Re-architecting Congestion Management in Lossless Ethernet

Wenxue Cheng and Kun Qian, *Tsinghua University and Beijing National Research Center for Information Science and Technology (BNRist);* Wanchun Jiang, *Central South University;* Tong Zhang, *Tsinghua University, Beijing National Research Center for Information Science and Technology (BNRist), and Nanjing University of Aeronautics and Astronautics;* Fengyuan Ren, *Tsinghua University and Beijing National Research Center for Information Science and Technology (BNRist)*

https://www.usenix.org/conference/nsdi20/presentation/cheng

This paper is included in the Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI '20)

February 25–27, 2020 • Santa Clara, CA, USA

978-1-939133-13-7

Re-architecting Congestion Management in Lossless Ethernet

Wenxue Cheng[†,‡] Kun Qian[†,‡] Wanchun Jiang[§] Tong Zhang[†,‡,*] Fengyuan Ren[†,‡]

[†] *Tsinghua University*

[‡] *Beijing National Research Center for Information Science and Technology (BNRist)*

[§]*Central South University* [*]*Nanjing University of Aeronautics and Astronautics*

Abstract

The lossless Ethernet is attractive for data centers and cluster systems, but various performance issues, such as unfairness, head-of-line blocking and congestion spreading, etc., impede its large-scale deployment in production systems. Through fine-grained experimental observations, we inspect the interactions between flow control and congestion control, and are aware that the radical cause of performance problems is the ineffective elements in the congestion management architecture for lossless Ethernet, including the improper congestion detection mechanism and inadequate rate adjustment law.

Inspired by these insights and findings obtained in experiment investigations, we revise the congestion management architecture, and propose the Photonic Congestion Notification (PCN) scheme, which consists of two basic components: (*i*) a novel congestion detection and identification mechanism to recognize which flows are really responsible for congestion; (*ii*) a receiver-driven rate adjustment method to alleviate congestion in as short as 1 RTT. We implement PCN using DPDK NICs and conduct evaluations using testbed experiments and simulations. The results show that PCN greatly improves performance under concurrent burst workload, and significantly mitigates PFC PAUSE messages and reduces the flow completion time under realistic workload.

1 Introduction

Recently, lossless network has become an attractive trend in data centers and cluster computing systems. Generally, retransmission caused by packet loss readily leads to goodput decrease, completion time increase, and even missing application deadlines [9, 10, 50]. In addition, scaling transport protocols such as Remote Direct Memory Access (RDMA) and Fibre Channel (FC) over data center requires reliable transmission without packet loss due to network congestion [3, 15].

The lossless InfiniBand (IB) [16] is popular in HPC (High performance Computing) cluster systems, but modern data center has already been built with IP/Ethernet technologies that are also dominated in traditional Internet. The data center operators and cloud builders may do some IB, but much less ubiquitous than Ethernet. Furthermore, they are reluctant to simultaneously deploy and manage two separate networks within the same data center [39, 49]. IEEE DCB (Data Center Bridging) [4] is naturally imparted appeal as an enhanced capability of Ethernet, which enables Ethernet to be a consolidated switching fabric that can replace traditionally separated fabrics for special purposes, such as FC for storage, IPC (Interprocess Communication) for HPC, and Ethernet for LAN traffic. Converged Ethernet has significant performance, cost, and management advantages over maintaining separate switching fabrics [8]. To enable lossless semantics for a consolidated Ethernet, both hop-by-hop flow control PFC (Priority-based Flow Control) [6] and end-to-end congestion control QCN (Quantized Congestion Notification) [5] are developed in the link layer to enhance traditional Ethernet. The scalable lossless Ethernet switching fabric is definitely one of the potential candidates for building future data centers to accommodate promising applications, such as RDMA over Converged Ethernet (RoCE) [15], NVMe Over Fabrics [42] and resource disaggregation [23], etc..

Over the last decade, the rise of various Online Data-Intensive (OLDI) applications [31] and virtualized services [40] generate increasingly diverse traffic patterns and specific characteristics, e.g., incast, burst and mixture of mice/elephant flows, etc. [12, 25, 44]. Because it is unclear whether the lossless Ethernet can work effectively in large-scale data centers with such complex traffic, we conduct empirical and experimental investigations to attain the in-depth understanding of congestion management (CM) architecture in lossless Ethernet. The detailed observation and conjoint analysis uncover the radical root of some performance issues, such as congestion spreading and being susceptible to burst traffic. In the light of these insights, we re-architect CM in lossless Ethernet. The key findings and main contributions are summarized as follows.

• Revealing the inadequate elements in existing CM architecture for lossless Ethernet, including: a) The congestion detection mechanism cannot exactly identify congested or uncongested flows when they are churned in the same queue, so that it is unlikely to notify different sources to make discriminative rate adjustments. b) The slow evolution-based rate adjustment of end-to-end congestion control mismatches the fast operations of hop-by-hop flow control.

• Developing a novel CM scheme named Photonic Congestion Notification (PCN), which includes: a) A subtle congestion detection and identification mechanism, which can distinguish real congested flows so as to make a proper rate adjustment for congested or uncongested flows even if they are churned in the same accumulated queue. b) A receiver-driven rate adjustment rule, which can speed up the convergence of rate regulation, and is robust to burst traffic and adaptable to link capacity.

• Implementing PCN using DPDK NICs and conducting evaluations using both testbed experiments and ns-3 simulations. Extensive simulations in the large-scale network with synthesized traffic from real workload show that PCN suppresses PFC PAUSEs by 12%, 47% and 90% compared to QCN, DCQCN and TIMELY respectively, and reduces latency by at most 10x, 11.3x and 13.2x.

2 Background

2.1 Traffic Features in Data Centers

A variety of applications in data centers generate flows with a wide spectrum of traffic patterns and distributions. For example, web search service usually generates short and burst flows. On the other hand, the log file processing introduces few but long-lived flows to transmit bulk of data. Investigations on traffic in many operation data centers show the wide distribution traffic patterns [41]. The size of flows may range from 0.05KB to more than 100MB, and the distribution is quite scattered. Among all traffic, mice flows, which finish sending all packets before receiving any ACK, cannot be adjusted by the end-to-end congestion control scheme. Furthermore, many measurements [18, 19, 33, 41] indicate that the occurrence of mice flow is not only frequent but also bursty. The highly dynamic entering/leaving of mice flows would greatly shock queue length in switches and then the end-to-end latency [12, 13, 44]. Although these flows do not react to the congestion control scheme, they severely disturb the normal operations of the congestion management of switching fabric in data centers or cluster systems.

2.2 Congestion Management in lossless Ethernet

To guarantee losslessness and provide satisfying job completion time under such diverse traffic patterns, congestion management becomes critical and challenging in lossless Ethernet. IEEE DCB [4] specifies a framework for CM consisting of two basic functions, including end-to-end congestion control and hop-by-hop flow control.

The end-to-end congestion control regulates source sending rate actively according to the congestion information reflected by measured variables, such as switch queue length or RTT. Representative solutions include QCN developed by IEEE 802.1 Qau [5], the Layer-3 scheme DCQCN [49], and the RTT-based scheme TIMELY [36]. Although these protocols can constrain the switch queue length and accordingly reduce the packet loss ratio, there is not enough guarantee of zero packet loss. Actually, the uncontrollable burst may be already lost before sources are aware of network congestion, especially when the congestion control loop delay is relatively large or the degree of burst and concurrency is heavy. What is worse, a large number of congestion control mechanisms [5, 21, 27, 36, 38, 49] start flows at the line rate to accelerate the completion of mice flows, which exacerbates the loss problem.

To avoid packet loss due to uncontrollable burst, Priority-based Flow Control (PFC) is defined by IEEE 802.1Qbb [6] to ensure losslessness. With PFC, a switch sends a PAUSE frame to its upstream device (a switch or a NIC) to stop transmission when the ingress queue length exceeds a certain threshold. And a RESUME frame is sent when the queue drains below another threshold. Although PFC can guarantee zero packet loss due to network congestion, it leads to some performance issues such as head-of-line blocking (HLB), unfairness and even deadlock [26, 28, 45, 46, 49]. When PFC is triggered incessantly, the local congestion spreads back to both congested and uncongested sources, and then the network throughput and flow completion time are drastically harmed. The fundamental solution for these performance issues is to eliminate persistent congestion by end-to-end congestion control schemes such that PFC is not triggered incessantly [46, 49].

In total, the end-to-end congestion control needs PFC to prevent packet loss due to transient congestion of uncontrollable burst, and PFC also needs end-to-end congestion control to eliminate persistent congestion. That is, the end-to-end congestion control and hop-by-hop lossless flow control are complementary to each other in lossless Ethernet.

3 Experimental Observation and Insights

3.1 Observations

Although both end-to-end congestion control and hop-by-hop flow control can meet their goals independently under the diverse traffic patterns, their interaction would induce unexpected issues. (1) When burst short flows enter into the network, existing flows in the network would still suffer from the PFC-related side-effects, i.e., congestion spreading and unfairness. (2) After burst leaving the network, congestion control would not efficiently and timely reallocate available bandwidth.

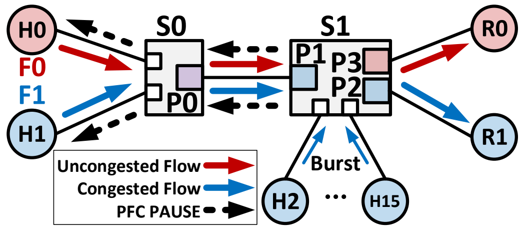
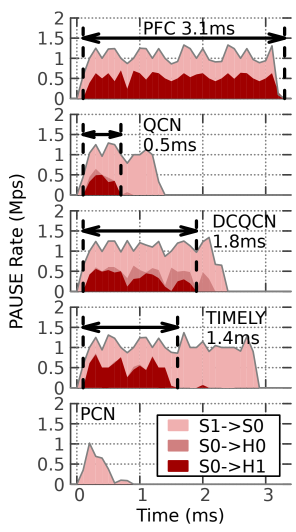


Figure 1: Compact and typical network scenario.

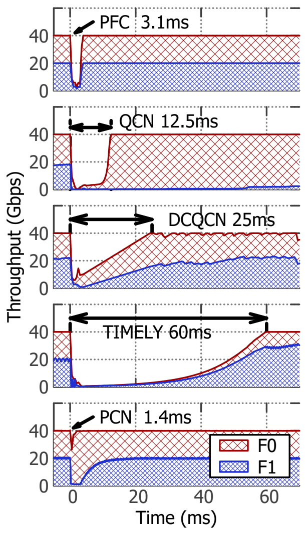
Most existing work considerably concerns the single element in the CM of lossless Ethernet (e.g., congestion control [36, 49]) or special symptoms (e.g., HLB [7], deadlock [28, 29, 45]), but unconsciously overlooks the interaction of congestion control and flow control under diverse traffic patterns, thus is likely to shield the essential cause of aforementioned performance issues. Subsequently, we first conduct a careful, fine-grained and multi-variable observation, and then infer the radical root of special symptoms and issues.

Specifically, we define a compact and typical network scenario, which is not too complex to hinder us capturing the basic principles of both key elements and core mechanisms in the CM of lossless Ethernet. At the same time, it should have sufficiently common features so as to ensure the obtained conclusions and insights are without loss of generality. As shown in Fig.1, we choose a basic unit of a typical network topology in data center, like Clos [22] and Fat-Tree [11], where 16 senders and 2 receivers are connected by two switches. All links are 40Gbps, with a propagation delay of 5μs. The traffic is a mixture of long-lived flows and concurrent burst mice flows. In detail, H0 and H1 start long-lived flows to R0 and R1, respectively. Assume that F0 and F1 achieve their fair bandwidth allocation of the 40Gbps bottleneck link from switch S0 to S1 at the beginning of simulation. At time 0, each sender of H2∼H15 generates 16 short flows to R1 at line rate (i.e., 40Gbps) simultaneously, and the size of each flow is 64KB. Since each mice flow only lasts for 12.8μs (<1 RTT), it is uncontrollable by the end-to-end congestion control mechanisms. These uncontrollable burst flows last for about 3ms in total. We conduct simulations with ns-3 to investigate various CM schemes including PFC, PFC+QCN, PFC+DCQCN, and PFC+TIMELY. All parameters are set to the default values recommended by the related standard [5, 6] and literature [36, 49], and the details are given in § 7. The results are presented in Fig.2.

When PFC is solely employed, the input port P1/S1 pauses its upstream port P0/S0 to avoid packet drops, and the port P2/S1 is congested by concurrent burst flows. Subsequently, "Pause" spreads upstream along with the long flow, and both H0 and H1 are eventually paused. We measure the *PAUSE Rate* (i.e., the rate of transmitting PAUSE messages), and the instantaneous throughput. As shown in Fig.2(a), a congestion tree, which roots from S1, spreads to H0 and H1, appears



(a) Pause rate



(b) Throughput

Figure 2: Interactions between PFC and existing congestion controls.

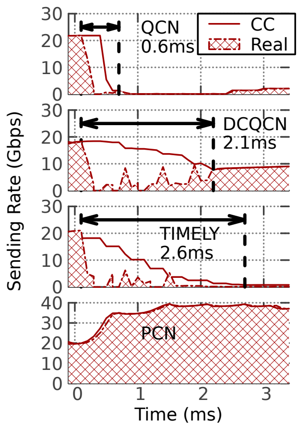
and lasts for 3.1ms (>100 RTT) until the burst mice flows finish. During this process, both the congested flow F1 and uncongested flow F0 face great throughput loss as shown in Fig.2(b), no matter whether they are responsible for the real congestion at port P2/S1.

When QCN, DCQCN or TIMELY works with PFC jointly, the congestion tree still appears, as shown in Fig.2(a). However, its lasting time is reduced to 0.5ms (≈17RTT), 1.8ms (≈57RTT) and 1.4ms (≈47RTT). Surprisingly, the two long flows F0 and F1 may fail to recover to their initial throughput quickly after both concurrent burst flows and congestion tree disappear. As illustrated in Fig.2(b), the throughput loss unexpectedly lasts for 12.5ms with QCN, 25ms with DCQCN and 60ms with TIMELY, respectively, even if the concurrent burst flows last for only 3ms. Totally, the performance of PFC+QCN, PFC+DCQCN and PFC+TIMELY is worse than PFC in this scenario.

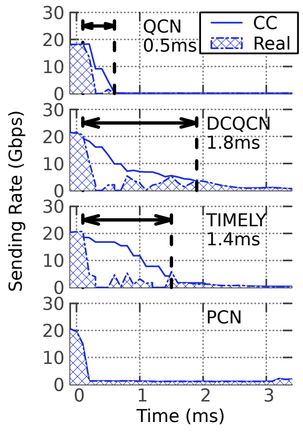
3.2 Interaction Issues

To understand the long duration of congestion tree and unexpected great throughput loss, we analyze the dynamic behaviors of flows in detail, and reveal the interaction issues between hop-by-hop flow control and end-to-end congestion controls. We believe that careful analysis and rigorous reasoning from interactive behavior could enlighten us the root causes of various performance issues reported by existing work [26, 34, 37].

1) PFC confuses congestion detection. In above experiments, an ideal end-to-end congestion control scheme should only throttle F1 to 2.5Gbps and allocate flow F0 the remaining



(a) Uncongested flow F0



(b) Congested flow F1

Figure 3: The responsiveness of different congestion controls.

bandwidth (37.5Gbps) of bottleneck link from S0 to S1. However, this ideal bandwidth allocation cannot be achieved even existing congestion controls (QCN, DCQCN and TIMELY) are employed. To explore the cause of this phenomenon, we record the sending rate regulated by the end-to-end congestion control and the real sending rate of the uncongested flow F0 and congested flow F1. The results are presented in Fig.3(a). When congestion tree exists, both the queue length and RTT increase at port P0/S0. Because senders infer congestion according to the feedback information (i.e., queue length or RTT), F0 is also regarded as congested. Hence, the sending rate of F0 is reduced by QCN, DCQCN and TIMELY, even if it contributes nothing to the real congestion point at P2/S1. Therefore, after the congestion tree disappears (as marked by the dash line in Fig.3(a)), the sending rate of F0 is very low although it escapes from the collateral damage of PFC.

In summary, it takes some time for congestion controls to eliminate congestion tree. In this transient process, the large queue length and RTT due to congestion spreading caused by PFC would mislead congestion controls to decrease the sending rate of victim flows (F0 in this example).

2) The slow evolution-based rate adjustment of end-to-end congestion control mismatches the fast hop-by-hop operations of PFC. Fig.3 also unveils the reason why the congestion tree is still created and lasts for tens of RTTs with QCN, DCQCN and TIMELY. Although F0 and F1 are throttled immediately when the concurrent burst mice flows enter, it takes a long time for QCN, DCQCN and TIMELY to reduce the sending rates (the regulation time of different congestion controls are marked in Fig. 3). However, PFC works hop-by-hop and thus the congestion spreads very fast. During the rate decrease of F0 and F1, PFC is triggered incessantly. So the real sending rates of F0 and F1 are mainly determined by PFC rather than end-to-end congestion control, thus the throughput of both F0 and F1 are small. This is why the congestion tree spreading still occurs even if the end-to-end congestion control is employed.

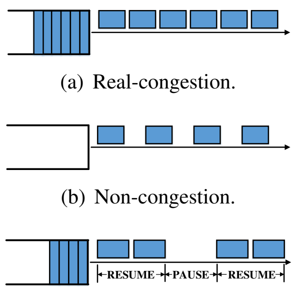
This problem is attributed to the mismatch between the slow evolution-based rate adjustment of end-to-end congestion control and the fast operations of hop-by-hop flow control. More specifically, when the available bandwidth reduces suddenly due to the concurrent burst mice flows, the end-to-end congestion control schemes have no idea of the target rate thus only make rate decrease based on the current sending rate step by step, which is at most 50% per update period. Moreover, the update period is about 20μs (time of transmitting 100 packets) for QCN, 50μs for DCQCN and at least 12.5μs (time of sending 64KB segment) for TIMELY. What's more, when the throughput of F0 and F1 is very small, DCQCN may not receive a single packet in one update period, and would start rate increase automatically. As a result, tens of update periods may be needed to decrease F1's rate to approach the remaining available bandwidth, as shown in Fig.3.

3) The rate increase is inadaptable to dynamic network conditions. After the concurrent burst mice flows vanish and the congestion tree disappears, the sending rates of both F0 and F1 have been throttled, and need to increase step by step. QCN and DCQCN increase the sending rate towards the target rate stored at previous rate decrease in a binary-search way and raise the target rate linearly with a pre-configured value periodically. TIMELY adds the sending rate with a fixed value in each update period. Briefly, all rate increasing methods are linear. Consequently, they fail to take full use of available bandwidth immediately after the disturbance of concurrent burst mice flows. This is why flows F0 and F1 need much longer time to recover to full throughput as presented in Fig.2(b). Moreover, the step of rate increase in each update period needs to be configured adaptively according to network bandwidth. For example, the parameters of QCN, DCQCN and TIMELY tuned for 40Gbps link may be too conservative for 100Gbps link, but too aggressive for 1Gbps link. The tuning of parameters would become difficult in practice.

4 Principles

The root cause of all aforementioned performance issues can be concluded as the existing end-to-end congestion control scheme cannot cooperate with hop-by-hop flow control well. To address these issues, we revisit the architecture of CM. We first present a discussion about which elements in existing congestion management introduce these performance issues, and then propose the ways to overcome these incongruities by re-architecting the CM for lossless Ethernet. Briefly, the principles are threefold.

1. The uncongested flow becomes a victim because the existing congestion management cannot identify real congested flows. The operation of PFC would back pressure congestion and contaminate current congestion signals



(a) Real-congestion.

(b) Non-congestion.

(c) Quasi-congestion.

Figure 4: Different states of egress port.

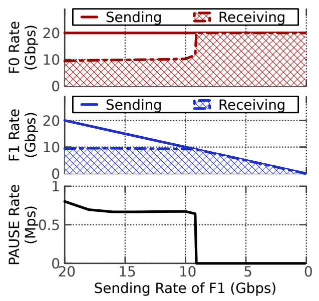


Figure 5: Relationship of PAUSE and receiving rate.

(i.e., queue length and RTT). We need to find out a new mechanism to properly distinguish which flows are really responsible for congestion.

2. Congestion spreading is caused by the slow evolution-based rate decreasing mechanism, thus a fast and accurate rate decreasing solution is indispensable.

3. When burst traffic vanishes, the long-lived flows mainly rely on the linear rate increase to share the released bandwidth, which leads to sluggish convergence and bandwidth waste. Therefore, a prompt rate increasing mechanism should be developed.

4.1 Congestion Detection and Identification

Traditionally, the end-to-end congestion control detects the network congestion based on the measured variables like switch queue length and RTT. However, this congestion detection mechanism is confused by PFC in lossless Ethernet. We need to revise the congestion detection and identification mechanism to avoid this confusion and then correctly identify which flows are really congested.

4.1.1 Detecting Congestion

To aid detecting congestion, we classify the egress ports of switches into the following three states.

Real-Congestion: The ports in real-congestion fully utilize the egress links and the excessive incoming packets accumulate in the buffer, as shown in Fig.4(a). For example, in the previous simulation, when the concurrent burst mice flows start, port P2/S1 is in real-congestion.

Non-Congestion: As illustrated in Fig.4(b), no packets accumulate in the buffer of egress ports, and thus the incoming packet is transmitted immediately. That is, the egress links work normally with utilization less than 100%. The port P3/S1 in above simulation is always in non-congestion.

Quasi-Congestion: The ports in quasi-congestion also keep certain queue length, but the associated egress link is not fully utilized due to PAUSE and RESUME, as depicted in Fig.4(c). Therefore, it is unknown whether the incoming rate of packets exceeds the link capacity or not. For example, in the previous simulation, port P0/S0 turns into quasi-congestion in face of PFC triggers. However, because flows passing through this port would suffer large queue length and delay, the congestion detection mechanism in existing congestion controls (e.g., QCN, DCQCN and TIMELY) dogmatically judges that these flows experience congestion.

Consequently, to distinguish these different states of the egress ports, especially the quasi-congestion state, the impacts of PFC should be taken into consideration when detecting congestion.

4.1.2 Identifying Congested Flows

Owing to the impact of PFC, packets from both congested and uncongested flows are likely to backlog in the same queue length in egress port, which is paused by its downstream ingress port. Therefore, it may be proper to predict potential congestion depending on the queue length of egress port, but indeed unwise to make congestion judgment and provide indiscriminate information to all flow sources, just like QCN and DCQCN. TIMELY also hardly distinguishes whether the flow actually traverses the real congested port by merely measuring RTT and its variations.

To avoid the confused congestion information in existing CM architecture to perturb the normal interaction between flow control and congestion control, and even lead to mutual damage, we advocate decoupling congestion detection and identification functions during re-architecting the CM of lossless Ethernet. The switch is responsible for detecting congestion and providing congestion signals through monitoring the related network status. The end systems synthesize relevant information to judge congestion and identify whether its flow is really congested.

4.2 Receiver-Driven Rate Decrease

The ideal congestion control scheme should throttle the congested flows towards a proper rate directly. To achieve this goal, we need to obtain this proper rate at first. In lossless Ethernet, the proper rate should not trigger PFC but can still keep high throughput. To find this rate, we should answer the following two sub-questions: 1) what is the minimum rate for congested flows to not lose throughput? 2) what is the maximum rate for congested flows to not trigger PFCs?

The first answer is intuitive. It should be the arrival rate of receiver. We define it as *Receiving Rate*. On one hand, the path of congested flows must have at least one real congested port, thus the sum of receiving rates of all flows just achieves the capacity of bottleneck link. On the other hand, if the congested flow decreases rate to less than its receiving rate, there must be idle bandwidth on the bottleneck link, which means that this flow can actually send more data. Thus, the receiving rate is the minimum rate for the congested flows to not lose throughput. The power of receiving rate has also introduced in recent designs like Fast TCP [32], NDP [27] and Homa [38]. They both take advantage of receiving rate to achieve fast convergence when detecting congestion.

Fortunately, the receiving rate is also the answer to the second sub-question. That is, when the sending rate does not exceed the receiving rate, the packets of congested flows do not accumulate at the ingress port of congested switches and then PFCs are not triggered. What's more, this phenomenon occurs regardless of whether the egress port of the same switch is congested or not.

To vividly illustrate this phenomenon, we repeat the experiment in the network scenario given in § 3.1. We start H2∼H4 at line rate to simulate uncontrollable bursts. And both the congestion-irrelevant flow F0 and congestion-relevant flow F1 are controlled by rate limiters. The sending rate of F0 is fixed at its fair allocation (i.e., 20*Gbps*), and the sending rate of F1 varies from 20*Gbps* to 0 step by step manually. When the simulation is running, both flows are throttled by their fixed rate limiters and PFC. The sending rates set in rate limiters and receiving rates at the receiver side for these two long flows, as well as the generating rate of PAUSE frames on ingress port p1/S1, are drawn in Fig.5. Obviously, when the sending rate of congestion-relevant flow F1 exceeds 9.5Gbps, its receiving rate is only 9.5Gbps. At the same time, the ingress port p1/S1 generates PFC PAUSEs persistently and the congestion-irrelevant flow F0 is collaterally damaged. On the contrary, when the sending rate of F1 does not exceed 9.5Gbps, no PAUSE frame is generated from port p1/S1 and the congestion-irrelevant flow F0 can achieve its expected throughput. This experiment indicates that throttling congested flows to their receiving rate can prevent more PFC triggers on the associated egress ports, and then suppress congestion spreading in this branch of congestion tree.

Consequently, we obtain a valuable insight, that is **decreasing the rate of congested flows to their receiving rate directly**. It inspires us to design a receiver-driven rate decreasing algorithm to work in harmony with PFC in lossless Ethernet, which will be elaborated in the following.

4.3 Gentle-to-Aggressive Rate Increase

The rate increase should accelerate non-congested flows to rapidly share available bandwidth and then keep at full utilization stably simultaneously. The rate-increase rule of a non-congested flow is needed in two cases.

1) The flow has just turned its state from congested to non-congested. According to our receiver-driven rate decrease principle, the flow rate has reduced to its receiving rate, which implies no PFC trigger and no throughput loss. Thus, the flow has little space for rate increase. Therefore, the rate of this flow should be increased gently.

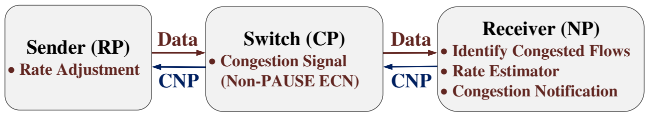

Sender (RP)
• Rate Adjustment
Data
CNP
Switch (CP)
• Congestion Signal (Non-PAUSE ECN)
Data
CNP
Receiver (NP)
• Identify Congested Flows
• Rate Estimator
• Congestion Notification

Figure 6: PCN framework.

2) The flow has remained in the non-congested state for several continuous update periods. In this case, the flow can increase more aggressively to occupy the available bandwidth. Since our receiver-driven rate-decrease rule can sharply reduce the overflowed traffic, the rate increasing mechanism can be designed more aggressively to fulfill network bandwidth quickly.

Therefore, we obtain a suggestion, that is **increasing the rate of non-congested flows gently at first and then aggressively**. It guides us to design a gentle-to-aggressive rate increasing algorithm that can guarantee stability and fast convergence simultaneously.

5 PCN

In this section, based on the principles in § 4, we re-architect congestion management for lossless Ethernet and propose Photonic Congestion Notification (PCN)[1], which is designed to be a rate-based, end-to-end congestion control mechanism to work with PFC in harmony. As shown in Fig.6, PCN is composed of three parts: reaction point (RP), congestion point (CP) and notification point (NP). In general, the CP, which always refers to the congested switch, marks passing packets using a Non-PAUSE ECN (NP-ECN) method to detect whether the egress ports are in real congestion. Notice that a packet marked with NP-ECN does not definitely mean encountering congestion, it requires NP to make the final decision. The NP, i.e., the receiver, identifies the congested flows, calculates their receiving rate and sends the congestion notification packets (CNP) to RP periodically. The RP, which is always the NIC of senders, adjusts the sending rate of each flow according to the information in CNPs. Subsequently, we introduce each part of PCN in details.

5.1 CP Algorithm

We develop the NP-ECN method to detect congestion and generate the congestion signal. The CP algorithm follows the state machine in Fig.7. Suppose that when one egress port of a switch receives a RESUME frame from its downstream

[1]We liken current schemes (e.g. QCN, DCQCN and TIMELY) to quantum, because they can only quantify the network congestion as a whole, but cannot provide different congestion notifications for congested flows and non-congested victim flows, which seems in *quantum entanglement*. And as an analogy, our design is like the photon, which breaks down the entanglement, i.e., directly recognizing the congested flows and allocating them the appreciate rates.

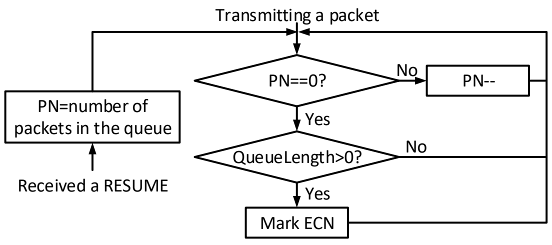


Figure 7: CP state machine.

node, there are N packets in the associated waiting queue. NP-ECN will set its counter PN=N. Then the port restarts to transmit packets. Each time a packet is transmitted, the counter is decremented by one, until all of the N paused packets have been transmitted. For these N packets, they will not be marked. And for the later packets that have not been paused and correspondingly PN=0, the switch will mark them with ECN in the traditional method, with the threshold as zero.

In this way, all packets in the real-congestion egress ports will be marked with ECN. On the contrary, the packets are never marked with ECN in the non-congestion ports. And for quasi-congestion ports, the paused packets are not marked with ECN. Meanwhile, when the queue of ingress port is not empty, the packets arriving and leaving the ports in RESUME status are marked with ECN, namely, packets in quasi-congestion ports are partially marked with ECN. In PCN, CP only works for marking a congestion signal on packets and lets the NP node finally determine whether the flow is congested.

It is noted that NP-ECN mechanism can be implemented easily based on the commercial switch equipped with both ECN and PAUSE functions. Compared to the traditional ECN method in commodity switches, the NP-ECN method of PCN requires one more counter per port, and several more lines of logic. The space and computing complexities of modification are both O(1).

5.2 NP Algorithm

The functions of NP include identifying congested flows, estimating receiving rate and sending Congestion Notification Packets (CNP) periodically. T denotes the CNP generation period.

Identifying congested flows: NP identifies the congested flows based on the ECN signal marked by the NP-ECN mechanism. A flow is regarded to be congested if 95% packets received in CNP generation period T are marked with ECN. The value 95% is set empirically to filter some tiny disturbances in practice, such as queue oscillation and priority schedule, which make that one or more packets of real-congested flows are unlikely marked with ECN.

Estimating receiving rate: The receiving rate is calculated directly with T divided by the total size of arrived packets. Noticeably, the receiving rate of flow may be so small that just one packet arrives in several CNP generation periods. To address this special case, PCN also records the inter-arrival time of packets at the NP. When the inter-arrival time is larger than T, NP estimates the receiving rate by replacing T by the inter-arrival time.

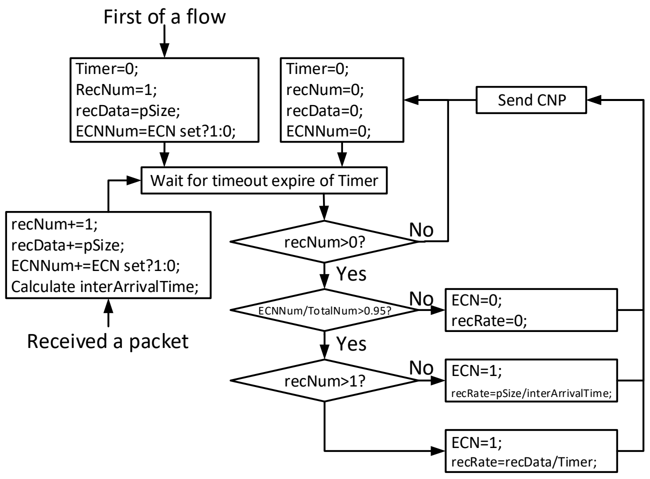


Figure 8: NP state machine.

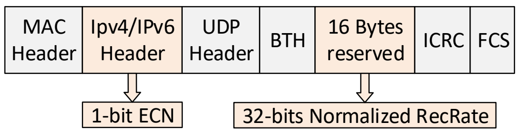


Figure 9: Packet format of CNP

Generating CNPs: The NP sends CNPs to notify the source of flow with the receiving rate in period T, which is set to be $50\mu s$ similar to DCQCN. Moreover, PCN generates CNP explicitly when the flow needs either rate-decrease or rate-increase, different from DCQCN which only generates CNPs to notify rate-decrease. And the CNP is not generated when none of its packets is received in period T. In details, the format of CNP packets is compatible with the CNP packet in RoCEv2 [15], as shown in Fig.9. The main information encapsulated by CNP includes 1-bit ECN in the IPv4/IPv6 header and 32-bit RecRate in the reserved segment, which carries the receiving rate normalized by $1Mbps$. The state machine of NP algorithm is summarized in Fig.8.

5.3 RP Algorithm

Algorithm 1 describes the pseudo code of how RP adjusts the sending rate according to the information in CNP. In the beginning, flows start at the line rate to improve flow completion time (FCT) for short flows.

Rate Decrease: When RP receives a CNP with ECN-marked, it conducts a rate decrease following the rule in line 6. Instead of resetting the sending rate to the receiving rate di-

Algorithm 1 PCN RP Algorithm.

```
1: $sendRate \leftarrow lineRate$
2: $w \leftarrow w_{min}$
3: repeat per CNP ($CE$, $recRate$)
4:   if $CE == 1$ then
5:     ($CNP$ notifies rate decrease)
6:     $sendRate \leftarrow \min\{sendRate,\ recRate \cdot (1 - w_{\min})\}$
7:     $w \leftarrow w_{\min}$
8:   else
9:     ($CNP$ notifies rate increase)
10:    $sendRate \leftarrow sendRate \cdot (1 - w) + lineRate \cdot w$
11:    $w \leftarrow w \cdot (1 - w) + w_{\max} \cdot w$
12: until End of flow
```

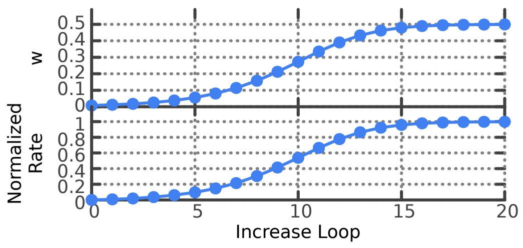


Figure 10: Evolution of w and rate when a sender receives continuous CNP notifying rate increase.

rectly as discussed in § 4.2, a small discount $w_{\min}$ is conducted, such that the build-up queue in switches can be drained out. Accordingly, during draining the build-up queue, the $recRate$ may be larger than the $sendRate$, the sending rate should be non-increasing in line 6.

Rate Increase: When RP receives a CNP without ECN-marking, it makes rate adjustments following the law in line 10 and 11. Specifically, the RP increases the sending rate by computing a weighted average of its current value and the line rate. This rate increase law is effective in multiple folds.

(1) The ideal sending rate can be reached as it always stays between the current sending rate and the line rate.

(2) Since the value of w is identical for all flows, the slow flows increase more aggressively than fast flows, which is beneficial to fairness.

(3) The weight w changes automatically from the minimum value $w_{\min}$ to the maximum value $w_{\max}$ such that PCN can realize the gentle-to-aggressive rate increase as discussed in § 4.3. For example, when $w_{\min} = 1/128$, $w_{\max} = 0.5$, and CNPs without ECN-marking are received successively, the evolution of w and the sending rate from 0 to the lineRate are presented in Fig.10. The sending rate grows by no more than 10% of the line rate in the first 5 CNPs, but increases to 95% of the line rate after only 15 CNPs.

(4) Any parameter configurations are not specially required to adapt to the upgrade of link capacity from 1Gbps to even 400Gbps.

5.4 Discussion

As discussed in §4, the main root of performance issues in current lossless Ethernet is the improper interaction between PFC and end-to-end congestion control schemes. We demonstrate that PCN solves the core problems in lossless Ethernet using a minimal implementation cost.

Implementation requirement: To implement PCN, a little switch modification is needed. Compared to the traditional ECN method in commodity switches, the NP-ECN method of PCN (see Fig.7) only requires one more counter per port, and several more lines of logic. The space and computing complexities of modification are both O(1).

Benefits: To demonstrate the advantages of PCN, we enable PCN and repeat the simulations in § 3.1, and the results are also inserted into Fig.2 and Fig.3, respectively. The results in Fig.2(a) tell that PAUSE in both S0->H0 and S0->H1 links are completely avoided and only a handful of PAUSE fleetingly appears in the S1->S0 link, but congestion spreading is quickly suppressed and congestion tree is not generated. The results in Fig.3 confirm that PCN can help the uncongested flows grab idle bandwidth quickly, and regulate the congested flows to proper rates correctly and promptly. PCN increases F0 to fully utilize network bandwidth during concurrent bursts. After the concurrent burst vanishes, F0 and F1 fairly share bandwidth without wasting network resources or triggering PFC PAUSEs as shown in Fig.2(b).

6 Theoretical Analysis and Parameter Setting

6.1 Theoretical Analysis

We build a fluid model of PCN and analyze its performance, including convergence, fairness, and stability. The main conclusions are summarized in the following propositions and the detailed analyses are listed in Appendix A.

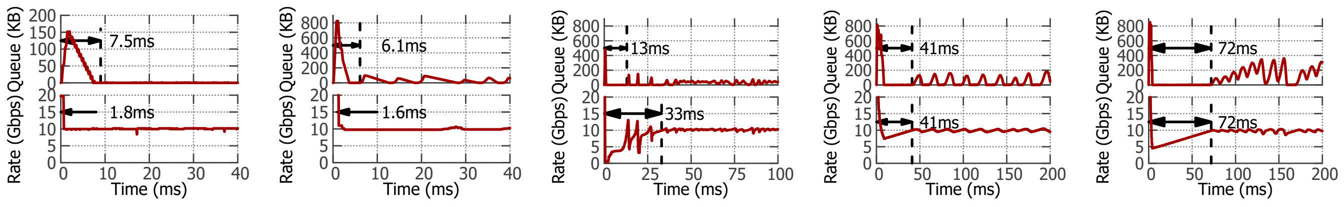
Proposition 1. *PCN can achieve convergence of total rate towards the bottleneck capacity as fast as in only one control loop, i.e., one RTT.*

Proposition 2. *PCN can always fairly share the bottleneck link, i.e., $R_i \rightarrow \frac{C}{N}$ regardless of the initial sending rates and parameter settings, where R_i is the receiving rate of flow i, N is the number of sources sharing the bottleneck link, and C is the link capacity.*

Proposition 3. *PCN is stable and the oscillation of both the queue length and rate are bounded in the steady state. The maximum oscillation ranges of queue length (ΔQ) and receiving rate of flow i (ΔR_i) are*

$$\Delta Q = (N - 2 + w_{\min}) w_{\min} CT \tag{1}$$

$$\Delta R_i \rightarrow w_{\min} C \tag{2}$$



(a) PCN (Testbed) (b) PCN (ns-3) (c) QCN (ns-3) (d) DCQCN (ns-3) (e) TIMELY (ns-3)

Figure 11: Dynamic behavior of PCN, DCQCN and TIMELY.

6.2 Parameter Settings

Based on the above conclusions, we can obtain some practical guidelines towards parameter settings, including the CNP generating period T, and the minimum and maximum value of weight factor w.

CNP generating period T: It should be identical for all flows. It is noteworthy that T is also the control loop period, thus a large T will damage the responsiveness of PCN. However, in practice, there exists an inherent control loop delay, i.e., RTT. If T is smaller than RTT, PCN is hardly aware of status change in the last control loop, which leads to overmuch adjustments and considerable oscillations. Therefore, the recommended T should be the maximum value of RTT in networks, which can be estimated in advance.

Minimum weight $w_{\min}$: The value of $w_{\min}$ should make a trade-off between fast convergence and stability. A large/small $w_{\min}$ will speed up/slow down the convergence of queue length, but make the flow oscillate more/less aggressively at steady state. According to Proposition 2, Equation (1) and (2), we recommend the proper value of w to be $0.1/N$, which limits the aggregate rate oscillation not exceeding $0.1C$ and the queue oscillation less than $0.1CT$.

Maximum weight $w_{\max}$: The value of $w_{\max}$ determines how aggressively a flow increases when the network is detected under-utilized continuously. Thus an aggressive $w_{\max}$ is recommended, i.e., $w_{\max} = 0.5$.

7 Evaluation

We evaluate the performance of PCN in a variety of settings using testbed experiments (§ 7.1) and ns-3 simulations (§ 7.1 $\sim$ 7.4), and compare it against QCN, DCQCN and TIMELY. The functional modules of our simulator are developed based on the open project for DCQCN [48] and code snippet (per-packet pacing version) for TIMELY [35], and all parameters are set to the default values recommended by the related literatures [36, 49]. All experiments enable PFC with $X_{OFF} = 512KB$.

7.1 Basic Properties

In this subsection, we verify the basic function of PCN using simple synthetic microbenchmarks.

Testbed setup: Since current commodity switches do not provide the interface to modify the ECN-marking logic, we implement PCN upon DPDK [1]. We plug two Intel 82599 NICs to one PowerEdge R530 server to act as PCN's CP. Each NIC has two 10Gbps Ethernet ports and the server is equipped with dual Intel Xeon E5-2620 v3 CPUs (6 cores, 2.4GHz). Thus, the server can work as a four-port switch. By deploying DPDK in the server, both PFC and NP-ECN are implemented based on the reference test-pipeline project [2].

DPDK also enables the implementation of our special packet processing required at NICs. On the sender side, the rate limiter at a per-packet granularity is employed for rate adjustment. On the receiver side, PCN receives packets, records ECN marking information, and sends back CNP packets periodically.

Scenario: We use a dumbbell topology where 3 pairs of sender and receiver share the same 10Gbps bottleneck link. Specially, the number of flows on one of three pairs is twice of that on other two. We run this experiment on both hardware testbed and ns-3 simulator for cross-validation. In both testbed experiments and simulations, the RTT is measured to be about $500\mu s$, thus the same configuration is kept in simulations.

Fine-grained observation: First, four long-lived flows are launched and the dynamic behaviors of PCN, QCN, DCQCN and TIMELY is observed. The evolutions of queue length in bottleneck and the aggregate sending rate are depicted in Fig.11. As illustrated in Fig.11(a) and 11(b), PCN exhibits the same performance on the testbed and simulator. Comparing Fig.11(b) to 11(c), 11(d) and 11(e), PCN outperforms QCN, DCQCN and TIMELY in terms of fast convergence and stability.

In both testbed experiment and simulation, PCN regulates the aggregate sending rate to the bottleneck capacity within 2ms (4 RTT), which is 20x, 25x and 45x faster than that with QCN, DCQCN and TIMELY, which is benefited from the receiver-driven rate-decrease method of PCN. It can throttle the incoming traffic to match the bottleneck capacity directly, rather than explore the available bandwidth round by round. Consequently, PCN can limit the bottleneck queue length

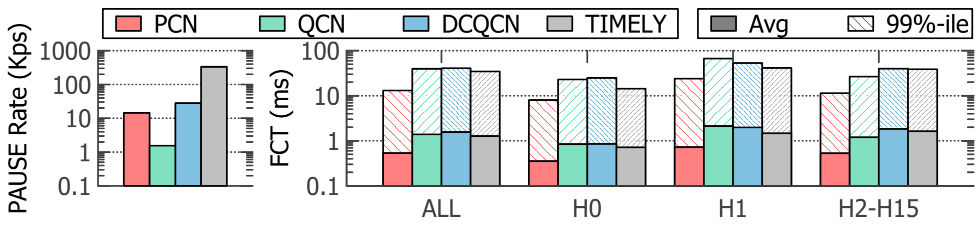


Figure 12: Generating rate of PAUSEs and FCT under concurrent burst.

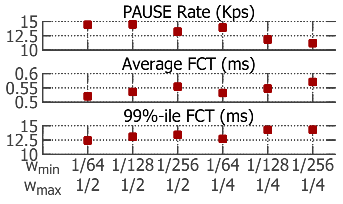


Figure 13: Performance of PCN with different $w_{\min}$ and $w_{\max}$.

at a very low level (about several packets) in no more than 7.5ms (∼15 RTT), while it costs 13ms (∼26 RTT) for QCN, 41ms (∼80RTT) for DCQCN and 72ms (∼144 RTT) for TIMELY.

In the steady state, PCN oscillates with low amplitude in both testbed experiment and simulation. The queue length almost approaches zero and the aggregate sending rate keeps near 10Gbps. QCN has the similar performance. However, both DCQCN and TIMELY lead to large oscillations and high buffer occupancy. This advantage comes from the congestion detection method of PCN. The threshold of queue length for ECN marking is set to zero, rather than a positive value.

7.2 Burst Tolerance

One advantage of PCN is robustness against PFC triggers caused by concurrent burst flows. Next, we use the basic scenario in Fig.1 to evaluate PCN in the typical head-of-line scenario. All links are 40Gbps with $5\mu s$ propagation delay, hosts $H0 \sim H15$ generate flows according to the heavy-tailed Hadoop workload [44] with exponentially distributed inter-arrival time. Specially, the workload generators at hosts $H2 \sim H15$ are set to be synchronous to simulate concurrent bursts. The target load at the two bottleneck links is set to 0.6. We measure the pause rate and flow completion time (FCT) of PCN and compare them with QCN, DCQCN and TIMELY.

The left subgraph in Fig.12 shows the generating rate of PFC PAUSEs. QCN triggers the smallest PAUSEs, and PCN can prevent at least 53% and 92% of PFC PAUSEs compared to DCQCN and TIMELY, respectively. And the average and 99_{th} percentile FCTs from different hosts are drawn in the right subgraph in Fig.12. The solid bar at the bottom indicates the average FCT and the upper stripe bar shows the 99_{th} percentile value. Clearly, PCN performs better than QCN, DCQCN and TIMELY for all kinds of hosts.

1) Actually, QCN avoids PAUSEs by drastically reducing the sending rate, which likely leads to poor link utilization and high FCT for long-lived flows. On the contrary, PCN can prevent PAUSEs without harming throughput, and then achieves 2.25x∼3.03x shorter FCT than QCN.

2) For the victim host H0, PCN achieves 2.4x and 2.0x faster average FCT compared to DCQCN and TIMELY, which is mainly benefited from a fact that PCN can mitigate PFC triggers between two switches. For the concurrent burst from $H2 \sim H15$, PCN can keep the buffer at egress port P2 nearly empty, and thus obtain an improvement of 3.5x and 3.4x in the 99_{th} percentile FCT compared to DCQCN and TIMELY. And for host H1, whose flows traverse two congested switches, the flow transmission speed of PCN is at least 2.2x of DCQCN and 1.7x of TIMELY.

Flow size	% of number		% of traffic	
	W1	W2	W1	W2
0KB-10KB (S)	80.14	70.79	3.08	0.22
10KB-100KB (M)	10.32	16.59	5.89	1.56
100KB-1MB (L)	9.12	3.52	83.8	1.53
1MB- (XL)	0.41	9.1	7.04	96.7

W1	Web-server rack at Facebook [44].
W2	Hadoop cluster at Facebook [44].

Table 1: Flow size distribution of realistic workloads.

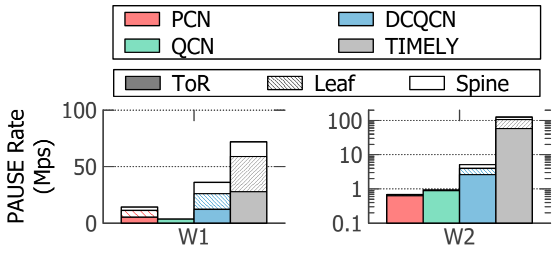
7.3 Parameter sensitivity

As discussed in § 6.2, the minimal and maximal of weight factor $w_{\min}$ and $w_{\max}$ determine the convergence speed and oscillation amplitude in steady state. To evaluate the parameter sensitivity, we repeat the concurrent burst simulation with different $w_{\min}$ and $w_{\max}$ values. Fig.13 shows the result. With the changes of $w_{\min}$ and $w_{\max}$, PCN can always achieve the satisfied performance. As $w_{\max}$ decreases, switch S0 receives fewer PFC PAUSEs from its downstream device, but the 99%-tile of FCT grows a little. Meanwhile, the value of $w_{\min}$ has almost no impact on pause rate, but the small $w_{\min}$ increases FCT slightly. The results indicate that our recommended parameter settings are proper.

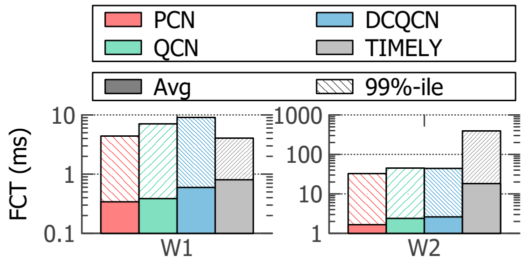
7.4 Realistic Workloads

In this subsection, we evaluate the performance of PCN with realistic workload.

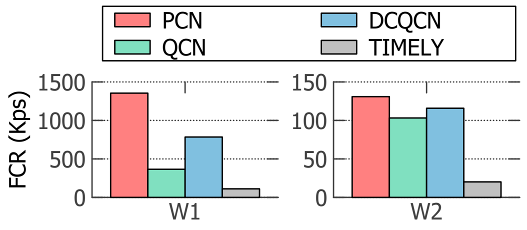
Scenario: We consider an 8-pod clos network. Each pod consists of 2 Leafs, 4 ToRs, and 64 hosts, and communicates with other pods through 8 spines. The link capability is 10Gbps below ToRs and 40Gbps above them, and the link delay is $5\mu s$. The over-subscription ratio is 1:1 at the ToR switch layer, so does in other layers. To support multi-path



(a) PAUSE Rate



(b) FCT



(c) FCR

Figure 14: Performance for realistic workloads.

capability, Equal Cost Multi Path (ECMP) routing scheme is used. In this configuration, the congestion tends to occur at the last hop. When PFC is employed to guarantee losslessness, the root congestion at the last hop may spread to the whole network.

Workloads: We choose two different workloads, whose flow size distribution is listed in Table 1. These two workloads are typical traffic pattern in operation data centers: (1) most flows are short, and (2) most traffic is constituted by few but large flows. The difference is that W2 contains more heavy-tailed flows.

We generate over 50 thousands of flows with exponentially distributed inter-arrival time, and configure the target load at 0.6 for ToR down-links. The source and destination of each flow are arbitrarily selected with a random in-cast ratio ranging from 1 to 15.

Fig.14 presents the results. The generating rates of PFC PAUSEs from different switch layers are drawn in Fig.14(a), where the solid bat at the bottom indicates the PAUSE rate from the ToRs, the middle stripe bar denotes that from the Leafs, and the top empty bar shows that from the Spines. In Fig.14(b), we draw the statistical FCT of all flows, and Fig.14(c) shows the flow completion rate (FCR) i.e., the number of completed flows per second. Subsequently, we compare the performance of PCN with QCN, DCQCN and TIMELY under different workloads.

(1) W1 contains the most S size flows in number and the most L size flows in bytes. Under this workload, the network congestion condition would change dramatically. Although PCN triggers 5.73x more PFC PAUSEs than QCN, it achieves 1.60x faster 99%-ile FCT and 3.70x larger FCR. This is because QCN reduces the sending rate of large flows so drastically that the network becomes seriously underload. Since PCN can rapidly detect the congestion point and adjust the rate of congested flows, short flows experience a low queuing delay and complete quickly. This can improve the overall FCT and increase FCR. Compared with DCQCN and TIMELY, PCN avoids 64% and 75% PFC PAUSEs, speeds up 1.75x and 2.35x in average FCT, and obtains 1.73x and 12.16x FCR, respectively.

(2) W2 is significantly heavy-tailed, where the S size flows occupy almost 80% of the number and less than 1% of the bytes, while the XL size flows only take less than 10% of the number but occupy almost all bytes. Under this workload, PCN suppresses 35%/89%/99% PAUSEs, speeds up 1.44x/1.57x/10.96x in average FCT and achieves 1.27x/1.13x/6.5x more FCR compared with QCN, DCQCN and TIMELY, respectively.

7.5 External Evaluations

Furthermore, we conduct external evaluations to explore PCN's performance in more scenarios. The detailed descriptions are in Appendix B, and the main findings are five folds.

Flow Scalability: PCN can hold as many as 1024 concurrent long flows, guaranteeing few PFC PAUSEs, low and stable queue length, near-full network utilization, as well as good fairness.

Adversarial Traffic: When facing dynamic flows entering and exiting with an interval of 10∼100 control loops, the end-to-end congestion control schemes fail to start the fast rate increasing algorithm. Compared with QCN, DCQCN and TIMELY, PCN can alleviate but not fully eliminate the interruption from adversarial traffic.

Multiple Bottlenecks: In the parking lot scenario with N bottlenecks, PCN allocates bandwidth following *proportional fairness*. That is, it allocates $\frac{1}{N+1}$ of capacity to the flow that passes all N bottlenecks.

Multiple Priorities: When concurrent burst in higher priority leads to severe oscillation of available bandwidth in lower priority, PCN triggers less PAUSE compared with DCQCN and TIMELY. Consequently, PCN outperforms other schemes in speeding up the overall flow completion.

Deadlock: PCN can not essentially prevent PFC deadlock, neither can other end-to-end congestion control schemes, but can significantly decrease the probability of deadlock. Compared to DCQCN and TIMELY, PCN can reduce 79.2% and 96.7% of deadlocks, respectively.

8 Related Work

The lossless switching fabric is a lasting topic. Here we only present a brief survey on the related work of lossless Ethernet and its congestion management, as well as receiver-driven rate control schemes.

Scaling RDMA over data centers. There are two lines in scaling RDMA over data centers. The first line, such as DCB [4] and RoCE [15, 26], attempts to enhance Ethernet with lossless property using PFC. It requires little modification to the well-tested RDMA transport stack but involves new issues caused by PFC. So an appropriate end-to-end congestion control scheme is needed. And the second line, such as Resilient RoCE [34] and IRN [37], tries to improve the RDMA transport stack to tolerate packet loss. Thus it can scale RDMA over lossy networks. We prefer the first line. We think the lossless Ethernet is more potential. On one hand, not just for RDMA, lossless Ethernet makes it easier to enable various well-tested transport protocols in data centers. It does not require NICs to support selective retransmission using the limited storage resources. On the other hand, lossless Ethernet can avoid retransmission of lost packets, and then can improve both network latency and throughput performance.

Lossless Ethernet switching fabric. It is always attractive to build cost-effective, efficient and large-scale lossless switching fabric leveraging commodity Ethernet chips. The related studies broadly follow three fundamental ways, which are reservation, explicit allocation, and congestion management. TDMA Ethernet [47] advocates reserving slots by deploying TDMA MAC layer. Fastpass [43] conducts explicit bandwidth allocations by a centralized arbiter to determine the time at which each packet should be transmitted and the path it should take. Whether TDMA Ethernet or Fastpass, they leverage non-conflict bandwidth access to build lossless Ethernet. However, due to slot wastage and unneglectable signal overheads, their flexibility and scalability in large-scale and ultra-high speed networks need to be further validated in practice. The third approach is to enhance traditional lossy Ethernet by introducing congestion management.

Congestion management for lossless Ethernet. IEEE DCB task group [4] defines the congestion management framework and develops concrete mechanisms, including PFC [6] and QCN [5], to enhance traditional Ethernet to be Converged Ethernet where losslessness should naturally be indispensable. To enable RoCE deployment in large-scale IP-routed data center networks, DCQCN [49] is developed through replacing the congestion notification mechanism defined in QCN with ECN in Layer 3, and then stitching together pieces of rate adjusting laws from QCN [5] and DCTCP [12]. TIMELY [36] follows the implicit congestion detection mechanism developed by TCP Vegas [20] and uses delay measurements to detect congestion, and then adjusts transmission rates according to RTT gradients. Both explicit and implicit congestion detection mechanisms in existing end-to-end congestion control schemes cannot identify the real congested flows, thus the performance issues in lossless Ethernet, such as HoL, congestion spreading and unfairness, are hardly solved essentially. In addition, IEEE 802.1 pQcz [7] has been supplemented to prevent PFC harming victim flows by isolating congestion. However, modification of current commodity switches is required to add more functions. In comparison, the congestion detecting mechanism in our PCN can correctly identify congested flows, moreover is practicable and back-compatible, which endows fundamental advantages for congestion management in lossless Ethernet.

Receiver-driven rate control. Recently, a series of receiver-driven rate control schemes have been proposed, such as ExpressPass [21], NDP [27] and Homa [38]. ExpressPass proactively controls congestion even before sending data packets by shaping the flow of credit packets in receivers. Both NDP and Homa also use the receiver-driven method to allocate priority to different flows in lossy data center networks. The receiver-driven rate adjustment in our PCN not only has the similar benefit of matching the incoming traffic load to the network capacity in one RTT, but also can drastically mitigate PFC triggers in one RTT as well, which is especially appropriate for lossless Ethernet.

9 Conclusion

This paper re-architects congestion management for lossless Ethernet, and proposes Photonic Congestion Notification (PCN), which is appropriate for lossless Ethernet by two ingenious designs: (*i*) a novel congestion detection and identification mechanism to recognize which flows are really responsible for congestion; (*ii*) a receiver-driven rate adjustment scheme to alleviate congestion in as fast as one loop control round, i.e., one RTT. PCN can be easily implemented on commodity switches with a little modification. Extensive experiments and simulations confirm that PCN greatly improves performance, and significantly mitigates PFC PAUSE messages and reduces the flow completion time under realistic workload.

Acknowledgments

The authors gratefully acknowledge the shepherd Rachit Agarwal and the anonymous reviewers for their constructive comments. This work is supported in part by the National Key Research and Development Program of China (No.2018YFB1700103), and by National Natural Science Foundation of China (NSFC) under Grant 61872208.

References

[1] DPDK. http://dpdk.org/.

[2] DPDK test-pipeline. https://github.com/DPDK/dpdk/tree/master/test/test-pipeline.

[3] FCoE (Fibre Channel over Ethernet). http://fcoe.com/.

[4] IEEE 802.1 Data Center Bridging Task Group. http://ieee802.org/1/pages/dcbridges.html.

[5] IEEE 802.1 Qau - Congestion Notification. https://1.ieee802.org/dcb/802-1qau/.

[6] IEEE 802.1 Qbb - Priority-based Flow Control. https://1.ieee802.org/dcb/802-1qbb/.

[7] IEEE P802.1 Qcz - Congestion Isolation. https://1.ieee802.org/tsn/802-1qcz/.

[8] Benefits of Storage Area Network/Local Area Network (SAN/LAN) Convergence, 2009. http://i.dell.com/sites/doccontent/business/smb/sb360/en/Documents/benefits-san-lan-convergence.pdf.

[9] Monitoring and troubleshooting, one engineer's rant, 2011. https://www.nanog.org/meetings/nanog53/presentations/Monday/Hoose.pdf.

[10] Keeping cloud-scale networks healthy, 2016. https://video.mtgsf.com/video/4f277939-73f5-4ce8-aba1-3da70ec19345.

[11] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 63–74. ACM, 2008.

[12] Mohammad Alizadeh, Albert G Greenberg, David A Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center tcp (dctcp). In *ACM SIGCOMM*, 2010.

[13] Mohammad Alizadeh, Abdul Kabbani, Tom Edsall, Balaji Prabhakar, Amin Vahdat, and Masato Yasuda. Less is more: Trading a little bandwidth for ultra-low latency in the data center. In *In Proceedings of USENIX NSDI*, 2012.

[14] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick Mckeown, Balaji Prabhakar, and Scott Shenker. pfabric: minimal near-optimal datacenter transport. In *ACM SIGCOMM*, 2013.

[15] InfiniBand Trade Association. Infiniband Architecture Specification Volume 1 Release 1.2.1 Annex A17: RoCEv2, 2014. https://cw.infinibandta.org/document/dl/7781.

[16] InfiniBand Trade Association. Infiniband Architecture Specification Volume 2 Release 1.3.1. 2016. https://cw.infinibandta.org/document/dl/8125.

[17] Wei Bai, Li Chen, Kai Chen, Dongsu Han, Chen Tian, and Hao Wang. Information-agnostic flow scheduling for commodity data centers. In *In Proceedings of USENIX NSDI*, 2015.

[18] Theophilus Benson, Aditya Akella, and David A Maltz. Network traffic characteristics of data centers in the wild. In *Proc. of the 10th ACM SIGCOMM conference on Internet measurement*, pages 267–280. ACM, 2010.

[19] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. Understanding data center traffic characteristics. In *Proc. of the 1st ACM workshop on Research on enterprise networking*, pages 65–72. ACM, 2009.

[20] Lawrence S Brakmo, Sean W O'Malley, and Larry L Peterson. *TCP Vegas: New techniques for congestion detection and avoidance*, volume 24. ACM, 1994.

[21] Inho Cho, Keon Jang, and Dongsu Han. Credit-scheduled delay-bounded congestion control for datacenters. In *Proc. of ACM SIGCOMM*, 2017.

[22] Charles Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, 32(2):406–424, 1953.

[23] Peter Gao, Akshay Narayan, Sagar Karandikar, Joao Carreira, Sangjin Han, Rachit Agarwal, Sylvia Ratnasamy, and Scott Shenker. Network requirements for resource disaggregation. In *In Proceedings of USENIX NSDI*, 2016.

[24] Peter Gao, Akshay Narayan, Gautam Kumar, Rachit Agarwal, Sylvia Ratnasamy, and Scott Shenker. phost: distributed near-optimal datacenter transport over commodity network fabric. In *ACM CoNEXT*, 2015.

[25] Albert G Greenberg, James R Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A Maltz, Parveen Patel, and Sudipta Sengupta. Vl2: a scalable and flexible data center network. In *ACM SIGCOMM*, 2011.

[26] Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni, Jianxi Ye, Jitu Padhye, and Marina Lipshteyn. Rdma over commodity ethernet at scale. In *ACM SIGCOMM*, 2016.

[27] Mark Handley, Costin Raiciu, Alexandru Agache, Andrei Voinescu, Andrew W Moore, Gianni Antichi, and Marcin Wojcik. Re-architecting datacenter networks and stacks for low latency and high performance. In *ACM SIGCOMM*, 2017.

[28] Shuihai Hu, Yibo Zhu, Peng Cheng, Chuanxiong Guo, Kun Tan, Jitendra Padhye, and Kai Chen. Deadlocks in datacenter networks: Why do they form, and how to avoid them. In *ACM HotNets*, 2016.

[29] Shuihai Hu, Yibo Zhu, Peng Cheng, Chuanxiong Guo, Kun Tan, Jitendra Padhye, and Kai Chen. Tagger: Practical pfc deadlock prevention in data center networks. In *ACM CoNEXT*, 2017.

[30] Raj Jain, Arjan Durresi, and Gojko Babic. Throughput fairness index: An explanation. 1999.

[31] Virajith Jalaparti, Peter Bodik, Srikanth Kandula, Ishai Menache, Mikhail Rybalkin, and Chenyu Yan. Speeding up distributed request-response workflows. 2013.

[32] Cheng Jin, David X Wei, and Steven H Low. Fast tcp: motivation, architecture, algorithms, performance. In *IEEE INFOCOM 2004*, volume 4, pages 2490–2501. IEEE, 2004.

[33] Rishi Kapoor, Alex C Snoeren, Geoffrey M Voelker, and George Porter. Bullet trains: a study of nic burst behavior at microsecond timescales. In *ACM CoNEXT*, pages 133–138. ACM, 2013.

[34] John F. Kim. Resilient roce relaxes rdma requirements. http://kr.mellanox.com/blog/2016/07/resilient-roce-relaxes-rdma-requirements/, 2016.

[35] Radhika Mittal. TIMELY algorithm to compute new rate, 2015. http://radhikam.web.illinois.edu/timely-code-snippet.cc.

[36] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. Timely: Rtt-based congestion control for the datacenter. In *ACM SIGCOMM*, 2015.

[37] Radhika Mittal, Alexander Shpiner, Aurojit Panda, Eitan Zahavi, Arvind Krishnamurthy, Sylvia Ratnasamy, and Scott Shenker. Revisiting network support for rdma. In *ACM SIGCOMM*, 2018.

[38] B Montazeri, Yilong Li, Mohammad Alizadeh, and John K Ousterhout. Homa: a receiver-driven low-latency transport protocol using network priorities. In *ACM SIGCOMM*, 2018.

[39] Timothy Prickett Morgan. The Tug of War between Infiniband and Ethernet, 2017. https://www.nextplatform.com/2017/10/30/tug-war-infiniband-ethernet/.

[40] Jayaram Mudigonda, Praveen Yalagandula, Jeffrey C Mogul, Bryan Stiekes, and Yanick Pouffary. Netlord: A scalable multi-tenant network architecture for virtualized datacenters. In *Proc. of ACM SIGCOMM*, 2011.

[41] Mohammad Noormohammadpour and Cauligi S Raghavendra. Datacenter traffic control: Understanding techniques and tradeoffs. *IEEE Communications Surveys & Tutorials*, 20(2):1492–1525, 2018.

[42] Jim O'Reilly. Future of the Ethernet drive may hinge on NVMe over Ethernet. http://searchstorage.techtarget.com/feature/Future-of-the-Ethernet.

[43] Jonathan Perry, Amy Ousterhout, Hari Balakrishnan, Devavrat Shah, and Hans Fugal. Fastpass: A centralized zero-queue datacenter network. *ACM SIGCOMM Computer Communication Review*, 44(4):307–318, 2015.

[44] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C Snoeren. Inside the social network's (datacenter) network. In *ACM SIGCOMM*, 2015.

[45] Alex Shpiner, Eitan Zahavi, Vladimir Zdornov, Tal Anker, and Matty Kadosh. Unlocking credit loop deadlocks. In *ACM HotNets*, 2016.

[46] B. Stephens, A. L. Cox, A. Singla, J. Carter, C. Dixon, and W. Felter. Practical dcb for improved data center networks. In *IEEE INFOCOM*, 2014.

[47] Bhanu Chandra Vattikonda, George Porter, Amin Vahdat, and Alex C Snoeren. Practical tdma for datacenter ethernet. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 225–238. ACM, 2012.

[48] Yibo Zhu. NS-3 simulator for RDMA over Converged Ethernet v2 (RoCEv2), 2016. https://github.com/bobzhuyb/ns3-rdma.

[49] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. Congestion control for large-scale rdma deployments. In *ACM SIGCOMM*, 2015.

[50] Yibo Zhu, Nanxi Kang, Jiaxin Cao, Albert Greenberg, Guohan Lu, Ratul Mahajan, Dave Maltz, Lihua Yuan, Ming Zhang, Ben Y Zhao, et al. Packet-level telemetry in large datacenter networks. In *ACM SIGCOMM Computer Communication Review*, volume 45, pages 479–491. ACM, 2015.

A Theoretical Analysis

We build a fluid model of PCN to exhibit PCN's good performance of fast convergence, fairness, and stability. The main symbols are summarized in Table 2.

A.1 Fluid Model

Suppose N sources share the bottleneck link with capacity C. For each source i ($i = 1, \cdots, N$) and each CNP generating time $t_k = kT (k = 1, 2, \cdots)$, $R_i(k), \widehat{R_i}(k)$ and $Q(k)$ denote the sending rate and receiving rate of source i, and the queue length in bottleneck link, respectively. Clearly, the queue length $Q(k)$ evolutes as follows

$$Q(k+1) = \max\{0, Q(k) + [\sum R_i(k) - C]T\} \tag{3}$$

As the associated egress port is not paused by its downstream device and excessive packets are accumulated in buffer, we regard the flow through this port is congested. Define the congestion indicator function $p(k)$

$$p(k) = \begin{cases} 0, & if\ Q(k+1) = 0 \\ 1, & if\ Q(k+1) > 0 \end{cases} \tag{4}$$

When N sources share the bottleneck capacity C by the sending ratio $\eta_i(k) = \frac{R_i(k)}{\sum_{j=1}^{N} R_j(k)}$. If the link is underflow, all incoming traffic can arrive their receiver side. Consequently, the receiving rate of each source satisfies

$$\widehat{R_i}(k) = p(k)\eta_i(k)C + (1 - p(k))R_i(k) \tag{5}$$

With probability $p(k)$ and receiving rate $\widehat{R_i}(k)$, source i will change its sending rate and the weight factor according to the corresponding adjustment rule. Thus, we have,

$$\begin{aligned} R_i(k+1) = p(k)\widehat{R_i}(k)(1 - w_{\min}) + \\ (1 - p(k))[R_i(k)(1 - w(k)) + Cw(k)] \end{aligned} \tag{6}$$

and

$$\begin{aligned} w(k+1) = p(k)w_{\min} + \\ (1 - p(k))[w(k)(1 - w(k)) + w_{\max} w(k)] \end{aligned} \tag{7}$$

The dynamic behavior of PCN congestion management system can be described using Equation (3), (4), (5), (6) and (7). Based on this fluid model, we analyze PCN's properties in terms of convergence, fairness, and stability.

Variable	Description
R_i	Sending rate of Flow i
η_i	Bandwidth allocation ratio, $\eta_i = \frac{R_i}{\sum R_i}$
$\widehat{R_i}$	Receiving rate of Flow i
w	Weight factor
Q	Bottleneck queue length
T	CNP generating period
k	Sequence of CNP generating periods
C	Bottleneck link capacity

Table 2: Variables of fluid model

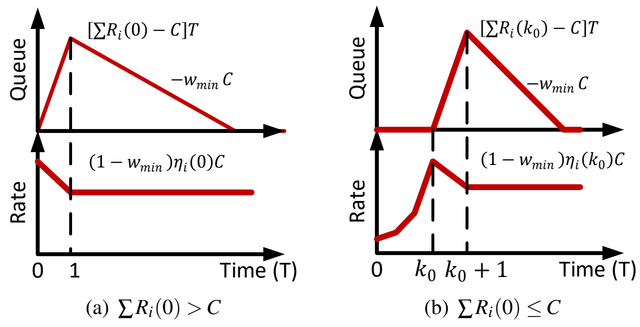


Figure 15: Convergence of PCN.

A.2 Performance Analysis

A.2.1 Convergence

Without loss of generality, assume the queue associated with bottleneck link is empty and the sending rate of N flows is arbitrary at initial time 0. Hence, there are two cases.

(1) $\sum R_i(0) > C$: If the total rate exceeds the bottleneck capacity, the corresponding queue increases and all flows conduct the rate-decrease adjustment, thus,

$$\begin{cases} Q(1) &= [\sum R_i(0) - C]T \\ R_i(1) &= (1 - w_{\min})\eta_i(0)C \end{cases}$$

Note that the total rate $\sum R_i(1) = (1 - w_{\min})C < C$, thus the buffer will be drained out in next several periods. When the port is in congestion, the rate-decrease algorithm takes effect. Since $\frac{R_i(1)}{\sum R_i(1)} = \frac{R_i(0)}{\sum R_i(0)}$, we have

$$\begin{cases} Q(k) &= [\sum R_i(0) - C]T - (k-1)w_{\min}CT \\ R_i(k) &= (1 - w_{\min})\eta_i(0)C \end{cases} \tag{8}$$

Equation (8) implies that the total sending rate $\sum R_i(k)$ converges to $(1 - w_{\min})C$ in one control loop, while the queue length approaches to zero after $\lceil 1 + \frac{\sum R_i(0) - C}{w_{\min}C} \rceil$ control loops. The detail evolutions are illustrated in Fig.15(a).

(2) $\sum R_i(0) \le C$: If the total rate is less than the bottleneck capacity, all flows run the rate-increase algorithm. Eventually, the total sending rate will exceed the link capacity after K_0 control loops, where $K_0 < 10$ according to Fig.10. Therefore, we have $\sum R_i(k_0) > C$ and $Q(k_0) = 0$. Subsequently, the dynamic behavior of both queue and aggregate rate drawn in Fig.15(b) are similar with those in above case.

In a word, PCN can achieve convergence of total rate towards the bottleneck capacity as fast as in only one control loop, and drain out backlog packets.

A.2.2 Fairness

Suppose the above convergence phase ends at the start of k_1 control loop, where the buffer has been drained out, and the sending rate of flow i is increased from $(1 - w_{\min})\eta_i(k_0)C$, then,

$$\begin{cases} Q(k_1) &= 0 \\ R_i(k_1) &= (1 - w_{\min})^2\eta_i(k_0)C + w_{\min}C \end{cases} \tag{9}$$

and

$$\eta_i(k_1) = \frac{R_i(k_1)}{\sum R_i(k_1)} = \frac{(1 - w_{\min})^2\eta_i(k_0) + w_{\min}}{(1 - w_{\min})^2 + Nw_{\min}} \tag{10}$$

Note that $\sum R_i(k_1) = [1 + (N - 2 + w_{\min})w_{\min}]C > C$, the bottleneck link becomes real congested and the RP conducts the rate-decrease adjustment in the next period, thus we have

$$\begin{cases} Q(k_1 + 1) &= (N - 2 + w_{\min})w_{\min}CT \\ R_i(k_1 + 1) &= (1 - w_{min})\eta_i(k_1)C \end{cases} \tag{11}$$

Since the aggregate sending rate will become below C, the backlog packets in queue will be drained out at a rate of $w_{\min}C$ per period and the sending rate is kept, then

$$\begin{cases} Q(k_1 + k) &= (N - k - 1 + w_{\min})w_{\min}CT \\ R_i(k_1 + k) &= (1 - w_{min})\eta_i(k_1)C \end{cases}$$

Obviously, the buffer will become empty again at the starting of $k_1 + N$ control loop, and the sending rate of flow i is increased from $(1 - w_{\min})\eta_i(k_1)C$, thus

$$\begin{cases} Q(k_1 + N) &= 0 \\ R_i(k_1 + N) &= (1 - w_{\min})^2\eta_i(k_1)C + w_{\min}C \end{cases} \tag{12}$$

and

$$\eta_i(k_1 + N) = \frac{(1 - w_{\min})^2\eta_i(k_1) + w_{\min}}{(1 - w_{\min})^2 + Nw_{\min}} \tag{13}$$

Comparing equation (9) and (12), we can find that PCN repeats the one period of rate-increase and N-1 periods of rate-decrease, as illustrated in Fig.16. And from equation (10) and (12), we also obtain the following dynamic evolution of bandwidth allocation ratio of each flow,

$$\begin{aligned} \eta_i(k_1 + kN) &= a\eta_i(k_1 + (k-1)N) + b \\ &= a^k\eta_i(k_1) + \textstyle\sum_{j=0}^{k-1} a^j b \\ &= a^{k+1}\eta_i(k_0) + \textstyle\sum_{j=0}^{k} a^j b \end{aligned}$$

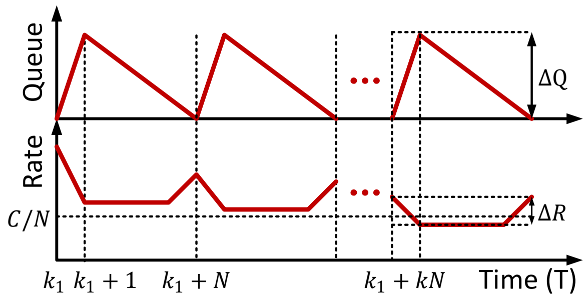


Figure 16: Dynamic behavior of PCN.

where $a = \frac{(1-w_{\min})^2}{(1-w_{\min})^2+Nw_{\min}} \in (0,1)$, $b = \frac{w_{\min}}{(1-w_{\min})^2+Nw_{\min}}$. Consequently, as $k \to \infty$, there is

$$\eta_i(k_1+Nk) \to \frac{b}{1-a} = \frac{1}{N} \qquad (14)$$

That is, PCN can always achieve fair bandwidth allocation regardless of the initial sending rates of flows and parameter settings.

A.2.3 Stability

Finally, we show the steady state behavior of PCN. As illustrated in Fig.16, the queue length varies between 0 and $Q(k_1+1)$ periodically. Based on equation (11), the maximal queue oscillation ΔQ satisfies

$$\Delta Q = Q(k_1+1) = (N-2+w_{\min})w_{\min}CT \qquad (15)$$

Similarly, the sending rate also changes in each N control loops. As Fig.16 shows, the aggregate rate increases in k_1+kN period and decreases in k_1+kN+1 period. Note that each flow achieves fair bandwidth allocation ratio in the steady state, i.e., $\eta_i \to \frac{1}{N}$, thus we can obtain the following derivation based on equation (11) and (12),

$$\begin{aligned} R_i(k_1+kN) &= (1-w_{\min})\eta_i C + w_{\min}C \\ &\to [1+(N-1)w_{\min}]\tfrac{C}{N} \\ R_i(k_1+kN+1) &= (1-w_{\min})\eta_i C \\ &\to (1-w_{\min})\tfrac{C}{N} \end{aligned}$$

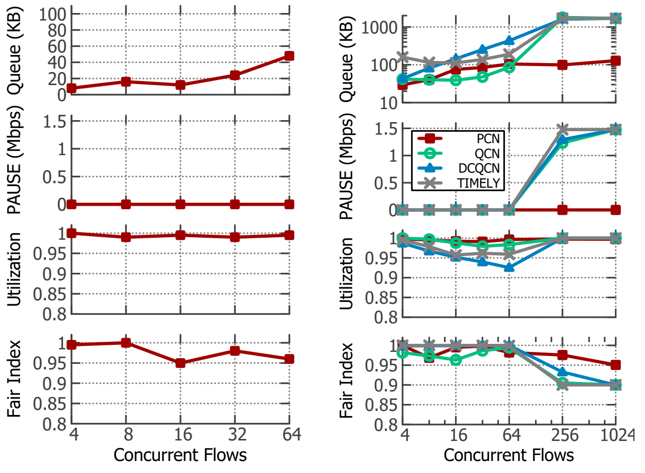
Therefore, the rate oscillation ΔR_i around the fair share $\frac{C}{N}$ satisfies

$$\Delta R_i \to w_{\min}C \qquad (16)$$

Equation (15) and (16) indicate the oscillations of both the queue and rate are bounded in steady state, i.e., the stability of PCN is fine.

B External Evaluations

In this section, we will explore how PCN performs in artificial cases, including *flow scalability*, *adversarial traffic*, *multiple bottlenecks*, *multiple priorities* and *deadlock*.



(a) PCN(Testbed) (b) PCN/QCN/DCQCN/TIMELY(ns-3)

Figure 17: Flow scalability test.

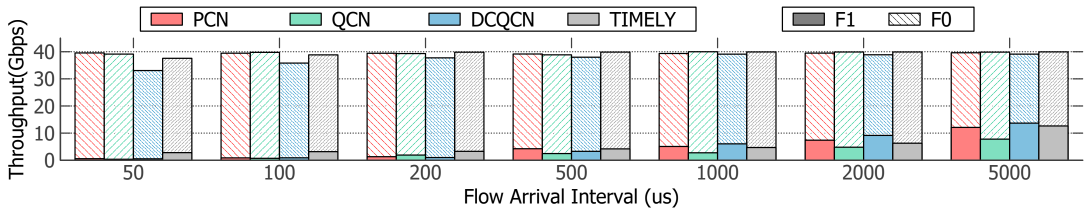
B.1 Flow Scalability

Using the simple 3-dumbbell topology in §7.1, we vary the number of flows from 4 to 64 (testbed) and 1024 (ns-3) to test the performance of PCN under more flows. The queue length, pause rate, link utilization and fairness are measured and calculated, and the results are presented in Fig.17.

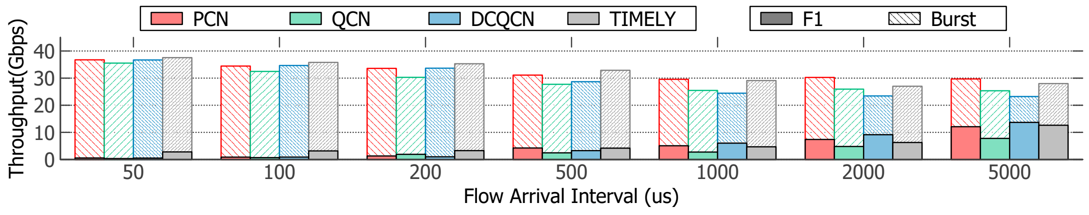
First, we measure the average queue occupancy and pause rate as the number of concurrent flows increases. In PCN, the average queue length is no more than 60KB and 100KB in the testbed and ns-3 simulator, respectively. At the same time, there are no PAUSE frames generated. In contrast, with $4 \sim 256$ flows, DCQCN's queue length grows with the number of flows, QCN and TIMELY keep the queue length around $50KB \sim 100KB$ and $100 \sim 200KB$, respectively. But QCN, DCQCN and TIMELY maintain a very high queue occupancy beyond 256 flows, which indicates the end-to-end congestion control fail to take effect. As for QCN, DCQCN and TIMELY, PFC is rarely triggered when the number of flows is less than 256, but persistent PAUSE frames are generated.

Second, we measure the utilization of bottleneck link. PCN achieves near 100% utilization in all case with both testbed and ns-3 simulator. QCN, DCQCN and TIMELY have a little under-utilization with the increase of concurrent flows, but recover full utilization with more than 256 flows. However, this recovery of link utilization is due to PFC rather than the end-to-end congestion control schemes.

Finally, we calculate the Jain's fairness index [30] using the throughput of each flow at $500ms$ interval. With a large number of flows, the fairness index of QCN, DCQCN and TIMELY drops significantly. Because they can not prevent PFC from persistent triggers, the inherent unfairness problem of PFC exhibits. On the opposite, PCN achieves good fairness in all cases.

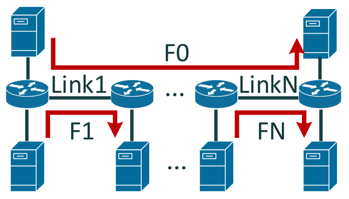


(a) Link S0→ S1

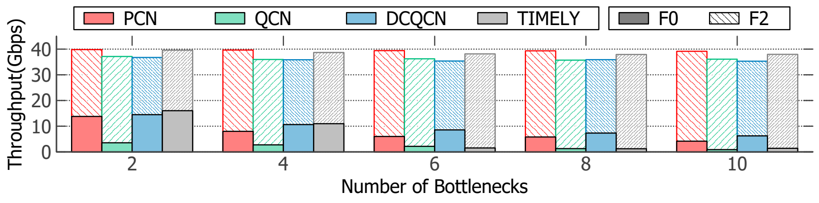


(b) Link S1→R1

Figure 18: Performance of various schemes under adversarial traffic.



(a) Parking-lot Topology



(b) Throughput of Link1

Figure 19: Performance of various schemes under multi-bottleneck scenario.

B.2 Adversarial Traffic

Subsequently, we test PCN using an adversarial traffic pattern. In the basic scenario in Fig.1, we set long flows F0 and F1 transmit persistently, and burst flows from H2∼H15 to R1 enter and exit the network at different intervals, varying from 50μs (1 control loop) to 5000μs (100 control loops). We simulate PCN, QCN, DCQCN and TIMELY, and measure the throughput of the two bottlenecks (link S0→S1 and S1→R1) and different flows. The results are drawn in Fig.18.

The first bottleneck, link S0→S1, is irrelative with the burst flows. Fig.18(a) shows that under PCN, QCN and TIMELY, link S0→S1 can achieve near-full utilization. But when the burst flows becomes more frequent, DCQCN trends to loss as high as 15% of throughput. This is because switch S1 pauses S0 when the burst flows make P2|S1 congested, and DCQCN conducts improper rate decrease for the victim flow F0.

The second bottleneck, link S1→R1, is frequently interrupted by the burst flows. Fig.18(b) exhibits that when the flow arrival interval shrinks, the congestion-relative flow F1 occupies lower throughput but the link utilization becomes larger. The performance issue occurs when the flow arrival interval is a little large (>500μs, 10 control loops). This means, their rate increase phase is interrupted by new-arrival flows. We can see that PCN keeps the link throughput at 30Gbps, while QCN, DCQCN and TIMELY remains at 23Gbps, 25Gbps and 29Gbps, respectively. That is, PCN can alleviate, but not eliminate, the interruption from adversarial traffic.

B.3 Multiple Bottlenecks

In a multi-bottleneck scenario, the NP-ECN method of PCN's CP may encounter several issues. On one hand, when the first congestion point marks ECN on all packets, the second congestion point may be paused, thus some flows are the victim but they have been marked with ECN already. On the other hand, flows through multiple congestion points may have a larger probability to be marked with ECN, resulting in unfairness. To test how PCN performs in multiple bottleneck scenario, we conduct a series of simulations using the parking lot topology in Fig.19(a). There are N bottlenecks and $N+1$ flows, where we set $N = 2,4,6,8,10$. F0 passes all the bottlenecks while other flows pass only one bottleneck. We measure the throughput of F0 and F1, and their sum is the throughput of link1. The result is drawn in Fig.19(b). Obviously, link1 achieves the similar utilization regardless of the number of bottlenecks. PCN can always provide more than 98% of link utilization, while the link utilization under QCN and DCQCN

Scheme	S(P1)	M(P2)	L(P3)	XL(P4)
PCN	0	0	0.10	171.49
QCN	0	0	0.28	110.49
DCQCN	0	0	0.26	175.02
TIMELY	0	0	1.09	210.36

Table 3: Generating rate of PFC PAUSEs for multiple priorities.

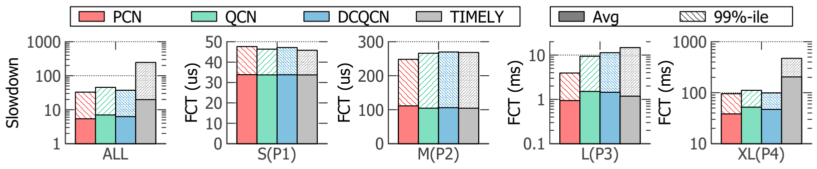


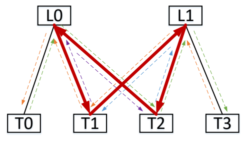
Figure 20: Average/99%-ile flow completion time for multiple priorities.

and TIMELY is 90%, 88% and 95%, respectively. Meanwhile, F0 is allocated less bandwidth than F2. Actually, the bandwidth allocation of PCN conforms to *proportional fairness*, where F0 obtains about $\frac{1}{N+1}$ of the capacity. QCN allocates F0 less than the proportional fairness. DCQCN allocates F0 more than the proportional fairness, but also less than the *max-min fairness*, i.e, half of the capacity.

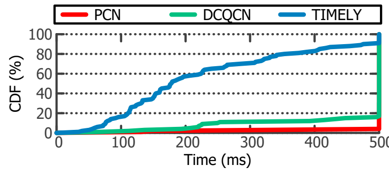
B.4 Multiple Priorities

The switching fabric in data center typically provides multiple priorities to improve performance, especially for minimizing flow completion time. The principle "short flow first" has been adopted in a series of works such as pHost [24], pFabric [14] and PIAS [17]. However, the concurrent burst in higher priority may trigger more PAUSE in lower priority, and impact the end-to-end congestion control schemes. To demonstrate and confirm this fact, we configure W1 and repeat the simulation in § 7.4, where the flows are classified into four priorities according to their size, namely, the S size flows are in the first priority and the XL flows are gathered in the fourth priority. The switches forward packets following the strict priority scheduling.

The generating rate of PFC PAUSEs and FCT for different priorities are listed in Table 3 and shown in Fig.20. For S and M flows in these two high priorities, few PFC PAUSE messages generate regardless of congestion control schemes. Thus, these flows can obtain almost the same FCT under three congestion control schemes. On the contrary, for the L and XL flows in the two low priorities, PFC PAUSEs can not be avoided. In this case, PCN triggers less PAUSE compared with DCQCN and TIMELY. QCN reduces PAUSE generated for XL flows by underutilizing available bandwidth. PCN outperforms the other three schemes in speeding up the overall flow completion time.



(a) Deadlock Topology



(b) Statistical Results

Figure 21: Performance of various schemes under deadlock scenario.

B.5 Deadlock Scenario

A common concern in Lossless Ethernet is that PAUSE can lead to deadlocks [28]. To explore PCN's effort to avoiding deadlock, we conduct a simple simulation using the topology illustrated in Fig.21(a). It comes from one pod in the clos network used in § 7.4, but link L0-T3 and link L1-T0 are failed, such that there is a *cycle buffer dependency* (CBD) as the red line draws. We simulate PCN, DCQCN and TIMELY with the W2 workload. The target load is 0.6 at ToR downlinks with in-cast ratio ranging from 1 to 15. Each scheme is tested for 1000 times and every simulation lasts for 500ms. We record the time when deadlock occurs, and draw the statistical results in Fig.21(b). Among the 1000 simulations, PCN only encounters with deadlock for 28 times, while DCQCN and TIMELY are deadlocked for 134 and 870 times, respectively. The advantage of PCN comes from the positive effect of mitigating PFC triggers and stopping congestion spreading.