LSCC: Link-Segmented Congestion Control for RDMA in Cross-Datacenter Networks

Minfei Long*, Jiangping Han†‡, Wentao Wang*, Jiayu Yang†, Kaiping Xue†‡
*Department of EEIS, University of Science and Technology of China, Hefei, Anhui 230027, China
†School of Cyber Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China
‡Corresponding author: J. Han (jphan@ustc.edu.cn), K. Xue (kpxue@ustc.edu.cn)

***Abstract*—As multiple datacenters are established in different geographical locations, some applications run on cross-datacenter networks. In order to improve the service quality, service providers establish dedicated links between datacenters to take advantage of the high performance of RDMA. However, existing RDMA congestion control algorithms are designed for intra-datacenter networks. Due to the long distance between datacenters, cross-datacenter networks have higher latency, which leads to long feedback loop that prevents timely adjustments at the traffic source. Meanwhile, excessive congestion signals are generated due to untimely adjustments, which makes existing RDMA congestion control algorithms unable to accurately deal with congestion like in cross-datacenter networks. In addition, the long-haul link connecting datacenters has large bandwidth delay product (BDP), which brings great buffer pressure to switches. In this paper, link-segmented congestion control (LSCC) is proposed to avoid congestion through segmented link control. LSCC builds a segmented feedback loop between egress switches connecting to the long-haul link, which provides timely congestion feedback and greatly reduces the buffer pressure of switches. Evaluations based on DPDK implementation and large-scale simulation show that LSCC can reduce the average flow completion time (FCT) by 30%-65% and 31%-56% in realistic datacenter load and cross-datacenter load, respectively.**

***Index Terms*—RDMA, cross-datacenter, congestion control**

I. Introduction

With the development of cloud computing and storage business, datacenters are more and more important and being build around the world. As datacenter scales up, some enterprises are establishing datacenters in different geographic locations, these distributed datacenters are often interconnected through wide area networks (WANs). However, WANs have higher latency and less bandwidth than the intra-datacenter network, and some network resource (e.g., guaranteed bandwidth) are even leased from internet service providers (ISPs) [1]. Due to the limitations of WANs, cloud operators have tried to establish dedicated links to connect datacenters to improve the quality of network services [2]. With the dedicated links, cross-datacenter traffic can take advantage of remote direct memory access (RDMA) technology, which has been widely used in internal datacenter networks.

Different from the dependence of traditional TCP/IP protocol on CPU resources [3], RDMA is used in datacenters to free up CPU computing resources. RDMA utilizes RDMA-enabled NIC (RNIC) to allow hosts to read and write directly to each other's memory address space without the intervention of the operating system. Nowadays, some enterprises such as Amazon, Microsoft, and Google are trying to enhance their network performance by deploying RDMA technology for transmission across datacenters [4].

However, when RDMA is deployed across datacenters, a series of new challenges are introduced, one of which is congestion control. Currently, the existing RDMA congestion control algorithms (e.g., DCQCN [3], TIMELY [5] and HPCC [6]) are mainly designed for the intra-datacenter network and we find that the performance of these algorithms is severely impaired in cross-datacenter networks. We summarize the causes of performance impairment as two points: 1) The long propagation delay of dedicated links. Usually, the distance between datacenters is tens or hundreds of kilometers, corresponding to a propagation delay of hundreds of microseconds or milliseconds [7]. A large RTT means a higher delay in congestion perceptron. On the one hand, flows cannot receive congestion feedback at the first RTT and greedily seizes bandwidth resource at the line rate. Especially for "small flows" whose size is smaller than bandwidth delay product (BDP), they finish before receiving any congestion feedback. On the other hand, untimely rate adjustment causes excessive congestion signals, which in turn leads to excessive rate reduction of long flows, reducing link utilization. 2) The buffer pressure of datacenter switches. In order to ensure low queuing delay, the switches in datacenters usually have shallow buffers [8], while the dedicated links have a large BDP, which can reach tens of megabytes. Lossless RDMA requires at least one BDP buffer space (headroom) to avoid packet loss, resulting in large buffer pressure on the egress switches connected to long-haul dedicated links.

In this paper, we innovatively propose LSCC, a link-segmented congestion control algorithm, to provide high bandwidth for cross-datacenter transmission. LSCC is deployed in the egress switch of datacenters, which connects other datacenters through long-haul links. Inspired by the advantages of proactive transport [9]–[13], LSCC establishes a feedback loop over the long-haul link, which divides the entire cross-datacenter transmission into three segments: sender - local egress switch, local egress switch - remote egress switch, and remote egress switch - receiver. This feedback loop not only provides timely congestion feedback to the traffic source but also effectively reduces buffer pressure on switches through a novel feedback mechanism.

The contributions of this paper are summarized as follows:

979-8-3503-5012-8/24/$31.00 ©2024 IEEE

- We reveal the limitations of existing RDMA congestion control algorithms in cross-datacenter transmission and analyze the root causes of performance impairments.
- We innovatively design LSCC from a link-segmented perspective according to the characteristics of cross-datacenter networks. Furthermore, we prove that LSCC can reduce the buffer pressure of the switch to a smaller order of magnitude than BDP through theoretical derivation.
- We deploy a prototype of LSCC based on DPDK implementation and evaluate LSCC through both testbed evaluation and large-scale simulation. LSCC can reduce average flow completion time (FCT) by 30%-65% and 31%-56% in realistic datacenter load and cross-datacenter load, respectively.

The rest of this paper is organized as follows. Section II introduces the relevant background knowledge and proposes the challenges of RDMA congestion control in the cross datacenter scenario through pre experiements. Section III illustrates the design rationale and details of LSCC. Section IV evaluates LSCC through DPDK testbed and NS-3 simulation. Section V summarizes related work and Section VI concludes this paper.

II. Background

In this section, we introduce the background of RDMA and cross-datacenter traffic, and discuss the challenges of RDMA congestion control in cross-datacenter networks.

A. RDMA in Datacenter

RDMA has emerged as a promising technology for high-performance data transfer in distributed computing and storage networks. It offers a paradigm shift by allowing direct memory access between servers, bypassing the complexities of the TCP/IP stack, this capability not only reduces latency but also improves overall network performance and resource utilization such as near-zero CPU overhead.

RoCE is the most widely used RDMA protocol in datacenters due to its compatibility with Ethernet standards. To make RoCE work better, priority flow control (PFC) [14] is designed by IEEE 802.1Qbb to avoid packet loss. Specifically, RoCEv1 sets the priority for flows through the priority code point (PCP) domain in the VLAN header, and RoCEv2 extends it to the DSCP domain in the IP header. For each communication entity, there exist logical egress ports and ingress ports, each port can have up to 8 queues, corresponding to different priorities, packets with different priorities are cached in different queues. PFC sets two thresholds X_{on} and X_{off} in the ingress queue. When the length of a receiver's ingress queue exceeds X_{off}, it sends a pause frame with the corresponding priority to the upstream egress queue. When the length of the receiver's ingress queue decreases below X_{on}, it sends a resume frame to resume the transmission. Considering that it takes a round trip time on the current link for a resume frame to take effect, there needs to be a cache space (headroom) for continuously received packets. The size of the headroom cannot be smaller than the BDP of the link, otherwise packet loss will occur. While PFC can provide lossless networking, it causes deadlock, head-of-line (HoL) blocking and PFC pause frame storm [15], [16].

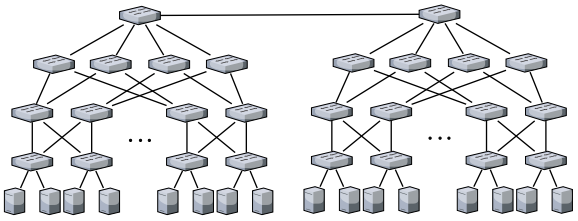


Fig. 1. Cross-datacenter networks topology.

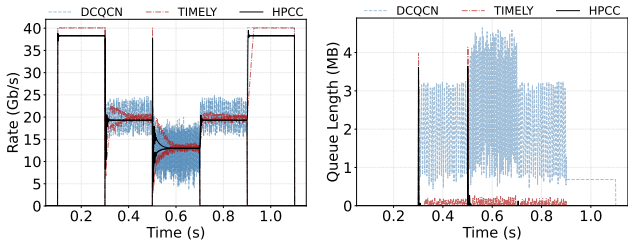
B. Cross-datacenter Traffic

As the scale of datacenter deployment increases, some applications run on different datacenters, for example, Microsoft Azure [7] has a requirement to expand RDMA support to a regional scale since the corresponding compute and storage clusters may be located in different datacenters. Moreover, multiple datacenters enable redundancy and fault tolerance. In the event of hardware failure in one datacenter, the other datacenters can provide services, ensuring business continuity and high availability. Furthermore, synchronization between datacenters ensures data security and recoverability. Furthermore, deploying cloud services across different datacenters offers geographic flexibility, enabling services to be tailored based on the user's geographic location. This helps reduce network latency, enhance service quality, and deliver a better user experience.

For cross-datacenter topologies, there are several methods proposed in [17] to connect datacenters. Regardless of the specific deployment of interconnections between datacenters, cross-datacenter traffic typically follows a consistent path. It originates from a sender, travels through the local egress switch, traverses a long-haul link to the egress switch of the remote datacenter, and finally reaches the intended receiver. Fig. 1 is a cross-datacenter topology based on the FatTree [18] architecture. For long-haul links, the base round-trip time (RTT) can range from hundreds of microseconds to milliseconds. Taking Azure as an example, the base RTT varies from a few microseconds within a datacenter to as large as 2 milliseconds within a region.

C. Challenges

Deploying RDMA across datacenter networks faces many new challenges, one of which is higher latency. In this scenario, it takes a much longer time to recover a lost packet, which is even worse than the TCP/IP stack [7]. Given this degradation, lossless RDMA with PFC mechanism may be the preferred solution. This imposes a requirement for improved RDMA congestion control. Based on our experiment, we find that the high latency of long-haul links severely impairs the performance of various existing RDMA congestion control algorithms. For detailed elucidation, we give evaluations of



(a) Rate (b) Queue Length

Fig. 2. Impaired congestion control algorithms.

representative algorithms, including DCQCN [3], TIMELY [5] and HPCC [6]. We use a simple many-to-one topology with a bandwidth of 40Gbps and a delay of 4us, except for the long-haul link, which has a bandwidth of 200Gbps and a delay of 400us. Three flows start sequentially with an interval of 0.2s. The threshold for ECN-marking is 2MB. To demonstrate the performance of each algorithm, we do not restrict the buffer size and adjust the parameters of each algorithm to more favourable values in this scenario.

Fig. 2 shows the sending rate of senders and the queue length of the congested point. For DCQCN, the large RTT slows down convergence and causes wide ranges of oscillating sending rates and queue length, accompanied by queue starvation that impairs throughput and queue buildup that can easily trigger PFC pause frames. Although TIMELY and HPCC can ensure low queueing delay, the large RTT prevents each flow from sensing network congestion at the start stage, resulting in sharp queue buildup, which is more serious than DCQCN. In dynamic loads, the sharp queue buildup can render TIMELY and HPCC useless, as will be demonstrated in Section IV-C. Based on the evaluation above, we conclude that the key to solving the problem is to avoid the impact of high latency congestion feedback. However, most congestion control algorithms are fundamentally based on end-to-end feedback, when the sender and receiver are far apart, high latency is an inherent attribute.

III. LSCC Design

In this section, we first give the framework of link-segmented congestion control (LSCC) algorithm. Then, we describe the detailed design of LSCC, and provide guidelines for setting parameters in LSCC. Finally, we introduce an improved HPCC algorithm based on our design.

A. Framework of LSCC

LSCC is a link-segmented congestion control algorithm. The framework of LSCC is shown in Fig. 3. A rate calculation module and a rate limiting module are deployed at the port of the egress switch connected to the long-haul link, and they collaborate to form a segmented feedback loop. The rate calculation module assesses the congestion level inside the datacenter based on received feedback signals and quantifies it into a virtual bottleneck rate, which is then fed back to the remote rate limiting module. The rate limiting module sends

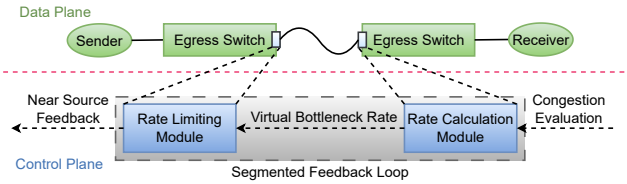
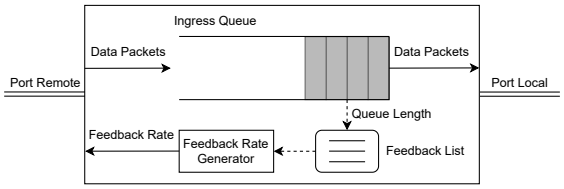


Fig. 3. Framework of LSCC.

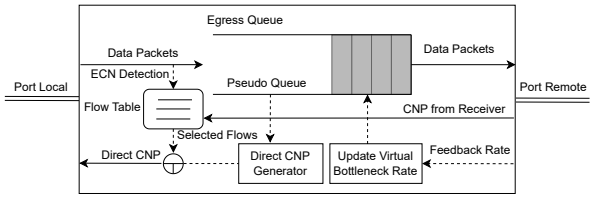
direct feedback signals to sources inside the local datacenter based on the received bottleneck rate. To address the instability caused by high latency, a dynamic feedback list and a pseudo queue are designed in the rate calculation module and the rate limiting module, respectively, while reducing the buffering pressure on the switch. To achieve flow-level fairness, a flow table is maintained in the rate limiting module. The rate calculation module and the rate limiting module are described in detail below.

B. LSCC Algorithm

Fig. 4 shows the overview of the two modules of LSCC. The core idea of LSCC is as follows. The rate calculation module at the remote egress switch periodically calculates and sends suitable feedback rate values to the local egress switch. The rate limiting module at the local switch limits the sending rate of the source by sending direct CNPs based on the length of a pseudo queue. The source adopts the same method in DCQCN [3] to adjust sending rate.



(a) Rate Calculation Module



(b) Rate Limiting Module

Fig. 4. Overview of LSCC design at the egress switch.

Rate calculation module. When congestion occurs in the remote datacenter, the congestion feedback has an inherent high latency attribute. Under dynamic load, the bottleneck link could be at any hop and congestion will spread upstream due to the PFC mechanism, ultimately causing the egress switch to intermittently pause sending. From the perspective of statistics, the egress switch has a real bottleneck. LSCC calculates this bottleneck rate and sends it to the upstream egress switch in the remote datacenter. Fig. 4(a) shows the design of the rate calculation module, which is deployed at the ingress queue

of the port connected to the long-haul link. In the RDMA NIC, a memory management unit (MMU) records all packets that come from each ingress port and are forwarded to the respective egress port. LSCC sets a Q_{ref} as the reference value of the ingress queue, if the queue length exceeds Q_{ref}, a smaller rate value will be sent to remove the excess. In contrast, a larger rate will be sent to fill in the gap. The most intuitive way to calculate the rate is:

$$R(t) = BR(t) - Ra(t), \tag{1}$$

where $Ra(t) = \frac{Q(t)-Q_{ref}}{T}$, the first term is the bottleneck rate, and the second term is used as an adjustment to maintain the queue length. T is the period for rate calculation. T is typically set to 1-2 times the round trip time (RTT) to allow any change in the rate calculation to go into effect before the next update of the feedback rate. However, this setting is not feasible as the RTT is very large. If T is set to be less than RTT, there will be an issue of overreaction. Innovatively, we design a dynamic feedback list (shown in Fig. 5) to record the rates that have been sent. The dynamic feedback list with length N is a first-in-first-out queue, each time a newly calculated feedback rate value goes to the end of the queue, the already existing elements in the queue move one bit to the head of the queue in turn, and the first element in the head of the queue goes out of the queue, which means that the element starts to take effect. As shown in Fig. 5, the feedback rate $R(t-2D)$ starts to take effect while the feedback rate $R(t)$ is calculated. Specifically, let's assume that the RTT (denoted as 2D) of the long-haul link is equal to N*T. Therefore, there are N feedback rates on the long-haul link that have been sent but are not in effect. To avoid overreaction, Eq. (1) is modified as:

$$R(t) = \frac{\int_{t-T}^{t} BR(t)dt}{T} - [\frac{Q(t) - Q_{ref}}{T} - \sum_{j=1}^{N} Ra(t - jT)]. \tag{2}$$

The first term is the statistical average bottleneck rate, which is the sum of the rates of all egress ports within T. The revised second term subtracts feedback that is not in effect on the long-haul link. In this regard, we justify the method by a simple derivation. Transforming modified $Ra(t)$ into another form, we get:

$$Q(t) = Q_{ref} + \sum_{j=0}^{N} Ra(t - jT)T. \tag{3}$$

Further, subtracting $Q(t-T)$ from $Q(t)$, we get:

$$\Delta Q(t) = [Ra(t) - Ra(t - 2D - T)]T. \tag{4}$$

Assuming that the remote egress switch adjusts to its value immediately upon receiving a feedback rate, we can get the import rate of the local egress switch:

$$R_i(t) = R(t - 2D). \tag{5}$$

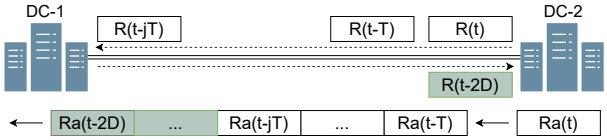


Fig. 5. Feedback list.

Then we can get another form of $\Delta Q(t)$:

$$\begin{aligned}\Delta Q(t) &= \int_{t-T}^{t} [R_i(t) - BR(t)]dt \\ &= \int_{t-2D-T}^{t-2D} R(t)dt - \int_{t-T}^{t} BR(t)dt.\end{aligned} \tag{6}$$

It is worth noting that during the period from $(t-2D-T)$ to $(t-2D)$, the actual feedback rate taking effect is $R(t-2D-T)$, so Eq. (6) is transformed as:

$$\begin{aligned}\Delta Q(t) = &\int_{t-2D-2T}^{t-2D-T} BR(t)dt - Ra(t-2D-T)T \\ &- \int_{t-T}^{t} BR(t)dt.\end{aligned} \tag{7}$$

Combining the Eq. (4) and Eq. (7), we get:

$$\begin{aligned}Ra(t) &= \frac{\int_{t-2D-2T}^{t-2D-T} BR(t)dt}{T} - \frac{\int_{t-T}^{t} BR(t)dt}{T} \\ &= \overline{BR}(t-2D-T) - \overline{BR}(t),\end{aligned} \tag{8}$$

Substituting the Eq. (8) into Eq. (3), we obtain another expression for $Q(t)$:

$$\begin{aligned}Q(t) =& Q_{ref} + \sum_{j=1}^{N+1} \overline{BR}(t-2D-jT)T \\ &- \sum_{j=0}^{N} \overline{BR}(t-jT)T \\ =& Q_{ref} + \int_{t-2D-T}^{t} [BR(t-2D-T) - BR(t)]dt.\end{aligned} \tag{9}$$

The meaning of Eq. (9) is clearly presented through Fig. 6, the red area is queue stacking and the blue area is queue draining. The size of $Q(t)$ is equal to Q_{ref} plus the difference between the queue stacking in the last two adjacent periods. In contrast, the deep-buffer switch solution requires a minimum buffer of two BDPs [19], meaning that the buffering pressure is largely relieved by LSCC.

Some practical issues need to be considered when deploying, one of which is that the remote egress switch does not immediately adopt the feedback rate but gradually adjusts to it, as will be explained in the next rate limiting module. Therefore, a maximum difference between successive feedback rate values should be limited to prevent the remote switch from failing to adjust to the target value in time. We will give the parameter guideline in Section III-C. Alg. 1 is the rate calculation algorithm.

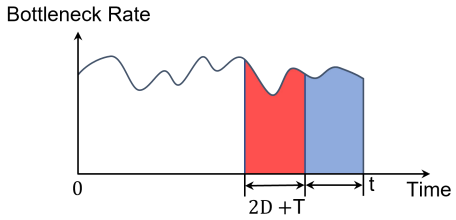


Fig. 6. Relationship between Q and BR.

Algorithm 1: Rate Calculation Module

Input: $Q_{cur}, Ra\_list, Q_{ref}, BR$
Output: R

$Ra\_sent \leftarrow \sum Ra\_list$
$Ra \leftarrow (Q_{cur} - Q_{ref})/T - Ra\_sent$
$R \leftarrow BR - Ra$
if $R - R\_last > \Delta R$ **then**
  $R \leftarrow R\_last + \Delta R$
else if $R\_last - R > \Delta R$ **then**
  $R \leftarrow R\_last - \Delta R$
if $R < R_{min}$ **then**
  $R \leftarrow R_{min}$
else if $R > R_{max}$ **then**
  $R \leftarrow R_{max}$
$R\_last \leftarrow R$
$Ra \leftarrow BR - R$
for $i = (N-1) : 1$ **do**
  $Ra\_list[i] \leftarrow Ra\_list[i-1]$
$Ra\_list[0] \leftarrow Ra$
return R

Rate limiting module. When a local egress switch receives a feedback rate from a remote egress switch, it should immediately adjust its rate to this feedback rate to comply with the schedule of the rate calculation module. Although some advanced switches such as programmable switches support this feature, we still insist on end-to-end feedback because directly modifying the rate is akin to PFC. A slightly larger rate difference can easily fill the shallow buffer. Our specific approach is to introduce a pseudo queue at the egress queue of the port connected to the long-haul link, as shown in Fig. 4(b). This pseudo queue takes the actual rate of incoming packets as the import rate and the received feedback rate as the export rate, defined as:

$$\Delta Q_e^*(t) = \int (R_{in} - R)dt. \tag{10}$$

We compare it with the actual egress queue:

$$\Delta Q_e(t) = \int (R_{in} - R_{out})dt. \tag{11}$$

R is unrestricted and can take any value below the bandwidth capacity R_{out} is not restricted and can be any value below the bandwidth capacity. R is always smaller than R_{out} (see line 10-11 in Alg. 1). Therefore, the length of the pseudo queue is always larger than that of the actual egress queue, i.e., a virtual bottleneck.

TABLE I
FLOW TABLE

$sip/dip/sport/dport$	CNP	$TxBytes$
96bit	1 bit	32bit

Similarly, we set a threshold Q_{ref}^* for this pseudo-queue. If the length of this pseudo-queue exceeds Q_{ref}^*, the local egress switch will send direct CNP to the source. In contrast, the source will increase the rate based on additive increase multiplicative decrease (AIMD). Each time the local egress switch receives a new feedback rate, the length of the pseudo-queue will fluctuate because the virtual bottleneck rate changes. However, if the pseudo-queue stabilizes before the next feedback rate arrives, i.e., the source rate has converged, it is equivalent to the local egress switch immediately adopting the feedback rate, as assumed in the rate calculating module. Therefore, the local egress switch has a duration of T to send feedback to the source, allowing the sending rate to gradually converge to the received feedback rate.

Since LSCC is deployed in the middle hop of the entire link, making it unable to achieve flow-level congestion control, a flow table is required to select specific sources when sending direct CNP. The flow table records each flow within the local datacenter and determines whether to send direct CNP to it or not so that innocent long flows can be filtered out. As shown in Table I, each entry in this flow table records information of a single internal flow, including 1) $[sip/dip/sport/dport]$, 2) $TxBytes$ and 3) CNP. The $[sip/dip/sport/dport]$ is the quaternions for RDMA flows, the $TxBytes$ records the size of bytes transferred, the CNP is used to determine whether to send direct CNP to the corresponding source. When the egress switch receives a CNP from the remote datacenter, the CNP for the corresponding flow is set to $true$. In addition, the CNP is used to reflect local congestion, while forwarding packets from local sources, the egress switch detects the ECN domain, and if marked, sets the CNP of the corresponding flow to $true$. To ensure stability, the threshold for identifying long flows should be set to be greater than BDP. This ensures that any flow is constrained by direct CNP during the start stage, avoiding queue proliferation. Alg. 2 shows the algorithm of rate limiting module.

With the introduction of near-source feedback, there is dual feedback in the network: the normal CNP from the receiver and the direct CNP from the local egress switch. We have proved that feedback in high-latency networks leads to instability, however, it is still necessary to ensure fairness. For example, the flow table requires normal CNP to filter innocent flows; when there are multiple bottlenecks in networks, normal CNP is required to ensure fair rate competition. It becomes a matter of trade-offs. Our experience suggests that direct CNP feedback needs to be fine-grained while normal CNP feedback needs to be coarse-grained because stability is more

important. LSCC sets the generation interval of normal CNP larger than that of direct CNP, which is a tenfold relationship in our setting. In a cross-datacenter network, most flows are "small flows" relative to the large BDP, they last for only a few RTTs or even less than one RTT. Therefore, the specific setting of normal CNP does not make much difference.

Algorithm 2: Rate Limiting Module

```
Procedure SendDirectCNP(P)
    if Q*_cur < Q*_ref then
        return;
    for each flow in FTb do
        if FTb[flow].CNP == True or
          FTb[flow].TxBytes < threshold then
            SendCNP(flow);
    ResetTable(T);
Procedure ReceivePacket(P)
    flow ← Parse(P) ;
    if flow is external then
        return;
    if P is CNP then
        FTb[flow].CNP ← True;
    else if P is DataPacket then
        FTb[flow].TxBytes+ = P.size;
        if P.ECN == 11 then
            FTb[flow].CNP ← True;
            P.ECN = 10;
```

Deployment. For the feedback rate message generated in the rate calculation module, which contains a suggested rate value, we define a minimum unit ΔF to save overhead. We use Internet control message protocol (ICMP) for this feedback rate message and prioritize it to minimize queuing delay. Considering that the rate calculation module maintains a Q_{ref} to improve bandwidth utilization, congestion marking needs to be disabled at this hop. To make LSCC easier to deploy in a real network, DCQCN is utilized in the rate limiting module because it has been widely commercially available and direct CNP does not require changes at the source.

C. Parameter Guideline

In this section, we give the setting guidelines of major parameters. Variables and parameters used in LSCC are listed in Table II.

$\mathbf{Q_{ref}}$. The value of Q_{ref} directly affects the utilization of the long-haul link. According to Eq. (9), the minimum value of queue is $Q_{ref} - \Delta\overline{BR}_{max} * (2D + T)$, where $\Delta\overline{BR}$ is the difference between the average bottleneck rates of two adjacent periods, as shown in Fig. 6. To ensure the queue not to be empty in the presence of bottleneck, it needs to satisfy $Q_{ref} > \Delta\overline{BR}_{max} * (2D + T)$, the $\Delta\overline{BR}_{max}$ is determined by the dynamic load of the link, a larger value ensures higher bandwidth utilization but increases the non-essential queuing time. In our configuration, it is set to 0.1 of the link bandwidth.

TABLE II
VARIABLES AND PARAMETERS

Symbol	Definition
R	Feedback rate
ΔR	Maximum difference in adjacent feedback rate
R_{max}	Maximum feedback rate
R_{min}	Minimum feedback rate
Q_{cur}	Current queue length
Q_{ref}	Reference of queue length
T	Cycle of rate calculation module
BR	Bottleneck rate
Q^*_{cur}	Current pseudo-queue length
Q^*_{ref}	Reference of pseudo-queue length

T. The value of T reflects the feedback granularity of the feedback loop. According to Eq. (9), A smaller T promotes stability of the queue, but if T is too small, the source may struggle to converge to the target rate within T. Based on experimental experience, we set T to 200us, which is several or tens of times the RTT within the datacenter. This allows the source to iterate so many times.

$\mathbf{\Delta R}$. The ΔR setting needs to take into account both the value of T and the convergence ability of DCQCN. To satisfy the rate convergence of sources within T to the target value, we have: $\Delta R < M * \Delta r$, Δr is the rate change capability of a single source within T, M is the number of coexisting flows. For DCQCN, rate-increase is slower than rate-decrease (AIMD), so Δr can be calculated as $\Delta r = \frac{T}{T_i} * R_{ai}$. ΔR should not be set too small, otherwise it will affect the feedback capability. According to Eq. (2), we consider the most extreme case, Q_{cur} is always smaller or larger than Q_{ref} in 2D, then the size of the adjustment term for feedback in 2D is: $(N + 1) * \Delta R$. Let $(N + 1) * \Delta R * T > \Delta Q$, then we have: $\Delta R > \frac{\Delta Q}{T(2D/T+1)}$, a greater ΔQ means a greater feedback strength. Thus, the constrain on ΔR is:

$$\frac{\Delta Q}{T(N+1)} < \Delta R < \frac{M * T * R_{ai}}{T_i}. \tag{12}$$

D. HPCC*

We combine the feedback loops with HPCC, (we named it HPCC*), and find that the performance is greatly improved. Specifically, the rate calculation module is consistent with LSCC, the rate limiting module replaces direct CNP with direct INT, and the flow table is not required. When a packet is forwarded at the local egress switch, the rate limiting module copies the INT information it carries into the direct INT and appends the length of the pseudo-queue ($qlen$) and the current received feedback rate (B) as the INT information for this hop to the direct INT. The INT in HPCC only retains 4 bits to record the link bandwidth, which is because the bandwidth of the datacenter link is fixed, e.g., 40Gbps, 100Gbps, etc. Considering that the received feedback rate is

dynamic, HPCC* adds an additional 24 bits to record B. In addition, the default maximum hop of HPCC is 5, HPCC* does not require modification of this setting because the rate limiting module copies the INT information of the internal link and then removes it, i.e., sets $nHop$ to 0 to avoid duplicate INT information carried by ACK. For the rate adjustment at traffic source, similar to LSCC, HPCC* controls dual loops. For the near-source loop, HPCC* independently computes congestion window W^* and only utilizes it when it is less than W calculated by the normal loop. Given the condition that the direct INT does not contain a sequence number, snd_una is introduced to achieve this. Specifically, HPCC* records both snd_nxt and snd_una like HPCC, when a subsequent arriving ACK acknowledges the sequence number (snd_una) as snd_nxt, an RTT has elapsed and snd_nxt is updated to a new sequence number. HPCC* records the snd_una at the update time, which is the previous snd_nxt. Therefore, if the sequence number acknowledged by ACK is $snd\_una + \frac{snd\_nxt - snd\_una}{N}$, the interval time is $\frac{RTT}{N}$. Compared to LSCC, HPCC* requires modifications to the source, but it can precisely control congestion like HPCC.

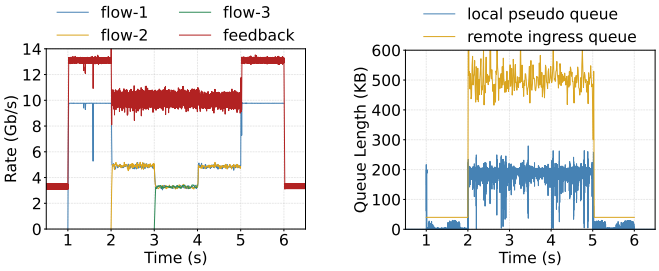
IV. Evaluation

In this section, we use DPDK [20] experiments with our prototype and NS-3 simulator [21] to evaluate the performance of LSCC and compare it to existing RDMA congestion control solutions. The DPDK evaluation demonstrates the feasibility and value of LSCC, while the simulations allow us to evaluate LSCC under conditions that are hard to replicate in realistic cross-datacenter networks. The simulations include micro-benchmarks and large-scale realistic load.

A. DPDK Evaluation

DPDK enhances the performance of data plane applications by bypassing the network stack of the operating system kernel and taking packet processing packets directly in user space. The topology comprises four servers and two switches. Three servers are connected to the same switch as senders, while one server is connected to another switch as a receiver, the two switches are connected by a virtual long-distance link. Similar to the Linux TC tool, we define a delay queue on the switch to simulate sending delayed congestion feedback in the long-haul link. Each node in the testbed is an ASUS host with 2.5GHz 64-bit 16-Core i5-13400 processor, equipped with an Intel E810 25GbE 4-port NIC supporting DPDK and RoCEv2. The bandwidth of each server is 10Gbps.

The delay of the long-haul link is set to 500us, corresponding to a distance of 100km [19]. Q_{ref} is set to 500KB that is approximately 16% of the BDP for the long-haul link, Q^*_{ref} is set to 200KB. The parameter settings for DCQCN are referenced from paper [3]. According to Eq. (12), ΔR is set to 50 Mbps. In the test scenario, the first sender generates a long-lived traffic load that is equal to the link bandwidth of 10Gbps, then the second and third senders start and stop their transmissions, spaced by one second. Fig. 7(b) shows the length of local pseudo-queue and remote ingress queue. Fig. 7(a) shows the rate of flows and the feedback rate received by the local egress switch. In the absence of flows, due to the difference between the empty queue and Q_{ref}, the remote egress switch constantly sends periodically changing feedback rate to the local egress switch to limit the line rate within the first RTT of a new flow, then it increases by 10Gbps as the first flow generates. Since the bottleneck rate is just reached and there is no queue buildup, the feedback rate is greater than the actual bottleneck rate to bring the queue up to Q_{ref}. With the generation of the second flow and third flow, the length of both the local pseudo-queue and remote ingress queue fluctuates around their reference values, and the rates of flows converge to the same fair value. From the DPDK evaluation, we conclude that LSCC can greatly reduce the impairment of high-delayed congestion feedback.



(a) Rate (b) Queue Length

Fig. 7. Evaluation on a DPDK testbed.

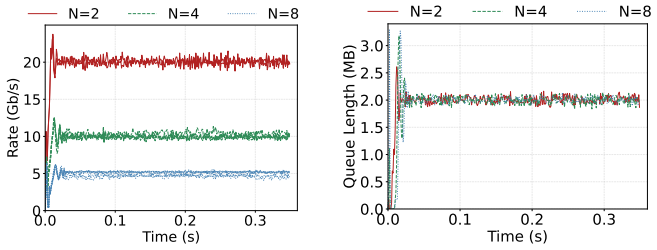
B. Micro-Benchmarks

For micro-benchmarks, we utilize specific traffic patterns to evaluate particular aspects of LSCC's performance with NS-3 simulator.

Setting. We use the topology shown in Fig. 1. Two datacenters are connected by the egress switch to a long-haul link with a bandwidth of 200Gbps and a delay of 1ms, corresponding to a distance of 200km. Each ToR switch is connected to two servers by a 40Gbps link, the bandwidth of all other links is 100Gbps, the latency of all links inside the datacenter is 1us. According to guideline in Section III-C, both Q_{ref} and Q^*_{ref} are set to 2MB, T is set to 200us, ΔR is set to 1Gbps. The parameters of DCQCN are set to favourable values according to Mellanox's recommendations [22] for better performance. X_{on} and X_{off} for PFC are set to 100KB and 300KB.

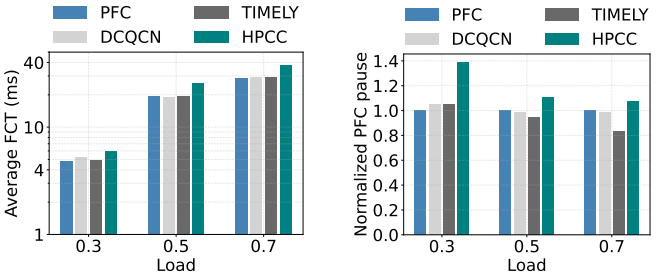
Stability. N source nodes at the same datacenter generate a long-lived flow simultaneously to a destination node at another datacenter, causing persistent congestion on the bottleneck link. Fig. 8(a) shows the rates of each source, Fig. 8(b) shows the ingress queue length at the remote egress switch. Within 50ms (i.e. 25 RTTs) after the start of N flows, the rates of all flows converge to the fair rate. Meanwhile, the ingress queue length in the bottleneck stabilizes around Q_{ref}. These results mean that each source reacts to congestion quickly due to the virtual bottleneck close to it.

Convergence. To show that LSCC makes flows converge quickly to their fair rate, N is set to 5 and each flow is



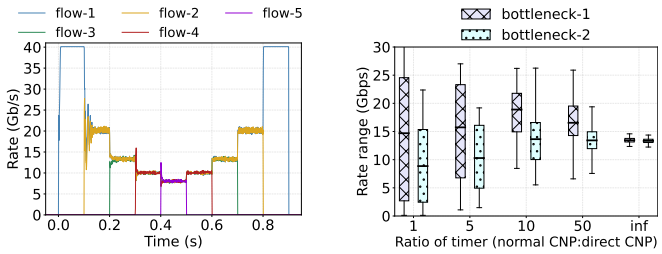
(a) Fair Rate (b) Queue Stability

Fig. 8. Stability.



(a) Average FCT (b) PFC Activation

Fig. 10. Ineffective congestion control algorithms.



(a) Convergence (b) Fairness

Fig. 9. Convergence and fairness.

generated and terminated at an interval of 0.5s. Fig. 9(a) shows the result, which is consistent with the DPDK evaluation. This result is similar to the performance of DCQCN on short-haul links, that is, LSCC achieves "synchronization" of congestion from the far-source to the near-source.

Multi-bottleneck (Fairness). To prove that LSCC can achieve fairness, we set two destination nodes, with two and three source nodes corresponding to them. Each source node generates a long-lived flow, and the generation interval of normal CNP is set to various multiples of direct CNP generation interval. Fig. 9(b) shows the result of rate variation for competitive flows at the two bottlenecks, respectively. The two bottlenecks should have fair rates of 20Gbps and 13.3Gbps. When the multiple is 1, a large number of lagging normal CNP reduce bandwidth utilization. When the multiple is infinity, i.e., there is no normal CNP, stability is achieved but all flows share the minimum bottleneck rate. When the multiple is 10, fairness and stability are well balanced, we use this setting in the subsequent evaluation.

C. Large-Scale Realistic Load

We use NS-3 simulator to evaluate LSCC and HPCC*, the comparative solutions are DCQCN, TIMELY, HPCC, SWING [19], and PFC, where PFC denotes that only using PFC in the network without any congestion control algorithms at end hosts. SWING is an improved PFC mechanism designed for long-range lossless RDMA, which deploys a PFC relay device at each end of the long-haul link close to the egress switch. The topology used is the same as in Section IV-B.

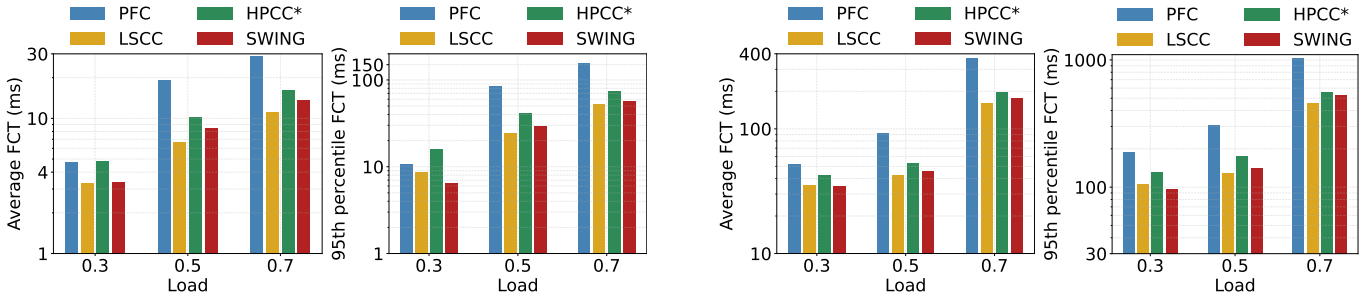
Parameter settings. For LSCC, Q_{ref} and Q^*_{ref} are set to 5MB, other parameters are the same as micro-benchmarks. For PFC, X_{on} and X_{off} of the ports that are connected to the long-haul link are set to 5MB, which is equal to Q_{ref}, the headroom is set to one-BDP of the link it is connected to. For TIMELY, the parameters are recommended values in [5]. For HPCC, we set η = 95%, W_{AI} = 500MBps. For SWING, we adjust the parameters to match our topology according to [19].

Traffic load. We use the publicly available datacenter traffic distributions WebSearch [23]. The WebSearch workload is characterized by small requests and large responses. Among them are mainly long flows, 95% of the flows are larger than 1MB [12]. In addition, we use an Alibaba cross-datacenter over WAN traffic distribution [24], characterized by mostly long flows, 40% of the flows are larger than 50MB, we can use it to evaluate the transport of large traffic between datacenters such as file backup and storage service [7]. In our evaluation, 16 servers in the same datacenter send traffic to 4 servers at 4 different racks in another datacenter, and the flow arrival times obey the Poisson distribution. The average load of the long-haul link is set to 30%, 50% and 70%.

Ineffective congestion control algorithms. As shown in Fig. 10(a). Under different loads, the average FCT of DCQCN, TIMELY and HPCC is close to PFC, or even worse. The long-delayed feedback signal prevents the source from sensing the congestion in time, so the PFC signal is triggered before the arrival of the feedback signal, and the rate reduction performed by the source after receiving the feedback signal is unnecessary, which instead reduces the link utilization and increases FCT. Fig. 10(b) shows the average pause time of the nodes, confirming that the long-delayed congestion feedback signal renders DCQCN, TIMELY and HPCC ineffective.

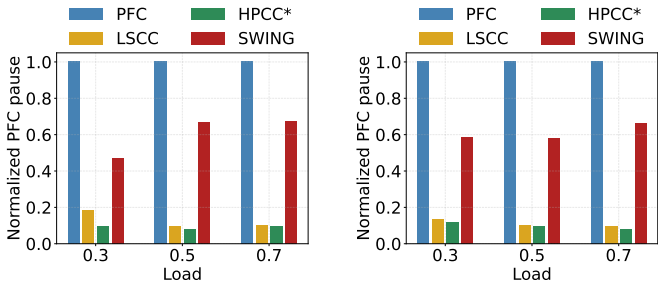
Less FCT. As shown in Fig. 11, compared to PFC, LSCC reduces the average FCT and 95th percentile FCT by 30%-65% and 22%-79% in different WebSearch load, by 31%-56% and 43%-58% in different Ali-WAN load, respectively. The FCT of HPCC* is slightly larger than that of LSCC due to the overhead of INT, but HPCC* still significantly improves performance. SWING also has good performance because it takes full advantage of the bandwidth of the long-haul link.

Less PFC pause. Fig. 12 is the normalized PFC pause time. LSCC and HPCC* can significantly reduce the triggering of PFC pause frames. Firstly, the near-source feedback can avoid congestion in the local datacenter timely, which can reduce



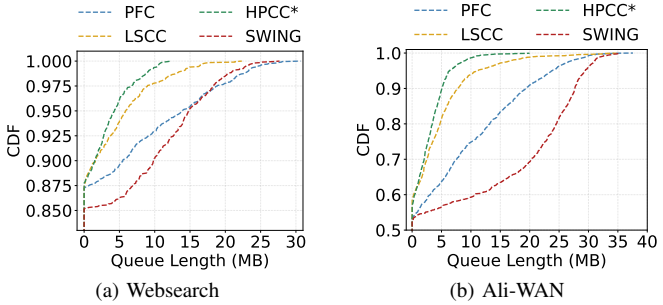
(a) Websearch (b) Ali-WAN

Fig. 11. Average FCT and 95th percentile FCT of PFC, LSCC, HPCC* and SWING in different loads and traffic.



(a) Websearch (b) Ali-WAN

Fig. 12. Normalized PFC pause time.



(a) Websearch (b) Ali-WAN

Fig. 13. CDF of buffer occupancy (50% average load).

PFC trigger. Secondly, the feedback loop ensures zero PFC triggering on the long-haul link. Thirdly, the feedback loop provides a reliable buffer to absorb congested packets in the remote datacenter, which can resume PFC pause. For SWING, it trades the frequent triggering of PFC pause frames on the long-haul link for high bandwidth utilization.

Less buffer occupancy. Fig. 13 shows the occupancy of the egress switch buffer in 50% load. LSCC and HPCC* occupy less buffer than PFC and SWING, about 95% of them are below 10MB. According to the theoretical analysis in Section III-B, the buffer occupancy fluctuates around Q_{ref} (5MB), and the experimental results match it well. SWING has a high buffer occupancy because its buffer requirement is positively proportional to the BDP (which is shown in [19]). Under Ali-WAN load, the comparison is more obvious, which means that our algorithms are suitable for cross-datacenter scenarios.

V. Related Work

Congestion control in datacenter. The congestion control algorithms in datacenter can be classified into two categories: reactive transport and proactive transport. Among reactive transport algorithms, DCTCP [23] is one of the most representative works. It adjusts the congestion window according to the fraction of ECN-marked ACK packets. DCQCN [3] is one of the most widely used RDMA congestion control algorithms, similar to DCTCP, it utilizes ECN to access network congestion level. Some works like TIMELY [5] and Swift [25] are based on network delay, they adjust rates using RTT gradients to achieve low queueing delay. HPCC [6] and PowerTCP [26] leverage INT to obtain accurate link information, achieving fine-grained congestion control. Proactive transport algorithms are usually based on receiver-driven or switch-assistance. The representative works consist of NDP [13], Expresspass [11], pHost [27] and Homa [12]. The basic idea of proactive transport is to send tokens, credit packets, or other personalized packets containing control information from the receiver or switch to the source, achieving precise congestion control.

Cross-datacenter transport over WAN. Different from datacenter networks, WANs have much higher latency of tens of milliseconds and lower bandwidth capacity. Moreover, WAN switches have deeper buffers and may be leased and not subject to configuration changes by operators. Therefore, inter-datacenter flows can have an impact on intra-datacenter flows. Gemini [1] integrates both ECN and delay signals to achieve low latency for intra-datacenter flows and high bandwidth for inter-datacenter flows. IDCC [28] utilizes INT and RTT to measure queueing delay inside datacenter and in the WAN respectively. Another work Annulus [29] utilizes two control loops to handle bottlenecks near the traffic source.

Improvement of RDMA. To enhance the performance of RDMA, some works try to optimize or replace the PFC mechanism to avoid relevant problems. TCD [30] introduces a new congestion detection mechanism for lossless RDMA. It can detect congestion ports accurately and identify flows that truly contribute to congestion. P-PFC [31] predicts PFC triggers in the future to reduce tail latency. IRN [32] introduces a selective retransmission mechanism for lossy RDMA. SWING [19] provides lossless RDMA for cross-datacenter networks by modifying the PFC mechanism on egress switches.

VI. Conclusion

In this paper, we designed LSCC, a link-segmentation congestion control algorithm for cross-datacenter networks. It deploys a rate calculation module and a rate limiting module at the egress switch of the datacenter, and the two modules collaborate to establish a feedback loop on the long-haul link. On the one hand, the feedback loop synchronizes the bottlenecks in the remote datacenter to the local egress switch, which feeds timely congestion signals to the traffic source. On the other hand, the feedback loop provides a reliable buffer for the remote datacenter, which interrupts the spreading of PFC signals and ensures the bandwidth utilization of the long-haul link. LSCC reveals the problem of repetitive feedback in high latency networks and devises a novel feedback mechanism for this purpose. Through DPDK small-scale testbed tests and NS-3 large-scale realistic load evaluation, we showed that LSCC can be deployed in real-world networks and achieve better performance than the state-of-the-art RDMA congestion control algorithms.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grant No. 62302472, and the Fundamental Research Funds for the Central Universities under grant No. WK2100000039, Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093.

References

[1] G. Zeng, W. Bai, G. Chen, K. Chen *et al.*, "Congestion control for cross-datacenter networks," *IEEE/ACM Transactions on Networking*, vol. 30, no. 5, pp. 2074–2089, 2022.

[2] M. Filer, J. Gaudette, Y. Yin *et al.*, "Low-margin optical networking at cloud scale," *Journal of Optical Communications and Networking*, vol. 11, no. 10, pp. 94–108, 2019.

[3] Y. Zhu, H. Eran, D. Firestone *et al.*, "Congestion control for large-scale RDMA deployments," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 523–536, 2015.

[4] I. Ivanov, "Interconnection: The next-generation data center business model," Accessed: Feb., 2024. [Online]. Available: https://www.datacenterknowledge.com/industry-perspectives/interconnection-next-generation-data-center-business-model

[5] R. Mittal, V. T. Lam, N. Dukkipati *et al.*, "TIMELY: RTT-based congestion control for the datacenter," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 537–550, 2015.

[6] Y. Li, R. Miao, H. H. Liu *et al.*, "HPCC: High precision congestion control," in *Proceedings of the 2019 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2019, pp. 44–58.

[7] W. Bai, S. S. Abdeen, A. Agrawal *et al.*, "Empowering azure storage with RDMA," in *Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2023, pp. 49–67.

[8] W. Bai, S. Hu, K. Chen *et al.*, "One more config is enough: Saving (DC) TCP for high-speed extremely shallow-buffered datacenters," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 489–502, 2020.

[9] P. Taheri, D. Menikkumbura, E. Vanini, S. Fahmy, P. Eugster, and T. Edsall, "RoCC: robust congestion control for RDMA," in *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2020, pp. 17–30.

[10] X. Zhong, J. Zhang, Y. Zhang, Z. Guan, and Z. Wan, "PACC: Proactive and accurate congestion feedback for RDMA congestion control," in *Proceedings of the 2022 IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2022.

[11] I. Cho, K. Jang, and D. Han, "Credit-scheduled delay-bounded congestion control for datacenters," in *Proceedings of the 2017 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2017, pp. 239–252.

[12] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout, "Homa: A receiver-driven low-latency transport protocol using network priorities," in *Proceedings of the 2018 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2018, pp. 221–235.

[13] M. Handley, C. Raiciu, A. Agache *et al.*, "Re-architecting datacenter networks and stacks for low latency and high performance," in *Proceedings of the 2017 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2017, pp. 29–42.

[14] "802.1qbb - priority-based flow control," Accessed: Feb., 2024. [Online]. Available: http://www.ieee802.org/1/pages/802.1bb.html

[15] C. Guo, H. Wu, Z. Deng *et al.*, "RDMA over commodity ethernet at scale," in *Proceedings of the 2016 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2016, pp. 202–215.

[16] K. Qian, W. Cheng, T. Zhang, and F. Ren, "Gentle flow control: avoiding deadlock in lossless networks," in *Proceedings of the 2019 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2019, pp. 75–89.

[17] A. Singh, J. Ong, A. Agarwal *et al.*, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," *ACM SIGCOMM computer communication review*, vol. 45, no. 4, pp. 183–197, 2015.

[18] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM computer communication review*, vol. 38, no. 4, pp. 63–74, 2008.

[19] Y. Chen, C. Tian, J. Dong *et al.*, "Swing: Providing long-range lossless RDMA via PFC-relay," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 1, pp. 63–75, 2022.

[20] "Intel dpdk," Accessed: Feb., 2024. [Online]. Available: https://www.dpdk.org/

[21] "NS-3 simulator," Accessed: Feb., 2024. [Online]. Available: https://www.nsnam.org/

[22] "Mellanox, "DCQCN parameters"," Accessed: Feb., 2024. [Online]. Available: https://enterprise-support.nvidia.com/s/article/dcqcn-parameters

[23] M. Alizadeh, A. Greenberg, D. A. Maltz *et al.*, "Data center tcp (dctcp)," in *Proceedings of the 2010 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2010, pp. 63–74.

[24] G. Zeng, J. Qiu, Y. Yuan, H. Liu, and K. Chen, "FlashPass: Proactive congestion control for shallow-buffered WAN," in *Proceedings of the 29th IEEE International Conference on Network Protocols (ICNP)*. IEEE, 2021, pp. 1–12.

[25] G. Kumar, N. Dukkipati, K. Jang *et al.*, "Swift: Delay is simple and effective for congestion control in the datacenter," in *Proceedings of the 2020 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2020, pp. 514–528.

[26] V. Addanki, O. Michel, and S. Schmid, "PowerTCP: Pushing the performance limits of datacenter networks," in *Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2022, pp. 51–70.

[27] P. X. Gao, A. Narayan, G. Kumar, R. Agarwal, S. Ratnasamy, and 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 (CoNEXT)*, 2015, pp. 1–12.

[28] Y. Geng, H. Zhang, X. Shi *et al.*, "Delay based congestion control for cross-datacenter networks," in *Proceedings of the 31st IEEE/ACM International Symposium on Quality of Service (IWQoS)*, 2023, pp. 1–4.

[29] A. Saeed, V. Gupta, P. Goyal *et al.*, "Annulus: A dual congestion control loop for datacenter and wan traffic aggregates," in *Proceedings of the 2020 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2020, pp. 735–749.

[30] Y. Zhang, Y. Liu, Q. Meng, and F. Ren, "Congestion detection in lossless networks," in *Proceedings of the 2021 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2021, pp. 370–383.

[31] C. Tian, B. Li, L. Qin *et al.*, "P-PFC: Reducing tail latency with predictive PFC in lossless data center networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1447–1459, 2020.

[32] R. Mittal, A. Shpiner, A. Panda *et al.*, "Revisiting network support for RDMA," in *Proceedings of the 2018 ACM Special Interest Group on Data Communication (SIGCOMM)*, 2018, pp. 313–326.