the SCE testing the server receiver was configured to change the TOS bits from 11 to 01 to emulate the switch in the middle setting “01” for congestion. (iptables -t mangle -I INPUT -p tcp -m tos --tos 0x03 -j TOS --set-tos 0x01)

Topology

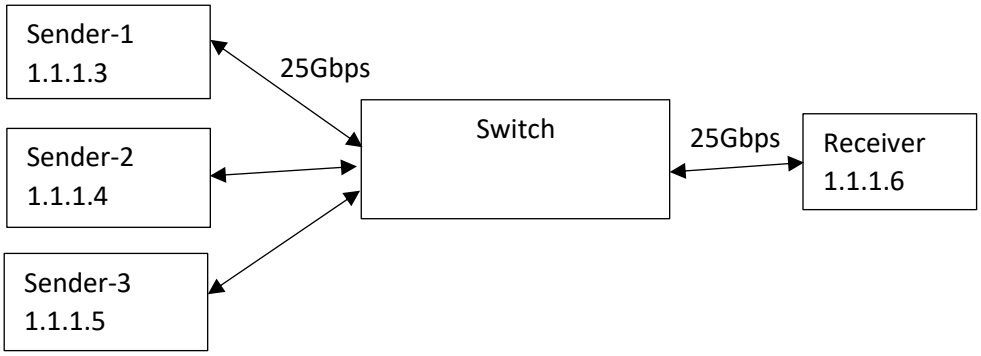


Fig 3. Network Topology

25Gbps DAC (Direct Attached Cables) 3m connecting the four servers to the switch in an isolated VLAN without any other traffic. The links MTU was left default (1500 bytes).

Iperf3 Testing

The Iperf3 application was used with 1.1.1.6 in server mode and the other machines in client mode.

Receiver commands:
iperf3 -s -p 5101&
iperf3 -s -p 5102&
iperf3 -s -p 5103&
Senders commands:
iperf3 -c 1.1.1.6 -p 510X –t <duration-seconds>, // X = 1,2,3

SCE Testing

On the receiver: *iptables -t mangle -I INPUT -p tcp -m tos --tos 0x03 -j TOS --set-tos 0x01*

One to one iperf3 data transfer no oversubscription.

Source	Destination	Duration (seconds)	Bandwidth (Gbits/sec)	Data Transfer (Gbytes)	TCP Retries	Switch Packet Drops	Switch ECN	Algo
1.1.1.3	1.1.1.6	20	23.3	54.3	0	0	ON	reno-sce
1.1.1.3	1.1.1.6	20	23.4	54.2	0	0	OFF	reno-sce
1.1.1.3	1.1.1.6	20	23.5	54.6	0	0	OFF	dctcp-sce
1.1.1.3	1.1.1.6	20	23.3	54.3	0	0	ON	dctcp-sce

Table 1. No oversubscription

The link was fully utilized achieving almost 100% throughput (40 bytes IPv4 header on 1500 MTU).

Two to one iperf3 data transfer.

Each pair of servers was sending to the same destination, one in continuous data transfer, and one for 20 seconds.

Source	Destination	Duration (seconds)	Bandwidth (Gbits/sec)	Data Transfer (Gbytes)	TCP Retries	Switch Packet Drops	Switch ECN	Algo
1.1.1.4	1.1.1.6	Cont.					OFF	reno-sce
1.1.1.3	1.1.1.6	20	10.9	25.4	8767	14930	OFF	
1.1.1.4	1.1.1.6	Cont.					OFF	dctcp-sce
1.1.1.3	1.1.1.6	20	11.4	26.5	9321	14551	OFF	
1.1.1.4	1.1.1.6	Cont.					ON	reno-sce
1.1.1.3	1.1.1.6	20	11.0	25.7	0	1	ON	
1.1.1.4	1.1.1.6	Cont.					ON	dctcp-sce
1.1.1.3	1.1.1.6	20	12.5	29.1	0	0	ON	

Table 2. 2:1 oversubscription with SCE

Three to one iperf3 data transfer

Three servers sending to one. Two in continuous data transfer, and one for 20 seconds.

Source	Destination	Duration (seconds)	Bandwidth (Gbits/sec)	Data Transfer (Gbytes)	TCP Retries	Switch Packet Drops	Switch ECN	Algo
1.1.1.3	1.1.1.6	20	7.44	17.3	14313	46,145	OFF	reno-sce
1.1.1.4	1.1.1.6	Cont.					OFF	
1.1.1.5	1.1.1.6	Cont.					OFF	
1.1.1.3	1.1.1.6	20	7.9	18.4	12557	36,405	OFF	dctcp-sce
1.1.1.4	1.1.1.6	Cont.					OFF	
1.1.1.5	1.1.1.6	Cont.					OFF	
1.1.1.3	1.1.1.6	20	8.22	19.1	0	1	ON	reno-sce
1.1.1.4	1.1.1.6	Cont.					ON	
1.1.1.5	1.1.1.6	Cont.					ON	
1.1.1.3	1.1.1.6	20	7.79	18.1	0	0	ON	dctcp-sce
1.1.1.4	1.1.1.6	Cont.					ON	
1.1.1.5	1.1.1.6	Cont.					ON	

Table 3. 3:1 oversubscription with SCE

Iperf3 CE testing

The Below tables show testing done without the system mangle table change. Hence, packets were seen as marked CE “11” at the receiver, with the same probability curve as before, and still starting remarking at anything greater than zero of memory utilization.

CE testing two-to one, remarking starting at greater than 0%

Source	Destination	Duration (seconds)	Bandwidth (Gbits/sec)	Data Transfer (Gbytes)	TCP Retries	Switch Packet Drops	Switch ECN	Algo
1.1.1.4	1.1.1.6	Cont.					ON	reno-sce
1.1.1.3	1.1.1.6	20	4.83	11.3	0	0	ON	

Table 4. 2:1 oversubscription with CE marking starting at 0%

Bandwidth went down from 11 Gb/sec per host to 4.83 Gb/sec per host, suggesting that the TCP stack reacted more drastically with CE indications on the wire.

CE testing three-to-one, remarking starting at greater than 0%

Source	Destination	Duration (seconds)	Bandwidth (Gbits/sec)	Data Transfer (Gbytes)	TCP Retries	Switch Packet Drops	Switch ECN	Algo
1.1.1.3	1.1.1.6	20	4.56	10.6	0	0	ON	reno-sce
1.1.1.4	1.1.1.6	Cont.					ON	
1.1.1.5	1.1.1.6	Cont.					ON	

Table 5. 3:1 oversubscription with CE

Bandwidth in the 3:1 test stays similar to above, with an overall link utilization of around 65%.

CE testing with remarking started at 30% of the queue

In this test, the switch was configured to start CE remarking at 30% of the queue buffering and keep a linear remarking curve.

CE testing at 30% start: two-to-one

Source	Destination	Duration (seconds)	Bandwidth (Gbits/sec)	Data Transfer (Gbytes)	TCP Retries	Switch Packet Drops	Switch ECN	Algo
1.1.1.4	1.1.1.6	Cont.					ON	reno
1.1.1.3	1.1.1.6	20	11.6	27.0	35	31	ON	
1.1.1.4	1.1.1.6	Cont.					ON	reno-sce
1.1.1.3	1.1.1.6	20	11.6	27.0	0	0	ON	

Table 6. 2:1 oversubscription with CE, remarking starting at 30%

CE testing at 30% start: three-to-one

Source	Destination	Duration (seconds)	Bandwidth (Gbits/sec)	Data Transfer (Gbytes)	TCP Retries	Switch Packet Drops	Switch ECN	Algo
1.1.1.3	1.1.1.6	20	7.38	17.2	0	0	ON	reno
1.1.1.4	1.1.1.6	Cont.					ON	
1.1.1.5	1.1.1.6	Cont.					ON	
1.1.1.3	1.1.1.6	20	7.98	18.6	0	0	ON	reno-sce
1.1.1.4	1.1.1.6	Cont.					ON	
1.1.1.5	1.1.1.6	Cont.					ON	

Table 7. 3:1 oversubscription with CE, remarking starting at 30%

With traditional CE remarking and starting at 30% of the queue the bandwidth and throughput observed was similar than with SCE remarking starting at 0%.

CE and SCE testing long Iperf3 runs

Two minutes test run with three-to-one oversubscription and dctcp-sce congestion management. The three senders were started almost at the same time.

On 1.1.1.3: iperf3 -c 1.1.1.6 -p 5101 -t 120
On 1.1.1.4: iperf3 -c 1.1.1.6 -p 5102 -t 120
On 1.1.1.5: iperf3 -c 1.1.1.6 -p 5103 -t 120

SCE remarking with linear slope starting at 0%, two minutes run

Same test as before but longer runs.

Source	Destination	Duration (seconds)	Bandwidth (Gbits/sec)	Data Transfer (Gbytes)	TCP Retries	Switch Packet Drops	Switch ECN	Algo
1.1.1.3	1.1.1.6	120	7.53	105	366	0	ON	dctcp-sce
1.1.1.4	1.1.1.6	120	7.74	113	0	0	ON	dctcp-sce
1.1.1.5	1.1.1.6	120	8.11	108	0	0	ON	dctcp-sce
	TOTAL		23.38					

Table 8. 3:1 oversubscription with SCE, 2 minutes

All 366 TCP retries occurred at the very first second of testing on one host. Usually they occurred when the senders started very close to each other. Other runs showed same initial burst of retries and then 0 if the servers started talking right away.

CE remarking with linear slope starting at 30%, two minutes run

Source	Destination	Duration (seconds)	Bandwidth (Gbits/sec)	Data Transfer (Gbytes)	TCP Retries	Switch Packet Drops	Switch ECN	Algo
1.1.1.3	1.1.1.6	120	7.92	111	0	0	ON	dctcp-sce
1.1.1.4	1.1.1.6	120	8.05	112	0	0	ON	dctcp-sce
1.1.1.5	1.1.1.6	120	7.75	108	0	0	ON	dctcp-sce
	TOTAL		23.72					
1.1.1.3	1.1.1.6	120	7.9	110	14	26	ON	reno-sce
1.1.1.4	1.1.1.6	120	7.94	111	0	0	ON	reno-sce
1.1.1.5	1.1.1.6	120	7.85	110	0	0	ON	reno-sce
	TOTAL		23.69					

Table 9. 3:1 oversubscription with CE at 30%, 2 minutes run

CE remarking with linear slope starting at 10%, two minutes run

Source	Destination	Duration (seconds)	Bandwidth (Gbits/sec)	Data Transfer (Gbytes)	TCP Retries	Switch Packet Drops	Switch ECN	Algo
1.1.1.3	1.1.1.6	120	7.83	109	0	0	ON	dctcp-sce
1.1.1.4	1.1.1.6	120	8.04	112	0	0	ON	dctcp-sce
1.1.1.5	1.1.1.6	120	7.62	106	0	0	ON	dctcp-sce
	TOTAL		23.49					
1.1.1.3	1.1.1.6	120	7.83	109	72	0	ON	reno-sce
1.1.1.4	1.1.1.6	120	7.92	108	22	0	ON	reno-sce
1.1.1.5	1.1.1.6	120	7.71	111	46	0	ON	reno-sce
	TOTAL		23.46					

Table 10. 3:1 oversubscription with CE at 10%, 2 minutes run

CE remarking with linear slope starting at 0%, two minutes run

Source	Destination	Duration (seconds)	Bandwidth (Gbits/sec)	Data Transfer (Gbytes)	TCP Retries	Switch Packet Drops	Switch ECN	Algo
1.1.1.3	1.1.1.6	120	4.6	65	0	0	ON	dctcp-sce
1.1.1.4	1.1.1.6	120	4.6	64	0	0	ON	dctcp-sce
1.1.1.5	1.1.1.6	120	4.9	68	0	0	ON	dctcp-sce
	TOTAL		14.1					
1.1.1.3	1.1.1.6	120	4.53	63.2	86	0	ON	reno-sce
1.1.1.4	1.1.1.6	120	4.88	68.2	97	0	ON	reno-sce
1.1.1.5	1.1.1.6	120	4.72	66	113	0	ON	reno-sce
	TOTAL		14.13					

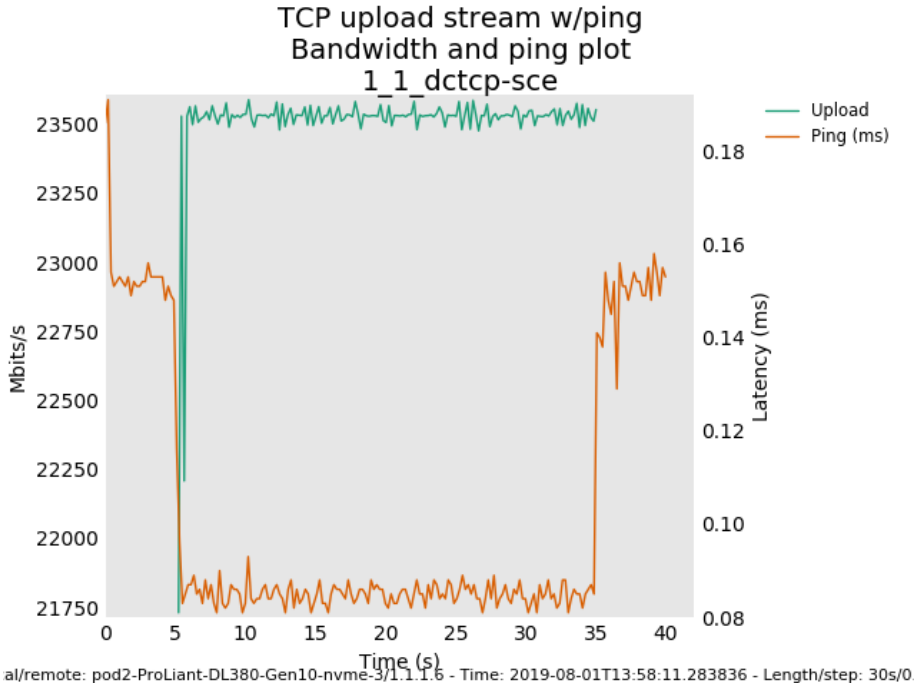
Table 11. 3:1 oversubscription with CE at 0%, 2 minutes run

As previously noticed, with CE remarking starting too early in the queue depth the link utilization drops considerably.

Netperf/Flent SCE Testing

System 6 (1.1.1.6) has *netserver* running and it was the receiver. Test used “tcp_upload”, command: flent tcp_upload -p totals -l 30 -H 1.1.1.6

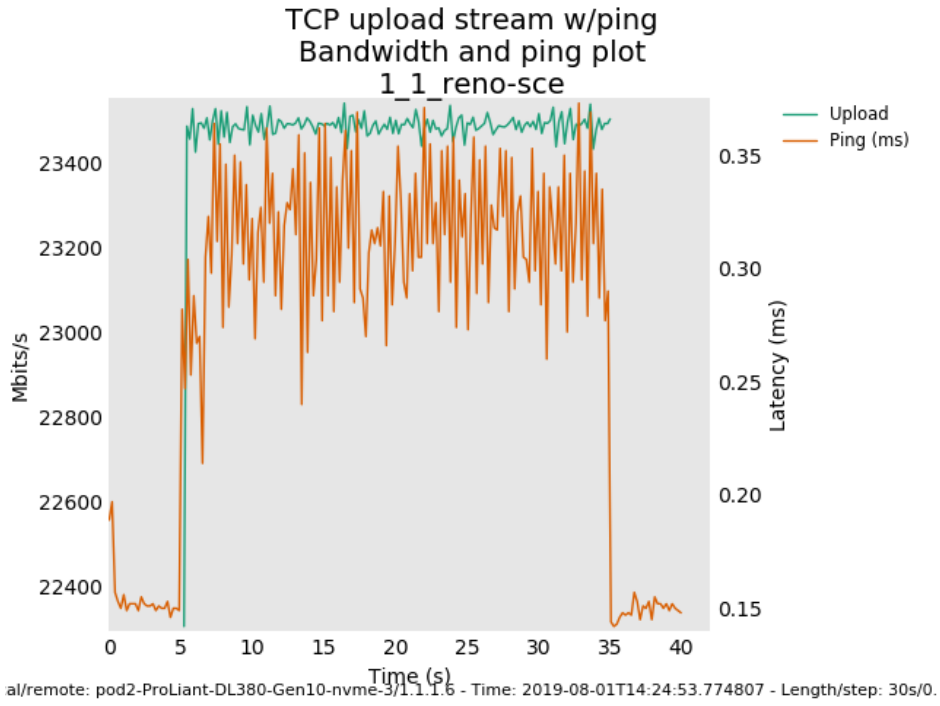
TCP Upload 1:1 no oversubscription 1.1.1.3 to 1.1.1.6 DCTCP-SCE



Pic. 1. TCP upload no oversubscription, dctcp

No Drops observed (no oversubscription).

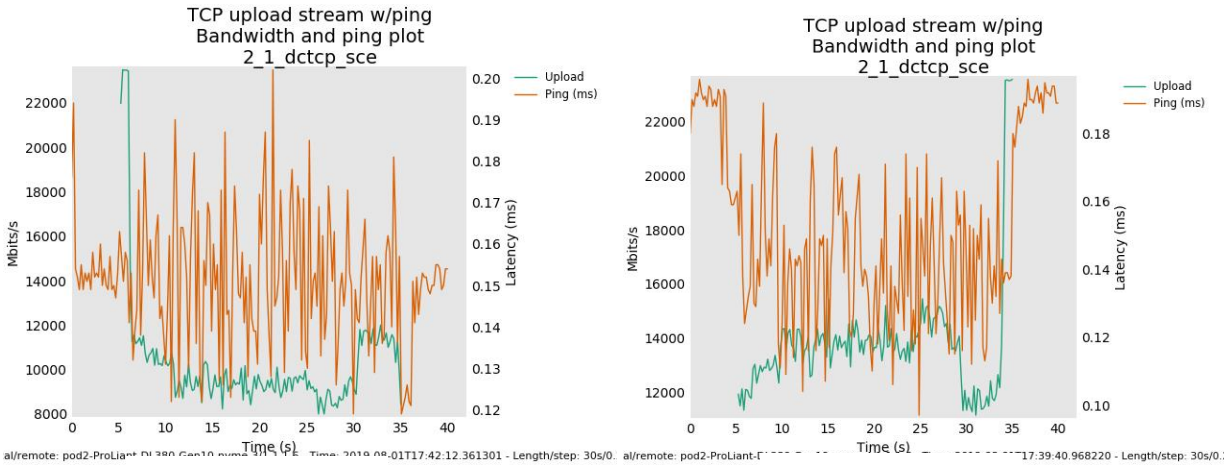
TCP Upload 1:1 no oversubscription 1.1.1.3 -> 1.1.1.6 RENO-SCE



Pic. 2. TCP upload no oversubscription, reno-sce

TCP Upload 2:1 Oversubscribed 1.1.1.3, 1.1.1.4 -> 1.1.1.6 DCTCP-SCE

The 30 seconds run were started within 1 second from each other.

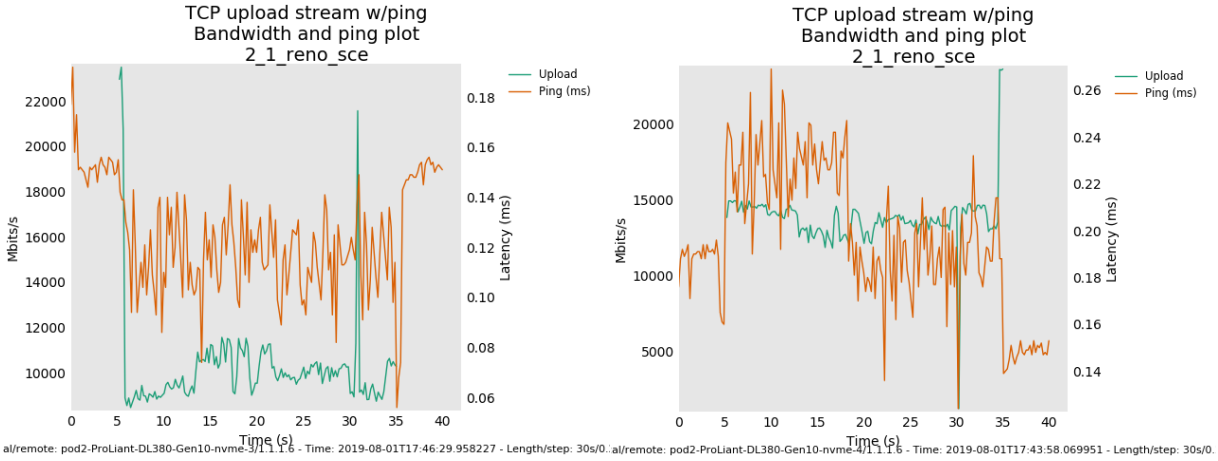


1.1.1.3 to 1.1.1.6 **1.1.1.4 to 1.1.1.6**

Pic. 3. 2:1 TCP upload dctcp-sce both senders

In the test above, sender 1.1.1.4 started half second after sender 1.1.1.3

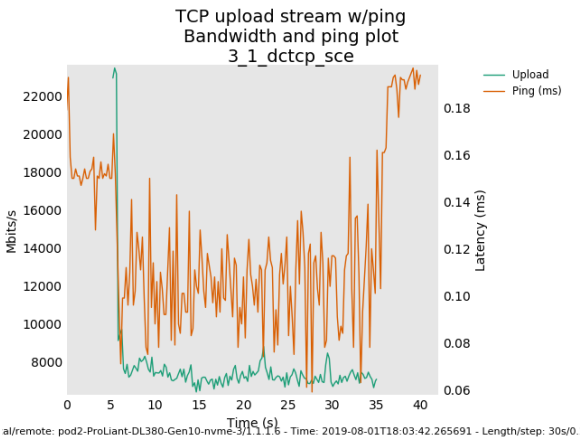
TCP Upload 2:1 Oversubscribed 1.1.1.3, 1.1.1.4 -> 1.1.1.6 RENO-SCE



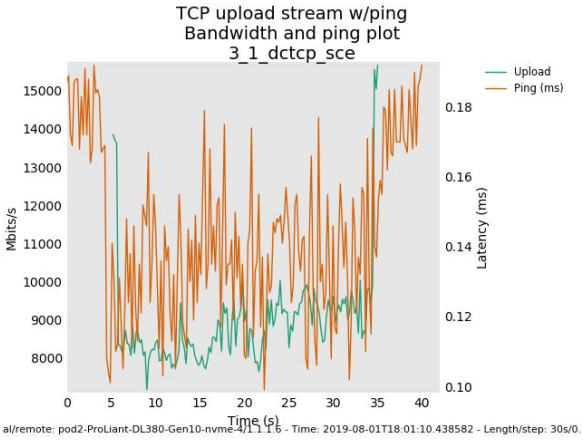
1.1.1.3 to 1.1.1.6 **1.1.1.4 to 1.1.1.6**

Pic. 4. 2:1 TCP upload reno-sce both senders

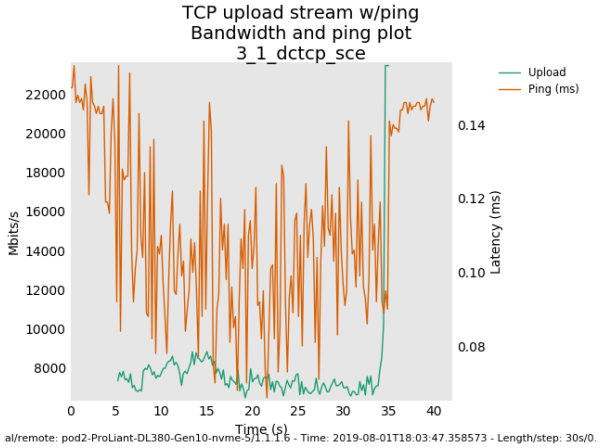
TCP Upload 3:1 Oversubscribed 1.1.1.3, 1.1.1.4, 1.1.1.5 -> 1.1.1.6 DCTCP-SCE



1.1.1.3 to 1.1.1.6



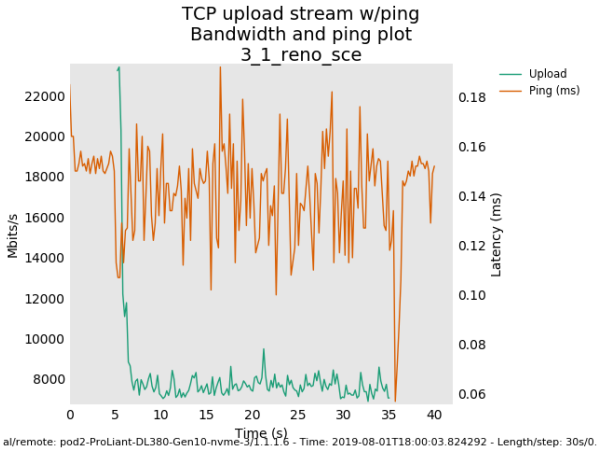
1.1.1.4 to 1.1.1.6



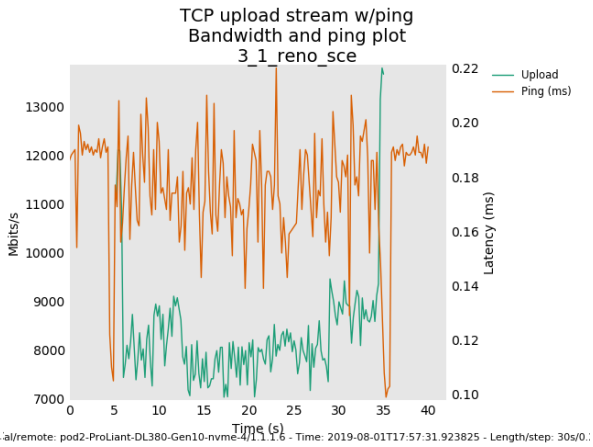
1.1.1.5 to 1.1.1.6

Pic. 5. 3:1 TCP upload dctcp-sce all three senders

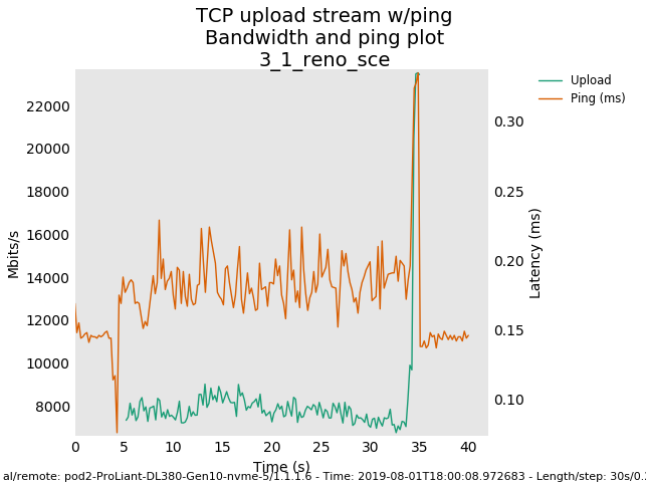
TCP Upload 3:1 Oversubscribed 1.1.1.3, 1.1.1.4, 1.1.1.5 -> 1.1.1.6 RENO-SCE



1.1.1.3 to 1.1.1.6



1.1.1.4 to 1.1.1.6



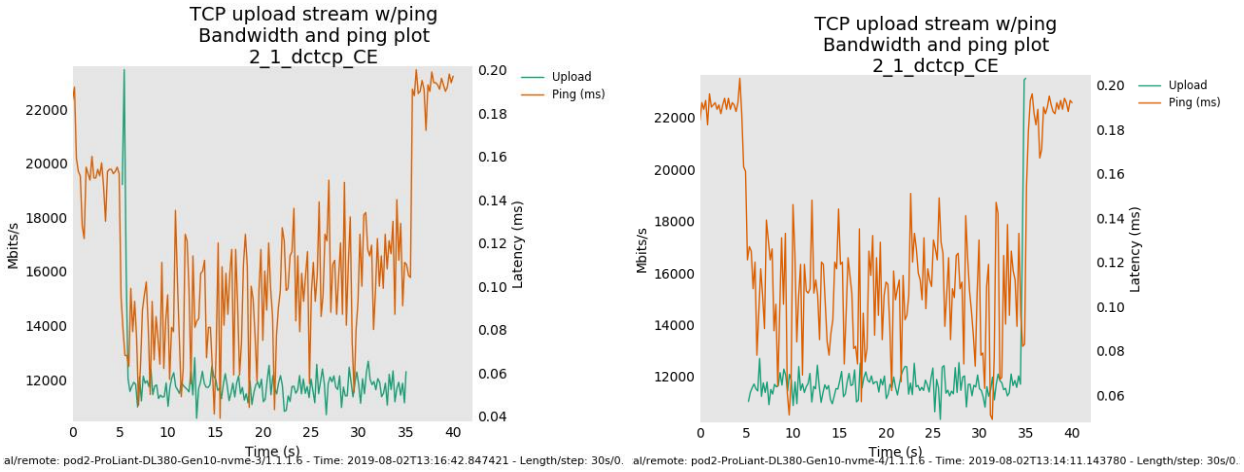
1.1.1.5 to 1.1.1.6

Pic. 6. 3:1 TCP upload reno-sce all three senders

Netperf/Flent CE Testing

Flent testing with just CE remarking and starting at 10% queueing start-point for remarking.

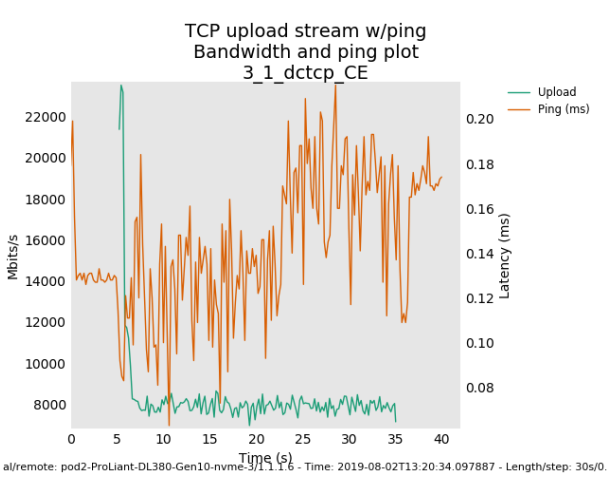
TCP Upload 2:1 Oversubscribed 1.1.1.3, 1.1.1.4 -> 1.1.1.6 DCTCP-SCE CE marking



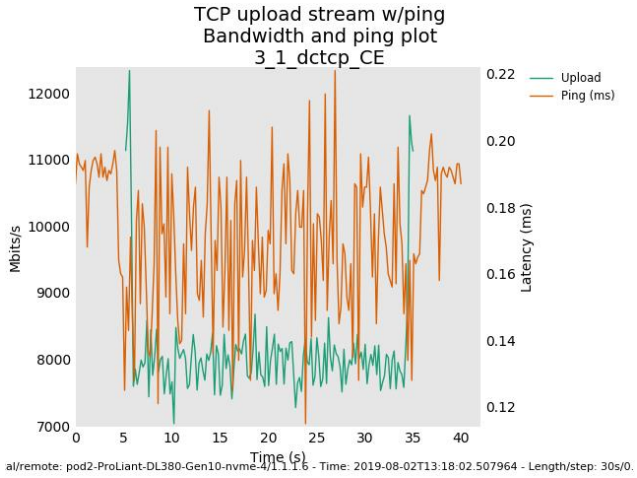
1.1.1.3 to 1.1.1.6 **1.1.1.4 to 1.1.1.6**

Pic. 7. 2:1 TCP upload CE remarking-10%-start

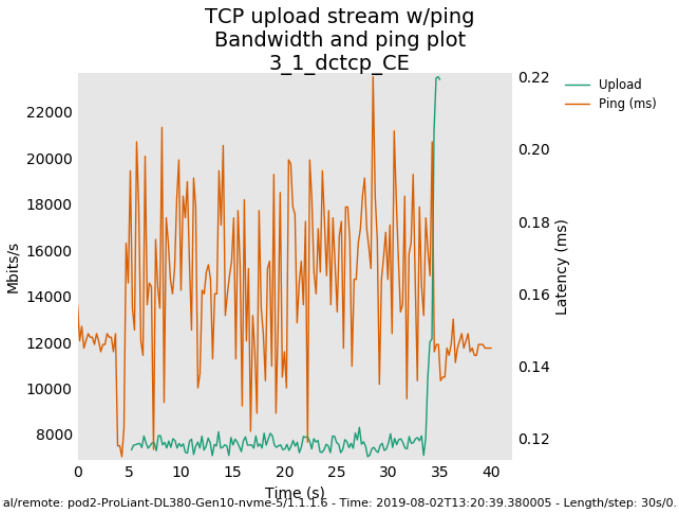
TCP Upload 2:1 Oversubscribed 1.1.1.3, 1.1.1.4, 1.1.1.5 -> 1.1.1.6 DCTCP-SCE CE Marking



1.1.1.3 to 1.1.1.6



1.1.1.4 to 1.1.1.6



1.1.1.5to 1.1.1.6

Pic. 8. 3:1 TCP upload CE remarking-10%-start

Flent Testing packet drops at the switch

Same flent test as above (tcp_upload), run with SCE and CE remarking and monitoring the packet drops at the switch ports.

Test	Packet drop at the switch
2:1 dctcp-sce SCE	5, 3
2:1 reno-sce SCE	45, 20
2:1 dctcp-sce CE 30%	3
2:1 reno-sce CE 30%	0
3:1 dctcp-sce SCE	11
3:1 reno-sce SCE	160, 198
3:1 dctcp-sce CE 30%	32
3:1 reno-sce CE 30%	0
2:1 reno-sce CE 10%	2
3:1 reno-sce CE 10%	0
2:1 dctcp-sce CE 10%	0
3:1 dctcp-sce CE 10%	0

Table 12. Flent testing packet drops

Some run were done 2 times to confirm, comma separated above.

Notes

Test runs were done only once or twice, and runs vary around 10~20%

Observations

The most noticeable result with Iperf3 testing was the very low number (almost zero) of packets dropped by the switch when the device was doing ECN remarking -- even with a 3:1 –almost 75 Gbps to 25 Gbps oversubscription. This seems to signify that the switch was properly marking the traffic proportionally of the queue depths and the TCP-SCE algorithm was able to adjust the rates in time. Another noticeable result was the low numbers of TCP retries when ECN was on, either CE or SCE remarking.

Since the servers were directly attached to the switch the TCP retries did not seem to impact that much on the overall performance. With a larger number of hops between sender and receiver, or overall greater distance between sender and receiver, the number of packet drops and TCP retries would have caused a much greater impact on the overall performance.

It seems like CE marking with queue tuning (starting point at a specific percentage greater than 0%) achieved same performance result with Iperf3 than SCE marking, and even better performance results with the Flent/netperf when marking started at 10% of the queue depth.