ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/326161720

Congestion control in high-speed lossless data center networks: A survey

Article *in* Future Generation Computer Systems · July 2018

DOI: 10.1016/j.future.2018.06.036

CITATIONS
8

READS
435

3 authors, including:

Shan Huang
National University of Defense Technology
7 PUBLICATIONS **8** CITATIONS
SEE PROFILE

Dezun Dong
Nationa University of Defense Technology
58 PUBLICATIONS **465** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project TH2 Supercomputer View project

All content following this page was uploaded by Shan Huang on 30 May 2019.

The user has requested enhancement of the downloaded file.

Future Generation Computer Systems 89 (2018) 360–374

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Congestion control in high-speed lossless data center networks: A survey

Shan Huang [a], Dezun Dong [a,*], Wei Bai [b]

[a] Department of Computer Science, National University of Defense Technology, Changsha, China
[b] Microsoft Research, Beijing, China

HIGHLIGHTS

- This paper introduces the network congestion in high-speed lossless networks.
- This paper presents the typical Link-layer Flow Control (LFC) schemes which can achieve network lossless.
- This paper surveys several congestion control schemes which are deployed in lossless networks.
- This paper compares the existing congestion control schemes and presents their merits and demerits.
- This paper puts forward several challenges and opportunities of designing congestion control schemes for high-speed lossless DCNs.

ARTICLE INFO

Article history:
Received 26 January 2018
Received in revised form 21 May 2018
Accepted 20 June 2018
Available online 3 July 2018

Keywords:
High-speed lossless data center network (DCN)
Network congestion
Congestion control

ABSTRACT

In data centers, packet losses cause high retransmission delays, which is harmful to many real-time workloads. To prevent packet losses, the lossless fabrics have been deployed in many production data centers. However, when network congestion happens, the lossless fabric also causes many problems like saturation tree and unfairness, which seriously degrade the performance of data center applications. Therefore, how to control the congestion in high-speed lossless data center networks is a significant problem.

In this paper, we first introduce link layer flow control schemes to provide lossless fabrics. Then we survey congestion control schemes in high-speed lossless data center networks. In particular, we classify existing congestion control schemes into two categories: reactive and proactive. Finally, we present the challenges and opportunities for future research in this area.

© 2018 Elsevier B.V. All rights reserved.

1. Introduction

Many companies, such as Microsoft, Google, Amazon and Alibaba, have built lots of data centers around the world to provide high-quality service to global users [1–6]. Inside a data center, the hundreds of thousands of servers are inter-connected by a data center network (DCN) with high link speed (10–100 Gbps) and low base latency (10–100 μs) [7,8].

Today's data centers host many real-time applications, e.g., web search, retail and social networking [9–14]. To serve a user request, these applications will generate lots of small latency-sensitive request and response messages in DCNs. The user experience is determined by how fast the application collects all (or most of) the response messages. Hence, the high delay in DCNs can seriously degrade the performance of these applications, resulting in poor user experience and operator revenue. However, data center traffic is well known for its burstiness and data center switches typically have very shallow buffers [1,15,16]. In DCNs, the bursty traffic is likely to overfill the shallow switch buffers, leading to high retransmission delay and poor user experience.

To minimize retransmission delay, lossless networks have been deployed in production data centers. In contrast to traditional lossy networks, lossless networks employ the hop-by-hop Link-layer Flow Control (LFC) to avoid packet congestion losses. However, like lossy networks, when the congestion happens, lossless networks also suffer from high queuing delay [17]. Furthermore, due to the LFC, lossless networks may spread the congestion, resulting in congestion tree [18], causing unfairness problem (e.g., victim flows) and degrading throughput [19] (more details in Section 3). To mitigate above problems, we need advanced congestion control solutions in lossless DCNs.

Various survey papers related to data center congestion control have been proposed. But most of them emphatically summarized

* Corresponding author.
E-mail addresses: huangshang12@nudt.edu.cn (S. Huang), dong@nudt.edu.cn (D. Dong), webai@microsoft.com (W. Bai).

https://doi.org/10.1016/j.future.2018.06.036
0167-739X/© 2018 Elsevier B.V. All rights reserved.

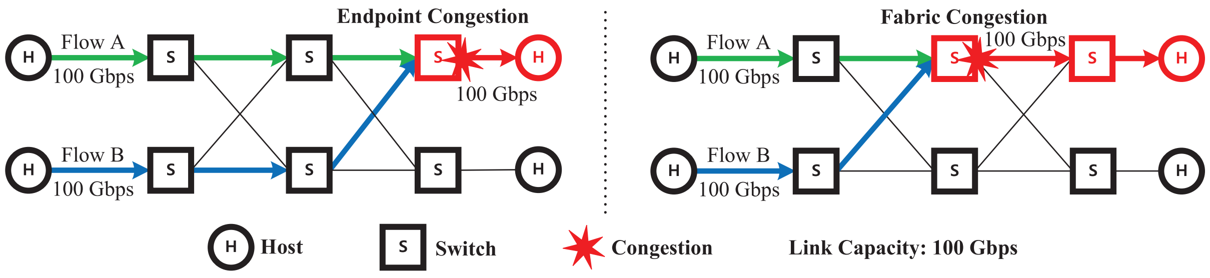

Fig. 1. Network congestion.

the techniques deployed in lossy data center networks. [20] presented a survey of end-to-end TCP congestion control schemes which do not require explicit feedbacks. [21] surveyed the congestion control schemes which aims at reducing the Flow Completion Time (FCT) and presented them according to four major techniques they used. [22] comprehensively surveyed several congestion control schemes for fast transmission of flows in DCNs, and compared the surveyed schemes based on their objectives, features and working mechanisms. Most recently, [23] surveyed a large number of traffic control schemes, and analyzed the characteristics, techniques and trade-offs of the surveyed schemes.

In this paper, we focus on the congestion control schemes for lossless data center networks. We classify the surveyed congestion control schemes into two categories: reactive congestion control (Section 4) and proactive congestion control (Section 5). Reactive congestion control schemes iteratively react to congestion signals, such as Explicit Congestion Notification (ECN) and delay. They are reactive in nature as they only take effect after the congestion happens. In contrast, proactive congestion control schemes only send packets when they think that the network pipe has enough capacity to hold them. We further classify the reactive schemes into two categories: switch-based, host-based, and classify the proactive schemes into three categories: centralized, end-to-end decentralized and hop-by-hop decentralized. We make comparisons among different schemes based on several properties, *e.g., facilitating small message*, *achieving fairness*, *implementation complexity*, *reaction speed* (for reactive schemes) and *preprocessing overhead* (for proactive schemes).

The rest of the paper is organized as follows. In Section 2, we introduce the network congestion, and present the criteria we used for comparing the surveyed congestion control schemes. In Section 3, we introduce some widely used LFC mechanisms, and analyze the defects of the LFC. In Section 4, we discuss the lossless reactive congestion control schemes. In Section 5, we discuss proactive congestion control schemes for lossless DCNs, and introduce some proactive schemes which are not based on the LFC for complementary. In Section 6, we present our understanding of the challenges and opportunities for congestion control in high-speed lossless DCNs. Finally, in Section 7, we conclude the paper.

2. Background

2.1. Classification of network congestion

Network congestion happens when the rate of the input traffic exceeds the link capacity. Typically, there are two types of network congestion, i.e., the endpoint congestion and the fabric congestion [24,25]. Endpoint congestion is caused by the over-subscribed receiver (end-point), whereas, fabric congestion is caused by the saturated link between two nearby switches. As Fig. 1 shows, when Flow A (100 Gbps) and Flow B (100 Gbps) simultaneously arrive at the same destination (with the capacity of 100 Gbps), endpoint congestion occurs. In the same manner, when they meet at the same link between two switches, fabric congestion occurs. In high-speed lossless DCNs, both endpoint congestion and fabric congestion can cause dramatical performance degradation [26].

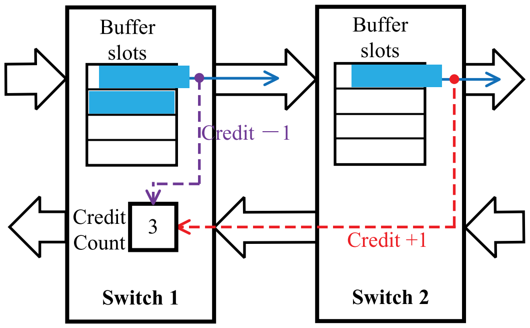

Fig. 2. CBFC mechanism.

2.2. Objectives of congestion control

The ultimate goal of congestion control is to regulate the traffic and manage congestion, furthermore, to mitigate the performance reduction caused by congestion. Hence, congestion control schemes should be designed to handle both endpoint congestion and fabric congestion, while either of them can severely affect the network performance. Additionally, to develop an applied congestion control scheme, some attached objectives should be taken into consideration, including low latency for short flows [27–30], fairness [31,32], applicability [33] etc. In DCNs, mice flows are more sensitive to latency than elephant flows, thus it is important to reduce the latency for short flows. Besides, long flows require high and stable throughput, fairness is significant for preventing the long flows from bandwidth loss and starvation caused by aggressive short flows. Moreover, the congestion control schemes should be cost-friendly, for widespread application.

3. Link-layer flow control

The Link-layer Flow Control (LFC) is a key technique to achieve lossless. However, it has some drawbacks since it may lead to congestion tree. In this section, we introduce the typical LFC schemes and analyze its defects.

There are three main categories of LFC: Credit-based flow control, ON/OFF flow control and ACK/NACK flow control [34]. Among these categories, credit-based flow control and ON/OFF flow control are the mainstream in lossless interconnections. Accordingly, we surveyed the Credit-Based Flow Control (CBFC), which belongs to the first category, and the Priority-based Flow Control (PFC), which belongs to the second category.

3.1. CBFC

The Credit-based Flow Control (CBFC) has been proposed for a long time. CBFC was first developed for ATM networks [35,36], and gradually employed by other fabrics, such as InfiniBand [37],

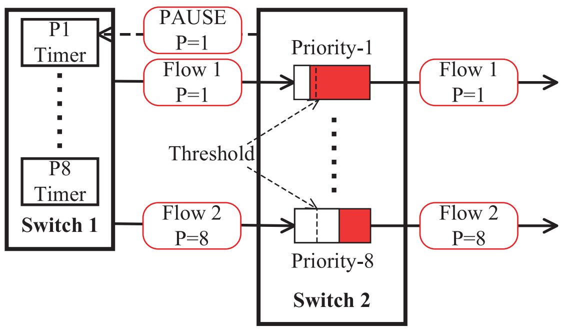

Fig. 3. PFC mechanism.

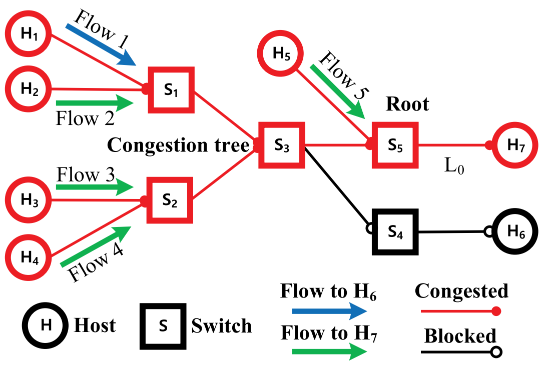

Fig. 4. Congestion tree.

PCIe [38] and Fiber Channel [39]. Fig. 2 illustrates the basic working mechanism of CBFC. Each switch maintains a credit count, and the credit count is dynamically updated with the data flows. When the downstream switch (switch 2) sends out a slot of data, i.e., a new buffer slot is made available, and a credit is sent to the upstream switch (switch 1) at the same time. When a credit arrives, the credit count of switch 1 will be increased by 1. Otherwise, when sends a slot of data to switch 2, the credit count of switch 1 will be decreased by 1. By doing this, the value of upstream switch's credit count is always equivalent to the available buffer slots of the downstream switch, and the switch buffer will not be overflowed.

3.2. PFC

IEEE 802.1Qbb, The Priority-based Flow Control (PFC) is an LFC mechanism proposed for the Converged Enhanced Ethernet (CEE) [40]. PFC employs eight virtual channels with eight different priorities, to distinguish different flow classes. For each priority, PFC uses IEEE 802.3 PAUSE frame [41] to prevent packets loss, and the PAUSE action in a VC does not affect other VCs. As shown in Fig. 3, in the VC with priority-1, queue length exceeds the threshold, and the switch 2 sends back a pause frame to the upstream switch. When receives a PAUSE frame, the P1 timer specifies how long the traffic should be paused, and the switch 1 pauses the transmission of the flow of corresponding priority. In each switch, there are eight timers controlling the pausing time for eight priority classes respectively. Note, to guarantee zero packet loss, an appropriate buffer headroom should be left for the packets which are injected during the time when a PAUSE frame is on the way to the previous switch.

3.3. Defects of LFC

LFC mechanisms have some limitations. Serious congestion could lead to tree saturation due to the LFC: congestion will spreads from the congested node (root) to the source nodes (leaves), forming a congestion tree. Once congestion tree is formed, a series of problems will arise: high queuing delay, low network throughput and unfairness. We take Fig. 4 as an example to explain these problems in detail, assuming that the link L_0 is congested and the LFC is triggered.

3.3.1. High queuing delay

Better than lossy networks, lossless networks can avoid retransmission delay. But when congestion occurs, the same as lossy networks, packets will also be queued in the switch leading to high queuing delay.

3.3.2. Decreasing network throughput

When the link L_0 is congested, the congestion first spreads from the switch S_5 to the switch S_3 and the source H_5. And then the congestion continuously spreads upstream until it reaches the source nodes H_1, H_2, H_3 and H_4. However, due to the congestion of S_3, all flows came from S_1 to S_3 are paused, including flow 1. Finally, this congestion tree blocks both S_4 and H_6, thus flow 1 is "accidentally injured" and unable to reach the destination H_6. Consequently, the link between S_3 and H_6 is idle and lead to a sharp decrease of network throughput.

3.3.3. Unfairness

The severe congestion of link L_0 leads to the congestion tree, and the transmissions of all upstream flows are paused, including flow 1 to flow 5. When the congestion is relieved, S_3 and H_5 restart their transmissions. And then, they equally divide the bandwidth of L_0 (flow 5 occupies 1/2 bandwidth and flow 2 to flow 4 divide the remaining 1/2 bandwidth). Finally, these flows share unequal bandwidth.

In addition, LFC also could lead to the Head of Line (HoL) blocking and deadlock [42,43], seriously affecting network performance. Therefore, for lossless networks, we need effective congestion control mechanism to avoid the side effects of LFC.

4. Reactive congestion control schemes

Reactive congestion control aims to handle the network congestion after it has occurred. It can be classified into two categories based on the position where congestion is detected, i.e., switch-based and host-based, as Fig. 5 shows.

The main idea of switch-based congestion control schemes is using explicit feedback to notify the source of network congestion, and then, the source accordingly reduces the sending rate of data which contribute to the congestion. When a switch detects the congestion, it can directly return the congestion information to the sender, or indirectly forward the congestion information to the receiver and then back to the sender by the receiver. We classify the former one as backward notification, conversely, the latter one as forward notification. The backward notification schemes react to congestion more quickly, while forward notification schemes put emphasis on end-to-end management. The typical lossy congestion control schemes DCTCP [16], HULL [44] and MQ-ECN [45] are switch-based forward notification schemes.

With host-based congestion control, the host-end directly determines congestion without the support of switches. That is, the whole switching network can be considered as a black box, so the source knows nothing about the network status. Normally, host-based congestion control schemes can be classified into delay-based and Loss-based. Delay-based schemes take transmission

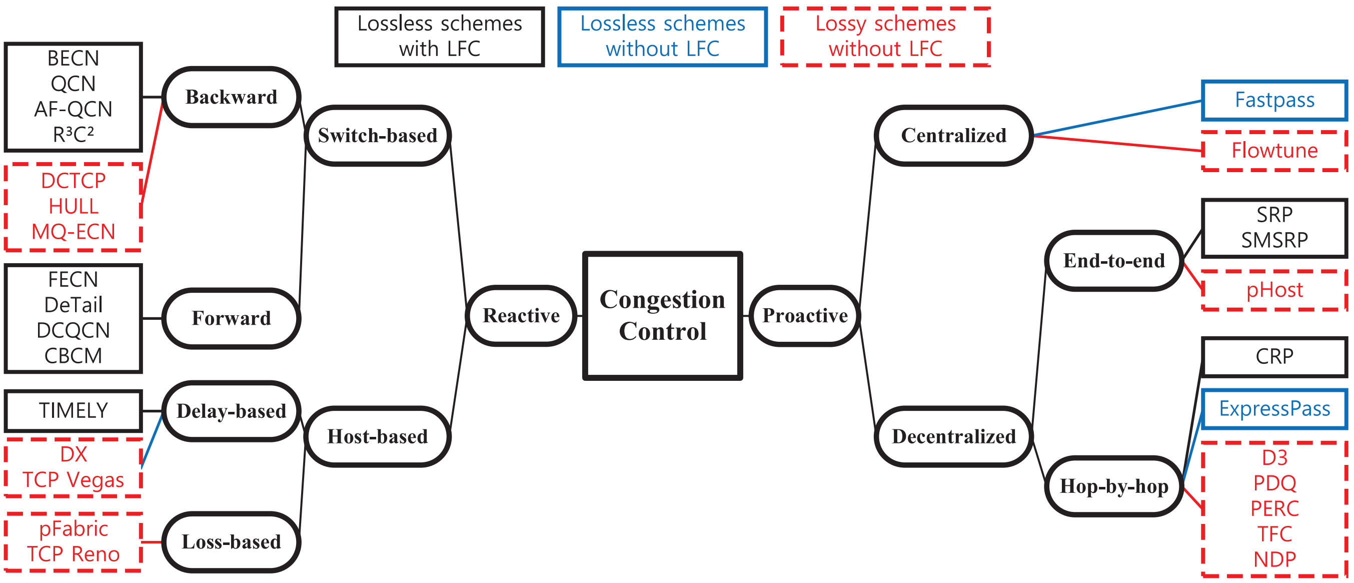

Fig. 5. Classifications of congestion control schemes: including both lossless schemes and some lossy schemes.

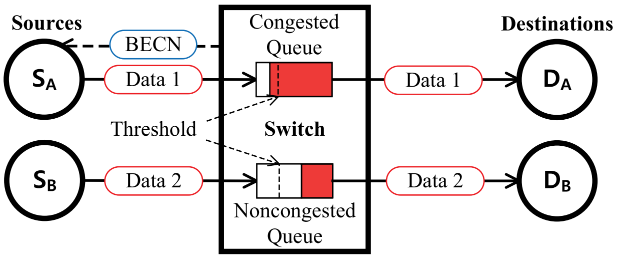

Fig. 6. BECN mechanism.

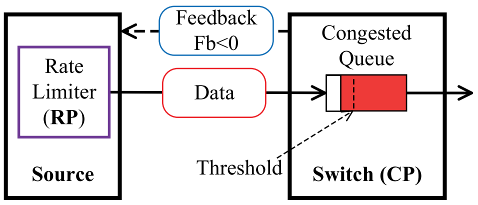

Fig. 7. QCN mechanism.

latency as the congestion signal, while loss-based schemes detect the congestion based on packet loss. Previous lossy host-based schemes DX [46], TCP Vegas [47] etc. react to congestion based on packet delay, while pFabric [48], TCP Reno [49] etc. take packet loss as the signal to determine congestion. In this paper, we only focus on the lossless schemes, so the packet drop cannot be a congestion signal.

In this section, we only discuss the reactive congestion control schemes deployed on lossless fabrics. According to the categories presented above, we introduce the surveyed schemes respectively and make some comparisons between them.

4.1. Switch-based congestion control

4.1.1. Backward notification

BECN. The Backward Explicit Congestion Notification (BECN) [50, 51] is the earliest backward notification scheme. With BECN, the congestion information is directly sent back to the source by the congested point (switch). As shown in Fig. 6, when the queue of a virtual channel exceeds a fixed threshold, the switch will directly send BECN cells to the source that contribute to the congestion. On each receiving of a BECN cell, the source cuts in half the sending rate of the specific virtual channel. If five successive BECN cells arrive, the source will pause the transmission. Otherwise, if no BECN cell arrives on the throttled virtual channel for a time period, the sending rate will be increased to the previous level, until regains the original rate.

Note, the BECN cell just likes a bit of congestion information, delivering whether the congestion occurs or not.

QCN. The Quantized Congestion Notification (QCN) [52,53], IEEE 802.1Qau, is a backward congestion control scheme developed for Ethernet networks. The main idea of QCN is that the switch directly sends back congestion message to inform sender to cut the sending rate. Unlike the BECN cell which only carries a bit of congestion information, the congestion message in QCN carries a quantized value of congestion degree to precisely control the injection rate.

QCN consists of two components: the Congestion Point (CP) and the Reaction Point (RP). CP represents the switch where congestion occurs and RP is the NIC rate limiter where the traffic comes from. The working mechanism of QCN is as shown in Fig. 7.

CP algorithm: The CP regularly calculates the $F_b = -(Q_{off} + wQ_{delta})$ to make sure whether the congestion occurs. Q_{off} represents the difference between current queue length and equilibrium queue length, i.e., $Q_{off} = Q - Q_{eq}$. Q_{delta} means the growth rate of the queue, that is, $Q_{delta} = Q - Q_{old}$, where Q_{old} is the queue length at previous processing. w is a fixed parameter usually set as 2. If the computed F_b is negative, meaning congestion exists, the switch will send this feedback to the source (RP). Otherwise, nothing would happen.

RP algorithm: When receives a feedback, the RP will decrease its sending rate based on the value of F_b When congestion disappears, RP will increase its rate following three phases: Fast Recovery (FR), Active Increase (AI) and Hyper-Active Increase (HAI). Every time after rate decreased, RP turns to the FR phase. The FR phase aims at rapidly recovering the current sending rate C_R to the target rate T_R (the rate before decreasing). There are five cycles in the FR phase, and at the end of each cycle, the sender increases the current rate according to the formula $C_R = (C_R + T_R)/2$, but the T_R is also increased as $T_R = T_R + R_{AI}$, where R_{AI} is a fixed value controls

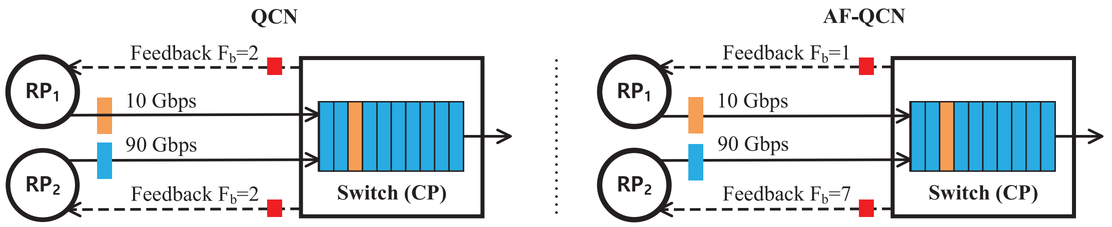

Fig. 8. The core idea of AFQCN.

the increasing rate of T_R. Moreover, HAI phase is more aggressive than the target rate increases as $T_R = T_R + iR_{HAI}$. In HAI phase, RP considers the link is so idle that the packets can be faster injected into the network, with the lack of negative F_b.

Given that some high-layer protocols, e.g., UDP, do not employ congestion control scheme, QCN can fill up their gaps by providing link-layer service. Likewise, QCN algorithm is designed to be stable and simple to implement. However, it cannot ensure the fairness, see Fig. 8.

AF-QCN. The Approximately Fair QCN (AF-QCN) [54] is a backward congestion control scheme, which aims to address the fairness issue of standard QCN. AF-QCN is developed based on the QCN and the Approximate Fair Dropping (AFD) algorithm [55], and it directly employs the Reaction Point (RP) of standard QCN at the source-end, without any modification.

The core of AF-QCN is to ensure a weighted fairness for congested flows by the AFD algorithm, basing on the transmitting rate of different classes of flows. As Fig. 8 shows, under this condition, standard QCN algorithm can result in unfairness between flow 1 and flow 2, while the AF-QCN algorithm provides favorable fairness. As we can see, the core of the AF-QCN algorithm is determining the value of F_b.

AF-QCN calculates the weighted F_b as follows:

$$F_b = (1-\alpha)F_{b-QCN} + \alpha F_{b-AF} \tag{1}$$

Where α is fixed as $0 < \alpha < 1$, and F_b is the feedback value determined by the original QCN algorithm. The calculating of F_{b-AF} is nearly identical to AFD, and it includes three steps:

Step 1: AF-QCN updates the arrival rate of each flow (or flow class) every T_s seconds as follows:

$$M_i = (1-\beta)M_i + \beta M_{i-new} \tag{2}$$

Where β is a fixed value, $0 < \beta < 1$, and M_i is the estimated data amount of the class i during the T_s, while M_{i-new} is the actual arrived data amount of class i.

Step 2: AF-QCN calculates the fair share M_{fair_i} of class i as follows:

$$M_{fair_i} = \frac{W_i}{\sum_j W_j}\sum_j M_j \tag{3}$$

Where W_i denotes the corresponding weight of class i, for allocating bandwidth.

Step 3: Similar to the AFD algorithm, AF-QCN calculates the value of F_{b-AF} as follows:

$$\begin{aligned} D_i &= (1 - M_{fair_i}/M_i)_+ \\ F_{b-AF} &= \lfloor 64D_i \rfloor \end{aligned} \tag{4}$$

Where D_i is the drop probability in AFD algorithm, and F_{b-AF} is obtained by encoding D_i into a equivalent 6 bit number.

AF-QCN achieves fast convergence to weighted fairness, taking just a few milliseconds. And AF-QCN can provide associated services to different classes of cloud applications (flows), therefore, benefits the Quality of Service (QoS). However, the defects of AFD algorithm are not addressed by AF-QCN: the AFD algorithm does not operate well on unresponsive flows, and it can be decreased when the network is with multiple congested links [56]. On the other hand, the sampling limitations of QCN are out of the consideration of AF-QCN [57].

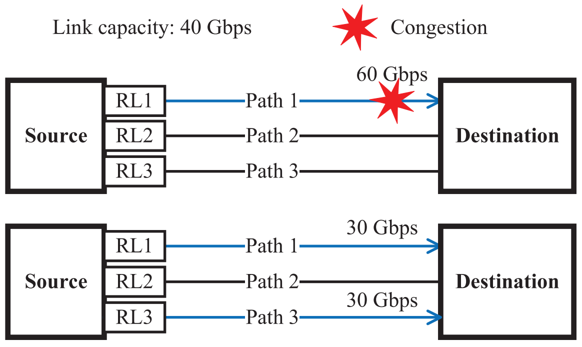

Fig. 9. R^3C^2 operating process.

R^3C^2. The Reactive Route & Rate Controller (R^3C^2) [58] is an integrated congestion control scheme, which combines the layer-3 source driven Reactive Route Control (R^2C^2) with the layer-2 QCN Reaction Point (RP) to utilize available multipaths.

The main idea of R^3C^2 is throttling the injection rate on the congested path and forwarding the congested packets to other available idle paths, as Fig. 9 depicts. In R^3C^2, each path maintains a Rate Limiter (RL). When the congestion occurs on the flow's original transmission path (path 1), the Rate Limiter 1 (RL1) will throttle the injection rate on the congested path, in accordance with the basic QCN algorithm. Meanwhile, the R^3C^2 adapter chooses another available path (path 3) to transmit the rest of flow without the operation of Rate Limiter 3 (RL3).

By employing load balance and rate control simultaneously, R^3C^2 can mitigate both endpoint congestion and fabric congestion. However, The fairness issue of QCN is not addressed by it.

4.1.2. Forward notification

FECN. The Forward Explicit Congestion Notification (FECN) [50,59] is an end-to-end congestion control scheme, which was first proposed for the Asynchronous Transfer Mode (ATM) networks. FECN uses the one-bit explicit forward congestion indication (EFCI) field to convey the congestion information, and its operating process is as Fig. 10 shows. A packet is injected into the network with the initial EFCI state of 0, and the value of EFCI should not be changed if there is no congestion. However, the packet's EFCI field will be marked with 1 by the switch when passes a congested queue, and then continue to be forwarded. Upon receiving a marked packet, it feedbacks a resource management (RM) cell to the correspond source-end. When receives the RM cell, the source starts to throttle the injection rate, just like the source-end in the BECN scheme.

There are some variants of FECN, e.g., the IP-header ECN (IP-ECN) [60,61] and the InfiniBand Congestion Control (IB CC) [62,63]. Ordinarily, the working principles of them are almost the same.

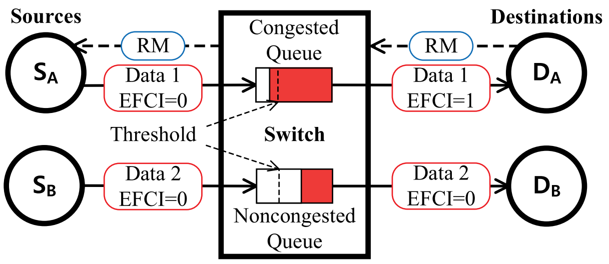

Fig. 10. FECN mechanism.

DeTail. DeTail [64] is an integrated reactive congestion control scheme which employs ECN mechanism and adaptive load balance to handle network congestion. Different layers of network stack are co-designed in the DeTail, and each layer has its own function. Fig. 11 shows the components of DeTail and their functions.

Link layer: DeTail employs PFC on link layer to achieve no packet loss. The switch continuously monitors its ingress port to detect the congestion. When congestion occurs, PAUSE frames will be sent back to the previous switch, thus avoiding the packet loss caused by congestion.

Network layer: In network layer, DeTail employs a forwarding engine to get the available paths for a packet. When a packet arrives at the switch, the forwarding engine will find its corresponding forwarding entry which is associated with a bitmap of the acceptable ports. The acceptable ports are leading to the shortest path for the packet to its destination. In addition, DeTail uses the egress port Virtual Channel (VC) occupancy to reflect the congestion degree of a path and generates a favored ports bitmap that reflects the ports with the lowest congestion degree. Through combining these two bitmaps, the switch can allocate a shortest and less congestion path for the packet, thus, achieving effective load balancing. Note, using VC occupancy as the congestion indicator, rather than port occupancy, provides support for PFC at the link layer.

Transport layer: In transport layer, DeTail develops a custom transport protocol to meet the network requirements. DeTail employs fast recovery and fast retransmit disabled TCP NewReno to avoid needless monitoring and to handle packet reordering due to adaptive routing. On the other hand, DeTail uses the ECN-supporting switch to mark the congested packet's IP header with ECN label, so that the ECN-supporting sender can cut the injection rate of the corresponding flow that contributes to the congestion. Note, DeTail only uses ECN mechanism for low priority flows, so it indirectly facilitates the transmission of high priority flows.

Application layer: In the DeTail stack, the application layer assigns priority to the flow, basing on its latency sensitivity. Although there are eight priorities of the PFC, DeTail only uses two of them, i.e., latency-sensitive or not.

The DeTail stack is an organic whole, it achieves high link utilization through per-packet load balancing. However, it is difficult to deploy since every layer of the network should be modified more or less.

DCQCN. DCQCN [19] is an end-to-end congestion control protocol enabling the deployment of the Remote Direct Memory Access (RDMA) in large-scale DCNs. DCQCN extends the QCN mechanism to IP-routed L3 networks by employing IP-ECN mechanism. DCQCN consists of three elements: the Congestion Point (CP), the Notification Point (NP), and the Reaction Point (RP). Further, CP is usually the switch, NP is the destination and RP is the source.

CP algorithm: The key mechanism of CP is performing RED-ECN functionality, that is, using the Random Early Detection (RED) mechanism to mark packets. Three parameters, K_{min}, K_{max} and P_{max}, are set to determine whether a packet should be marked with an ECN tag. Specifically, if the buffer occupancy is between K_{min} and K_{max}, the switch will mark an arriving packet with CE codepoint proportionally. If the occupancy exceeds K_{max}, the switch will certainly mark the packet. Otherwise, packets will not be marked, as is shown in Fig. 12.

NP algorithm: An NP reacts to the marked packets by sending congestion notification packets (CNPs), which have been defined by the RDMA over Converged Ethernet version 2 (RoCEv2) standard [65]. The NP algorithm mainly focuses on rightly generating and transmitting the CNPs. If there arrives an ECN-marked packet, at first, the NP will check which flow the packet belongs to: if no CNP generated due to the flow in the last time slot (N microseconds, usually 50us), a CNP will be generated and sent by the NP immediately, a new time slot begins. Otherwise, if any marked packet arrives within the current time slot (N microseconds), the NIC only generates one CNP packet at the beginning of next time slot. As a consequence, if a long-term congestion exists, a congested flow will bring the CNPs about "per CNP per N microseconds".

RP algorithm: The main function of RP is to adjust the transmission rate according to the received CNPs. There are three steps of RP algorithm: Rate Decrease, Rate Increase, and Fast Recovery.

When the sender receives a CNP, it turns to the period of rate decrease. At the beginning, the sender remembers the current rate as the target rate (R_T) for fast recovery, like basic QCN. Then, it reduces its current rate (R_C) and updates the rate reduction factor (α), as follows:

$$\begin{aligned} R_T &= R_C \\ R_C &= R_C(1-\alpha/2) \\ \alpha &= (1-g)\alpha + g \end{aligned} \tag{5}$$

Note the NP will not generate CNP if it does not receive any ECN-marked packets within a time slot of N microseconds. Correspondingly, the RP will update the α according to another Eq. (6) when no CNP arrives in a fixed time slot K (usually 55us), which must be larger than the CNP generating time slot at NP.

$$\alpha = (1-g)\alpha \tag{6}$$

The second step, RP increases its ingress rate with three sequential phase, fast recovery, additive increase and hyper increase.

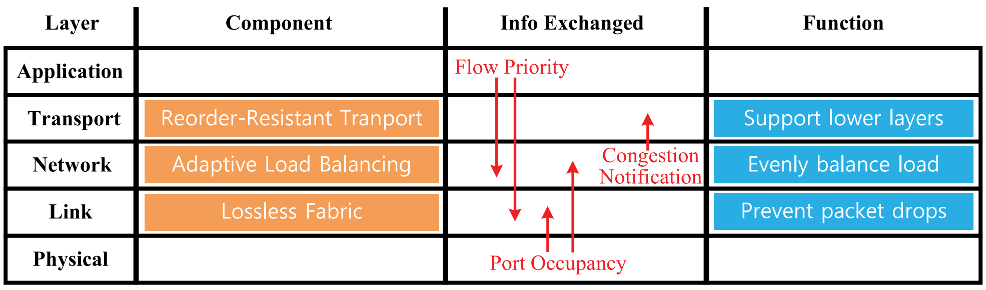

Fig. 11. DeTail Architecture.

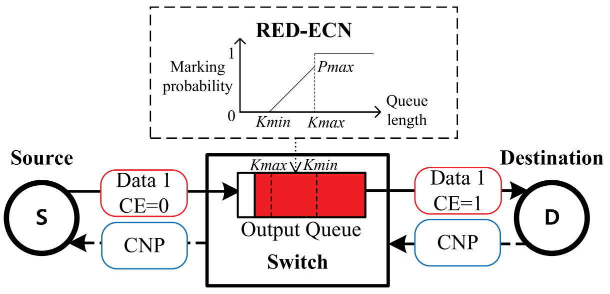

Fig. 12. DCQCN mechanism.

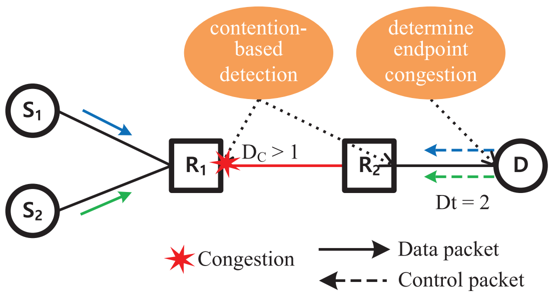

Fig. 13. CBCM mechanism.

In the fast recovery phase, the sending rate increased by every fixed cycle that is defined by a timer and a byte counter. The timer increases rate every T time units, while the byte counter increases rate for every B bytes. These two components are complementary that the timer ensures the quick recovery of flow even if its transmitting rate has dropped to extremely low, whereas the byte counter comes into effect when the transmitting rate reaches a high level.

During the phase of fast recovery, RP rapidly increases its sending rate towards the fixed target rate (R_T) in $F = 5$ cycles which are counted by the timer and byte counter. Eq. (7) presents the method of fast recovery:

$$R_C = (R_C + R_T)/2 \tag{7}$$

The Additive increase, where the current rate slowly draws near the target rate, closely follows the fast recovery, and the target rate is increased by R_{AI}, as follows:

$$\begin{aligned} R_T &= R_T + R_{AI} \\ R_C &= (R_C + R_T)/2 \end{aligned} \tag{8}$$

In addition, to achieve faster rate gaining, there is a further phase of hyper increase. Ordinarily, hyper increase aims to obtain more throughput when bandwidth frees up.

When congestion happens, the basic QCN directly throttles the sender who contributes to the congestion (at the link layer). As a sender may transmit several flows at the same time, consequently, some flows which do not contribute the congestion are accidentally injured by the QCN mechanism. By marking the ECN in IP header, DCQCN can throttle the congested flows respectively at the transport layer. Therefore, DCQCN can achieve the fairness which QCN mechanism does not care about.

DCQCN was improved by Guo et al. after one-year deployment [17]. They proposed DSCP-based PFC to enhance the compatibility of basic VLAN-based PFC in layer-3 networks. Besides, they addressed four practical problems of basic DCQCN: RDMA transport livelock, PFC deadlock, PFC pause frame storm and slow receiver symptom.

CBCM. The Contention-based Congestion Management (CBCM) [24] is a reactive congestion control scheme upon the lossless network. CBCM mainly aims at addressing the endpoint congestion by throttling the source according to the value of contention degree in the router, and it adopts load balance scheme as supplementary to mitigate the fabric congestion. Given that the load balance mechanism may lead to congestion spreading when the endpoint congestion exists [24], CBCM distinguishes between fabric congestion and end-point congestion and manages them separately.

Fig. 13 depicts the basic framework of CBCM, where two sources (S1 and S2) send data to a single target node (D) through two routers (R1 and R2). In the router, the competition degree D_C for each output port is calculated each cycle. If $D_C > 1$, it is determined that competition occurs at the output port, i.e., congestion happens; otherwise when $D_C \leq 1$, there is no competition. When the packet passes through the congested output port (e.g., the output port of *R1* is congested), its header is marked by the router. Then, the marked packet is forwarded to the destination.

At the destination, when the endpoint congestion is detected, the control packet, which carries a value of D_t, is generated and sent to each source that contributes to the endpoint congestion. When a source receives a control packet, the rate of flow towards the hot destination is adjusted to $1/D_t$, where the D_t (e.g., $D_t = 2$) is the throttle value equals to the number of source-ends that contribute to congestion. And then, the throttled data packet is isolated from the normal packet, only using the minimal routing mechanism to the destination through a separate VC. When the source no longer contributes to the endpoint congestion, the throttled transmission is recovered. Meanwhile, the source notifies the destination by sending an *unthrottle* packet. Note, adaptive routing is only used for the flows that do not contribute to the endpoint congestion.

CBCM proposed a contention-based detection scheme which is different from the schemes based on buffer occupancy, and it produces lower queuing delay than these schemes. CBCM distinguishes the fabric congestion from endpoint congestion, and resolves them by injection throttling and adaptive routing, respectively. By doing this, CBCM can efficiently manage the congestion and mitigate the congestion spreading. However, CBCM requires great modification of the switch, so it is hard to deploy in large commercial data centers.

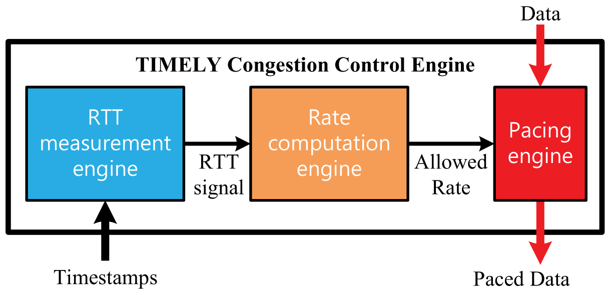

Fig. 14. TIMELY overview.

4.2. Host-based congestion control

4.2.1. Delay-based congestion control

TIMELY. TIMELY [66] is a Delay-based reactive congestion control scheme taking RTT as congestion signal, and it is implemented in the context of RDMA. End-to-end transmission latency is mainly caused by the queuing delay in network nodes. That is, the Round-Trip Time (RTT) of a packet can embody the queuing delay of all the channels it passes through and, further, reflect the congestion status in the network. The RTT is an effective congestion

Table 1
Comparison of reactive congestion control schemes.

Categories		Reactive schemes	Mechanisms of handling fabric congestion	Facilitating small message	Achieving fairness	Implementation complexity	Reaction speed
Switch-based	Backward Notification	BECN	Injection throttling			Low	Fast
		QCN	Injection throttling			Low	Fast
		AF-QCN	Injection throttling	√	√	Medium	Fast
		R^3C^2	Exploiting multi-path			Medium	Fast
	Forward Notification	FECN	Injection throttling			Low	Medium
		DeTail	Exploiting multi-path	√		High	Medium
		DCQCN	Injection throttling		√	Medium	Medium
		CBCM	Exploiting multi-path		√	High	Medium
Host-based	Delay-based	TIMELY	Injection throttling		√	Low	Slow

signal without any requirement of the switch feedback, and it is estimated per completion event which signaled by the successful transmission of a segment (16–64 KB) of data.

TIMELY mainly focuses on achieving the RTT-based rate control in NIC without relying on transport protocol, and its framework mainly composed of three components: (1) RTT measurement engine; (2) Rate computation engine; and (3) Rate control engine, as Fig. 14 shows.

At first, when the NIC receives an ACK for a segment of data, the measurement engine will precisely measure its RTT according to Eq. (9). Where $t_{completion}$ represents the arriving time of ACK, and t_{send} is the sending time of the first packet of the segment.

$$RTT = t_{completion} - t_{send} - \frac{seg.size}{NIC\ line\ rate} \tag{9}$$

Then, the measurement engine delivers the measured RTT to the rate computation engine, which runs the core congestion control algorithm to generate an updated target rate. If the received RTT is less than T_{low}, TIMELY will additively increase the ingress rate by δ, where δ is the bandwidth additive increment constant. If the RTT is more than T_{high}, the rate will be decreased multiplicatively by β, where β is the multiplicative decrement factor. Otherwise, the RTT is between T_{low} and T_{high}, the sending rate will be adjusted depending on the gradient of RTT. The gradient is defined as the normalized change between two successive arriving RTT samples. If the gradient is positive (i.e., RTT is increasing), the sending rate will be reduced multiplicatively in direct proportion to it. Otherwise, the rate will be increased additively by δ.

Ordinarily, in the rate control engine, messages and flows are broken into segments for transmission. The segments of different flows are appropriately scheduled for transmitting, according to the target rate provided by rate computation engine.

Other than most of the congestion control algorithms which require the switch to provide congestion detection function, like ECN, QCN, DCQCN, CBCM, etc. TIMELY uses delay measurements to detect congestion. Notwithstanding, it achieves low network latency and high throughput via the RTT-based congestion control algorithm.

4.3. Comparisons

We make a comparison of the surveyed reactive schemes from five aspects: Mechanisms of handling fabric congestion, facilitating small message, achieving fairness, implementation complexity and reaction speed. We use Table 1 to show their differences, and discuss them as follows.

4.3.1. Mechanisms of handling fabric congestion

By combining adaptive routing mechanisms and congestion control mechanisms, R^3C^2, DeTail, CBCM manage both fabric congestion and endpoint congestion. BECN, QCN, AF-QCN, FECN, DCQCN and TIMELY throttle injection rate when congestion occurs, so they can mitigate congestion tree. However, they injure the flows which have no contribution to endpoint congestion, when these flows can be routed to other idle links.

4.3.2. Facilitating small message

In DeTail, short messages are prioritized to bypass the ECN mechanism, so they are transmitted without throttling. AF-QCN employs a weighted congestion feedback and less reduces the transmitting rate of small messages under congestion conditions.

4.3.3. Achieving fairness

By using the AFD algorithm, AF-QCN addresses the unfairness issue of standard QCN. DCQCN uses RED-like marking mechanism and fast rate recovery to achieve fairness. TIMELY employs AIMD algorithm to achieve fairness across connections. CBCM employs separated VCs to isolate different flow classes and throttles all the congested flows to adjust the VCs' transmitting rate, so as to mitigate the unfairness between different hotspot senders.

4.3.4. Implementation complexity

BECN and FECN have been supported by the commercial switches, and the functionality of QCN has been provided by the Ethernet switches, so they hold almost zero implementation complexity. TIMELY is easy to implement, it does not need the support of switch. R^3C^2 and AF-QCN are QCN-based schemes requiring the support of QCN mechanism. DeTail and DCQCN need the switch to provide ECN support. In CBCM, the router should be modified to provide the functionalities of contention detection and packet marking. Besides, DeTail and CBCM employ switch (or router) based load balance mechanisms, more complex to implement. The Reactive congestion control mechanisms are relatively simple to implement, and they have been widely deployed in most of the networks. But their performance could be dramatically decreased without appropriate parameter configuration.

4.3.5. Reaction speed

Reaction speed is reflected by the time interval between when the congestion occurs and when the sender begins to manage the congestion. Among the surveyed schemes, BECN, QCN, AF-QCN and R^3C^2 reacts fastest to congestion, because the congestion feedback is directly sent back by the switch. The forward notification schemes, i.e., FECN, DeTail, DCQCN and CBCM are slower than the backward schemes, they rely on the receiver to feedback the congestion information. Besides, TIMELY, as a host-based scheme, aims to reduce end-to-end latency by taking RTT as feedback, but it spends longer time reacting to congestion than the switch-based schemes.

5. Proactive congestion control schemes

Contrary to the reactive congestion control schemes which manage the network congestion after it has occurred, proactive congestion control schemes aim to prevent congestion from the formation. Proactive congestion control schemes can be divided into centralized and decentralized. The centralized schemes mainly rely on a centralized scheduler to manage and allocate network resources; decentralized schemes distribute this task to

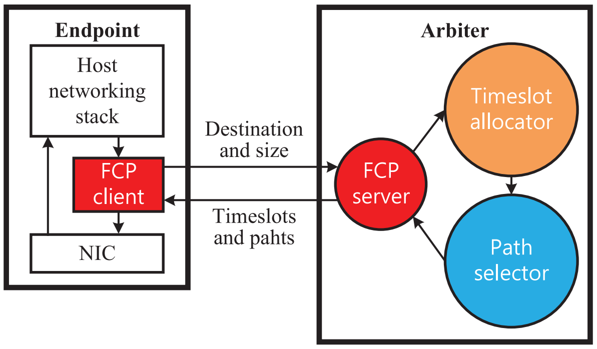

Fig. 15. Fastpass overview.

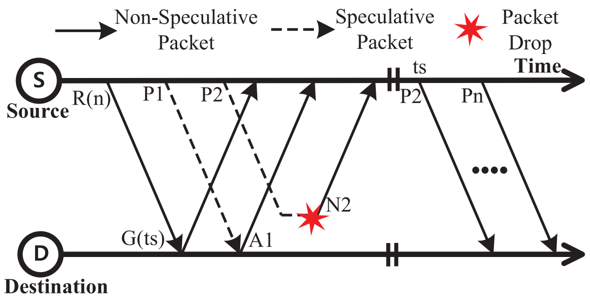

Fig. 16. SRP operation diagram.

multiple network nodes (hosts and switches). Further, the decentralized schemes can be subdivided into two categories: end-to-end and hop-by-hop, as Fig. 5 shows. In the end-to-end schemes, the sender directly requests the resources from the receiver, without the participation of switch. The hop-by-hop schemes require the sender, the receiver and the switches which the data flow passes through to work together, so as to regulate the network resources.

Many proactive congestion control schemes have been proposed for lossy data center networks, e.g, (1) centralized schemes: FastPass [67], Flowtune [68]; (2) end-to-end decentralized schemes: pHost [69]; (3) hop-by-hop decentralized schemes: D^3 [70], PDQ [28], PERC [71], TFC [72], NDP [73], ExpressPass [74]. In FastPass, the centralized arbiter allocates the time slots for transmitting packets to avoid endpoint congestion, and selects the transmission paths for packets to avoid fabric congestion as well as to fully utilize available links. Similar to Fastpass, Flowtune manage the network resources using a centralized allocator, but it regulates the flows at the granularity of the flowlet [75]. With D^3, the sender requests a series of allocated rates from the routers which the flow passes by, and determines the flow sending rate as the minimum one of the allocated rates to avoid congestion. PERC is similar to D^3, and the flow is transmitted based on the available capacity of links which it passes through. PDQ also regulates the traffic based on the rate, but it has a specific goal of meeting flow deadlines and minimizing mean FCT. TFC is a window-based transport protocol. ExpressPass is a credit-based congestion control scheme, where each arrived credit packet triggers the sender to transmit a fixed-size data frame. In NDP, each *pull* packet, which is sent by the receiver, will pull a data packet from the sender.

Some proactive schemes can achieve near zero congestion packet losses. Hence, many of them may not be used in combination with the LFC. In this section, we mainly present the proactive schemes deployed in lossless DCNs, and meanwhile introduce some lossless proactive schemes which not rely on the LFC.

5.1. Centralized schemes

5.1.1. Fastpass

Fastpass [67] is a centralized proactive congestion control scheme, it achieves nearly-zero queue without the support of LFC. Fastpass deploys a global arbiter that connected to the ToR to regulate the flow transmission. In Fastpass, end-hosts communicate with the arbiter using the reliable Fastpass control protocol (FCP). The arbiter comprises the communication module, i.e., the FCP server, and the functional modules, i.e., the timeslot allocator and the path selector. In the data transmission, the host first provides the destination address and flow size information to the FCP server. And then, the FCP server deliver the knowledge to the timeslot allocator, and the allocator allocates the timeslot for the flow. After that, the transmission path of the flow is decided by the path selector, and finally the FCP server returns the timeslot and path to FCP client, i.e., the end-host. Fig. 15 depicts the framework of Fastpass. The operational process of Fastpass is as follows.

Timeslot allocation: In order to fully utilize the link, the size of timeslot is based on the bandwidth and the Maximum Transmission Unit (MTU), and the arbiter allocates the destination's resource for the sources according to "one timeslot one MTU-sized packet". The arbiter can achieve different allocation policies via changing the allocating order of source–destination pairs. For instance, it can achieve max–min fairness by "least recently allocated first" and min FCT by "fewest remaining MTUs first".

Path selection: Path selector allocates the transmission paths for packets, with the purpose of only one packet can transmit on the path in a single timeslot. By doing this, Fastpass can ensure no packet queuing in the network. The functionality of path selection can be performed with graph edge-coloring algorithm [76].

Fastpass can avoid congestion with zero queue status, reduce packet delay, gain bandwidth utilization and reduce the flow completion time. However, there are several drawbacks of Fastpass: the communication of the arbiter and hosts may occupies the bandwidth that provided for data transmission; the arbiter needs to consume large amounts of memory to maintain the global knowledge in the modern large data centers; the transmission of control information may result in great overhead; some client applications cannot provide the flow size information required by timeslot arbiter; the switch requires modification to provide support. Therefore, the applying and deploying of Fastpass are limited.

5.2. Decentralized end-to-end schemes

5.2.1. SRP

SRP [77] is a decentralized end-to-end congestion control scheme aims at reserving the available time slots of the destination to transmit data within its capacity, so as to avoid endpoint congestion.

Fig. 16 shows the process of SRP under the congestion condition. Before the connection between the source node S and the destination node D is established, a reservation packet R is sent firstly carrying a small amount of data. In order to ensure fast arrival, the R is assigned with the highest priority and transmitted through a separate virtual channel. After sending the reservation packet, the source node S begins to send speculative packets $P1$ and $P2$ to the destination node D. These speculative packets are transmitted through low priority virtual channels, with a limited transmission Time-to-Wait (TTW). If a speculative packet accumulates the queue waiting time exceeding the preset TTW value, it will be discarded by the router. Like $P2$, the router then sends back a *NACK* to the source node S. S stops sending packets in the speculative

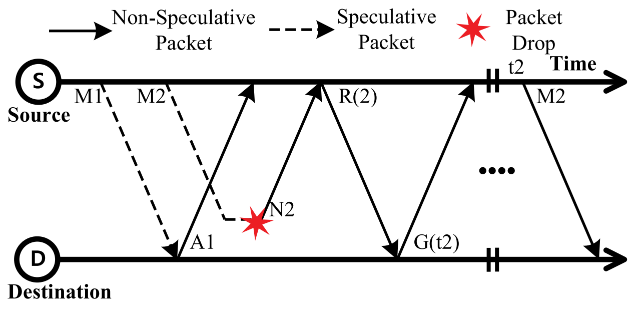

Fig. 17. Basic SMSRP operation diagram.

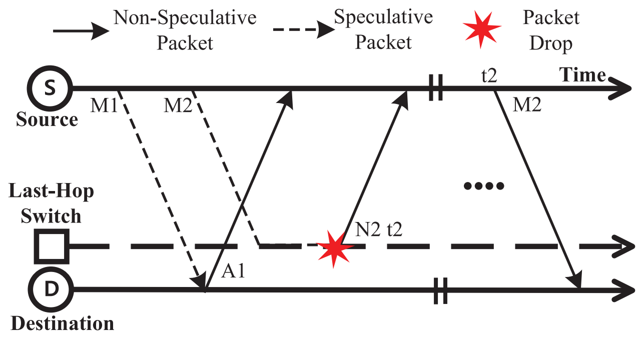

Fig. 18. LHRP-based SMSRP operation diagram.

mode when it receives *NACK*, waiting for an appropriate time to resume transmission. In order to achieve the timeout mechanism, each packet carries a timestamp in the header, and the router checks the timestamp at the head of the input port. Note, if there is no congestion, all the speculative packets can arrive destination smoothly. However, because of the unreliability of low-priority speculative packets, an *ACK* is also required to inform the source node whether a speculative packet has been transmitted successfully.

In SRP, each endpoint maintains a reservation schedule table. When the reservation packet arrives at the destination node *D*, it will send a grant packet to the source node *S* according to its own reservation schedule table. The grant packet $G(ts)$ contains the time *ts* when the next data packet can be sent. Meantime, the destination node *D* updates the reservation schedule table, so the next reservation packet from any other node will be assigned with a time point no earlier than the transfer time occupied by the previous source node. If the source node *S* receives the grant packet, the speculative packet will be paused until the time reaches *ts*. When *S* begins to send data packets in non-speculative mode, any packets that were previously dropped will be re-transmitted. Finally, when the source node *S* completes the transmission, the whole SRP process will be repeated.

Basic SRP protocol works well with the large and medium message transmissions. In SRP, a lightweight reservation handshake is adopted between the source and the destination to ensure no over-subscription at the destination node. In order to reduce the delay of the reservation procedure, the source sends out the speculative packets to cover the delay overhead. However, when facing fine-grained communication load, the payload size of speculative packets will be not enough to recover the cost of the reservation control packets.

5.2.2. SMSRP

Basic SMSRP. SMSRP [30] is a decentralized end-to-end scheme designed for counteracting the endpoint congestion caused by both long and short messages, it is an enhancement of basic SRP. Similar to the basic protocol principle, the reservation mechanism is used to avoid congestion, and speculative packets are used to cover the overhead caused by reservation. The design conception of SMSRP is as follows: if the endpoint is not congested, there is no need to make redundant reservation handshakes before every message transfer; in other words, it is not late to activate reservation mechanism when the congestion is detected.

Fig. 17 presents the operation diagram of SMSRP. The source node sends two small messages ($M1$ and $M2$) to the endpoint which may be congested, each of their sizes are the same as a packet. When the messages are ready to leave, the source immediately turns to the speculative mode. Then, the messages are transmitted through a low priority virtual channel, so that the routers can drop them when congestion occurs. If the packet successfully transmitted, as shown by $M1$, the reservation is not required anymore, and a positive response $A1$ will be sent back to the source. Therefore, when the endpoint is idle, the SMSRP protocol generates no additional overhead.

If the network endpoint is congested, the router drops the speculative packet after the queuing delay timeout. And then the router will send the corresponding *NACK* back to the source side. When receives a *NACK*, the source generates a reservation ($R(2)$) packet and sends it to the destination node to request the retransmission time. When receives the grant packet ($G(t2)$), the source will be paused and wait for the arrival of $t2$ to resend $M2$. Non-speculative messages are lossless and transmitted through a separate high priority VC, ensuring that they are not blocked by speculative messages.

SMSRP can achieve low latency in the non-congested network, and it can be easily integrated with the basic SRP protocol through very small hardware modification. Compared with the basic SRP, it mainly needs to modify the Network Interface Card (NIC) of the endpoint, and exchange the order of reservation handshake and speculative packet transmission.

LHRP-based SMSRP. The working principle of the LHRP-based SMSRP [30] is similar to the basic SMSRP, but it improves the basic SMSRP by using the Last-Hop Reservation Protocol (LHRP).

When a reservation is needed to resolve the congestion, the reservation control packet must arrive at the reservation scheduler of the destination NIC. Nevertheless, it may lead to a bandwidth contention between the reservation packet and the data packet transmitted in the last hop switch output channel. Intuitively, the bandwidth between the last hop Ingress port and the terminal card is the key resource that should be mainly used for transmitting data packets and as little as possible to be consumed by the transmission of the control message. By migrating the reservation scheduler from NIC to the last-hop switch, last-hop reservation protocol (LHRP) successfully addresses this problem.

The working process of switch-based SMSRP is shown in Fig. 18. The source directly sends two short messages to the destination in the speculative mode without reservation. If the transmission is successful, as in the case of $M1$, no further reservations are required and the protocol does not generate additional overhead. Under the condition of congestion, without using LHRP, the congested speculative message may be pushed into the previous switch and results in tree saturation; With LHRP-based SMSRP, the speculative packet will be discarded, so that it does not affect the transmission of non-speculative data packets, such as $M2$. Moreover, when the last hop switch drops a speculative packet, the reservation scheduler in the switch will allocate the retransmission time and return *NACK* ($N2t2$) to the source. When receives the *NACK* of $M2$, the source node resends message $M2$ at the time $t2$, in the non-speculative mode of ensuring lossless transmission.

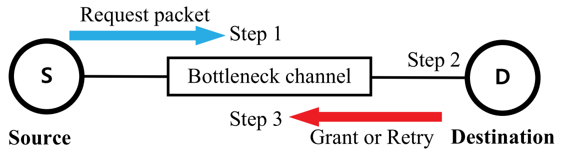

Fig. 19. CRP mechanism.

5.3. Decentralized hop-by-hop schemes

5.3.1. CRP

CRP [25] is a decentralized hop-by-hop scheme, with the goal of eliminating both channel congestion and endpoint congestion. In CRP, each endpoint and critical channel employs a reservation table, and each cell of the table represents the available bandwidth of the channel (or endpoint) within a certain time slot. The working process of CRP can be divided into three steps, as Fig. 19 depicts.

Reservation handling in channels: The source-end first sends a *request* packet to the destination, and the *request* carries a reservation vector. Each bit of the vector represents the availability of a time slot, and every bit of the vector is initiated with *TRUE*. When the *request* arrives a critical channel, the router compares the reservation table with the vector of *request* packet. If the available bandwidth of two adjacent time slots (the current one and the following one) is not enough to transmit the data, the router will set the corresponding bit to *FALSE*. Otherwise, the router does not modify the vector. Every time slot, which the *request* carries, is compared by the router.

Reservation handling in destinations: When the *request* packet arrives at the destination, the NIC compares its reservation table with the vector just like what routers do. Then, the destination generates a *grant* packet which carries the earliest time available for transmitting, and sends the *grant* back to the source using the reverse path.

Grant and retry operation: Further, when *grant* reaches the critical link on the reverse path, another comparison is needed. Within the granted time slots, if the critical link does not have enough resources to satisfy the request, the *grant* response will be converted to a *retry*. Or else, the associated cells of the reservation table will be decreased by the equivalent value of requested bandwidth, to prevent other flows reserving these time slots.

CRP employs speculative packets to override the extra latency and bandwidth of the transmission of *request* and *grant* packets. The drop of a speculative packet will feedback a *NACK* to the source, and the successful transmitting of a speculative packet will generate an *ACK* returning to the source. Besides, CRP uses four VCs to transmit reservation requests, response control packets, speculative packets and data packets, respectively.

CRP prevents the over-subscription of all critical network resources, thus avoids both endpoint congestion and fabric congestion. In addition, the reservation overhead is covered by the speculative packets, and the transmission delay of small messages can be mitigated by skipping the reservation. However, the reservation control mechanism is hard to implement: it requires extensive modification of the switches; and a large amount of network status information is needed to be maintained and synchronized.

5.3.2. ExpressPass

ExpressPass [74] is a decentralized hop-by-hop congestion control scheme based on credit-scheduling, it achieves bounded queue without the LFC. Before sending the data, the sender delivers a credit request to inform receiver that there are flows waiting to be transmitted. When receives the request, the receiver sends credit packets to the sender. The credits are discardable and would be

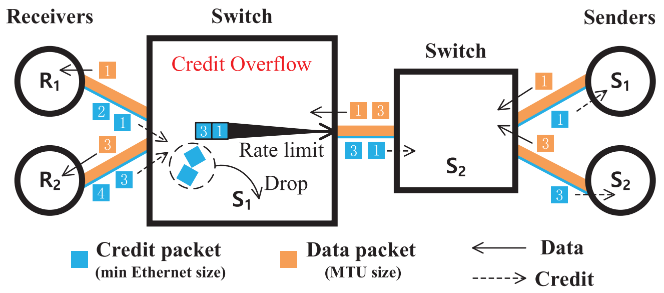

Fig. 20. ExpressPass overview.

dropped at the congestion point. Then the credit packets reach the sender, and the sender determines the transmission rate of the flow according to the amount of received credits. If the data packets are exhausted, the sender will send a CREDIT_STOP message to the receiver so that no more credit will be injected into the network. The overview of ExpressPass is as Fig. 20 shows, the S_1 sends data to the R_1 and the S_2 sends data to the R_2.

The switches forward the data packets and guarantee the symmetry of the paths of credits and data via the hash algorithm. As the size of credit packet is the min Ethernet size (84B) and size of data packet is the MTU size (1538B), switches and host NICs throttle the rate of credit packets to 5% bandwidth ($84/(84 + 1538)$) to ensure that the flow is transmitted within the link capacity, see Fig. 20.

Credit feedback control: There are some problems with naive credit-based approaches, such as unfairly shared and underutilized [74]. To address these problems, ExpressPass employs the credit feedback control, using the credit loss as the indicator. Each credit carries a sequence number, and the data packet brings the same number of its corresponding credit. Thus, the gap between two successive arrived packets at receiver indicates the number of discarded credits. The main goal of credit feedback control is to dynamically adjust the injecting rate of credit on different links based on the credit congestion status.

ExpressPass just needs to simply configure the priority queuing and rate limiting at the switch, and does not need to make any other modifications to the switching chip, thus it greatly reduces the implementation complexity. However, the reservation overhead is without its consideration, and it increases the latency to some degree.

5.4. Comparisons

We compare the surveyed reactive schemes from five aspects: Mechanisms of handling fabric congestion, facilitating small message, achieving fairness, implementation complexity and preprocessing time (see Table 2). Note, it is unnecessary to compare the reaction speed of proactive schemes, and we use the criterion of preprocessing time instead.

5.4.1. Mechanisms of handling fabric congestion

SRP and SMSRP avoid endpoint congestion by using bandwidth reservation at the destination, but they overlook the fabric congestion caused by bottleneck links, e.g., inter-cluster links [25]. Considering this problem, CRP uses channel reservation mechanism to avoid both endpoint congestion and fabric congestion. However, CRP does not exploit multi-path to full utilize idle links, and it injures the flows which only contribute to the fabric congestion. Since the credits are discarded at the congested link, ExpressPass can avoid fabric congestion by its credit-based scheduling mechanism. Fastpass uses its path selector to select available paths for flows, so as to avoid fabric congestion.

Table 2
Comparison of proactive congestion control schemes.

Categories		Proactive schemes	Mechanisms of handling fabric congestion	Facilitating small message	Achieving fairness	Implementation complexity	Preprocessing time
Centralized		Fastpass	Path selection	√	√	High	Long
Decentralized	End-to-end	SRP	None	√	√	Medium	Short
		SMSRP	None	√	√	Medium	Short
	Hop-by-hop	CRP	Channel reservation	√	√	High	Medium
		ExpressPass	Credit-based scheduling	√	√	Medium	Long

5.4.2. *Facilitating small message*

SRP and CRP allow small messages to bypass the reservation protocol by separated VCs, so as to avoid the relatively large overhead of handshake. But under heavy traffic condition, a large number of small messages will be out of control and then cause congestion. By making a little change in the basic SRP, SMSRP controls the transmission of small messages and further addresses the above issue. ExpressPass reduces the FCT for small flows by achieving fast convergence, thus it facilitates small message. Fastpass can facilitate small message by performing "the fewest remaining MTUs the first allocated", via the timeslot allocator.

5.4.3. *Achieving fairness*

Both SRP and SMSRP provides fairness by their equal share reservation mechanism. CRP divides reservation vectors among different traffic classes to guarantee fairness. Notwithstanding, using a separated VC without any throttling can result in throughput unfairness between hotspot senders, like the small message transmissions in SRP and CRP. ExpressPass solves the unfairness problem by its credit feedback control mechanism. Fastpass achieves fairness by performing "the least recently the first allocated" through the timeslot allocator.

5.4.4. *Implementation complexity*

SRP and basic SMSRP are implemented on end-hosts, farther, LHRP-based SMSRP needs a slight modification of the last-hop switch. Compare to SRP and SMSRP, CRP needs to implement on every critical network nodes, thus increases the reservation overhead and implementation complexity. ExpressPass is relatively easy to implement, as it requires no modification on the switch chip. Fastpass has high implementation complexity, because it needs a centralized arbiter to manage the whole network.

5.4.5. *Preprocessing time*

Preprocessing time is the overhead time that produced by the proactive schemes during reservation or pre-allocation. SRP and SMSRP use speculative packets to cover the reservation overhead, so the preprocessing time is short. Though CRP also leverages speculative packets, it needs longer time than SRP and SMSRP for preprocessing as it has more resources to reserve. Besides, Fastpass and ExpressPass need relatively longer time for preprocessing.

6. Opportunities and challenges

There are many trade-offs for congestion control design in lossless data center networks. Link-layer Flow Control (LFC) can prevent congestion packet losses, but cause many other problems, such as unfairness. Proactive schemes perform better than reactive schemes on the aspects of facilitating small messages and achieving fairness. But they are difficult to implement and have extra preprocessing time overhead. Although these trade-offs are hard to balance, there are still many opportunities and challenges for further research.

6.1. *Improving the LFC*

Current LFC mainly causes two problems: HoL blocking and deadlocks. HoL blocking occurs at the switch's ingress port, where the head packet blocks all the packets behind it. The deadlock only exists in lossless networks. We assume that a group of flows is transmitted in the network. When each flow in the group is waiting for the transmission of the other flows in this group, all flows will be paused, and the flow group enters the deadlock state.

6.1.1. *Mitigating the HoL blocking*

HoL can be mitigated by isolating different flow classes [40,78], such as the PFC and the Virtual Channels (VCs). However, it cannot be completely addressed by isolating flow classes, especially at the VCs where two (or more) flows pass through. Thus this is worthwhile to research.

6.1.2. *Eliminating the deadlock*

Guo et al. [17] proposes to drop blocked packets to mitigate the deadlock problem in lossless data center networks. However, discarding packet is costly as the packets need to be retransmitted, thus causing high retransmission latency. Therefore, how to efficiently handle the deadlock is still an open question in lossless data center networks.

6.2. *Exploiting multi-path properly*

Today's data centers provide multiple shortest paths for each host pair [79,80]. Many data center load balance schemes have been proposed to mitigate fabric congestion. Most of them focus on lossy networks. However, CBCM [24] points out that, since the congested packets cannot be discarded in lossless networks, existing load balancing solutions may cause congestion spreading when the endpoint congestion exists. To solve this problem, CBCM simply deactivates the adaptive routing mechanism when the endpoint congestion exists. We think load balancing in lossless networks still has large research space.

6.3. *Accurately and quickly reacting to congestion*

In recent years, the link speed of data centers has been significantly increased [15,71]. Given this trend, existing reactive congestion control schemes are faced with a serious problem: most of the flows can complete in very few RTTs, but reactive congestion control can only respond to network congestion in the second RTT. Therefore, how to respond to congestion accurately and quickly is the key challenge of the reactive schemes.

6.4. *Developing proactive schemes to avoid congestion*

Proactive congestion control can avoid congestion, and it does not need response time reacting to congestion. However, it is challenging to design a perfect proactive congestion control scheme, as follows.

6.4.1. Reducing the preprocessing time

Proactive schemes need preprocessing time to prepare the data transmission, and this generally costs an RTT. When the network load is light, the overhead of preprocessing can prolong the FCT. To cover the preprocessing time, some schemes (SRP, SMSRP and CRP) use speculative packets, and this may be helpful.

6.4.2. Precisely allocating the bandwidth

As for the SRP, SMSRP and CRP where the sources reserve the idle time of the destinations (and bottleneck channels), it is hard for the sources to perfectly schedule the sending time of different flows: a source may have several flows requesting to be transmitted to different destinations, and the destinations may grant overlapping time to these flows. Besides, as for the ExpressPass where the receivers returns credits to the senders, it is difficult for a sender to full utilize the scheduled bandwidth: a sender may have no data to send when the credits arrive, because the CREDIT_STOP packet needs time to be transmitted from the sender to the receiver. Therefore, it is a great challenge to precisely allocate the bandwidth.

6.4.3. Co-existing with other transport traffic

In order to maximize the link utilization, a proactive congestion control scheme needs to hold all the network information which it requires to make full use of the network resources. However, it is hard to guarantee that no other transport traffic will be injected into the network (such as internet traffic), especially in the commercial data centers. Once these intruders sneak into the network, a series of problems will be brought. Therefore, it is challenging to develop a proactive scheme to co-exist with other transport traffic.

6.5. Cross-layer co-design

In DCNs, various applications have different requirements, e.g., low latency, high throughput, fairness, reducing FCT, meeting deadlines etc. It is beneficial to design unique schemes for different specific applications, as it can provide associated performance requirements. Therefore, Cross-layer co-designing can be a worthwhile opportunity for developing congestion control schemes.

7. Conclusion

In this paper, we survey several congestion schemes for lossless DCNs. We classify the surveyed schemes into reactive and proactive. Besides, We make comparisons of different schemes based on five criteria: the mechanisms of handling fabric congestion, facilitating small message, achieving fairness, implementation complexity and reaction speed (or preprocessing time).

More deeply and broadly, by reviewing the research status of the lossless congestion control and the existing problems of high-speed lossless DCNs, we present some challenges and opportunities for designing congestion control schemes, including (1) improving the LFC, i.e., mitigating the HoL blocking and eliminating the deadlock, (2) exploiting multi-path properly, (3) accurately and quickly reacting to congestion (for reactive schemes), (4) developing proactive schemes to avoid congestion, i.e., reducing the preprocessing time, precisely allocating the bandwidth and co-existing with other transport traffic, (5) cross-layer co-design.

How to design an excellent congestion control scheme for a high-speed lossless DCN? There is no certain answer yet.

Acknowledgment

We sincerely appreciate Prof. Jiao Zhang (BUPT) for her constructive suggestions and the reviewers for their valuable comments. Personally, I want to thank Hanyi Shi and Yiming Li for their encouragements. They inspire me to complete this paper.

References

[1] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, R. Chaiken, The nature of data center traffic: measurements &amp; analysis, in: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference, ACM, 2009, pp. 202–208.
[2] A. Greenberg, J. Hamilton, D.A. Maltz, P. Patel, The cost of a cloud: research problems in data center networks, ACM SIGCOMM Comput. Commun. Rev. 39 (1) (2008) 68–73.
[3] S. Di, D. Kondo, F. Cappello, Characterizing cloud applications on a google data center, in: Parallel Processing, ICPP, 2013 42nd International Conference on, IEEE, 2013, pp. 468–473.
[4] G. Wang, T.E. Ng, The impact of virtualization on network performance of amazon ec2 data center, in: Infocom, 2010 Proceedings IEEE, IEEE, 2010, pp. 1–9.
[5] A. Kalia, M. Kaminsky, D.G. Andersen, Using rdma efficiently for key-value services, in: ACM SIGCOMM Computer Communication Review, vol. 44, no. 4, ACM, 2014, pp. 295–306.
[6] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, S. Yekhanin, et al. Erasure coding in windows azure storage, in: Usenix Annual Technical Conference, Boston, MA, 2012, pp. 15–26.
[7] J. D'ambrosia, 40 gigabit ethernet and 100 gigabit ethernet: The development of a flexible architecture [commentary], IEEE Commun. Mag. 47 (3) (2009).
[8] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, A. Vahdat, Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network, SIGCOMM Comput. Commun. Rev. 45 (4) (2015) 183–197.
[9] A. Roy, H. Zeng, J. Bagga, G. Porter, A.C. Snoeren, Inside the social network's (datacenter) network, in: ACM SIGCOMM Computer Communication Review, vol. 45, no. 4, ACM, 2015, pp. 123–137.
[10] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[11] D. Abts, J. Kim, High performance datacenter networks: Architectures, algorithms, and opportunities, Synthesis Lect. Comput. Architect. 6 (1) (2011) 1–115.
[12] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, J.M. Hellerstein, Distributed graphlab: a framework for machine learning and data mining in the cloud, Proc. VLDB Endowment 5 (8) (2012) 716–727.
[13] C. Evangelinos, C. Hill, Cloud computing for parallel scientific hpc applications: Feasibility of running coupled atmosphere-ocean climate models on amazons ec2, Ratio 2 (2.40) (2008) 2–34.
[14] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al., A view of cloud computing, Commun. ACM 53 (4) (2010) 50–58.
[15] W. Bai, K. Chen, S. Hu, K. Tan, Y. Xiong, Congestion control for high-speed extremely shallow-buffered datacenter networks, in: Proceedings of the First Asia-Pacific Workshop on Networking, ACM, 2017, pp. 29–35.
[16] M. Alizadeh, A. Greenberg, D.A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, M. Sridharan, Data center tcp (dctcp), in: ACM SIGCOMM Computer Communication Review, vol. 40, no. 4, ACM, 2010, pp. 63–74.
[17] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, M. Lipshteyn, Rdma over commodity ethernet at scale, in: Proceedings of the Conference on ACM SIGCOMM 2016 Conference, ACM, 2016, pp. 202–215.
[18] G.F. Pfister, V.A. Norton, Hot spot contention and combining in multistage interconnection networks, IEEE Trans. Comput. 100 (10) (1985) 943–948.
[19] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M.H. Yahia, M. Zhang, Congestion control for large-scale rdma deployments, in: ACM SIGCOMM Computer Communication Review, vol. 45, no. 4, ACM, 2015, pp. 523–536.
[20] A. Afanasyev, N. Tilley, P. Reiher, L. Kleinrock, Host-to-host congestion control for tcp, IEEE Commun. Surv. Tutor. 12 (3) (2010) 304–342.
[21] S. Liu, H. Xu, Z. Cai, Low latency datacenter networking: A short survey, 2013. ArXiv preprint arXiv:1312.3455.
[22] R. Rojas-Cessa, Y. Kaymak, Z. Dong, Schemes for fast transmission of flows in data center networks, IEEE Commun. Surv. Tutor. 17 (3) (2015) 1391–1422.
[23] M. Noormohammadpour, C.S. Raghavendra, Datacenter traffic control: Understanding techniques and trade-offs, 2017. ArXiv preprint arXiv:1712.03530.
[24] G. Kim, C. Kim, J. Jeong, M. Parker, J. Kim, Contention-based congestion management in large-scale networks, in: Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on, IEEE, 2016, pp. 1–13.
[25] G. Michelogiannakis, N. Jiang, D. Becker, W.J. Dally, Channel reservation protocol for over-subscribed channels and destinations, in: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, ACM, 2013, p. 52.
[26] P.J. Garcia, F.J. Quiles, J. Flich, J. Duato, I. Johnson, F. Naven, Efficient, scalable congestion management for interconnection networks, IEEE Micro 26 (5) (2006) 52–66.

[27] N. Dukkipati, N. McKeown, Why flow-completion time is the right metric for congestion control, ACM SIGCOMM Comput. Commun. Rev. 36 (1) (2006) 59–62.
[28] C.-Y. Hong, M. Caesar, P. Godfrey, Finishing flows quickly with preemptive scheduling, in: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, ACM, 2012, pp. 127–138.
[29] A. Munir, I.A. Qazi, S.B. Qaisar, On achieving low latency in data centers, in: Communications, ICC, 2013 IEEE International Conference on, IEEE, 2013, pp. 3721–3725.
[30] N. Jiang, L. Dennison, W.J. Dally, Network endpoint congestion control for fine-grained communication, in: High Performance Computing, Networking, Storage and Analysis, 2015 SC-International Conference for, IEEE, 2015, pp. 1–12.
[31] T. Bonald, L. Massoulié, A. Proutiere, J. Virtamo, A queueing analysis of max–min fairness, proportional fairness and balanced fairness, Queueing Syst. 53 (1) (2006) 65–84.
[32] T. Lan, D. Kao, M. Chiang, A. Sabharwal, An Axiomatic Theory of Fairness in Network Resource Allocation, IEEE, 2010.
[33] A. Greenberg, P. Lahiri, D.A. Maltz, P. Patel, S. Sengupta, Towards a next generation data center architecture: scalability and commoditization, in: Proceedings of the ACM Workshop on Programmable Routers for Extensible Services of Tomorrow, ACM, 2008, pp. 57–62.
[34] W.J. Dally, B.P. Towles, Principles and Practices of Interconnection Networks, Elsevier, 2004, pp. 245–250.
[35] H. Kung, T. Blackwell, A. Chapman, Credit-based flow control for atm networks: credit update protocol, adaptive credit allocation and statistical multiplexing, in: ACM SIGCOMM Computer Communication Review, vol. 24, no. 4, ACM, 1994, pp. 101–114.
[36] T. Blackwell, K. Chang, H. Kung, D. Lin, Credit-based flow control for atm networks, in these Proceedings, 1994.
[37] G.F. Pfister, An introduction to the infiniband architecture, in: High Performance Mass Storage and Parallel I/O, vol. 42, 2001, pp. 617–632.
[38] D. Mayhew, V. Krishnan, Pci express and advanced switching: evolutionary path to building next generation interconnects, in: High Performance Interconnects, 2003 Proceedings. 11th Symposium on, IEEE, 2003, pp. 21–29.
[39] W. Jiang, F. Ren, J. Wang, Survey on link layer congestion management of lossless switching fabric, Comput. Standards Interfaces (2017).
[40] H. Barrass, et al. Proposal for priority based flow control, vol. 2, 2008, pp. 1–9.
[41] H. Barrass, et al. Definition for new pause function, 2007.
[42] D.R. Pannell, Network switch with head of line input buffer queue clearing, Jan. 21 2003, uS Patent 6,510,138.
[43] D. Lee, S.J. Golestani, M.J. Karol, Prevention of deadlocks and livelocks in lossless, backpressured packet networks, Feb. 22 2005, uS Patent 6,859,435.
[44] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, M. Yasuda, Less is more: trading a little bandwidth for ultra-low latency in the data center, in: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, USENIX Association, 2012, pp. 19–19.
[45] W. Bai, L. Chen, K. Chen, H. Wu, Enabling ecn in multi-service multi-queue data centers, in: NSDI, 2016, pp. 537–549.
[46] C. Lee, C. Park, K. Jang, S.B. Moon, D. Han, Accurate latency-based congestion feedback for datacenters, in: USENIX Annual Technical Conference, 2015, pp. 403–415.
[47] L.S. Brakmo, S.W. O'Malley, L.L. Peterson, TCP Vegas: New Techniques for Congestion Detection and Avoidance, vol. 24, no. 4, ACM, 1994.
[48] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, S. Shenker, pfabric: Minimal near-optimal datacenter transport, in: ACM SIGCOMM Computer Communication Review, vol. 43, no. 4, ACM, 2013, pp. 435–446.
[49] K. Fall, S. Floyd, Simulation-based comparisons of tahoe, reno and sack tcp, ACM SIGCOMM Comput. Commun. Rev. 26 (3) (1996) 5–21.
[50] P. Newman, Traffic management for atm local area networks, IEEE Commun. Mag. 32 (8) (1994) 44–50.
[51] P. Newman, Backward explicit congestion notification for atm local area networks, in: Global Telecommunications Conference, 1993, Including a Communications Theory Mini-Conference. Technical Program Conference Record, IEEE in Houston. GLOBECOM'93., IEEE, IEEE, 1993, pp. 719–723.
[52] R. Pan, B. Prabhakar, A. Laxmikantha, Qcn: Quantized congestion notification, IEEE802, vol. 1, 2007.
[53] M. Alizadeh, B. Atikoglu, A. Kabbani, A. Lakshmikantha, R. Pan, B. Prabhakar, M. Seaman, Data center transport mechanisms: Congestion control theory and ieee standardization, in: Communication, Control, and Computing, 2008 46th Annual Allerton Conference on, IEEE, 2008, pp. 1270–1277.
[54] A. Kabbani, M. Alizadeh, M. Yasuda, R. Pan, B. Prabhakar, Af-qcn: Approximate fairness with quantized congestion notification for multi-tenanted data centers, in: High Performance Interconnects, HOTI, 2010 IEEE 18th Annual Symposium on, IEEE, 2010, pp. 58–65.
[55] R. Pan, L. Breslau, B. Prabhakar, S. Shenker, Approximate fairness through differential dropping:(summary), ACM SIGCOMM Comput. Commun. Rev. 32 (1) (2002) 72–72.
[56] R. Pan, L. Breslau, B. Prabhakar, S. Shenker, Approximate fairness through differential dropping, ACM SIGCOMM Comput. Commun. Rev. 33 (2) (2003) 23–39.
[57] N. Chrysos, F. Neeser, R. Clauberg, D. Crisan, K.M. Valk, C. Basso, C. Minkenberg, M. Gusat, Unbiased quantized congestion notification for scalable server fabrics, IEEE Micro 36 (6) (2016) 50–58.
[58] M. Gusat, D. Crisan, C. Minkenberg, C. DeCusatis, R3c2: reactive route and rate control for cee, in: High Performance Interconnects, HOTI, 2010 IEEE 18th Annual Symposium on, IEEE, 2010, pp. 50–57.
[59] K.-Y. Siu, H.-Y. Tzeng, Intelligent congestion control for abr service in atm networks, ACM SIGCOMM Comput. Commun. Rev. 24 (5) (1994) 81–106.
[60] K. Ramakrishnan, S. Floyd, A Proposal to Add Explicit Congestion Notification (Ecn) to Ip, Tech. Rep., 1998.
[61] S. Floyd, Tcp and explicit congestion notification, ACM SIGCOMM Comput. Commun. Rev. 24 (5) (1994) 8–23.
[62] T. InfiniBand, Architecture specification, volume 1, release 1.2. 1, 2007.
[63] E.G. Gran, S.-A. Reinemo, O. Lysne, T. Skeie, E. Zahavi, G. Shainer, Exploring the scope of the infiniband congestion control mechanism, in: Parallel &Amp; Distributed Processing Symposium, IPDPS, 2012 IEEE 26th International, IEEE, 2012, pp. 1131–1143.
[64] D. Zats, T. Das, P. Mohan, D. Borthakur, R. Katz, Detail: reducing the flow completion time tail in datacenter networks, ACM SIGCOMM Comput. Commun. Rev. 42 (4) (2012) 139–150.
[65] I.T. Association, et al. Rocev2, 2014.
[66] R. Mittal, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, D. Zats et al, Timely: Rtt-based congestion control for the datacenter, in: ACM SIGCOMM Computer Communication Review, vol. 45 no. 4, ACM, 2015, pp. 537–550.
[67] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, H. Fugal, Fastpass: A centralized zero-queue datacenter network, ACM SIGCOMM Comput. Commun. Rev. 44 (4) (2015) 307–318.
[68] J. Perry, H. Balakrishnan, D. Shah, Flowtune: flowlet control for datacenter networks, in: NSDI, 2017, pp. 421–435.
[69] P.X. Gao, A. Narayan, G. Kumar, R. Agarwal, S. Ratnasamy, S. Shenker, phost: Distributed near-optimal datacenter transport over commodity network fabric, in: Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies, ACM, 2015, p. 1.
[70] C. Wilson, H. Ballani, T. Karagiannis, A. Rowtron, Better never than late: Meeting deadlines in datacenter networks, in: ACM SIGCOMM Computer Communication Review, vol. 41, no. 4, ACM, 2011, pp. 50–61.
[71] L. Jose, L. Yan, M. Alizadeh, G. Varghese, N. McKeown, S. Katti, High speed networks need proactive congestion control, in: Proceedings of the 14th ACM Workshop on Hot Topics in Networks, ACM, 2015, p. 14.
[72] J. Zhang, F. Ren, R. Shu, P. Cheng, Tfc: token flow control in data center networks, in: Proceedings of the Eleventh European Conference on Computer Systems, ACM, 2016, p. 23.
[73] M. Handley, C. Raiciu, A. Agache, A. Voinescu, A.W. Moore, G. Antichi, M. Wójcik, 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, ACM, 2017, pp. 29–42.
[74] I. Cho, K. Jang, D. Han, Proceedings of the Conference of the, ACM Special Interest Group on Data Communication, ACM, 2017, pp. 239–252.
[75] S. Sinha, S. Kandula, D. Katabi, Harnessing tcp's burstiness with flowlet switching, in: Proc. 3rd ACM Workshop on Hot Topics in Networks, Hotnets-III, 2004.
[76] F. Hwang, Control algorithms for rearrangeable clos networks, IEEE Trans. Commun. 31 (8) (1983) 952–954.
[77] N. Jiang, D.U. Becker, G. Michelogiannakis, W.J. Dally, Network congestion avoidance through speculative reservation, in: High Performance Computer Architecture, HPCA, 2012 IEEE 18th International Symposium on, IEEE, 2012, pp. 1–12.
[78] W.J. Dally, Virtual-channel flow control, IEEE Trans. Parallel Distrib. Syst. 3 (2) (1992) 194–205.
[79] A. Greenberg, J.R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D.A. Maltz, P. Patel, S. Sengupta, Vl2: a scalable and flexible data center network, in: ACM SIGCOMM Computer Communication Review, vol. 39, no. 4, ACM, 2009, pp. 51–62.
[80] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, in: ACM SIGCOMM Computer Communication Review, vol. 38, no. 4, ACM, 2008, pp. 63–74.

Shan Huang received his B.S. degree from the National University of Defense Technology (NUDT), Changsha, in 2016. He is a second-grade postgraduate in the College of Computer, NUDT, advised by Prof. Dezun Dong. His research interests include data center networks and high speed interconnect networks.

Dezun Dong received his B.S., M.S., and Ph.D. degrees from the National University of Defense Technology (NUDT), Changsha, in 2002, 2004, and 2010, respectively. He is a professor in the College of Computer, NUDT. His research interests range across high performance computer systems, high speed interconnect networks, wireless networks, and distributed computing algorithms. He has published over 40 peer-reviewed papers in international journals and conferences. Currently, he focuses on performance evaluation of interconnection networks for supercomputers and data centers. He is a member of the ACM, IEEE.

Wei Bai is an Associate Researcher 2 at Microsoft Research Asia. He received his Ph.D. from Department of Computer Science and Engineering, Hong Kong University of Science and Technology in 2017, advised by Prof. Kai Chen. He was also a recipient of Microsoft Research Asia Fellowship (2015). Before that, He received his B.E. in Information Security from Shanghai Jiao Tong University in 2013.

Wei is broadly interested in computer networking with a special focus on data center networking. His research work has been published in many top conferences and journals, such as SIGCOMM, NSDI, CoNext and ToN.

View publication stats