

Congestion Detection in Lossless Networks

Yiran Zhang, Yifan Liu, Qingkai Meng, Fengyuan Ren

Tsinghua university

Beijing National Research Center for Information Science and Technology (BNRist)

Beijing, China

ABSTRACT

Congestion detection is the cornerstone of end-to-end congestion control. Through in-depth observations and understandings, we reveal that existing congestion detection mechanisms in mainstream lossless networks (i.e., Converged Enhanced Ethernet and InfiniBand) are improper, due to failing to cognize the interaction between hop-by-hop flow controls and congestion detection behaviors in switches. We define ternary states of switch ports and present Ternary Congestion Detection (TCD) for mainstream lossless networks. Testbed and extensive simulations demonstrate that TCD can detect congestion ports accurately and identify flows contributing to congestion as well as flows only affected by hop-by-hop flow controls. Meanwhile, we shed light on how to incorporate TCD with rate control. Case studies show that existing congestion control algorithms can achieve 3.3× and 2.0× better median and 99th-percentile FCT slowdown by combining with TCD.

CCS CONCEPTS

• Networks → Transport protocols.

KEYWORDS

Lossless Networks, Congestion Detection, Flow Control

ACM Reference Format:

Yiran Zhang, Yifan Liu, Qingkai Meng, Fengyuan Ren. 2021. Congestion Detection in Lossless Networks. In *ACM SIGCOMM 2021 Conference (SIGCOMM '21), August 23–27, 2021, Virtual Event, USA*. ACM, New York, NY, USA, 14 pages. <https://doi.org/10.1145/3452296.3472899>

1 INTRODUCTION

With the advantages of zero packet loss and low latency, lossless network has become an attractive trend in cluster systems. Packet loss can significantly affect application tail latency and throughput, thus revenue [12, 37, 57]. Besides, Remote Direct Memory Accesses (RDMA) technology also requires no packet loss to achieve the best performance [41]. Today two mainstream lossless networks are Converged Enhanced Ethernet (CEE) and InfiniBand. It is reported that InfiniBand is the most used interconnect in the TOP500 list and is deployed in 70% of High Performance Computing (HPC) systems in academic, research, and institutions [3, 5]. CEE is increasingly

adopted for data storage and latency-sensitive services in enterprise datacenters [24, 34, 38].

Lossless networks rely on hop-by-hop flow controls to guarantee zero packet loss under normal operations. InfiniBand employs Credit Based Flow Control (CBFC) [13], while Priority Flow Control (PFC) is developed to enable RDMA over Converged Ethernet (RoCE) [2]. However, hop-by-hop flow controls can cause collateral damages, including head-of-line blocking, unfairness and even deadlock [24, 28, 50, 56]. Therefore, end-to-end congestion control is needed and has received significant attention recently [16, 23, 38, 39, 51, 56].

Congestion detection is the cornerstone of end-to-end congestion control. Both Data Center Bridging (DCB) Task Group [1] and InfiniBand Specification [13] specify congestion management in the individual network but endow a similar framework where switches detect congestion and endpoints conduct rate control. Through in-depth observations and analysis, we reveal that existing congestion detection mechanisms in lossless networks are improper due to failing to cognize the impact of hop-by-hop flow controls. In lossless networks, switch ports can alternate between sending (ON) and pausing (OFF). Specifically, the ON-OFF sending pattern can impose unexpected effects on congestion detection behaviors in switches, including causing queue buildup and affecting the real input rate of pausing ports.

In the light of our observations and understandings, we define ternary states of switch ports and propose Ternary Congestion Detection (TCD) for lossless networks. The ternary states are *congestion* (1), *non-congestion* (0), and *undetermined* (∅). The port in a congestion state is where congestion occurs, with queue buildup not caused by OFF. We name the state of ports in the ON-OFF sending pattern as *undetermined* because its real input rate may be masked due to the ON-OFF sending pattern of its upstream ports. We elaborate on the state transitions among ternary states, especially transitions from the undetermined state to the congestion state, which is the key for congestion detection in lossless networks.

We present a uniform design of TCD for mainstream lossless networks, including CEE and InfiniBand. TCD can be implemented on switches with low overhead. Further, TCD can notify both congested flows (i.e., flows passing through congestion ports) and undetermined flows (i.e., flows only passing through undetermined ports) to endpoints.

Our contributions are summarized as follows:

- Providing an in-depth understanding of the impact of hop-by-hop flow controls on congestion detection in lossless networks and defining ternary states.
- Developing a novel congestion detection mechanism named TCD, which utilizes the ON-OFF sending pattern and the feature of queue length evolution to detect the transitions among ternary

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGCOMM '21, August 23–27, 2021, Virtual Event, USA

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-8383-7/21/08...\$15.00

<https://doi.org/10.1145/3452296.3472899>

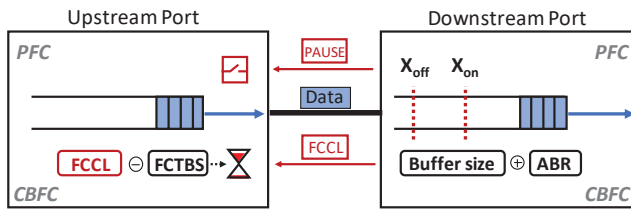


Figure 1: Flow controls for lossless networks.

states. Testbed and extensive simulations show that TCD can detect congestion ports accurately, and identify congested flows and undetermined flows.

- Incorporating TCD with existing congestion controls, including DCQCN [56], TIMELY [43], and IB CC [13]. Case studies show that existing congestion controls can achieve better performance by performing aggressive rate adjustment for congested flows while gentle rate adjustment for undetermined flows. Simulations with realistic workloads show that existing congestion controls combining TCD can improve the median and 99th-percentile FCT slowdown by 3.3× and 2.0×, respectively.

This work does not raise any ethical issues.

2 BACKGROUND

Generally, congestion detection is conducted by switches in existing lossless networks. Congestion management developed by DCB Task Group includes two significant components: Congestion Point (CP) responsible for congestion detection, and Reaction Point (RP) responsible for rate control. Analogously, InfiniBand Specification specifies the framework of congestion control in InfiniBand networks, where the switch detects congestion and Channel Adapter (CA) conducts injection throttling.

2.1 Congestion Detection Mechanisms in Lossless Networks

In CEE, DCB Task Group specifies QCN in IEEE 802.1 Qau [1]. In QCN, CP computes a congestion measure by sampling queue size. Because QCN does not support for L3 networks, DCQCN [56] is developed and widely adopted in CEE. Similar with QCN, DCQCN detects congestion based on queue size. CP marks packets with ECN [46] according to the RED algorithm [20]. A single bit ECN marking indicates the presence of congestion.

In InfiniBand, InfiniBand Specification specifies IB CC [13]. Switches are supposed to identify two cases: (a) If the queue of an output port exceeds a threshold and there are available credits to send packets, then it is the root cause; (b) If the queue of an output port exceeds a selected threshold and packets are delayed due to lack of credits, then it is the victim. Finally, a switch port is considered as congested and marks packets with Forward ECN (FECN) bit when it is a root cause of congestion. For CA, a single bit FECN marking indicates the presence of congestion.

To conclude, existing congestion detection mechanisms in CEE and IB essentially detect incipient congestion by queue size, which following the practice in traditional lossy networks. Besides utilizing the queue length as the primary congestion indicator, IB CC combines credits information to detect congestion.

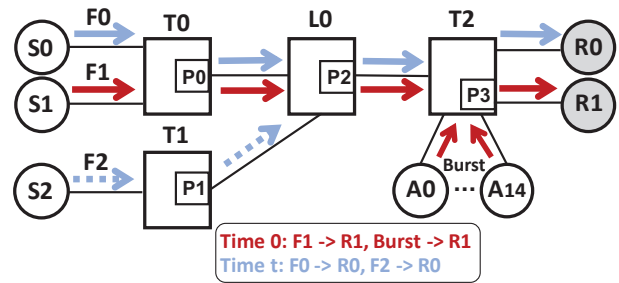


Figure 2: Typical network scenario.

2.2 Flow Controls for Lossless Networks

To guarantee the drop-free property, CEE employs Priority Flow Control (PFC), and InfiniBand employs Credit-Based Flow Control (CBFC), as shown in Figure 1. Here we briefly describe the operations of each flow control.

In PFC, the downstream switch sends a PAUSE frame to the upstream switch when the ingress queue exceeds a threshold X_{off} , and sends a RESUME frame when the ingress queue decreases to another threshold X_{on} . The upstream switch can only send when the egress queue is not paused.

In CBFC, the downstream switch maintains an Adjust Block Register (ABR) to record total received blocks. Besides, the downstream switch sends a Flow Control Credit Limit (FCCL) message to the upstream switch periodically, which contains the sum of allocated buffer size and ABR. The upstream switch maintains a Flow Control Total Block Sent (FCTBS) register to record total sent blocks. After receiving the FCCL message, the number of available credits is the difference between FCCL and FCTBS. Then the upstream switch can only send a packet when there are available credits.

PFC operates on a per-priority level. CBFC operates on a per Virtual Lane (VL) level. The number of priority queues and VLs is both limited in CEE switches and InfiniBand switches. In essence, PFC and CBFC function in the same way to avoid buffer overflow: the switch ports alternate between sending (ON) and pausing (OFF). If the port is pausing (OFF), the subsequent packets are queued to wait for available buffer space at the downstream switch.

3 OBSERVATIONS AND INSIGHTS

When hop-by-hop flow controls take effect (which is inevitable in lossless networks), the alternation between ON and OFF of switch ports may impose unexpected problems on congestion detection behaviors in switches. In this section, we conduct fine-grained simulations in a single congestion point scenario and multiple congestion points scenario to investigate the interaction issue between hop-by-hop flow controls and congestion detection.

3.1 Experimental Observations

3.1.1 Simulations setup. We adopt a topology as shown in Figure 2, which is a common unit in today's multi-rooted topologies (Fat-Tree [9], Leaf-Spine [11]). All links are 40Gbps with $4\mu s$ propagation delay. S1 sends long-lived flow F1 to R1. A0-A14 send concurrent 64KB bursts to R1. Assume F1 achieves 40Gbps at the beginning

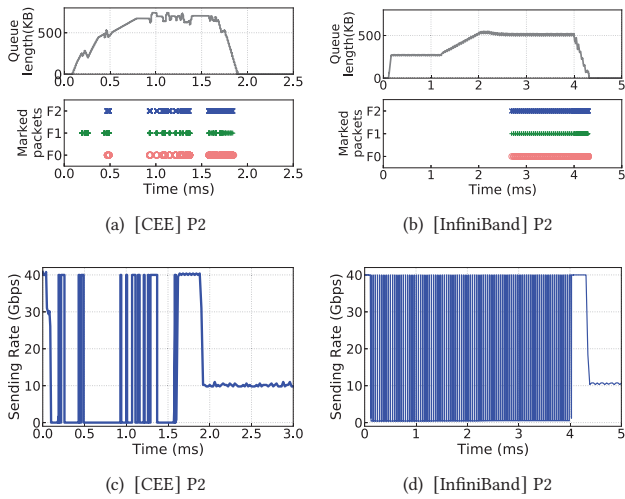


Figure 3: Single congestion point scenario. Queue length and sending rate.

of the simulation. In both single congestion point and multiple congestion points scenarios, bursts launch at time 0 and last for about 3ms. After bursts start, long-lived flow F0 and F2 send at a constant rate to R0 simultaneously. The bursts cannot be controlled by end-to-end congestion controls since the size is smaller than Bandwidth Delay Product (BDP). ECN (DCQCN [56]) and FECN (IB CC [13]) is the congestion detection mechanism in CEE and InfiniBand switches, respectively. The CC parameters are the recommended values in the literature [22, 56], and details are given in § 5.2. For CEE, the PFC threshold is 320KB. For InfiniBand, the ingress buffer size is 280KB.

3.1.2 Single congestion point scenario. In this scenario, port P3 is the only congestion point. F1 achieves 40Gbps at the beginning of the simulation and bursts launch at time 0. After bursts start, F0 and F2 send at constant 5Gbps. The rate of F1 has decreased below 15Gbps when F0 and F2 start, so port P0 is never congested. During bursts launching, the ideal bandwidth allocation of F1 is about 2.5 Gbps. Figure 3(a) and Figure 3(b) illustrate the queue length and marked packets at port P2 for CEE and InfiniBand.

For ports P0, P1 and P2, each input rate does not exceed the line rate. Hence, the three ports should not detect the presence of congestion. However, due to congestion spreading from port P3, three ports are paused intermittently. With constant input rate 5Gbps of F0 and F2, port P2 has a maximum queue length of over 500KB. As shown in Figure 3, both ECN and FECN detect congestion improperly at port P2. F0 is mistakenly marked with ECN at ports P2, although ports P2 is not the congestion point. Meanwhile, the congested flow F1 should only be marked with ECN at port P3. While partial packets of F1 are marked with ECN at ports P2, even it is not the real bottleneck. We omit the results at ports P1 and P0 as the marking behavior is similar.

In CEE, queue-based congestion detection alone is essentially inadequate. In lossy networks, generally, congestion is detected when switch ports have queue buildup. However, when PFC

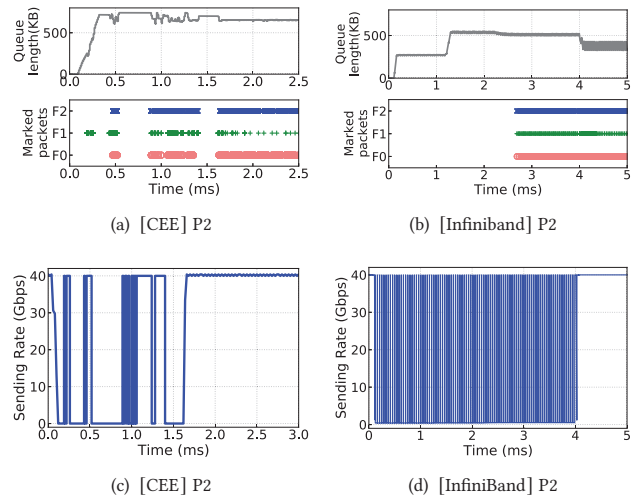


Figure 4: Multiple congestion points scenario. Queue length and sending rate.

takes effect in CEE, a port will receive PAUSEs intermittently, which may also lead to queue buildup. With dequeue marking [49, 54], CEE switches check the queue size and mark ECN when each packet dequeues. If the current egress queue length exceeds a threshold K_{max} (i.e., 200KB), the packet is marked with ECN. Because ECN detects congestion solely based on current queue length without considering whether the port receives PAUSEs before, switches can not distinguish between queue buildup caused by congestion and PAUSEs. As a result, an uncongested port may be mistakenly regarded as a congested port.

In InfiniBand, periodical updated credits of CBFC also confuse congestion detection. In InfiniBand, switches detect congestion combining credit information. Switches mark FECN when the queue length exceeds the queue threshold (i.e., 50KB) and the packet is not delayed due to lacking credit (i.e., not paused). However, as shown in Figure 3(b), partial packets of F0 and F2 are still marked with FECN at ports P2. Partial packets of F2 are marked at ports P1 (not shown in the figure). This is because IB CC overlooks the periodicity of credits in CBFC, which may mislead congestion detection. The ports affected by hop-by-hop flow controls can also receive a few credits periodically. Packets arriving after the arrival of new credits may be perceived to have sufficient credits to send out, resulting in improper detection behaviors.

3.1.3 Multiple congestion points scenario. In this scenario, port P2 becomes the second congestion point besides port P3. F1 achieves 40Gbps at the beginning of the simulation and bursts launch at time 0. After bursts start, F0 and F2 send at constant 25Gbps. When F0 and F2 start, the rate of F1 has decreased below 15Gbps, so port P0 is never congested. For CEE and InfiniBand, the sending rate of port P2 is depicted in Figure 4(c) and Figure 4(d), respectively. The queue length evolution is demonstrated in Figure 4(a) and Figure 4(b), respectively.

Due to congestion spreading from port P3, the sending rate of port P2 alternates between ON and OFF till 1.6ms/4ms (CEE/InfiniBand),

and the queue builds up. At the same time, port P2 is the other congestion point. After 1.6ms/4ms (CEE/InfiniBand), port P2 recovers to send normally, which indicates that congestion at port P3 is mitigated. Afterward, port P2 still has persistent queue accumulation. Note that this is different from the single congestion point scenario, where the accumulated queue at port P2 is drained out. The sending rate drops to 10Gbps after congestion at port P3 disappears (see Figure 3(c) and Figure 3(d)). Ideally, switches should be able to detect congestion when the sending rate alternates between ON and OFF. However, we observe that whether congestion happens at a port with an ON-OFF sending pattern may be unknowable. As illustrated in Figure 3(a) and Figure 4(a), the queue length evolutions are the same from 1.0ms to 1.5ms in CEE. Similarly, the queue length evolutions in InfiniBand are the same from 2.6ms to 4ms in Figure 3(b) and Figure 4(b).

The ON-OFF regulation of hop-by-hop flow controls masks the real input rate. Hop-by-hop flow controls are endowed with the propagation property. Once PAUSEs/no credits propagate to the upstream switches, the sending rate of the upstream ports is regulated to an ON-OFF pattern. Concretely, when upstream ports receive RESUME/credits, previously accumulated packets are sent out at line rate immediately. From the switch's view, even the real input rate is different, the incoming traffic rate is both line rates in a specific time scale. As a result, the ON-OFF arriving pattern may induce the same queue length evolutions.

Summary: Our detailed observations and analysis reveal that existing congestion detection mechanisms fail to cognize the impact of ON-OFF sending patterns, leading to inaccurate congestion detection results in lossless networks, such as CEE and InfiniBand. A closer observation on the port with an ON-OFF sending pattern tells that a port may have the same queue length evolution and sending pattern, but the real congestion state is diverse.

3.2 Understanding Port States in Lossless Networks

3.2.1 Definition of ternary states. Based on our observations in § 3.1, we define ternary states of switch ports in lossless networks as follows:

- **Non-congestion (0):** The port is persistently ON and without queue buildup.
- **Congestion (1):** The port is persistently ON. The output rate is at full rate with queue buildup not caused by OFF. For instance, in two scenarios in § 3.1, port P3 is the congestion port with full output rate. The accumulated queue is solely caused by congestion.
- **Undetermined (/):** The output rate is in an ON-OFF style. In two scenarios in § 3.1, ports P0, P1 and P2 are in the undetermined state. Note that the port may have queue buildup, but the cause of queue buildup is ambiguous. For instance, in the single congestion point scenario, the queue buildup in ports P0, P1 and P2 is owing to receiving PAUSEs/no credits. In the multiple congestion points scenario, the queue buildup in port P2 is also attributed to the excessive input traffic.

The undetermined state is a brand-new state as to traditional lossy networks due to the ON-OFF regulation of hop-by-hop flow controls. Generally, assume a port outputs freely with a maximum rate of the full rate in the beginning. Once the port is paused due

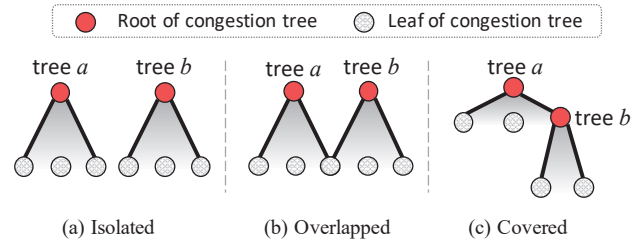


Figure 5: Congestion trees in lossless networks.

to receiving PAUSEs/no credits, it enters the undetermined state. The sending rate is regulated to an ON-OFF pattern. During each OFF period, the port is pausing, and all arriving packets are queued. During each ON period, the port sends at the full rate, dequeuing the accumulated packets. Note that packets may arrive at any moment during the undetermined state. Finally, the port recovers to send packets in a continuous ON pattern and leaves the undetermined state. After releasing from the undetermined state, a port may transform to a non-congestion or congestion state.

3.2.2 State transitions from the undetermined state. To further elaborate on the port states after a port leaves the undetermined state, we use the example of congestion trees and discuss port states under typical scenarios of multiple congestion trees, as shown in Figure 5.

(1) *Isolated.* Two congestion trees coexist in the network without overlapped leaves or roots. The roots are in the congestion state. If any congestion tree disappears, the leaves transit from the undetermined state to the non-congestion state, and the root transits from the congestion state to the non-congestion state. In the single congestion point scenario (§ 3.1.2), port P2 recovers to a non-congestion port after congestion spreading is eliminated.

(2) *Overlapped.* Two congestion trees have overlapped leaves but with different roots. Roots of congestion tree *a* and congestion tree *b* are in the congestion state. If one congestion tree disappears first, the overlapped leaves are still undetermined. Non-overlapped leaves will transit to the non-congestion state.

(3) *Covered.* This is a sophisticated case that is adopted in the multiple congestion points scenario (§ 3.1.3). A deeper congestion tree *a* covers another congestion tree *b*. The root of congestion tree *b* is also a leaf of congestion tree *a*. In the multiple congestion points scenario, port P2 is the covered root. Port P3 is the root of a deeper congestion tree. The covered root of congestion tree *b* is in an undetermined state. After congestion tree *a* disappears, the covered root of congestion tree *b* emerges, transforming into a congestion port.

The goal of congestion detection in lossless networks is to detect congestion ports in switches. Congestion ports are roots of congestion trees, where flows passing through them are the real culprits of congestion. The leaves of congestion trees are in an undetermined state. The actual state of leaves emerges after hop-by-hop flow controls cease to take effect. Besides, the chances are that new congestion tree forms right after a port dismisses from the undetermined state. Therefore, the key to congestion detection is to capture state transitions among ternary states.

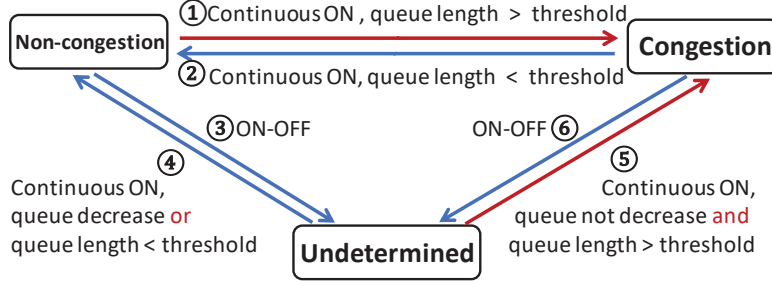


Figure 6: State transitions among ternary states.

4 TERNARY CONGESTION DETECTION

This section first presents the transition conditions among ternary states (§ 4.1). Then we give the conceptual ON-OFF model and uniform design of TCD (§ 4.2). We introduce the practical designs and parameter settings for CEE (§ 4.3) and InfiniBand (§ 4.4). Last, some implementation issues are discussed in § 4.5.

4.1 Congestion State Transitions

Figure 6 illustrates state transitions among ternary states in switch ports. Overall, state transitions ① and ② are the same as in lossy networks, where the queue length is the primary transition condition as the port is continuous ON. Since state transitions ③ to ⑥ are introduced by the new undetermined state in lossless networks, the crucial task is to figure out the transition conditions correlated with the undetermined state.

Assume the duration of an ON period is T_{on} . Our key insight is that a port in a continuous ON sending pattern has infinite T_{on} , while a port in an ON-OFF sending pattern usually has a limited T_{on} since the port is paused intermittently. Therefore, we can rely on a $\max(T_{on})$ to distinguish between the continuous ON sending pattern and ON-OFF sending pattern.

State transitions to the undetermined state (③ and ⑥). As soon as the port enters an ON-OFF sending pattern, the port transits to the undetermined state. The beginning of an ON-OFF sending pattern is paused (OFF) by hop-by-hop flow controls. When each packet dequeues (the port is ON), switches calculate the current T_{on} , which is the difference between the current timestamp and the timestamp when the latest OFF period ends. If the current T_{on} is smaller than $\max(T_{on})$, the port is considered as entering an ON-OFF sending pattern. Otherwise, the port does not transit to an undetermined state.

State transitions from the undetermined state to non-congestion or congestion state (④ and ⑤). The transition conditions from the undetermined state to non-congestion or congestion state are two-folds:

(1) The condition of leaving the undetermined state: from the ON-OFF sending pattern to the continuous ON pattern. Similar to state transitions ③ and ⑥, if T_{on} is larger than $\max(T_{on})$, we consider the port has been released from an undetermined state.

(2) The condition of entering congestion or non-congestion state after leaving the undetermined state: the feature of queue length evolution. After $\max(T_{on})$ expires, it may take some time for the

port to drain out the accumulated packets caused by OFF. If the queue length decreases after releasing packets from the undetermined state, indicating that the actual input rate does not exceed the line rate, so the port transits to the non-congestion state. Otherwise, the port transits to the congestion state. To obtain the trend of queue length evolution, switches can check the queue size every period T . Once the queue size increases in the current period T and exceeds the threshold, the current state transits to congestion. The current state changes to non-congestion if the queue size has already decreased to a low threshold value.

Congestion marking. Switches should conduct packet marking to notify endpoints when detecting transitions to the congestion state. We advocate that switches should also inform the transitions to the undetermined state to endpoints. Specifically, flows only passing through undetermined ports may be only victim flows. By providing information about the undetermined state, switches enable end-to-end congestion controls to conduct different rate adjustments for undetermined flows and congested flows according to different requirements and trade-offs. To this end, TCD supports ternary congestion notification, as shown in Table 1. In detail, packets may go through multiple ports with different states along the path. If a packet first passes through an undetermined port, then a congestion port, this packet should be considered as experiencing congestion. If a packet only goes through an undetermined port, the corresponding flow is undetermined. We use code point 10 to indicate the undetermined state encountered (UE). UE can only be marked when the current code point is not CE. Switches mark CE whenever the port is in a congestion state.

Based on the state transitions among ternary states, we propose Ternary Congestion Detection (TCD) in lossless networks, which can detect congestion, non-congestion, and undetermined state of switch ports. The state transitions ③, ④, ⑤, and ⑥ indicate that the foundation of TCD is the parameter $\max(T_{on})$. It determines when entering and leaving the undetermined state, hence the transition to the congestion state. The crucial problem is to determine the value of $\max(T_{on})$.

4.2 Conceptual ON-OFF Model

We build a conceptual ON-OFF model to describe T_{on} in lossless networks, as shown in Figure 7. During each ON period, the ingress queue length in the downstream port continues to increase. When the queue length increases to an upper bound B_1 , the upstream port is supposed to pause, and the ON period ends. The upstream port

Table 1: TCD marking scheme.

Code points	Meaning
00	Non TCD-Capable Transport
01	TCD-Capable Transport
10	Undetermined Encountered (UE)
11	Congestion Encountered (CE)

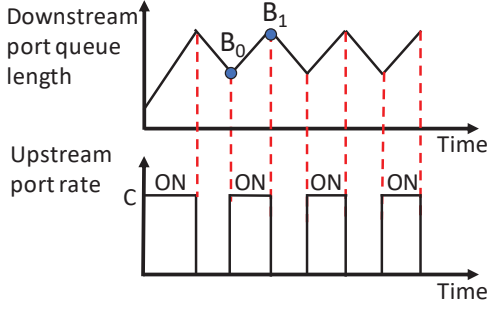


Figure 7: Conceptual ON-OFF model.

Table 2: Parameters of conceptual ON-OFF model

Parameter	Definition
C	Link capacity.
B_1/B_0	The ingress queue length to trigger OFF/ON.
R_i	Average input rate of a congested flow.
R_d	Average draining rate of a congested flow.
τ	The response time to start/stop transmitting.
ε	Congestion degree of a congested flow, which is defined as $(R_i - R_d)/C$.

is supposed to recover ON when the ingress queue length in the downstream port decreases to B_0 . The dynamic behavior of queue length and ON-OFF sending style repeat in the steady state. Key parameters are listed in Table 2.

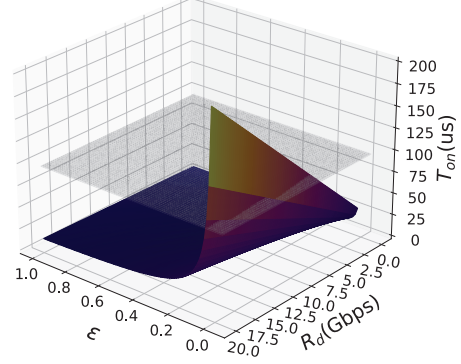
Ideally, the upstream port ceases transmitting when the downstream ingress queue length exceeds B_1 and recovers transmitting when the downstream ingress queue length is below B_0 . However, it takes some time for ON and OFF messages to take effect on the upstream port. Assume τ is the response time to cease transmitting (or start transmitting). Then the actual maximum and minimum of ingress queue length are related to τ . We have

$$T_{on} = \frac{B_1 - B_0 + \tau R_d}{R_i - R_d} + \tau \quad (1)$$

R_d is the average draining rate of a congested flow, which is the allocated bandwidth at the congested egress port. In the ON-OFF sending pattern, R_i is always larger than R_d . We define ε to represent the congestion degree experienced by a congested flow, with a larger value representing a larger extent of congestion. Substituting it into Eqn (1), we have

$$T_{on} = \frac{B_1 - B_0 + \tau R_d}{\varepsilon C} + \tau \quad (2)$$

The key of T_{on} is R_d and ε , which may vary under different congestion scenarios. To obtain $\max(T_{on})$, we first consider the upper bound of R_d . The general congestion scenario is multiple senders sending to one receiver simultaneously. Since the simplest case is two flows contending for one bottleneck link, the maximum bandwidth a congested flow can allocate among all congestion scenarios is $C/2$. Hence R_d satisfies $R_d \leq C/2$. Then the upper bound of T_{on} can be further obtained as follows:

Figure 8: The relationship between ε , R_d and T_{on} . The z-value of the flat plane is T_{on} when $\varepsilon = 0.05$.

$$T_{on} \leq \frac{2(B_1 - B_0) + \tau C}{2\varepsilon C} + \tau \quad (3)$$

According to Eqn (3), $\max(T_{on})$ is determined by ε , which can be close to zero as $R_i > R_d$. We conclude that if the difference between B_1 and B_0 is independent to ε , T_{on} is unbounded. For example, the recommended value of the difference between B_1 and B_0 is $2MTU$ in PFC [56]. However, a reasonable ε can adapt to most cases, which is enough for switches to distinguish between the ON-OFF sending pattern and continuous ON pattern. Figure 8 demonstrates the values of T_{on} when $\tau = 8\mu s$, $C = 40Gbps$. T_{on} increases first slowly then rapidly as ε decreases. Note that as ε decreases, the frequency that the hop-by-hop flow control is triggered also decreases. In practice, a minimal ε is improper since it leads to a very large $\max(T_{on})$. Too large $\max(T_{on})$ may defer the detection of the congestion state when a port is leaving an undetermined state. We empirically recommend ε to 0.05 because it can already cover most values of T_{on} . The recommended setting is also verified through simulations in § 5.1.4. We will further elaborate on the value of $\max(T_{on})$ in the practical designs for CEE and InfiniBand.

Figure 9 demonstrates the flowchart of TCD. The parameter $LAST_STATE$ records the port state detected at the latest moment. When each packet dequeues, the switch calculates current T_{on} and compares it with the pre-configured $\max(T_{on})$. If current T_{on} is smaller than $\max(T_{on})$, the port is detected as in an undetermined state. Otherwise, the switch further checks $LAST_STATE$. If $LAST_STATE$ is a non-congestion or congestion state, the switch detects congestion according to queue size, which is the same as in the lossy network. If $LAST_STATE$ is the undetermined state, which indicates that the port has just released from an ON-OFF sending pattern, the switch may also check whether the queue length increases or decreases during the last T period. At the end of every T period, if the queue length decreases and $LAST_STATE$ is the undetermined state, no packets are marked. If the queue length increases and is larger than the threshold, the switch detects the transition to the congestion state. As soon as the queue decreases to a low threshold value or detects the transition to the congestion state, $LAST_STATE$ is updated.

4.3 TCD for CEE

In PFC, the egress port ceases transmitting when receiving PAUSE from the downstream port and recovers to transmit after receiving

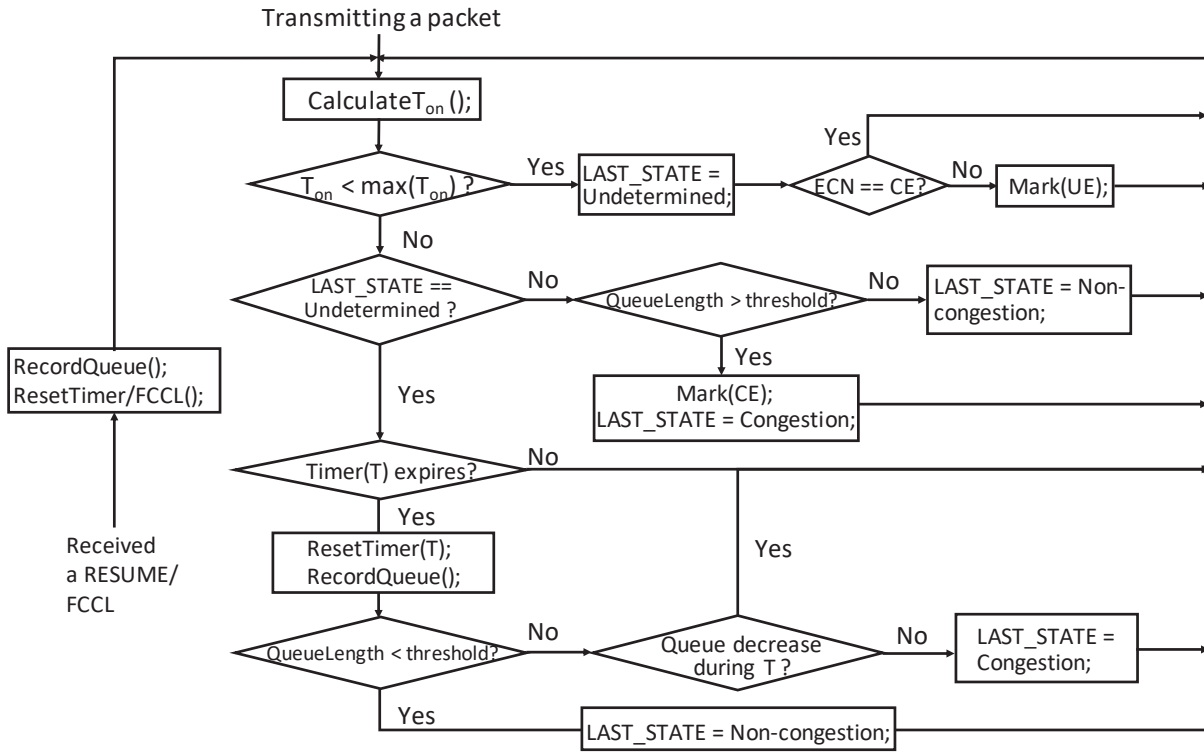


Figure 9: Flowchart of TCD.

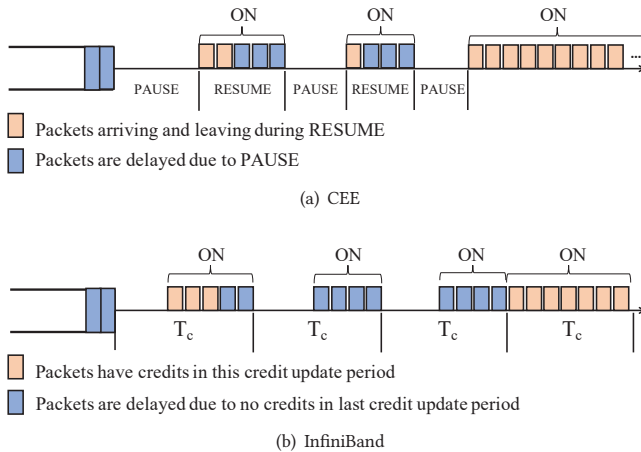


Figure 10: Practical ON periods.

RESUME. If neither PAUSE nor RESUME is received, the egress port transmits packets normally at line rate. As illustrated in Figure 10(a), the practical ON period is precisely the RESUME period. During each RESUME period, packets that are paused by the previous PAUSE are transmitted first, then new arriving packets may also be transmitted. Since PFC is triggered by the ingress queue length, we can directly adopt Equation (3) to obtain the value of $max(T_{on})$.

Parameter settings. In PFC, $B_1 - B_0$ is the difference between X_{off} and X_{on} , which is recommended as $2MTU$ [56]. The parameter τ is composed of several components. Briefly, when the receiver is ready to emit a feedback message, this message cannot interrupt

on-going packet transmission on the port. In the worst case, the message will be delayed for MTU/C . After processing the feedback message, the sender changes the output rate to ON or OFF. In the worst case, it needs to wait for another MTU/C . Finally, τ is also composed of propagation delay t_p . We have $\tau = 2MTU/C + 2t_p$.

For the period T to check queue decrease/increase after releasing from the undetermined state, we recommend to set T equal to $max(T_{on})$. When $\epsilon = 0.05$, $MTU = 1000B$ and $t_p = 1\mu s$, the typical values of $max(T_{on})$ for 40Gbps/100Gbps/200Gbps network is $34.4\mu s/26.96\mu s/24.48\mu s$, respectively.

4.4 TCD for InfiniBand

In CBFC, the egress port ceases transmitting when credits are not available, and recovers transmitting when current credits are at least one packet size. The number of credits are carried in FCCL message, which is updated to the upstream port periodically. As shown in Figure 10(b), when a port is regulated by CBFC, the practical ON period is the duration when packet are transmitting in each credit update period T_c . During each ON period, packets that have no credits at the previous T_c are transmitted first, then new arriving packets may also be transmitted. Since the flow control message FCCL is triggered by time rather than ingress queue length, the deduced $max(T_{on})$ in (3) can not be directly adopted in InfiniBand. The actual B_1 and B_0 vary under different congestion situations. Therefore, we extend the ON-OFF model by incorporating T_c .

Assume the total ingress buffer size is B , and it must be larger than CT_c to ensure CBFC working correctly. In detail, T_{on} is determined by the remaining buffer size $B - B_0$ in the beginning of

T_c . Then the maximum queue length B_1 is achieved at the end of T_{on} . Due to the introduce of T_c , CBFC regulates the sending rate in a more fine-grained style than PFC. The actual ON periods can be regarded as cutting in the unit of T_c . In the steady state, T_{on} is proportionate to T_c :

$$T_{on} = \frac{R_d T_c}{R_i} = \frac{R_d T_c}{R_d + \varepsilon C} \quad (4)$$

Given that $\varepsilon > 0$, the actual T_{on} is smaller than T_c when CBFC takes effect. We can utilize T_c as the upper bound of T_{on} .

Parameter settings. $\max(T_{on})$ is correlated with T_c . IB specification recommends T_c no larger than 65536 symbol times.¹ For the period T to check queue decrease/increase after releasing from an undetermined state, we recommend to set T equal to $\max(T_{on})$.

4.5 Implementation Discussions

We discuss implementation issues of TCD.

Hardware implementation feasibility. TCD is simple and inexpensive to implement in switches. TCD only requires additional registers to implement its functions, which are plentiful resources in today's commodity switches. For instance, to calculate current T_{on} , only one register to record the end time of the latest OFF period is needed per port per priority. TCD also needs registers to record $LAST_STATE$ and queue length. Comparing with the traditional marking scheme, TCD adds the procedure of checking the difference between timestamps and checking $LAST_STATE$ when each packet dequeues the egress, which is similar to checking MMU queue occupancy. $\max(T_{on})$ can be pre-configured since all parameters (ε , C , τ and T_c) are known in advance or configurable in lossless networks. The overall computational complexity is $O(1)$. With more switches supporting programmable and open data planes, we believe the procedure of checking timestamps and queue increase/decrease can be implemented at the data plane at line rate without the involvement of the switch control plane. For example, today Tofino ASIC already provides the metadata of timestamp and queue size at the egress pipeline [8].

Multiple priorities/VLs. In CEE switches, each port usually contains 8 (or less) priority queues [24, 29]. The ON-OFF model applies to each priority queue independently. However, with strict priority scheduling, if a low priority queue is paused and happens to be preempted by high priority traffic during RESUME, the actual RESUME period may fluctuate. However, it will not impose a significant impact on congestion detection since the deduced $\max(T_{on})$ is the upper bound of RESUME period. In practice, fewer priority queues are employed in actual deployments. In RoCE, only two or three priorities are supported by commodity switches [24, 29, 45].

In InfiniBand switches, InfiniBand Specification specifies that each port should contain 16 VLs, while actual implementations only offer up to 8 VLs [13, 18, 52]. By default, each VL is configured with a weight and a priority (low or high), where the weight is the proportion of link bandwidth that the VL is allowed to use. If multiple VLs are employed, $\max(T_{on})$ can be changed to the expected proportion of link bandwidth accordingly.

¹The symbol time is different at different link speeds. For example, the symbol time is 4ns for 10Gbps and 1ns for 40Gbps [13].

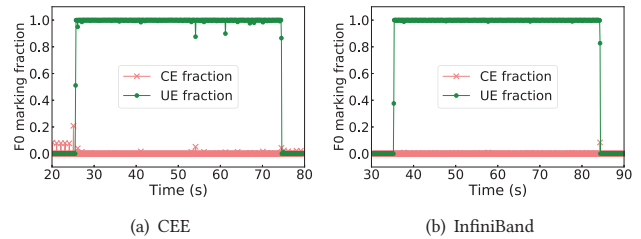


Figure 11: Testbed. F0 marking behaviors.

5 EVALUATION

5.1 Microbenchmarks

5.1.1 Testbed implementation. We build a compact topology of Figure 2 (with switch T0 directly connecting to switch T2) and implement TCD in DPDK [6]. Two servers, each equipped with dual Intel Xeon E5-2620 v3 CPUs (6 cores, 2.4GHz), are used to build software switches. Each server is plugged with 2 dual-port Intel 82599 10G NICs, working as a four-port switch. The basic L2 switch is built based on the reference test-pipeline project [7]. To achieve line-rate sending/receiving operation at 10Gbps, each RX/TX module is implemented on an individual core. We implement PFC according to IEEE 802.1Qbb [2] and CBFC according to InfiniBand Specification [13]. Although the testbed is built based on Ethernet, the physical layer media hardly affects the process of hop-by-hop flow controls.

In TCD under PFC, X_{off} is set to 800KB and X_{on} is 770KB. The parameter ε is 0.04. We move ε to 0.04 owing to the uncertain delay introduced by the software-based processing in DPDK. We find that in our DPDK implementation, there is a non-negligible random difference between the response time of PAUSE and RESUME frame, which drives the ON periods longer than expected. In TCD under CBFC, the credit update period is $60\mu s$ and the ingress buffer size is 800KB. The delay τ is about $20\mu s$ in our testbed. We let F0 start at 1Gbps and F1 start at 8Gbps. Then A0 starts at line rate. In this way, port P0 is in an undetermined state during A0 launching. F1 passes through the undetermined port P0 and also the congestion port. We focus on the received marked packets observed at the destination. Figure 11 illustrates the F0 marking fraction calculated every 100 ms. For both CEE and InfiniBand, after A0 starts, port P0 is detected as the undetermined state, and packets are marked with UE. After A0 ends, port P0 recovers to non-congestion, and no F0 packets are marked. We omit the results of F1. The marking fraction of F1 is inverse to F0, with all packets marked with CE during A0 launching.

5.1.2 Typical scenarios validation. Next, we validate the fined-grained congestion detection behavior of TCD in the typical scenarios of Section §3.1. Queue length evolution and marking behaviors in each scenario are illustrated in Figure 12 and Figure 13.

In the single congestion point scenario, port P2 and port P1 experience the transition from the undetermined state to the non-congestion state. Port P2 has more queue accumulation than port P1. After P2 releases from the undetermined state and recovers sending normally, it takes some time to drain out the accumulated

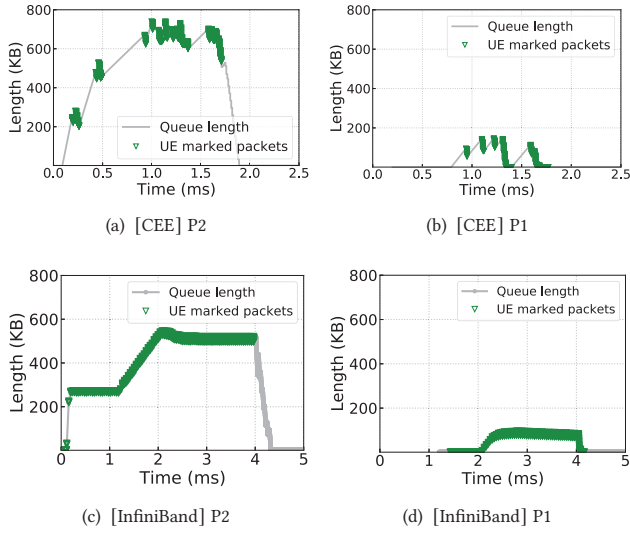


Figure 12: Single congestion point scenario. Port P2 and port P1: undetermined to non-congestion state.

queue. During this period, the switch detects that the queue length decreases continuously after an undetermined state. Hence packets are not marked with CE even the queue length is over the threshold.

In the multiple congestion points scenario, port P2 experiences the transition from the undetermined state to the congestion state. After P2 releases from the undetermined state and recovers sending normally, P2 still has persistent queue buildup. Hence P2 is detected as congested, and packets are marked with CE. Port P1 is still in an undetermined state because of congestion spreading from port P2.

5.1.3 Victim flows scenario. TCD can benefit victim flows by detecting congestion and undetermined state accurately. With ECN or FECN, victim flows may be mistakenly marked and are considered congested. We use the topology in Figure 2 to evaluate TCD in the typical head-of-line scenario. We set the link speed of links $S_0 - T_0$ and $S_1 - T_0$ to 20Gbps, and all remaining links are 40Gbps with a delay of $4\mu s$. Note that no flows are sent from host S_2 in this scenario. Then all flows of S_0 are potential victim flows and should not be detected as congested. If the number of packets marked with CE is non-zero, we consider the flow is mistakenly detected as congested. For CEE, hosts $S_0 \sim S_1$ and $A_0 \sim A_{14}$ generate flows according to the heavy-tailed Hadoop workload [48] with exponentially distributed inter-arrival time. The workload generators at hosts $A_0 \sim A_{14}$ are set to be synchronous to simulate the concurrent bursts. The default congestion control algorithm is DCQCN. For InfiniBand, hosts $S_0 \sim S_1$ and $A_0 \sim A_{14}$ generate MPI and I/O messages in typical sizes [15]. As shown in Table 3, in both networks, there are victim flows detected as congested. For CEE, about 26% of flows are mistakenly marked with ECN. With TCD, no victim flows are detected as congested.

5.1.4 Parameter sensitivity of ϵ . ϵ determines the value of $\max(T_{on})$. $\max(T_{on})$ gets larger as ϵ decreases. Because TCD detects the release from the undetermined state as soon as $\max(T_{on})$ expires, too

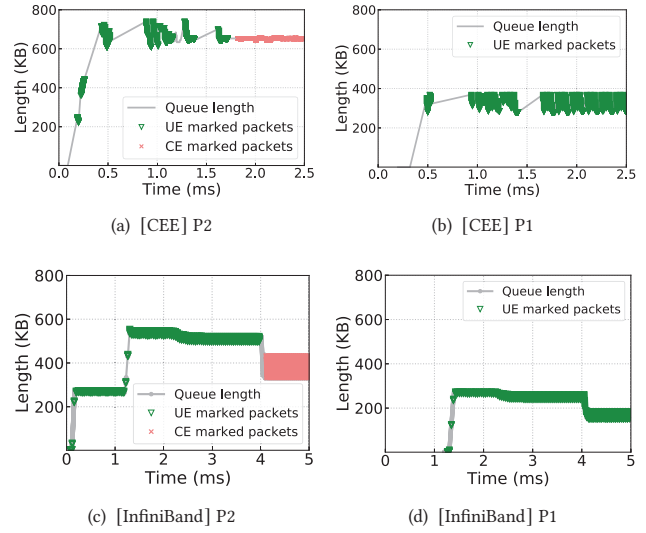


Figure 13: Multiple congestion points scenario. Port P2: undetermined to congestion state. Port P1: undetermined state.

Scheme	Fraction
ECN (CEE)	26.6%
TCD (CEE)	0%
FECN (IB)	13.5%
TCD (IB)	0%

Table 3: Victim flows marked with CE.

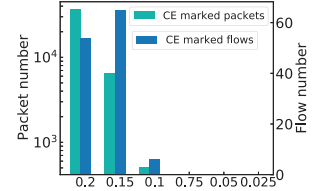


Figure 14: Parameter sensitivity of ϵ .

small $\max(T_{on})$ may result in detecting undetermined state mistakenly as congestion state or non-congestion state. On the other hand, too large $\max(T_{on})$ can also defer the detection of a congestion state. To evaluate the parameter sensitivity of ϵ , we repeat the concurrent burst simulation with different ϵ values. Figure 14 shows the result. In total, a larger ϵ yields more mistakenly marked packets. Packets of victim flows are not mistakenly marked with CE when ϵ is smaller than 0.1. Considering various positive and negative factors, our recommended value of 0.05 is proper.

5.2 Case Study

In this section, we conduct case studies to show how TCD incorporates existing congestion control algorithms. Our primary goal is not to propose the optimal congestion control algorithm in lossless networks but to underscore the significance of detecting congestion states accurately. It is now possible to simply consider different congestion states to improve end-to-end congestion controls. The primary principles are as follows:

- 1) Congested flows should decrease their rate aggressively because they are real contributors to congestion.
- 2) Flows only passing through undetermined ports (i.e., undetermined flows) should perform gentle rate adjustment. On the one

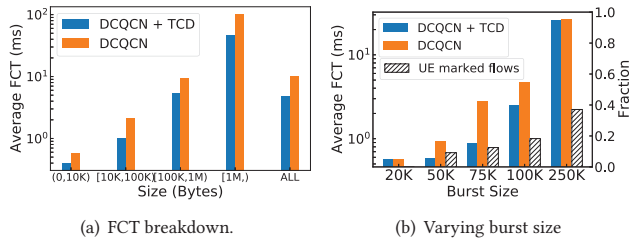


Figure 15: FCT performance for victim flows (DCQCN+TCD).

hand, undetermined flows may be victims that should not back off. On the other hand, blindly increasing the rate of undetermined flows may exacerbate congestion spreading. Specifically, if a packet is marked with UE, we advocate keeping the flow rate until it becomes uncongested or congested.

5.2.1 DCQCN. With TCD, NP conveys CE as well as UE information back to RP. Senders do not update the sending rate when receiving a CNP marked with UE. We change the rate reduction factor α from default 0.5 to 1.2 in order to decrease the rate of congested flows aggressively. Our simulator is developed based on the open-source project for DCQCN [55]. The PFC threshold X_{off} and X_{on} is 320KB and 318KB, respectively. Remaining parameters are set to the default values recommended in [56].

Victim flows scenario: We first evaluate DCQCN with TCD in the victim flow scenario. Figure 15(a) shows the average FCT breakdown of victim flows. On the whole, DCQCN with TCD achieves a better average FCT than DCQCN. For flows with a size smaller than 10KB, the faster flow completion mainly benefits from aggressive rate decrease of congested flows because their rate is not regulated by end-to-end congestion control. With a more aggressive rate decrease, the congestion spreading is less and fewer flows suffer from less queueing delay. For medium and large flows, the faster flow completion time mainly benefits from detecting congestion accurately. DCQCN does not mistakenly throttle these flows. Without TCD, some victim flows are detected as congested.

To further study the performance under small flows, we let $A_0 \sim A_{14}$ generate concurrent bursts with varying sizes. The inter-arrival time of each round of concurrent burst follows an exponential distribution. Hosts $S_0 \sim S_1$ generate flows according to *Hadoop* workload as before. The BDP is 80KB in this scenario. Figure 15(b) shows the average FCT and TCD marking behaviors of victim flows. When the burst size is small, there is almost no congestion spreading. As the burst size increases, it is harder for end-to-end congestion control to regulate the rate of flows of S_1 promptly, introducing more queue buildup and a larger extent of congestion spreading. As a result, more victim flows are marked as undetermined flows. When burst size is medium (e.g., 250KB), the traffic is overloaded, and congestion is severe. Overall, DCQCN combining with TCD can improve the FCT performance of victim flows, especially when congestion is caused by interference of small flows.

Realistic workloads: We choose two realistic heavy-tailed workloads: *Hadoop* workload [48] and *WebSearch* workload [12]. 90% flows of the *Hadoop* workload are less than 120KB. The *WebSearch*

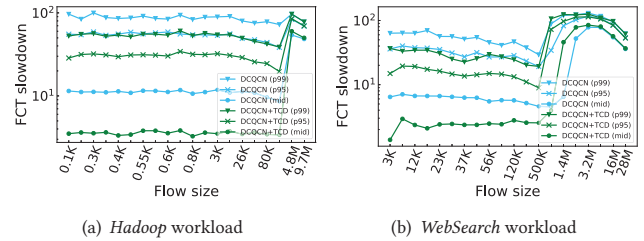


Figure 16: Overall FCT slowdown (DCQCN+TCD)

workload is heavier, with 90% flows less than 5MB. The network is a Fat-Tree [9] network ($k = 10$) with 250 servers. The link capability is 40Gbps with $4\mu s$ link delay. We adjust the flow generation rates to set the average link loads to 60%. We generate over 40 thousand flows with exponentially distributed inter-arrival time. Figure 16(a) and Figure 16(b) demonstrate FCT slowdown in the median, 95th and 99th percentile. FCT slowdown is calculated by the ratio between real FCT and baseline FCT.

For *Hadoop* workload, DCQCN with TCD achieves a much better FCT slowdown, especially for small flows. DCQCN with TCD reduces the median FCT slowdown from 10.8 to 3.6 for flows smaller than 80KB. For flows smaller than 50KB, DCQCN with TCD improves the 99th-percentile FCT slowdown by at most 1.7 \times . For flows larger than 100KB, DCQCN with TCD achieves comparable performance with DCQCN.

For *WebSearch* workload, DCQCN with TCD achieves a better FCT slowdown, especially for small and medium flows. DCQCN with TCD reduces the median FCT slowdown from 4.6 to 2.5 for flows smaller than 500KB and achieves 2 \times better 99th-percentile FCT slowdown. The 99th-percentile FCT slowdown is almost the same with DCQCN for flows larger than 1MB.

5.2.2 IB CC. With TCD, the receiver CA conveys CE as well as UE information back to the sender CA. Sender CA does not update the sending rate when receiving a CNP marked with UE. We change the rate reduction step from default 1 to 2 to decrease the rate of congested flows aggressively. Our simulator is developed based on the open-source project for InfiniBand [4] released by Mellanox. We extend this model by adding support for IB CC and TCD. The switch architecture is virtual cut-through, input buffering with virtual output queues (VoQ). Each switch input port is equipped with 280KB of buffer space.

Victim flows scenario: Figure 17(a) shows the average message completion time (MCT) breakdown of victim flows. On the whole, IB CC with TCD achieves a better average MCT than IB CC. Since the message sizes are larger than BDP, the performance improvement benefits from accurate congestion detection. I/O messages can fully utilize the available bandwidth without being throttled innocently. We also evaluate the average MCT of victim flows under varying burst sizes, and have similar findings with ECN-based DCQCN (results are not presented due to space limitation).

Synthetic workloads: We choose the synthetic communication patterns of typical MPI and I/O jobs to simulate HPC scenarios where multiple jobs share the network [15]. The network is a Fat-Tree [9] network ($k = 16$) with 1024 servers. The routing algorithm

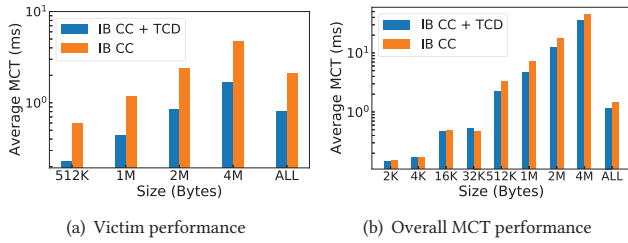


Figure 17: MCT performance (IB CC+TCD).

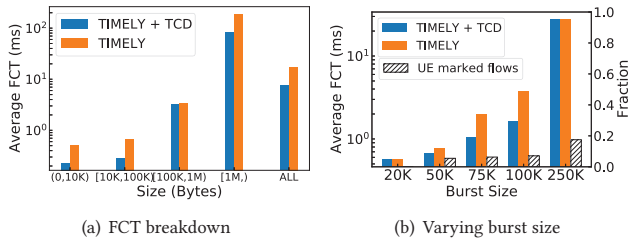


Figure 18: FCT performance for victim flows (TIMELY+TCD).

is the static D-mod-k scheme [21] in InfiniBand network. For each rack, we randomly select four servers as I/O servers to receive I/O traffic from I/O clients. Then 25% nodes are randomly selected as I/O clients. The remaining nodes are MPI clients and MPI servers. In total, we generate over 80 thousand messages, of which 10% are I/O messages. I/O messages are with sizes randomly from arrays [512KB, 1MB, 2MB, 4MB]. MPI messages range from 2KB to 32KB, with over 50% of MPI messages are 2KB. Figure 17(b) shows the average MCT breakdown. IB CC with TCD can achieve 1.22× better overall average FCT than IB CC. IB CC with TCD especially benefits I/O messages. For example, improving average MCT by 1.5× for 512KB messages. We consider that mistakenly throttling of victims can affect more I/O messages because multiple I/O messages are sending in one connection.

5.2.3 *TIMELY*. TCD can also benefit delay-based congestion control. *TIMELY* [43] uses RTT gradient as the congestion signal, which can not distinguish between delay increase caused by congestion and delay increase caused by PAUSES. With TCD, endpoints are aware of whether a flow only passes through an undetermined port. In detail, senders do not update the sending rate if the gradient is above zero ($T_{low} < RTT < T_{high}$) and the packet is marked with UE simultaneously. We change the rate reduction factor β from default 0.8 to 1.6 to decrease the rate of congested flows aggressively. Our simulator is developed based on the code snippet for *TIMELY* [42], and the remaining parameters are recommended values in [43].

Victim flows scenario: With *TIMELY*, we do observe that the sending rate of victim flows is reduced due to increased values of RTT samples. With TCD, sending rate of victim flows is not reduced when RTT sudden increases due to PAUSES. Figure 18(a) shows the average FCT breakdown of victim flows. *TIMELY* with TCD can

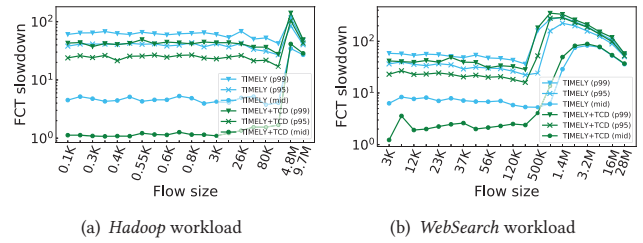


Figure 19: Overall FCT slowdown (TIMELY+TCD).

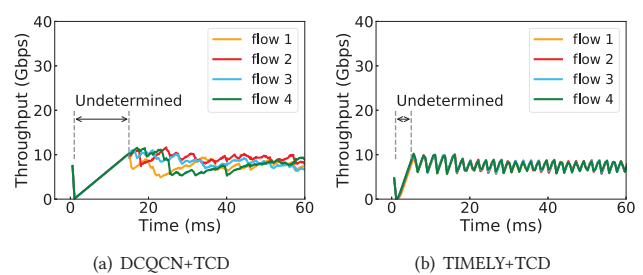


Figure 20: Fairness with TCD.

achieve 2.2× and 2.3× better average FCT for flows smaller than 10KB and flows larger than 1MB, respectively. We also evaluate the performance under small flows, where A0 ~ A14 generate concurrent burst with varying sizes. Figure 18(b) presents the average FCT performance of victim flows and the fraction of undetermined flows. Similarly, as the burst size increases, more flows are victimized and marked with UE.

Realistic workloads: Figure 19 presents the overall FCT slowdown for the *Hadoop* and *WebSearch* workload (we adopt the same network settings as in DCQCN case). For both workloads, *TIMELY* with TCD improves the median and 99th-percentile FCT slowdown especially for small and medium flows. For instance, under the *Hadoop* workload, *TIMELY* with TCD can reduce the 99th-percentile FCT slowdown for flows smaller than 50KB from 50.3 to 36.6.

5.2.4 *Fairness with TCD*. To evaluate how our recommended rate adjustment rules for ternary states affect the fairness property, we adopt a modified topology in Figure 2. We add host B0 ~ B3 connected to the switch L0. S1 sends long-lived flow F1 to R1. A0 ~ A14 generate concurrent 64KB burst to R1, which last for about 3ms. B0 ~ B3 send four long-lived flows to R0 at the same time when bursts start. Figure 20 demonstrates the throughput of these four flows. During burst launching, the congestion spreads to port P2. Port P2 becomes an undetermined port. Under the gentle rate adjustment rule, the CC rate of four flows is kept unchanged because they only go through the undetermined port P2. The throughput drop is because of head-of-line blocking at link L0-T2 as hop-by-hop flow control kicks in. Consequently, TCD can achieve the same fairness as the hop-by-hop flow control for flows only passing through an undetermined port. After congestion at port P3 disappears, port P2 becomes a congestion port. Then four flows transit to congested

flows. For ECN-based DCQCN and delay-based TIMELY, congested flows can achieve fair-share rate allocation (8Gbps) with TCD.

6 DISCUSSION

Design tradeoff. Adaptively adjusting $\max(T_{on})$ (e.g., predicting T_{on} based on historical values and real-time queue length) is another design choice to detect state transitions. TCD relies on a pre-configured $\max(T_{on})$ to determine whether to enter or leave the undetermined state. We believe a proper static $\max(T_{on})$ is enough for most practical scenarios according to the ON-OFF model and evaluations. Further, adaptively adjusting $\max(T_{on})$ increases the design complexity and cost while may only introduce marginal gains. With a pre-configured $\max(T_{on})$ in TCD, if an anomalous traffic pattern drives T_{on} to be much smaller than $\max(T_{on})$, it will not affect the detection on transitions to the undetermined state, only leading to limited late detection on the leave from the undetermined state (no more than $\max(T_{on})$ time). On the other hand, if an anomalous traffic pattern drives T_{on} to be much larger than the pre-configured $\max(T_{on})$, the undetermined port may switch between the undetermined state and non-congestion/congestion state at a low frequency. TCD makes acceptable compromise under these corner cases for overall design and implementation simplicity.

Cooperation with congestion controls. Decreasing the rate of congested flows aggressively and adjusting the rate of undetermined flows gently is our general recommendation for cooperating TCD with congestion controls in lossless networks. The detailed rate adjustment rules in § 5.2 are only simple cases to validate our insights. However, if PFC/CBFC is not triggered in lossless networks, simply adopting a more aggressive reduction for congested flows than existing congestion controls may harm the link utilization and throughput. Given that PFC/CBFC cannot be avoided totally from production experience [10, 24, 31], we think determining the proper individual rate adjustment rules for congested flows and undetermined flows is a valuable problem that needs to be studied in-depth in the future.

7 RELATED WORK

Congestion detection is a classic research topic. We classify existing congestion detection mechanisms into three categories.

Switch congestion detection. Switches are in charge of detecting congestion and providing explicit congestion signals. Switches detect incipient congestion based on average or instant queue size and perform packet dropping/marking to notify endpoints [20, 30, 40]. TCP and its variants [19, 25, 30] utilize packet loss as the congestion signal. With ECN [46] support, switches can mark CE instead of dropping packets. Although a single marking packet is intended to cause the transport layer to respond, several congestion controls propose to react to congestion based on the multi-bit congestion information provided in consecutive ECN. For example, DECBIT [47] reacts to congestion when the ratio of marked packets is over a certain parameter. DCTCP [12] reacts to the extent of congestion according to the fraction of marked packets.

Switch and endpoint collaborated congestion detection. Recent congestion control algorithms in lossless networks detect congestion in a collaboration of switches and endpoints, which requires tightly coupled design with rate control at endpoints. PCN [16]

discovers that PFC can affect congestion detection and develops Non-PAUSE ECN (NP-ECN). Switches maintain a counter to record the number of paused packets. Only non-paused packets are eligible to be marked with ECN. The receiver NICs detect and identify a congested flow when the fraction of marked packets exceeds a threshold (i.e., 95%) during a period T . HPCC [38] aims at controlling inflight bytes for the most congested link, relying on in-band network telemetry (INT) to obtain related information of inflight bytes. INT headers carry transmitted bytes and queue length information. Senders combine all information to calculate the current inflight bytes. In a word, both NP-ECN and INT are not independent congestion detection mechanisms in switches.

End-to-end congestion detection. Endpoints can also infer congestion based on end-to-end delay without any in-network support. Several delay-based congestion control algorithms in lossy networks leverage RTT or RTT variations [14, 33, 36, 43, 53] to detect congestion. Recent work also differentiates delays caused by in-network congestion and delay caused by endpoint congestion [35]. However, in lossless networks, with only end-to-end delay, it is tough for endpoints to distinguish between delay increase caused by congestion and delay increase caused by the ON-OFF regulation of hop-by-hop flow controls.

Receiver-driven congestion controls. Recently several receiver-driven congestion controls have been proposed, such as SRP [32], ExpressPass [17], NDP [26], and Homa [44]. The core of receiver-driven congestion controls is proactive transport operating in a "request and allocation" style: explicitly allocating the bandwidth of bottleneck link(s) and proactively preventing congestion [27]. These schemes usually let new flows blindly transmit unscheduled packets in the first RTT, where congestion may also occur with the risk of triggering hop-by-hop flow controls. Currently, TCD is suitable for end-to-end congestion controls that react to congestion signals (e.g., ECN or delay).

8 CONCLUSION

This paper re-understands congestion detection in lossless networks and proposes ternary congestion detection (TCD) for CEE and InfiniBand. We reveal a new port state called the *undetermined* state and define ternary states. Testbed and extensive simulations demonstrate that TCD can accurately detect congestion ports and identify congested flows as well as undetermined flows. Case studies show that existing congestion control algorithms can achieve better performance by combining with TCD, confirming that accurate congestion detection is significant for congestion controls. We envision that TCD will motivate further exploration of congestion controls considering ternary states.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the shepherd Vishal Misra and the anonymous reviewers for their constructive comments. The authors also thank Wenxue Cheng and Kun Qian for insightful discussions on this work. This work is supported in part by the National Key Research and Development Program of China (No.2018YFB1700203, 2018YFB1700103), and by National Natural Science Foundation of China (NSFC) under Grant 61872208, as well as gifts from MSRA.

REFERENCES

- [1] 2010. IEEE 802.1 Qau - Congestion Notification. <http://www.ieee802.org/1/pages/802.1au.html>
- [2] 2010. IEEE 802.1 Qbb - Priority-based Flow Control. <http://www.ieee802.org/1/pages/802.1bb.html>
- [3] 2012. InfiniBand Most Used Interconnect on the TOP500. <https://www.infinibandta.org/infiniband-most-used-interconnect-on-the-top-500/>
- [4] 2013. InfiniBand Flit Level Model. <https://omnetpp.org/download-items/InfiniB-and-FlitSim.html>
- [5] 2016. Life in the Fast Lane: InfiniBand Continues to Reign as HPC Interconnect of Choice. <https://www.infinibandta.org/life-in-the-fast-lane-infiniband-continues-to-reign-as-hpc-interconnect-of-choice/>
- [6] 2020. Intel DDPK. <https://www.dpdk.org/>
- [7] 2020. Test-pipeline. <https://github.com/DPDK/dpdk/tree/main/app/test-pipeline>
- [8] 2021. In-band Network Telemetry in Barefoot Tofino. <https://www.opencompute.org/files/INT-In-Band-Network-Telemetry-A-Powerful-Analytics-Framework-for-your-Data-Center-OCP-Final3.pdf>
- [9] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A Scalable, Commodity Data Center Network Architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication* (Seattle, WA, USA) (SIGCOMM '08). Association for Computing Machinery, New York, NY, USA, 63–74. <https://doi.org/10.1145/1402958.1402967>
- [10] Fatma Alali, Fabrice Mizero, Malathi Veeraraghavan, and John M. Dennis. 2017. A measurement study of congestion in an InfiniBand network. In *2017 Network Traffic Measurement and Analysis Conference (TMA)*, 1–9. <https://doi.org/10.23919/TMA.2017.8002911>
- [11] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, and George Varghese. 2014. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters. In *Proceedings of the 2014 ACM Conference on SIGCOMM* (Chicago, Illinois, USA) (SIGCOMM '14). Association for Computing Machinery, New York, NY, USA, 503–514. <https://doi.org/10.1145/2619239.2626316>
- [12] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data Center TCP (DCTCP). In *Proceedings of the ACM SIGCOMM 2010 Conference* (New Delhi, India) (SIGCOMM '10). Association for Computing Machinery, New York, NY, USA, 63–74. <https://doi.org/10.1145/1851182.1851192>
- [13] InfiniBand Trade Association. 2020. InfiniBandTM Architecture Specification Volume 1 Release 1.4. (2020). <https://www.infinibandta.org/document/dl/8567>
- [14] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson. 1994. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications* (London, United Kingdom) (SIGCOMM '94). Association for Computing Machinery, New York, NY, USA, 24–35. <https://doi.org/10.1145/190314.190317>
- [15] Kevin A. Brown, Nikhil Jain, Satoshi Matsuoka, Martin Schulz, and Abhinav Bhatele. 2018. Interference between I/O and MPI Traffic on Fat-Tree Networks. In *Proceedings of the 47th International Conference on Parallel Processing* (Eugene, OR, USA) (ICPP 2018). Association for Computing Machinery, New York, NY, USA, Article 7, 10 pages. <https://doi.org/10.1145/3225058.3225144>
- [16] Wenxue Cheng, Kun Qian, Wanchun Jiang, Tong Zhang, and Fengyuan Ren. 2020. Re-architecting Congestion Management in Lossless Ethernet. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA, 19–36. <https://www.usenix.org/conference/nsdi20/presentation/cheng>
- [17] Inho Cho, Keon Jang, and Dongsu Han. 2017. Credit-Scheduled Delay-Bounded Congestion Control for Datacenters. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) (SIGCOMM '17). Association for Computing Machinery, New York, NY, USA, 239–252. <https://doi.org/10.1145/3098822.3098840>
- [18] HPCAC Technical Community. 2019. Understanding Basic InfiniBand QoS. <https://hpcadvisorycouncil.atlassian.net/wiki/spaces/HPCWORKS/pages/1178075141/Understanding+Basic+InfiniBand+QoS>
- [19] S. Floyd. 2003. RFC3649: HighSpeed TCP for Large Congestion Windows. (2003).
- [20] S. Floyd and V. Jacobson. 1993. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking* 1, 4 (1993), 397–413. <https://doi.org/10.1109/90.251892>
- [21] C. Gomez, F. Gilbert, M.E. Gomez, P. Lopez, and J. Duato. 2007. Deterministic versus Adaptive Routing in Fat-Trees. In *2007 IEEE International Parallel and Distributed Processing Symposium*, 1–8. <https://doi.org/10.1109/IPDPS.2007.370482>
- [22] Ernst Gunnar Gran, Magne Eimot, Sven-Arne Reinemo, Tor Skeie, Olav Lysne, Lars Paul Huse, and Gilad Shainer. 2010. First experiences with congestion control in InfiniBand hardware. In *2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, 1–12. <https://doi.org/10.1109/IPDPS.2010.5470419>
- [23] Ernst Gunnar Gran, Sven-Arne Reinemo, Olav Lysne, Tor Skeie, Eitan Zahavi, and Gilad Shainer. 2012. Exploring the Scope of the InfiniBand Congestion Control Mechanism. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium*, 1131–1143. <https://doi.org/10.1109/IPDPS.2012.104>
- [24] Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni, Jianxi Ye, Jitu Padhye, and Marina Lipshteyn. 2016. RDMA over Commodity Ethernet at Scale. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) (SIGCOMM '16). Association for Computing Machinery, New York, NY, USA, 202–215. <https://doi.org/10.1145/2934872.2934908>
- [25] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: A New TCP-Friendly High-Speed TCP Variant. *SIGOPS Oper. Syst. Rev.* 42, 5 (July 2008), 64–74. <https://doi.org/10.1145/1400097.1400105>
- [26] Mark Handley, Costin Raiciu, Alexandru Agache, Andrei Voinescu, Andrew W. Moore, Gianni Antichi, and Marcin Wójcik. 2017. Re-Architecting Datacenter Networks and Stacks for Low Latency and High Performance. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) (SIGCOMM '17). Association for Computing Machinery, New York, NY, USA, 29–42. <https://doi.org/10.1145/3098822.3098825>
- [27] Shuihai Hu, Wei Bai, Gaoxiong Zeng, Zilong Wang, Baochen Qiao, Kai Chen, Kun Tan, and Yi Wang. 2020. Aeolus: A Building Block for Proactive Transport in Datacenters (SIGCOMM '20). Association for Computing Machinery, New York, NY, USA, 422–434. <https://doi.org/10.1145/3387514.3405878>
- [28] Shuihai Hu, Yibo Zhu, Peng Cheng, Chuanxiong Guo, Kun Tan, Jitendra Padhye, and Kai Chen. 2016. Deadlocks in Datacenter Networks: Why Do They Form, and How to Avoid Them. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks* (Atlanta, GA, USA) (HotNets '16). Association for Computing Machinery, New York, NY, USA, 92–98. <https://doi.org/10.1145/3005745.3005760>
- [29] Shuihai Hu, Yibo Zhu, Peng Cheng, Chuanxiong Guo, Kun Tan, Jitendra Padhye, and Kai Chen. 2019. Tagger: Practical PFC Deadlock Prevention in Data Center Networks. *IEEE/ACM Transactions on Networking* 27, 2 (2019), 889–902. <https://doi.org/10.1109/TNET.2019.2902875>
- [30] V. Jacobson. 1988. Congestion Avoidance and Control. In *Symposium Proceedings on Communications Architectures and Protocols* (Stanford, California, USA) (SIGCOMM '88). Association for Computing Machinery, New York, NY, USA, 314–329. <https://doi.org/10.1145/52324.52356>
- [31] Saurabh Jha, Archit Patke, Jim Brandt, Ann Gentile, Benjamin Lim, Mike Showerman, Greg Bauer, Larry Kaplan, Zbigniew Kalbarczyk, William Kramer, and Ravi Iyer. 2020. Measuring Congestion in High-Performance Datacenter Interconnects. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA, 37–57. <https://www.usenix.org/conference/nsdi20/presentation/jha>
- [32] Nan Jiang, Daniel U. Becker, George Michelogiannakis, and William J. Dally. 2012. Network congestion avoidance through Speculative Reservation. In *IEEE International Symposium on High-Performance Comp Architecture*, 1–12. <https://doi.org/10.1109/HPCA.2012.6169047>
- [33] Cheng Jin, D.X. Wei, and S.H. Low. 2004. FAST TCP: motivation, architecture, algorithms, performance. In *IEEE INFOCOM 2004*, Vol. 4. 2490–2501 vol.4. <https://doi.org/10.1109/INFCOM.2004.1354670>
- [34] Anuj Kalia, Michael Kaminsky, and David G. Andersen. 2014. Using RDMA Efficiently for Key-Value Services (SIGCOMM '14). Association for Computing Machinery, New York, NY, USA, 295–306. <https://doi.org/10.1145/2619239.2626299>
- [35] Gautam Kumar, Nandita Dukkipati, Keon Jang, Hassan M. G. Wassel, Xian Wu, Behnam Montazeri, Yaogong Wang, Kevin Springborn, Christopher Alfeld, Michael Ryan, David Wetherall, and Amin Vahdat. 2020. Swift: Delay is Simple and Effective for Congestion Control in the Datacenter. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication* (Virtual Event, USA) (SIGCOMM '20). Association for Computing Machinery, New York, NY, USA, 514–528. <https://doi.org/10.1145/3387514.3406591>
- [36] Changhyun Lee, Chunjong Park, Keon Jang, Sue Moon, and Dongsu Han. 2015. Accurate Latency-based Congestion Feedback for Datacenters. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*. USENIX Association, Santa Clara, CA, 403–415. <https://www.usenix.org/conference/atc15/technical-session/presentation/lee-changhyun>
- [37] Yuliang Li, Rui Miao, Changhoon Kim, and Minlan Yu. 2016. LossRadar: Fast Detection of Lost Packets in Data Center Networks. In *Proceedings of the 12th International Conference on Emerging Networking Experiments and Technologies* (Irvine, California, USA) (CoNEXT '16). Association for Computing Machinery, New York, NY, USA, 481–495. <https://doi.org/10.1145/2999572.2999609>
- [38] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, and Minlan Yu. 2019. HPC2: High Precision Congestion Control. In *Proceedings of the ACM Special Interest Group on Data Communication* (Beijing, China) (SIGCOMM '19). Association for Computing Machinery, New York, NY, USA, 44–58. <https://doi.org/10.1145/3341302.3342085>
- [39] Qian Liu, Robert D. Russell, and Ernst Gunnar Gran. 2016. Improvements to the InfiniBand Congestion Control Mechanism. In *2016 IEEE 24th Annual Symposium on High-Performance Interconnects (HOTI)*, 27–36. <https://doi.org/10.1109/HOTI.2016.018>

- [40] A. Mankin. 1990. Random Drop Congestion Control. *SIGCOMM Comput. Commun. Rev.* 20, 4 (Aug. 1990), 1–7. <https://doi.org/10.1145/99517.99521>
- [41] Mellanox. 2018. Benefits of Remote Direct Memory Access Over Routed Fabrics. <https://www.mellanox.com/related-docs/solutions/benefits-of-RDMA-over-routed-fabrics.pdf>
- [42] Radhika Mittal. 2015. TIMELY algorithm to compute new rate. (2015). <http://radhikam.web.illinois.edu/timely-code-snippet.cc>
- [43] Radhika Mittal, Vinh The Lam, Nandita Dukkkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. 2015. TIMELY: RTT-Based Congestion Control for the Datacenter. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) (SIGCOMM '15). Association for Computing Machinery, New York, NY, USA, 537–550. <https://doi.org/10.1145/2785956.2787510>
- [44] Behnam Montazeri, Yilong Li, Mohammad Alizadeh, and John Ousterhout. 2018. Homa: A Receiver-Driven Low-Latency Transport Protocol Using Network Priorities. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (Budapest, Hungary) (SIGCOMM '18). Association for Computing Machinery, New York, NY, USA, 221–235. <https://doi.org/10.1145/3230543.3230564>
- [45] Kun Qian, Wenxue Cheng, Tong Zhang, and Fengyuan Ren. 2019. Gentle Flow Control: Avoiding Deadlock in Lossless Networks. In *Proceedings of the ACM Special Interest Group on Data Communication* (Beijing, China) (SIGCOMM '19). Association for Computing Machinery, New York, NY, USA, 75–89. <https://doi.org/10.1145/3341302.3342065>
- [46] K Ramakrishnan, Sally Floyd, and D Black. 2001. RFC3168: The addition of explicit congestion notification (ECN) to IP.
- [47] K. K. Ramakrishnan and Raj Jain. 1990. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks. *ACM Trans. Comput. Syst.* 8, 2 (May 1990), 158–181. <https://doi.org/10.1145/78952.78955>
- [48] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren. 2015. Inside the Social Network's (Datacenter) Network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) (SIGCOMM '15). Association for Computing Machinery, New York, NY, USA, 123–137. <https://doi.org/10.1145/2785956.2787472>
- [49] Danfeng Shan and Fengyuan Ren. 2018. ECN Marking With Micro-Burst Traffic: Problem, Analysis, and Improvement. *IEEE/ACM Transactions on Networking* 26, 4 (2018), 1533–1546. <https://doi.org/10.1109/TNET.2018.2840722>
- [50] Alex Shpiner, Eitan Zahavi, Vladimir Zdornov, Tal Anker, and Matty Kadosh. 2016. Unlocking Credit Loop Deadlocks. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks* (Atlanta, GA, USA) (HotNets '16). Association for Computing Machinery, New York, NY, USA, 85–91. <https://doi.org/10.1145/3005745.3005768>
- [51] Brent Stephens, Alan L. Cox, Ankit Singla, John Carter, Colin Dixon, and Wesley Felter. 2014. Practical DCB for improved data center networks. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. 1824–1832. <https://doi.org/10.1109/INFOCOM.2014.6848121>
- [52] Hari Subramoni, Ping Lai, Sayantan Sur, and Dhabaleswar K. Panda. 2010. Improving Application Performance and Predictability Using Multiple Virtual Lanes in Modern Multi-core InfiniBand Clusters. In *2010 39th International Conference on Parallel Processing*. 462–471. <https://doi.org/10.1109/ICPP.2010.54>
- [53] K. Tan, J. Song, Q. Zhang, and M. Sridharan. 2006. A Compound TCP Approach for High-Speed and Long Distance Networks. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*. 1–12. <https://doi.org/10.1109/INFOCOM.2006.188>
- [54] Haitao Wu, Jiabo Ju, Guohan Lu, Chuanxiong Guo, Yongqiang Xiong, and Yongguang Zhang. 2012. Tuning ECN for Data Center Networks. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies* (Nice, France) (CoNEXT '12). Association for Computing Machinery, New York, NY, USA, 25–36. <https://doi.org/10.1145/2413176.2413181>
- [55] Yibo Zhu. 2016. NS-3 simulator for RDMA over Converged Ethernet v2 (RoCEv2). (2016). <https://github.com/bobzhuyb/ns3-rdma>
- [56] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. 2015. Congestion Control for Large-Scale RDMA Deployments. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) (SIGCOMM '15). Association for Computing Machinery, New York, NY, USA, 523–536. <https://doi.org/10.1145/2785956.2787484>
- [57] Yibo Zhu, Nanxi Kang, Jiaxin Cao, Albert Greenberg, Guohan Lu, Ratul Mahajan, Dave Maltz, Lihua Yuan, Ming Zhang, Ben Y. Zhao, and Haitao Zheng. 2015. Packet-Level Telemetry in Large Datacenter Networks. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) (SIGCOMM '15). Association for Computing Machinery, New York, NY, USA, 479–491. <https://doi.org/10.1145/2785956.2787483>