Short and Fat: TCP Performance in CEE Datacenter Networks

Daniel Crisan, Andreea S. Anghel, Robert Birke, Cyriel Minkenberg and Mitch Gusat
IBM Research, Zürich Research Laboratory
Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland
{dcr,aan,bir,sil,mig}@zurich.ibm.com

***Abstract*—One of the consequential new features of emerging datacenter networks is losslessness, achieved by means of Priority Flow Control (PFC). Despite PFC's key role in the datacenter and its increasing availability – supported by virtually all Converged Enhanced Ethernet (CEE) products – its impact remains largely unknown. This has motivated us to evaluate the sensitivity of three widespread TCP versions to PFC, as well as to the more involved Quantized Congestion Notification (QCN) congestion management mechanism.**

As datacenter workloads we have adopted several representative commercial and scientific applications. For evaluation we employ an accurate Layer 2 CEE network simulator coupled with a TCP implementation extracted from FreeBSD v9. A somewhat unexpected outcome of this investigation is that PFC significantly improves TCP performance across all tested configurations and workloads, hence our recommendation to enable PFC whenever possible. In contrast, QCN can help or harm depending on its parameter settings, which are currently neither adaptive nor universal for datacenters. To the best of our knowledge this is the first performance evaluation of TCP performance in lossless CEE networks.

I. Introduction and Motivation

Upcoming datacenter networks based on 802 Converged Enhanced Ethernet (CEE) are short and fat: up to one million nodes are connected in a single Layer 2 domain with abundant multipathing across 10-100 Gbps links of a few tens of meters (at most). Typical round-trip times (RTTs) range from 1-2 μs up to a few tens of μs, except under hotspot congestion, when the end-to-end delays can grow by several orders of magnitude, reaching into tens of ms [1]. Unlike wide area networks, the datacenter RTT is dominated by queuing delays, which under bursty workloads [2], [3], [4], [5], [6], [7], [8], [9], lead to a difficult traffic engineering and control problem. Hence the recent surge in research and standardization efforts addressing the new challenges in datacenter virtualization, flow and congestion control, routing and high performance transports.

One of CEE core new features is Layer 2 (L2) losslessness, achieved via per priority link-level flow control as defined by 802.1Qbb PFC [10]. It enables convergence of legacy and future datacenter applications, such as Fibre Channel over Ethernet (FCoE), business analytics, low latency algorithmic trading, high performance network storage, and MPI workloads currently still running on Myrinet and InfiniBand networks. However, the benefits of PFC come at a price: besides the potential for deadlock in certain topologies and switch architectures, it introduces exposure to saturation tree congestion. To counteract the potentially severe performance degradation due to such congestion, IEEE has recently standardized a new L2 congestion control scheme, Quantized Congestion Notification (QCN, 802.1Qau) [11].

Since the bulk of datacenter communications is based on Layer 4 (L4) transports, i.e., predominantly TCP with some notable RDMA, SCTP and UDP exceptions, an overarching question is: How disruptive are the new CEE features? Are PFC and QCN, beneficial, neutral or detrimental for the typical datacenter application? Concretely, assuming that TCP will remain the paramount transport, we ask three TCP-related questions:

1) How does TCP perform over CEE networks?
2) Is PFC – with its potential saturation trees – beneficial or detrimental to TCP, which traditionally has relied on *loss* as congestion feedback from *uncorrelated single* bottlenecks?
3) Is QCN beneficial or detrimental to TCP?

In addressing these questions we hope to provide some useful guidance to datacenter architects, network vendors, as well as operating system, hypervisor and application designers. Our contributions are twofold, structured as follows:

a) Methodology: We have adopted three datacenter workloads, including commercial workloads (see Section III-C). We extracted three TCP versions from FreeBSD v9, each representing a different class of TCP implementations, and ported them to our simulation environment: New Reno, Vegas and Cubic. For performance evaluation we have mixed hardware and software experiments with detailed L2 simulation models of CEE switches and adapters.

b) Results: In Section III-B we contribute the necessary parameter changes for TCP over 10Gbps CEE. In Section IV we present evidence of PFC's benefits to TCP, leading to our recommendation to *enable PFC also for TCP* applications. In addition, we identify cases in which QCN is beneficial, respectively detrimental, to L4 transports.

The paper is organized as follows. The datacenter network stack is briefly described in Section II. In Section III, we present the simulation environment, network models, and the workloads used in our evaluation. In Section IV, we present simulation results and discuss their implications. Related work is presented in Section V, after which we conclude in Section VI.

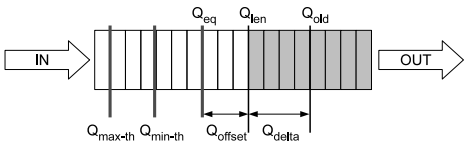


Figure 1. (i) Q_{max-th} is the PFC maximum threshold and Q_{min-th} is the PFC minimum threshold. (ii) The QCN load sensor measures the current queue occupancy level (Q_{len}) and computes the queue occupancy excess ($Q_{offset} = Q_{eq} - Q_{len}$), the queue rate excess ($Q_{delta} = Q_{old} - Q_{len}$) and the feedback value ($F_b = Q_{offset} + w \cdot Q_{delta}$). Q_{old} is the occupancy level at the previous sample and Q_{eq} is the target queue equilibrium (setpoint). If the computed feedback is negative, a 64 B congestion notification frame, including a 6-bit hotspot severity value, is sent back directly to the source of the sampled frame. A QCN-compliant source, upon receiving the congestion notification, reacts by instantiating a rate limiter that adjusts its transmission rate: the higher the feedback value, the higher the rate reduction according to the rate decrease control law. The load sensor detects and signals explicitly only the negative feedback. Therefore, the rate increase control law of the source rate limiter is responsible for autonomously recovering its injection rate, a three-phase process somewhat similar to Cubic.

II. Datacenter Network Stack

A. Layer 2 - Converged Enhanced Ethernet (CEE)

There is a growing interest in consolidated network solutions that meet the requirements of datacenter applications, i.e., low latency, no loss, burst tolerance, energy efficiency etc. One possible answer to the universal datacenter fabric is CEE with the following key features: (i) per-priority link-level flow-control and traffic differentiation, i.e., Priority Flow Control (PFC; 802.1Qbb) [10]; (ii) congestion management, i.e., Quantized Congestion Notification (QCN, optional in CEE; 802.1Qau) [11]; (iii) transmission scheduling: Enhanced Transmission Selection (ETS; 802.1Qaz).

1) PFC: Traditionally Ethernet does not guarantee lossless frame reception; instead, packets will be dropped whenever a receive buffer reaches its maximum capacity. Reliable upper-layer transports such as TCP interpret this event as implicit congestion feedback triggering window or injection rate corrections. This lossy network behavior, however, does not meet the semantics of applications such as Fibre Channel over Ethernet, MPI or low-latency messaging for Business Analytics.

PFC divides the traffic into 8 priorities based on the IEEE 802.1p Class of Service field. Within each priority, PFC acts as the 802.3x PAUSE, except that a PAUSEd priority will not affect the others.

2) QCN: QCN is an L2 congestion management scheme for controlling the injection rate of long-lived flows. It adapts the source injection rate to the available network capacity, the complement of which is signalled via explicit congestion feedback. QCN accomplishes this by using a primal-dual algorithm similar to TCP Cubic and Random Early Detection (RED) combined with Random Exponential Marking (REM). An *instantaneous* queue load sensor is implemented at each potential congestion point, e.g. switch buffer. The congestion point sensor randomly samples with a variable sampling probability P_s the queue characterized by two state variables, position and velocity $\{q, q'\}$, defined with respect to a pre-established setpoint Q_{eq}. A rate control algorithm is implemented in rate limiters at the source (Reaction Point), which reacts to the congestion feedback (or lack thereof) by reducing/increasing its transmission rate. See FIGURE 1 for details.

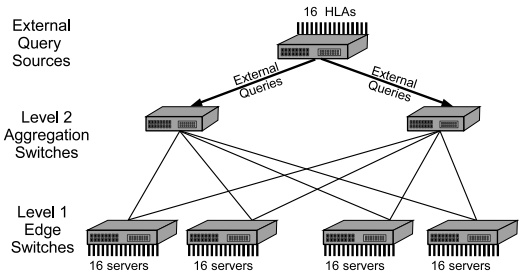


Figure 2. Two-tiered data center network topology with edge and aggregation switches and 64 end-nodes distributed in 4 racks. This topology is an *XGFT(2;16,4;1,2)*. The 16 external query sources are acting as the HLAs for the Scatter/Gather communication pattern generated by the commercial application traffic. These external sources inject TCP queries in the datacenter network through the Level 2 aggregation switches.

B. Layer 3 - ECN - RED

Random Early Detection [12] (RED) is an established Layer 3 (L3) Active Queue Management method. It detects congestion based on the *average* queue length. Unlike the $\{q, q'\}$ QCN load sensor, when properly configured (not always a trivial task [7]), RED is intrinsically more tolerant to bursts, as will be confirmed later. Explicit Congestion Notification (ECN) is an L3-L4 end-to-end congestion management protocol defined in RFC 3168. It provides congestion feedback without necessarily dropping packets.

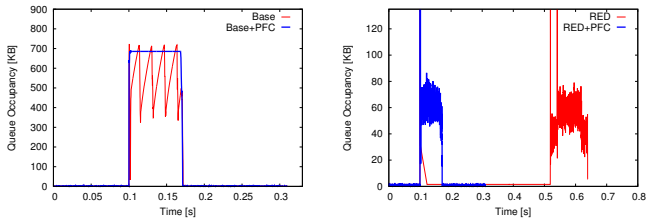
C. Layer 4 - TCP Versions

We have selected three common TCP versions. We use New Reno (RFC 3782) as it is both the best known and the most implemented TCP version; its congestion feedback is either packet loss and/or ECN-marked packets. Next, given the prevalence of queuing delay in the datacenter RTT, we have selected two contrasting and representative TCP variants. Using the RTT delay as the primary congestion measure (price), Vegas [13] stands for the delay-probing class of TCPs, including Compound TCP and Adaptive Reno. From the other extreme we have chosen Cubic [14], which represents the class of RTT-independent TCPs.

III. Evaluation Methodology: Environment, Models, Workloads

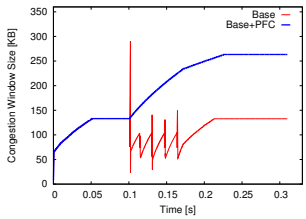
A. Simulation Environment

The first question for our performance evaluation is whether to rely on simulations or on hardware experiments? Hardware experiments are accurate and two-three orders of magnitude

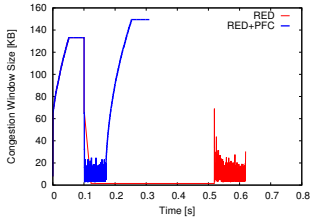


(a) Queue Base.

(b) Queue RED.

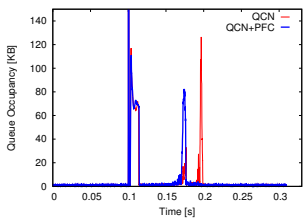


(c) Cwnd Base.

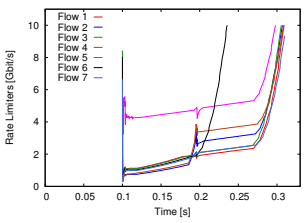


(d) Cwnd RED.

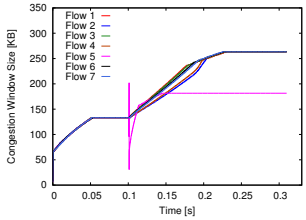
Figure 3. Congestive synthetic traffic: many to one. 7 TCP sources send to the same destination. From $t_0 = 0$ ms to $t_1 = 100$ ms admissible offered load. t_1 to $t_2 = 110$ ms burst: all 7 sources inject a $4\times$ overload of the destination sink capacity. After t_2 admissible load. The congestive event extends past t_2 due to backlog draining.



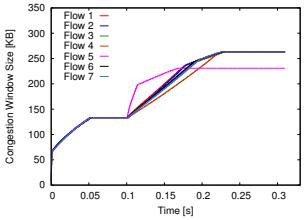
(a) Queue QCN 66.



(b) Rate limiters QCN 66.



(c) Cwnd QCN 66.



(d) Cwnd QCN 66 + PFC.

Figure 4. Congestive synthetic traffic: many to one. Same traffic pattern as in FIGURE 3. The rate limiters for the QCN+PFC configuration are not shown because they exhibit the same unfairness as those without PFC i.e. flow 5 gets more than 40% of the bandwidth.

faster than simulations. However, they are not: (a) *Observable*: not every queue and link is measurable; (b) *Repeatable*: scheduling of multicore multithreaded CPUs and asynchronous events (e.g. interrupts and IO) is not deterministic for our purpose; (c) *Flexible*: protocol, scheduling, architectural changes are difficult, if possible at all [7]; (d) *Available*: CEE-enabled switches with configurable PFC, QCN, ETS and OpenFlow are not yet widely available. These deficiencies are readily corrected in simulation, provided that sufficiently detailed and realistic models are implemented for the datacenter network components, TCP transports and the L2-3 protocols. Hence we have chosen to use both methods in our investigation, by calibrating our simulation models using hardware experiments.

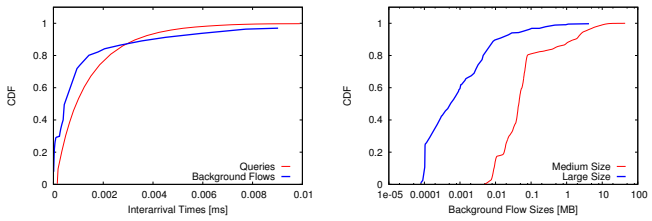
Having chosen simulation as our main instrument, the next question is why not use NS-3? While NS has a large library of supported L3 and L4 protocols, including a number of TCP versions, and is well established in the community, NS suffers from: (i) *Lack of L2 functionality*: none of the new CEE standards are supported, including PFC, ETS, QCN. (ii) *Insufficient detail*: whereas queuing, buffering, scheduling, link-level flow control, memory management etc. (switch and adapter micro-architecture) are essential and distinguishing features of modern datacenter switches and adapters, they are not reflected in NS, which focuses on higher abstraction layers. This makes it impossible to study L4's sensitivity to L2 protocols in NS. (iii) *Idealized L4 protocols*: compared to the actual TCP versions implemented in BSD, AIX and Linux kernels, the NS-3 TCP libraries are streamlined and simplified, thus trading accuracy for simulation efficiency. As our TCP simulations are to be calibrated against the actual OS stacks running on hardware, we have decided to port the above mentioned TCP stacks from the BSD kernel directly to our environment, which entails two simulators coupled in an end-to-end framework: Dimemas and Venus [15].

B. Network Models

1) Datacenter Topology: For commercial workload evaluations we use two practical, albeit scaled down in size, extended generalized fat tree (XGFT) [16] topologies: *XGFT(2;32,4;1,2)* and *XGFT(2;16,4;1,2)*. The latter is shown in FIGURE 2. In the first scenario, described in SECTION III-C1 we inject solely TCP traffic in the *XGFT(2;16,4;1,2)* network. In the second scenario, used in SECTION IV-C we inject both TCP and UDP traffic in the *XGFT(2;32,4;1,2)* network. In the following chapters we will outline the impact that the newly introduced CEE protocols have on TCP performance in these two scenarios.

For the MPI workloads using scientific application traffic we used two slightly different topologies: *XGFT(2;16,7;1,2)* and *XGFT(2;32,7;1,2)*.

2) TCP Transport Model: We extended the existing Venus network simulator with a model of the TCP transport. To be as close as possible to reality, we ported the TCP implementation code directly from the FreeBSD v9 kernel into our simulation framework, performing only compulsory (minimal) changes. They are mostly related to the allocation and deallocation of segments. The FreeBSD v9 kernel has two important features: (i) *connection cache*: the congestion window and the RTT estimation are inherited from one connection to the next; (ii) *adaptive buffers*: the receive and transmit buffers are increased in response to an RTT increase. During the calibration runs we noticed that the RTT was (much) smaller than the kernel timer quanta – by default 1 ms. With this setup both the Retransmission Time-Out (RTO) estimator and the TCP Vegas RTT measurement were ineffective. Therefore for all the experiments we reduced the timer granularity to 1 μs.



(a) Inter-arrival distributions. (b) Flow size distributions.

Figure 5. Flow inter-arrival and size distributions. For background flows we use the inter-arrival time and flow size distributions given in [2], [4]. The queries (foreground traffic) follow the inter-arrival time distribution from [2] accelerated 100×.

Next, we reduced the value of the minimum RTO to 2 ms, based on our network measurements and [8], [5]. In the absence of updated information, the kernel defaults to RTO = 3 s. Thus a loss of the initial SYN segment will drastically penalize the flow completion time, which occurred with PFC disabled. We reduced the default RTO to 10 ms, larger than the maximum RTT of our network. The RTO is computed using Jacobson's estimator; a constant term is added to the estimation, accounting for the variance in segment processing at the end-point kernels. In FreeBSD this term is conservatively set to 200 ms, accommodating slower legacy machines. We set the RTO variance to 20 ms to match current processors.

3) TCP Stack Delay Evaluation: We determined the delay of the TCP stack by modifying a Linux 2.6.32.24 kernel running on an Intel i5 3.2GHz machine with 4 GB of memory. We instrumented the E1000e Ethernet device driver of an Intel Gigabit Ethernet controller 82578DM. We measured the time spent between an application send and the moment a frame is enqueued in the device driver queue, and the time spent from the moment a frame is received by the device driver until the application receives the data. TABLE I summarizes the main parameters for TCP and ECN/RED as obtained through these experiments.

4) Simulation Parameters: The network link speed is 10 Gbps. We use an ideal output-queued switch with the following characteristics: (i) N-fold speedup: The write bandwidth into an output queue is N times the line rate. (ii) Full buffer sharing: the size of an output queue is bounded only by the sum of all the input buffers for all ports. In a real system, however, a single output queue can not monopolize the full memory. In addition, each input adapter provided one virtual output queue (VOQ) for each destination. This avoids most of the head-of-line blocking, which can be further exacerbated by the QCN rate limiters. We changed the scheduling discipline between the TCP sockets to prevent one socket from monopolizing the entire adapter memory. The QCN algorithm implemented is based on version 2.4. TABLE I contains the key parameters of the network.

C. Applications, Workloads and Traffic

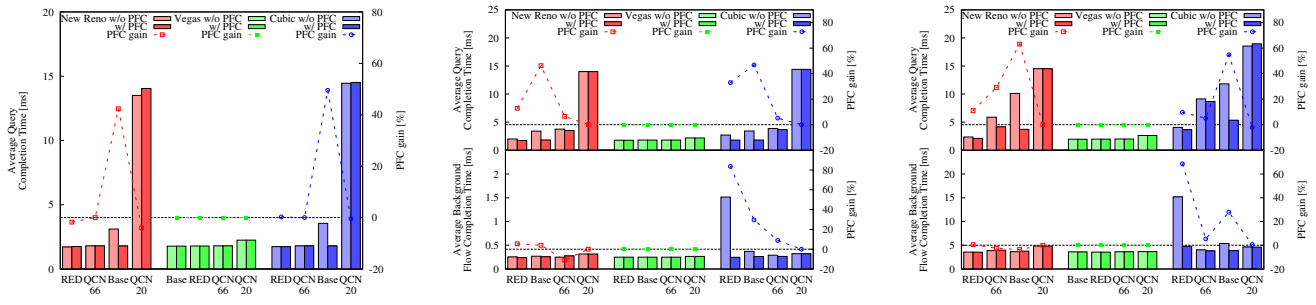
As stated at the onset, we aim to evaluate the impact of different L2, L3 and L4 protocols on performance at the application level, centered on revealing the TCP sensitivities to the two new CEE features: PFC and QCN. Therefore we have selected a few key datacenter applications, divided in two groups: commercial and scientific workloads.

Table I
SIMULATION PARAMETERS

Parameter	Value	Unit	Parameter	Value	Unit
TCP					
buffer size	128	KB	TX delay	9.5	μs
max buffer size	256	KB	RX delay	24	μs
default RTO	10	ms	timer quanta	1	μs
min RTO	2	ms	reassembly queue	200	seg.
RTO variance	20	ms			
ECN-RED					
min thresh.	25.6	KB	W_q	0.002	
max thresh.	76.8	KB	P_{max}	0.02	
QCN					
Q_{eq}	20 or 66	KB	fast recovery thresh.	5	
W_d	2		min. rate	100	Kb/s
G_d	0.5		active incr.	5	Mb/s
CM timer	15	ms	hyperactive incr.	50	Mb/s
sample interval	150	KB	min decr. factor	0.5	
byte count limit	150	KB	extra fast recovery	enabled	
PFC					
min thresh.	80	KB	max thresh.	97	KB
Network hardware					
link speed	10	Gb/s	adapter delay	500	ns
frame size	1500	B	switch buffer size/port	100	KB
adapter buffer size	512	KB	switch delay	100	ns

1) Commercial Applications: We have designed our commercial traffic generator based on findings from a few recent papers. In [3] the authors instrumented 19 datacenters to find evidence of ON-OFF traffic behavior. In [4] they perform an in-depth study of the spatial and temporal distribution of the flows in 10 production datacenters. In [2] the authors use a similar approach to measure the size and inter-arrival time distribution of the flows. Another study [6] observed that modern applications use a Scatter/Gather communication pattern. The traffic study from [2] confirms that finding.

We place the High Level Aggregators (HLA) as in FIGURE 2. The HLAs execute queries triggered from external HTTP requests. The queries have an inter-arrival time as in [2]. When an HLA launches a query it contacts some randomly chosen Mid-Level Aggregators (MLA) – one in each rack – and sends them a subquery. An MLA that receives a subquery will distribute it to all the other servers in the same rack – and then, later, it will collect the partial results. When *all* the results have been received, the MLA will send back its aggregated response to the HLA. Using the real-life data from [2], [4] we have created a traffic generator that injects a *foreground traffic* matrix of queries ('mice') on top of a *background traffic* matrix of longer lived flows ('elephants'). The queries are generated as outlined in the previous paragraph, have a fixed size of 20 KB and the inter-arrival time distribution shown in FIGURE 5(a). For the background flows each source randomly chooses a destination in order to match the ratio of intra-rack to inter-rack traffic of 30%. Then each source draws from the inter-arrival time (FIGURE 5(a)) and flow size distributions (FIGURE 5(b)) and sends the data. For the queries as well as



(a) No background flows. (b) Medium sized background flows. (c) Large sized background flows.

Figure 6. Commercial Workload with TCP Background Traffic. The upper part of the graphs shows the average query completion time, while the lower part shows the average completion time of the background flows. Exception: the first graph where there are no background flows. The bars are grouped in three categories, based on the TCP version. Within a category bars are sorted increasing with average query completion time without PFC.

for the background flows we collect the completion time as an application level metric [17].

2) Scientific Applications: We have selected nine MPI applications. Five of them belong to the NAS Parallel Benchmark [18]: BT, CG, FT, IS and MG. This benchmark aims to measure the performance of highly parallel supercomputers. In addition we used another 4 applications for weather prediction (WRF), parallel molecular simulations (NAMD) and fluid dynamics simulations (LISO and Airbus). All the above applications were run on the MareNostrum cluster at the Barcelona Supercomputing Center; during the run, the MPI calls of the applications were recorded into trace files. The collected traces were then fed into our end-to-end simulation environment; for a detailed description of this methodology, please refer to [15]. We assume that the MPI library on each processing node uses TCP sockets as underlying transport. The collected traces are in the order of a few seconds. Therefore, the TCP socket between a source and a destination of an MPI communication is opened only once, when the first transfer occurs and it is kept open during the entire run of the trace.

IV. Simulation Results and Discussion

In this section we use the following notations: *Base* – no congestion management scheme; *QCN 20/66* – Quantized Congestion Notifications (QCN) enabled with Q_{eq} = 20KB or 66KB respectively; *RED* – Random Early Detection with Explicit Congestion Notifications (ECN-RED) enabled. We run each of these congestion management schemes with or without PFC enabled and with different TCP congestion control schemes: New Reno, Vegas, Cubic.

A. Congestive Synthetic Traffic

To debug the simulation model and calibrate our expectations, we initially use TCP New Reno in a congestive synthetic traffic scenario, described in Figure 3, derived from the 802.1Qau input-generated hotspot benchmark.

1) Base: Figure 3(a) shows the evolution of the congested queue. Figure 3(c) shows the congestion window of one of the TCP sources. Without PFC (red) we observe the typical sawtooth graph. TCP increases the window size until the first segment is dropped; detected via duplicate ACKs and handled by the Fast Retransmit. With PFC (blue) there are no losses, therefore at t_1 the RTT increases abruptly; detected by the automatic buffer resizing mechanism that increases the receiver buffer size to allow further augmentation of congestion window.

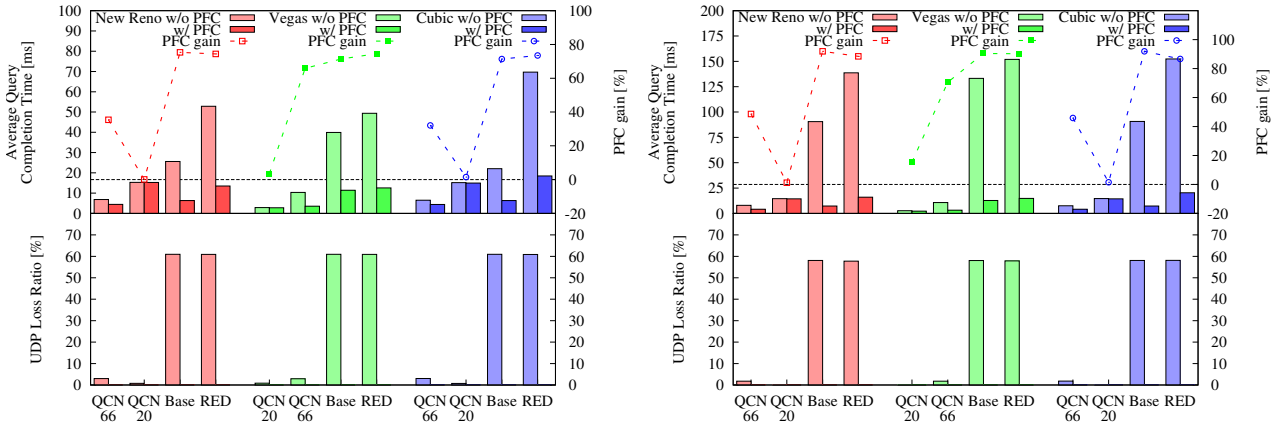
2) ECN-RED: Figure 3(b,d) show the congested queue and one TCP source congestion window evolution, respectively. With PFC enabled (red) the behavior is similar to the base. The difference is in the much lower queue occupancy during the congestive event which extends past t_2 due to backlog draining. Source reduces the injection rate based on ECN feedback. Disabling PFC leads to lower ECN-RED performance. When the queue fills, the TCP sources receive ECN feedback and enter Congestion Recovery. This however is too late to avoid loss, as the load has already been injected. The received duplicate ACKs are ignored since the sources are already in recovery, hence no Fast Retransmit before the retransmission timeout, resulting in throughput loss between 0.17s and 0.53s. This is a situation in which additional feedback, i.e., ECN leads to a wrong decision.

3) QCN 66: Figure 4(a) shows the congested queue, while Figure 4(b) shows the evolution of the QCN rate limiters. QCN's unfairness [19] causes a single flow to monopolize more than 40% of the capacity: the 'winner' flow ends its transmissions first. However, the other flows still cannot increase their injection rate because of the recovery phase. Figure 4(c,d) show the congestion window with and without PFC, respectively. QCN per se is capable of avoiding all the losses but one (of the 'winner').

B. Commercial Workload with TCP Background Traffic

The traffic pattern is described in Section III-C1. Figure 6(a), 6(b) and 6(c) show the average flow completion time using 3 background traffic flow sizes. Figure 6(a) corresponds to query traffic ('mice') without background flows; Figure 6(b) and Figure 6(c) show the same query traffic with medium, large sized 'elephants' respectively, as background traffic.

1) TCP Vegas: Despite the different configurations, Vegas does not reveal significant impact in flow completion time. It adjusts the congestion window based on the measured delays.



(a) Medium sized background flows.

(b) Large sized background flows.

Figure 7. Commercial Workload with UDP Background Traffic. The upper part of the graphs shows the average query completion time, while the lower part shows the loss ratios of the background UDP flows. Bars are grouped in three categories based on the TCP version. Within a category bars are sorted increasing with average query completion time without PFC.

Since PFC is effective only when flows experience drops which Vegas avoids, PFC plays a secondary role here. Ditto for the other L2 and L3 congestion management schemes.

2) TCP Cubic: Cubic [14] performs worse than New Reno and Vegas in this environment. We observed that the aggressive increases in the congestion window generate more losses than New Reno, therefore penalizing the query completion time. This is corroborated with Cubic's RTT independence, leading to increased losses and poor performance in a datacenter environment.

3) PFC: In all the tests, PFC reduces the flow completion time, with the exception of QCN 20 configuration. We attribute the PFC gains to avoiding TCP stalls waiting for retransmissions. They are caused by Jacobson's RTO estimator, in the datacenter environment, where the RTT is dominated by queuing, instead of flight delays [1]. Whereas link delays are constant, the datacenter's queuing delays are extremely dynamic: they can increase 100 to 1000 fold within milliseconds. The original RTO estimator, however, reacts slowly. This is compounded with its kernel calculation: a constant term is added that accounts for the cumulated variances in the segment processing at the two end-point kernels. This constant is orders of magnitude higher than the typical datacenter RTT.

4) QCN: For comparison with [2] we choose two Q_{eq} setpoints : (i) the value recommended by the IEEE standards committee – 20% of the queue size, i.e. 20 KB, (ii) an experimental value i.e. 66 KB. QCN 66 is always better than QCN 20 in FIGURE 6 mostly due to its higher tolerance to the intrinsic burstiness of the transport layer. The TCP source sends a burst of segments until either the congestion window or the receiver window is exhausted. The first burst of segments will trigger a reverse burst of ACKs, which in turn will produce a second burst of segments etc. This is supported by measurements in [3]. Generally QCN is highly sensitive to burstiness.

5) ECN-RED: With this workload ECN-RED delivers the best performance, further improved by enabling PFC. ECN-RED outperforms QCN because:

(i) *burst sensitivity* - ECN-RED congestion feedback is based on averaged, whereas QCN is based on instantaneous, queue length. Therefore a transient burst will not trigger a reduction of the injection rate with RED.

(ii) *interaction with L4* - the congestion notifications generated by RED are processed directly at the transport layer which adjusts the congestion window accordingly. On the other hand, TCP remains oblivious of Layer 2 congestion feedback.

(iii) *data/control differentiation* - RED can generate congestion notifications only for segments carrying data. The reduction of the congestion window only affects the data flow while the control segments can still move freely. On the other hand, QCN's rate limiters can not distinguish between control and data. For example, we found that some queries were delayed because the initial SYN segments were throttled by the rate limiter.

C. Commercial Workload with UDP Background Traffic

We also tested mixed TCP-UDP performance. In addition to the previous section, TCP has to compete against aggressive UDP background sources ('elephants'). Therefore, we double the number of end-nodes: half of the end-nodes are TCP, while the other half are UDP sources. UDP sources inject bursty traffic with average burst sizes of 28 KB and 583 K. The UDP burst sizes are selected according to the background flow size distributions from FIGURE 5(b).

The average flow completion time for the TCP queries are shown in FIGURE 7. We also measure the loss ratio for TCP and UDP flows – the loss ratio is computed as the percentage of dropped bytes vs. the total injected bytes (see FIGURE 7 – lower half).

Most of the dropped bytes are UDP. This is because TCP reduces its window whenever losses are detected. In contrast

with the previous section, here we observe that Vegas is sensitive to ECN and QCN. Again, enabling PFC improves performance. Overall, the best performer is QCN 66. When we introduce non-cooperative UDP sources, only QCN's rate limiters can restore some of the fairness lost by TCP in competing against UDP.

D. Scientific Workload

The simulated MPI traces are described in SECTION III-C2. Initially we run each benchmark on a reference system where we assume we have a perfect hardware accelerated transport and lossless network. We run every benchmark on each configuration while measuring the execution times. Then we compute the relative slowdown of each benchmark vs. the ideal reference. Finally we average all the slowdowns across the nine benchmarks, plotted in FIGURE 8.

Enabling PFC improves performance across all the configurations. The previous observations from SECTION IV-B apply also to this workload. The best performer is ECN-RED with PFC enabled. With PFC disabled, however, QCN 20 produces better results. In contrast with the commercial traffic, QCN 20 was the worst performer. Commercial workloads exhibit only sparse transient congestive events, whereas in the scientific workload the congestive events are sustained and involve all the end-nodes. The MPI applications use barriers to synchronize between execution phases. All the nodes start communicating almost at the same time and this generates heavy congestion. The aggressive Q_{eq} setpoint of QCN 20 effectively mitigates such congestive cases.

V. SELECTED RELATED WORK

Our work is at the confluence of established, e.g. TCP, and emerging research areas, such as datacenter workload analysis and new L2 networking protocols. Our commercial workload traffic generator is based on [3], [4], [2], [6].

QCN is defined in [11] and further analyzed in [20]. Its unfairness and lack of RTT adaptivity [2] have been addressed by E^{2}CM [21]. An alternative solution is proposed in [22].

While to the best of our knowledge this is the first comparative evaluation of 'short-fat' TCP for datacenters, a few performance reviews of modern TCP variants for 'long-fat' networks are available, e.g. in [23], [24].

The TCP Incast problem has been analyzed in [5], [8], where a $10 - 1000\times$ RTO reduction and high resolution timers have been proposed. Another TCP Incast solution is DCTCP [2], using a modified RED/ECN and a new multibit feedback estimator that filters the incoming single-bit ECN stream. This compensates the stiff adaptive queue management setup in the DCTCP congestion point (partly similar to QCN's sensor) with a smooth congestion window reduction function, reminiscent of QCN's rate decrease – hence departing from TCP's halving of the congestion window. Closely related is [19] which analyzes the TCP Incast problem in a QCN-enabled *lossy* network – arguably in conflict with default assumption of lossless CEE. The main drawbacks are the use of NS-2 simulations and the overly aggressive sampling proposal. TCP Incast is studied in [7] using SCTP and ECN with hardware experiments. The performance improvements range from 5.5% to 7.9%, limited by the experimental platform.

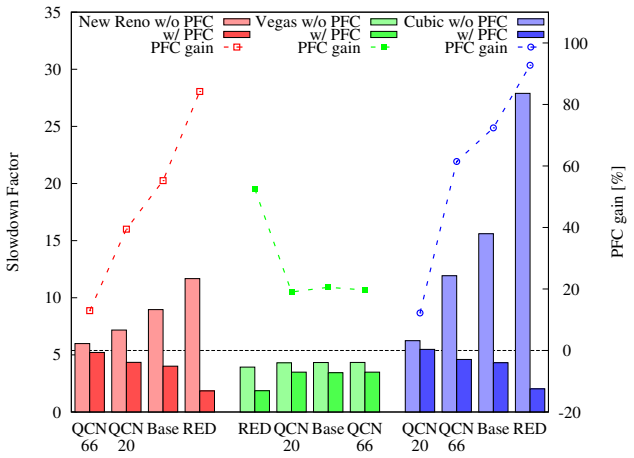


Figure 8. Scientific Workload: MPI Traces relative slowdowns. Bars are grouped in three categories based on the TCP version. Within a category bars are in increasing order of the relative slowdown factors with PFC disabled.

VI. CONCLUSIONS AND FINAL REMARKS

We summarize the results discussion by answering our initial questions from SECTION I: (**Q1**) How does TCP perform over CEE networks? (**Q2**) Is PFC beneficial or detrimental to TCP? (**Q3**) Is QCN beneficial or detrimental to TCP?

With respect to **Q1**, the delay-probing TCP Vegas performs the best, requiring arguably minimal changes, i.e., high resolution timers. By contrast, RTT-independent TCP Cubic entails the most adaptation effort for datacenter environments, eliciting exhaustive parameter retuning and potentially core algorithm changes. In our experiments, Cubic suffers from aggressivity and slow convergence of congestion windows. New Reno lies in between, requiring more parametrical retuning than Vegas, but no invasive changes such as Cubic. Whether RTT independence, as in BIC and Cubic, is actually harmful in CEE networks with a wide dynamic range of queuing delays (sub-μs up to tens of ms) remains an open research problem.

Moving to **Q2**, despite our contrary expectations, PFC has consistently improved the performance across *all* the tested configurations and benchmarks. The commercial workload completion time improves by 27% on average, and up to 91%. Scientific workloads show higher gains by enabling PFC: 45% on average, and up to 92%.

The answer to **Q3**, however, must be more carefully qualified, depending on the proper parameter settings per each case and application – somewhat similar to Cubic. In datacenter environments, both L4 Cubic and L2 QCN seem negatively impacted by their relative independence of the queuing dominated RTT, and lack of adaptation to a much wider dynamic range – up to a few magnitude orders more – of operation vs. the Internet.

On the *positive* side, properly tuned for commercial TCP

with UDP applications, QCN 66 *with* PFC improves performance on average by 49%, up to 70%. When we introduce non-cooperative UDP flows in the network QCN 66 keeps congestion under control regardless of the upper layer protocols. For scientific workloads, QCN 20 *without* PFC – currently an uncommon HPC configuration – improves performance on average by 31%, up to 59%. HPC applications typically exhibit alternating phases of computation and communication. During the latter, typically all nodes start communicating quasi-simultaneously, which can generate overload episodes and hotspots – especially in slim networks as reproduced here. The aggressive Q_{eq} setpoint of QCN 20 effectively mitigates such congestive events.

On the *negative* side, mistuned QCN can *severely* degrade performance. E.g., in commercial workloads relying exclusively on TCP – without competing UDP traffic sources – QCN 20 *without* PFC degrades performance on average 131%, up to 311% for New Reno and 321% for Cubic. For scientific workloads QCN 66 *with* PFC degrades performance on average by 5.4%, up to 8.2% – hence leaving QCN enabled is acceptable whenever its Q_{eq} is set $2\times$ to $4\times$ higher than the standard recommendation.

Our results show that RED handles the transient congestion episodes generated by commercial applications better than QCN. This reveals a preventable (by careful tuning) QCN weakness: burst sensitivity. A properly configured RED is less sensitive to burstiness, mainly because it relies on smoothed (low pass filtered) queue length. This can reduce the query completion time by up to 76%. Aggravating the performance penalty with bursty commercial workloads, QCN suffers from inherent unfairness: it tends to arbitrarily favor some 'winner' flows over the others, harming the average completion time. The final answer is that QCN elicits further investigation and improvement, particularly with respect to its lack of adaptivity and fairness. Meanwhile we recommend that QCN should be conservatively tuned and enabled whenever heterogeneous transports – e.g., TCP with UDP, RDMA, FCoE, RoCEE etc. – will be expected to share, even briefly, the same hardware priority in the CEE datacenter network.

Future Work: The main limitation of our model is the canonical implementation of queries with a strict synchronization barrier; this exacerbates QCN's unfairness, thus further degrading performance. To alleviate this problem we are currently implementing a relaxed version of the barrier. Also we plan to analyze Compound TCP and E^{2}CM.

ACKNOWLEDGMENTS

We are deeply indebted to C. Basso, C. DeCusatis, J. Kouloheris, F. Neeser, V. Pandey, R. Recio and G. Rodriguez for their contributions.

REFERENCES

[1] M. Gusat *et al.*, "Delay-based Cloud Congestion Control," in *Proc. IEEE GLOBECOM 2009 Global Communications Conference*, Honolulu, HI, USA, December 2009.

[2] M. Alizadeh *et al.*, "DCTCP: Efficient Packet Transport for the Commoditized Data Center," in *Proc. ACM SIGCOMM 2010 Conference on Data Communication*, New Delhi, India, August 2010.

[3] T. Benson *et al.*, "Understanding Data Center Traffic Characteristics," in *Proc. ACM SIGCOMM Workshop for Research on Enterprise Networks (WREN 2009)*, Barcelona, Spain, August 2009.

[4] ——, "Network Traffic Characteristics of Data Centers in the Wild," in *Proc. Internet Measurement Conference (IMC 2010)*, Melbourne, Australia, November 2010.

[5] Y. Chen *et al.*, "Understanding TCP Incast Throughput Collapse in Datacenter Networks," in *Proc. 1st ACM Workshop on Research on Enterprise Networking (WREN 2009)*, Barcelona, Spain, August 2009.

[6] S. Kandula *et al.*, "The Nature of Datacenter Traffic: Measurements & Analysis," in *Proc. Internet Measurement Conference (IMC 2009)*, Chicago, IL, USA, November 2009.

[7] R. R. Stewart *et al.*, "An Investigation into Data Center Congestion with ECN," in *Proc. 2011 Technical BSD Conference (BSDCan 2011)*, Ottawa, Canada, May 2011.

[8] V. Vasudevan *et al.*, "Safe and Effective Fine-grained TCP Retransmissions for Datacenter Communication," in *Proc. ACM SIGCOMM 2009 Conference on Data Communication*, Barcelona, Spain, August 2009.

[9] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *Proc. 6th Symposium on Operating System Design and Implementation (OSDI 2004)*, San Francisco, CA, USA, December 2004.

[10] *P802.1Qbb/D1.3 Virtual Bridged Local Area Networks - Amendment: Priority-based Flow Control*, IEEE Draft Standard, 2010. [Online]. Available: http://www.ieee802.org/1/pages/802.1bb.html

[11] *P802.1Qau/D2.4 Virtual Bridged Local Area Networks - Amendment: Congestion Notification*, IEEE Draft Standard, 2009. [Online]. Available: http://www.ieee802.org/1/pages/802.1au.html

[12] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.

[13] L. Brakmo *et al.*, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," in *Proc. ACM SIGCOMM 1994 Conference on Data Communication*, London, UK, August 1994.

[14] I. Rhee and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," in *Proc. PFLDnet*, Lyon, France, February 2005.

[15] C. Minkenberg and G. Rodriguez, "Trace-driven Co-simulation of High-Performance Computing Systems using OMNeT++," in *Proc. SIMUTools 2nd International Workshop on OMNeT++*, Rome, Italy, March 2009.

[16] S. R. Öhring *et al.*, "On Generalized Fat Trees," in *Proc. 9th International Parallel Processing Symposium (IPPS 1995)*, Santa Barbara, CA, USA, April 1995.

[17] N. Dukkipati and N. McKeown, "Why Flow-Completion Time is the Right Metric for Congestion Control," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 59–62, January 2006.

[18] D. Bailey *et al.*, "The NAS Parallel Benchmarks," NASA Ames Research Center, Moffett Field, CA, NASA Technical Report RNR-94-007, March 1994.

[19] P. Devkota and A. L. N. Reddy, "Performance of Quantized Congestion Notification in TCP Incast Scenarios of Data Centers," in *Proc. 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2010)*, Miami Beach, FL, USA, August 2010.

[20] C. Minkenberg and M. Gusat, "Congestion Management for 10G Ethernet," in *Proc. Second Workshop on Interconnection Network Architectures: On-Chip, Multi-Chip (INA-OCMC 2008)*, Goteborg, Sweden, January 2008.

[21] M. Gusat *et al.* (2007, March) Extended Ethernet Congestion Management (E2CM): Per Path ECM - A Hybrid Proposal. [Online]. Available: http://ieee802.org/1/files/public/docs2007/au-sim-IBM-ZRL-E2CM-proposal-r1.09.pdf

[22] A. Kabbani *et al.*, "AF-QCN: Approximate Fairness with Quantized Congestion Notification for Multi-tenanted Data Centers," in *Proc. 18th Annual IEEE Symposium on High-Performance Interconnects (HOTI 2010)*, Mountain View, CA, USA, August 2010.

[23] Y.-T. Li *et al.*, "Experimental Evaluation of TCP Protocols for High-Speed Networks," *IEEE/ACM Transactions on Networking*, vol. 15, no. 5, pp. 1109–1122, October 2007.

[24] H. Bullot *et al.*, "Evaluation of Advanced TCP Stacks on Fast Long-Distance Production Networks," *Journal of Grid Computing*, vol. 1, no. 4, pp. 345–359, December 2003.