Minimizing Buffer Utilization for Lossless Inter-DC Links

Chengyuan Huang, *Member, IEEE*, Feiyang Xue, *Member, IEEE*, Peiwen Yu, Xiaoliang Wang, Yanqing Chen, Tao Wu, Lei Han, Zifa Han, Bingquan Wang, Xiangyu Gong, Chen Tian, *Senior Member, IEEE*, Wanchun Dou, Guihai Chen, *Fellow, IEEE*, and Hao Yin

***Abstract*— RDMA over Converged Ethernet (RoCEv2) has been widely deployed to data centers (DCs) for its better compatibility with Ethernet/IP than Infiniband (IB). As cross-DC applications emerge, they also demand high throughput, low latency, and lossless network for cross-DC data transmission. However, RoCEv2's underlying lossless mechanism Priority-based Flow Control (PFC) cannot fit into the long-haul transmission scenario and degrades the performance of RoCEv2. PFC is myopic and only considers queue length to pause upstream senders, which leads to large queueing delay. This paper proposes Bifrost, a downstream-driven lossless flow control that supports long distance cross-DC data transmission. Bifrost uses *virtual incoming* packets, which indicates the upper bound of in-flight packets, together with buffered packets to control the flow rate. It minimizes the buffer space requirement to one-hop bandwidth delay product (BDP) and achieves low one-way latency. Moreover, we extend Bifrost and propose BifrostX, to accommodate the multi-priority queue of the current switch implementation. BifrostX enables flow control for each queue separately while maintaining low buffer reservation, no throughput loss, and no packet loss. Real-world experiments are conducted with prototype switches and 80 kilometers cables. Evaluations demonstrate that compared to PFC, Bifrost reduces average/tail flow completion time (FCT) of inter-DC flows by up to 22.5%/42.0%, respectively. Bifrost is compatible with existing infrastructure and can support distance of thousands of kilometers.**

Manuscript received 9 November 2023; revised 29 May 2024; accepted 5 August 2024; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor C. Peng. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB2702800; and in part by the National Natural Science Foundation of China under Grant 62325205, Grant 62072228, and Grant 62172204. The preliminary version of this paper was presented at IEEE ICNP 2023, Reykjavik, Iceland. *(Corresponding authors: Xiaoliang Wang; Chen Tian.)*

Chengyuan Huang, Feiyang Xue, Peiwen Yu, Xiaoliang Wang, Yanqing Chen, Chen Tian, Wanchun Dou, and Guihai Chen are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (e-mail: ryanhuang1014@gmail.com; xue.fyang@foxmail.com; pavin0702@foxmail.com; waxili@nju.edu.cn; dz1933003@smail.nju.edu.cn; tianchen@nju.edu.cn; douwc@nju.edu.cn; gchen@nju.edu.cn).

Tao Wu, Zifa Han, Bingquan Wang, and Xiangyu Gong are with Huawei Technologies Company Ltd., Shenzhen 518063, China (e-mail: wutao26@huawei.com; hanzifa@huawei.com; wangbingquan4@huawei.com; gongxiangyu@huawei.com).

Lei Han is with the School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China (e-mail: hanleinjupt@163.com).

Hao Yin is with the Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084, China (e-mail: h-yin@mail.tsinghua.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2024.3443600, provided by the authors.

Digital Object Identifier 10.1109/TNET.2024.3443600

***Index Terms*— Data center, RoCEv2, flow control, cross-DC data transmission.**

I. Introduction

In recent years, the scale of data centers (DCs) infrastructure has expanded massively to support the demand for scientific research, big data processing, and artificial intelligence. Applications that rely on cross-DC networks are emerging to solve new problems. Large cloud service providers (CSPs) deploy multiple small DCs across a region to serve the densely populated area due to the limited resources such as land, energy, and connectivity [1], [2], [3], [4], [5]. Besides, many data-intensive applications, *e.g.*, data analytics [6], [7], [8], machine learning [9], graph processing [10], [11] and super computing [12], [13], [14], [15], [16] involve large sets of data spread across DCs. These cross-DC applications also have to consider data privacy regulations which may prevent data movement across regions.

Remote Direct Memory Access (RDMA) has been widely used in DCs that achieves high performance, which benefits from kernel bypass technique and the support of an underlying lossless network [17], [18], [19], [20]. Infiniband (IB) [21] and RDMA over Converged Ethernet (RoCEv2) [22] are the state-of-the-art RDMA protocols. IB is designed as an independent dedicated protocol and has been deployed in high performance computing (HPC) [13]. However, IB is expensive and hard to deploy to heterogeneous network systems. On the other hand, RoCEv2 has a similar performance as IB but with the merits of better compatibility with Ethernet/IP protocols and better portability. As a result, it is a trend for RoCEv2 to become a widely used RDMA protocol [23], [24], [25], [26].

As cross-DC applications emerge, they also demand a high throughput, low latency, and lossless network that supports cross-DC data transmission. It is a natural design choice to apply RoCEv2 to the cross-DC scenario to extend its high performance while benefiting from its economy, compatibility and portability with the current intra-DC applications, *e.g.*, Microsoft Azure [27] deploys RoCEv2 across DCs to improve the performance of its storage traffic.

RoCEv2 cannot be migrated to cross-DC efficiently because the underlying flow control cannot fit into the long distance transmission. The long-haul high bandwidth link introduces a large round-trip-time (RTT) and bandwidth delay product (BDP), which delay the network signals and cause large queueing delay. Priority-based Flow Control (PFC) is used in RoCEv2 which checks the queue length at the downstream port and sends pause frames to the upstream when congested.

Our observation (§ III) is that PFC is myopic, which only considers queue length to reflect congestion and controls the sending rate accordingly. As a result, the downstream port of PFC cannot accurately determine the current network status, leading to the demand for ample buffer space to maintain high throughput and increasing queueing delay consequently.

In light of the above observation, this paper extends RoCEv2 to long distance inter-DC link by proposing Bifrost, a downstream-driven lossless flow control that uses in-flight packets together with queueing packets to control the flows. In-flight packets serve as foresight for Bifrost to make precise control decisions and thus achieve high performance in long distance transmission. Moreover, by considering the multi-priority queue scenario, we follow the idea of Bifrost and introduce BifrostX, which has the capability of controlling the sending rate of each priority queue separately.

We solve several challenges when making the design decisions (§ IV-A).

1) How to obtain in-flight packets at the downstream port without specific upstream information? Bifrost downstream port takes full control of upstream sending rate and proposes an idea of *virtual in-flight* packets to indicate the upper bound of in-flight packets by maintaining a history of previous pause decisions (§ IV-B).
2) How to achieve the minimum buffer usage with the lossless feature and high performance at the same time? We theoretically prove that Bifrost can minimize buffer usage to almost one BDP when fitting the restrictions of the lossless feature and no throughput loss (§ IV-C).
3) How to implement and deploy Bifrost to be compatible with the Ethernet/IP protocol stack? Bifrost can be implemented by making a slight modification to the downstream port of PFC without changing the pause frame format, which means it does not conflict with the Ethernet/IP protocol stack (§ V).
4) How to extend Bifrost to the current switch implementation of the multi-priority queue? BifrostX takes a step further and utilizes two kinds of credits to control the upstream traffic, simultaneously. The per-link credit is produced in the way same as Bifrost and is used to achieve no packet loss and no throughput loss. Meanwhile, the per-queue credit is computed based on the real-time queue length change for each queue and is used to control the rate for each queue (§IV-E).

We have implemented a prototype switch of Bifrost (§ V) and deployed it to a real long distance transmission scenario to verify the feasibility and performance. We also evaluate the performance of Bifrost in cross-DC environment with ns-3 simulations (§ VI). Compared to PFC with the same buffer size, Bifrost reduces average flow completion time (FCT) and tail FCT of inter-DC flows by up to 46.8% and 56.3% and reduces the overall average and tail FCT by 55.2% and 63.5% under the production workload in the simulations. Bifrost does not restrict the link length and can support the DCI for thousands of kilometers. Bifrost extends RoCEv2 to the cross-DC transmission without performance compromise.

II. Background

A. The Emerging Cross Data Center Application

With the rapid development of cloud services and high performance computing (HPC), interconnection between data centers (DCs) is emerging to support diverse applications. The need for cross-DC applications comes from several reasons.

First, limited lands, power, and connectivity have restricted the scaling of large DCs. Large cloud service providers (CSPs) have taken the strategy to employ a collection of DCs connected through dedicated fibers to serve a region instead of one mega-DC, which is coined data center interconnections (DCI) [4], [5], [28]. The DCI fiber length could range from tens of kilometers for city-wide interconnection to thousands of kilometers for nationwide interconnection. China also proves a national computing system that interconnects supercomputing hubs and DC clusters across the country to solve the issue of imbalanced resource distribution in the country [16], [29]. It leverages computing hubs in the west, where land prices and electricity costs are low, to serve the computing demand from the east. Second, many data-intensive applications involve large sets of data spread across DCs and adopt a "move computation to data" paradigm, *e.g.*, cloud services [30], [31], [32], data analytics [6], [7], [8], machine learning [9] and graph processing [10], [11]. Besides, the Energy Sciences Network [12], InfiniCortex [13], [14] and InfiniCloud [15] are built to connect supercomputing centers across the nation and continents. Third, federated cloud computing is proposed for data privacy management and international digital sovereignty requirements [33], [34]. Many countries and organizations have set up regulations of data privacy protection that restrict data movement and in turn require more design for these geo-distributed systems [35], [36], [37].

The traffic across the DCs can be classified into three categories: (1) *interactive* traffic that is sensitive to delay (*e.g.*, user request in distributed services and databases across DCs that triggers cross-DC communication); (2) *elastic* traffic that requires delivery within seconds or minutes (*e.g.*, data updating across DCs); and (3) *background* traffic without strict requirements but tends to desire high throughput (*e.g.*, data backup for fault tolerance and provisioning activities for performance) [1], [2], [3], [38]. These traffic patterns over the long-haul link demand a well-designed underlying network that provides low latency and high throughput.

B. Flow Control for RDMA

Remote Direct Memory Access (RDMA) is a technology that achieves high throughput, low latency, and less CPU consumption through kernel bypass technique and has been widely used in DCs for HPC. Infiniband (IB) [21] and RDMA over Converged Ethernet (RoCEv2) [22] are the state-of-the-art RDMA protocols. IB is designed as a dedicated protocol and is popular in HPC. However, IB is expensive, and the dedicated protocol leads to poor compatibility that cannot be easily deployed to heterogeneous network systems. RoCEv2 is designed to be compatible with Ethernet/IP protocols without performance compromise. Consequently, it is a trend to deploy more RoCEv2 in DCs rather than IB [23], [24], [25], [26].

TABLE I
THE COMPARISONS OF DIFFERENT FLOW CONTROL SCHEMES

	Ethernet/IP compability	Buffer saving	Fine-grained control	High throughput in WAN
PFC	✔	✘	✘	✘
CBFC	✘	✔	✔	✔
SWING	✔	✔	✘	✘
Bifrost	✔	✔	✔	✔

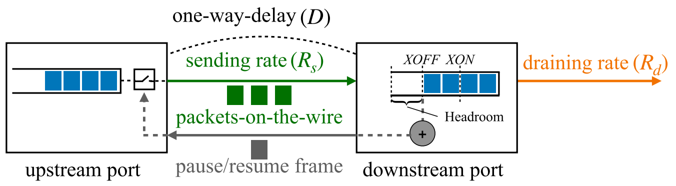


Fig. 1. PFC controls flow rate according to the queue length.

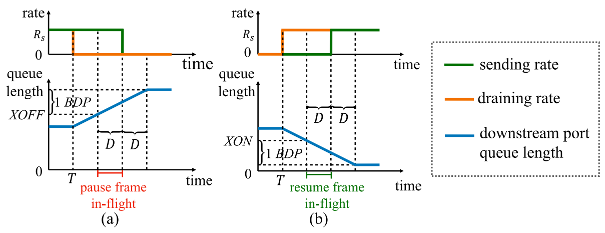


Fig. 2. PFC downstream port queue length in the worst case.

RoCEv2 requires a lossless network (*i.e.* no switch buffer overflow) to guarantee the performance [39], [40], [41], which is supported by Priority-based Flow Control (PFC) [42] by default. As shown in Figure 1, for each priority queue, the downstream port generates a pause frame when the ingress queue length exceeds a high threshold (*XOFF*) and sends it to the upstream port to pause the flow. When the ingress queue length decreases below a low threshold (*XON*), the downstream port sends a resume frame to the upstream to resume the transmission. The difference between the total ingress buffer size and *XOFF* is the headroom to accommodate the in-flight packets. If the headroom is larger than the one-hop bandwidth-delay product (BDP), buffer overflow can be avoided.

SWING [43] is proposed to mitigate the issue caused by PFC. It decouples the switch buffer and the flow control signal by adding an extra relay device which has a large DRAM, at the DCI. The DCI switch is only responsible to trigger flow control signal, while the relay device accommodates packets and forwards the signal.

CBFC [21] is the lossless flow control integrated with IB. The downstream port maintains the sum of total blocks received and available buffer space, named Flow Control Credit Limit (FCCL), where one block is 64 bytes [44]. The upstream port maintains the total blocks sent, named Flow Control Total Blocks Sent (FCTBS). The difference between FCCL and FCTBS is the available credits for the upstream to decide whether to send out a packet. Finally, we summarize the characteristics of different schemes in Table I.

III. MOTIVATION

A. Extending RoCEv2 to Cross-DC

With the increasing demand for cross-DC communication, network for DC interconnection has become a hot research topic. The need for high-performance cross-DC applications implies the underlying support of a high-throughput, low-latency network. Since RDMA has gained popularity in the intra-DC environment, we raise the idea of applying RDMA to the cross-DC scenario to extend its high performance. Specifically, we choose RoCEv2 over IB as the basis.

There are several reasons for this decision. First, RoCEv2-based transfer is not bound to the CPU computation limit but to the host memory bandwidth, which is readily scalable. Very high capacity flows can efficiently transfer between server nodes as a result [13]. We expect RoCEv2 to bring high performance in the cross-DC scenario. Second, it would take little effort to migrate RoCEv2 to the cross-DC scenario since it is well compatible with Ethernet/IP network. Third, using RoCEv2 in both intra-DC and cross-DC applications keeps the software development consistent and portable. Fourth, RoCEv2 hardware that has been deployed in DCs can be directly utilized for cross-DC data transmission so that no update is required for intra-DC devices. To summarize, it would be cheap and easy to develop high performance cross-DC applications by extending RoCEv2 to inter-DC links.

IB is not preferred to RoCEv2 not only because of the poor compatibility, but the flow control message's format also limits the distance it could support [45]. There are 12 bits in the CBFC message for the FCCL, which implies the available credits. The credits should be larger than one BDP for correctness. This format limits IB to at most 1 kilometer for a 100 Gbps link. Though it is possible to change the block size or the format to support a longer distance, it would increase the cost of configurations and management.

B. Lossless Flow Control Is Not Good Enough

PFC, the underlying flow control that provides a lossless network for RoCEv2, is responsible for supporting high throughput and low latency data transmission. However, the long-haul links with large round-trip-time (RTT) introduce several issues to PFC. We use the model in Figure 1 to illustrate the problems. R_s is the bandwidth of the long-haul link, and R_d is the full draining rate of the ingress buffer. D is the one-way-delay of the long distance link.

We first clarify the two goals when considering cross-DC transmission. First, the downstream buffer is not allowed to overflow because zero packet loss is the native goal of PFC. Second, the downstream switch should encounter no *throughput loss*. We define *throughput loss* as the egress of one switch is ready to send packets, but the queue contains no packets, leading to the draining rate drops to zero. The reason for throughput loss is that the upstream of this switch has been excessively paused by the flow control. In the case of long distance transmission, it takes a long delay for the resume message to transfer from the downstream to the upstream. During this period, the downstream ingress queue could be drained so that it has to wait for the data packets to arrive. Link bandwidth is not fully utilized when throughput loss occurs.

How to avoid buffer overflow? When the queue length exceeds *XOFF* threshold, the downstream port sends a pause frame to the upstream. Consider the worst case when the upstream port is sending packets at its line rate while the downstream switch stops draining at time T, as shown in Figure 2(a). In this case, the queue builds up at the downstream switch, and soon the queue length exceeds *XOFF* threshold. Then, the downstream switch emits the pause frame to its upstream to avoid the overwhelming data injection. It takes a one-way-delay D for the pause frame to arrive at the upstream and take effect, during which time the downstream queue will receive R_sD packets. The upstream port pauses the flow immediately, but there have already been R_sD in-flight packets. Thus, it takes $2D$ time for the downstream port to receive all the packets since it sent pause frames, so the downstream port needs a headroom of at least $2R_sD$ to accommodate all the packets, i.e., one BDP.

How to ensure no throughput loss? When the queue length decreases below *XON* threshold, the downstream port sends a resume frame to the upstream. Similarly, the worst case is when the upstream port has been completely paused and the downstream link starts to send at line rate at time T, as shown in Figure 2(b). After the queue length is below *XON* threshold, the downstream switch starts to emit the resume frame to the upstream. It takes D time for the resume frame to arrive at the upstream and takes D time for the data injected by the upstream to arrive at the downstream. Hence, the packets buffered at downstream should be at least $2R_dD$, i.e., one BDP. Usually, the bandwidth capacity of multiple links forming a path remains consistent, i.e., $R_d = R_s$, which avoids bandwidth waste and physical hotspots. Thus, to avoid buffer overflow and throughput loss, the total buffer reservation for PFC is at least 2 BDP.

The critical reason PFC reacts untimely and needs a deep buffer is that the downstream port makes a choice only by considering the current queue length. The queue length reflects the current congestion status of the network but does not contain other information about the network. The downstream port is unable to know whether the queue is increasing or decreasing. As a result, the configured threshold has to consider both cases, leading to a long queue and large buffer requirement. This issue is summarized as *myopic*.

Microsoft Azure [27] tackles the problem by simply adding off-chip DRAMs and leveraging shared memory between switch ports. However, the off-chip DRAM cannot meet the requirement of high performance transmission of DCI switch in high load and it does not reduce the queueing delay. SWING [43] addresses the issue of PFC by decoupling the flow control signal from the packet buffer. However, SWING's flow control is still triggered by queue length threshold, which cannot react to congestion swiftly because of the same *myopic* issue as PFC. As a result, SWING has sub-optimal performance as it is not fine-grained.

C. Lossy Solutions Have Poor Performance

Besides the lossless flow controls shown in Section II-B, there are also lossy solutions for RoCEv2, which can be adopted in the long-haul WAN. Improved RoCE NIC (IRN) [46] is an approach designed to be compatible with RoCEv2 and it uses an end-to-end flow control with selective retransmission to deal with packet loss. Contrary to PFC, IRN sender emits all packets within a sliding window (*i.e.*, a bitmap), and the switches drop packets when encountering buffer overflow. The fixed-size sliding window bounds the number of inflight packets and influences the performance of data transmission. As discussed by [43], IRN can achieve good performance with a well-designed topology where RTTs between hosts are close. However, inter-DC long-haul transmission introduces extremely larger RTT than intra-DC links. IRN's static bitmap is too rigid to balance the intra-DC and inter-DC traffic simultaneously.

TABLE II
IRN CONFIGURATIONS

	IRN-intra	IRN-inter
bitmap size (packets)	934	26800
$RTO_{high}(\mu s)$	175	1664
$RTO_{low}(\mu s)$	28	804

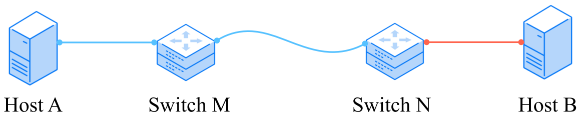


Fig. 3. Topology to evaluate the performance of IRN.

Furthermore, lossy solutions like IRN and TCP are not suitable for cross-DC transmission by their nature. It is difficult for the sender to perceive packet loss in time when congestion occurs at downstream DC. Packet retransmission also introduces high flow latency due to the large inter-DC RTT. Lossy solutions can not meet the requirements of high throughput and low latency for cross-DC applications.

For example, IRN configurations include the bitmap size that depends on end-to-end BDP, and RTO_{high} and RTO_{low} that depend on flow RTT. It has two configurations for two kinds of flows, intra-DC flow and inter-DC flow in cross-DC scenario. We denote them as IRN-intra and IRN-inter. We follow the recommendation of IRN configurations [46] as shown in Table II. IRN-inter has a much larger bitmap size, RTO_high, and RTO_low than IRN-intra to adapt to the inter-DC network condition.

A simulation is conducted to understand IRN. We build a linear topology as shown in Figure 3. The propagation delay between two hosts is configured to 14 μs and 402 μs respectively to simulate intra-DC and inter-DC scenarios. The latter indicates a long-haul link of 80 kilometers. The switch buffer is configured to 6 MB and 40 MB, respectively. Every link has a bandwidth of 400 Gbps. The link between switch N and host B is congested with different available bandwidths ranging from 50 Gbps to 400 Gbps.

We start a single long flow from Host A to Host B. Figure 4a shows that with IRN-intra, the goodput of inter-DC traffic remains almost unchanged when we increase the congestion bandwidth of the bottleneck link. This is because the static bitmap configuration is too small and it restricts the traffic injection. On the other hand, Figure 4b shows that, when utilizing IRN-inter for the intra-DC traffic, a large bitmap of IRN-inter allows for excessive traffic transmission, which

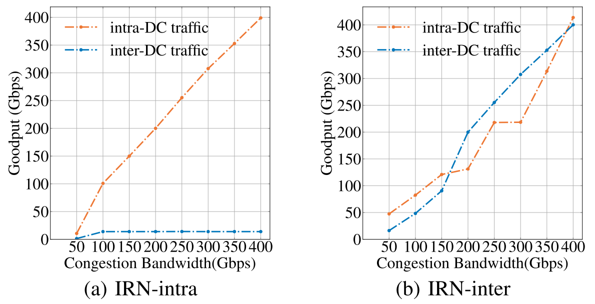


Fig. 4. Goodput with IRN-intra and IRN-inter under different congestion bandwidth.

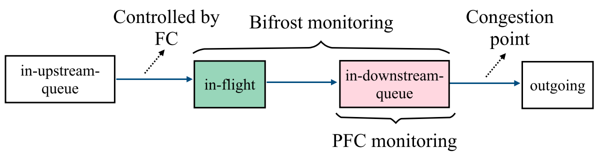


Fig. 5. Packets transition through 4 states when transmitting from upstream to downstream.

leads to frequent congestion and retransmission, hurting the goodput. Moreover, the inflated RTO configuration of the IRN-inter also delays the data retransmission, contributing to goodput degradation.

IV. Design

We propose a new lossless flow control, Bifrost, for long distance cross-DC links. Bifrost aims to use the minimum buffer while fitting the restrictions of lossless feature and no throughput loss at the same time. Moreover, the multi-priority queue extension, BifrostX is proposed to accommodate the practical multi-priority queue scenario.

A. Insight and Challenges

Packets transition through four states when transmitted from the upstream to the downstream, as shown in Figure 5: 1) in-upstream-queue, 2) in-flight, 3) in-downstream-queue, and 4) outgoing. The transition from *in-downstream-queue* to *outgoing* at the downstream egress depends on the congestion status, while the transition from *in-upstream-queue* to *in-flight* at the upstream egress is controlled by flow control. Flow control is responsible for adjusting the flow rate between upstream egress and downstream egress. Therefore, insights are that *in-flight* packets should be considered together with *in-downstream-queue* packets to make rate control decisions, rather than just queue length like PFC.

An ideal flow control should keep the upstream sending rate consistent with the downstream draining rate with a delay of one D (*i.e.*, one-way-delay). One D is required for the downstream's signal to arrive at the upstream. Similar to the analysis in Section III-B, there needs to be at least one BDP of buffer space to accommodate the inflight packets, which is the minimum buffer required for any lossless flow control based on downstream port signals.

Here we face several challenges in designing a new flow control. First, we need to infer the in-flight packets at the downstream port when making control decisions since it lacks the upstream information. Second, we need to guarantee a minimum buffer usage of one BDP to fit the requirements of lossless feature and no throughput loss at the same time. This indicates that the flow sending rate should be precisely controlled according to the draining rate. Third, Bifrost should be compatible with the existing Ethernet/IP architecture and support a wide range of distances.

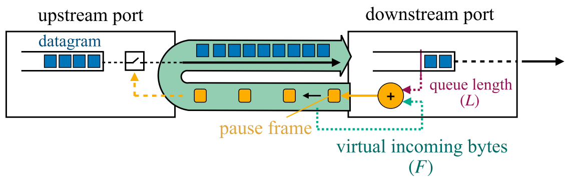


Fig. 6. Bifrost leverages virtual incoming bytes and queue length to make a pause decision to control the flow rate.

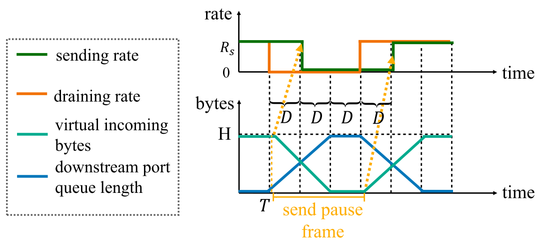


Fig. 7. Bifrost downstream port queue length in the worst case.

Instead of attempting to obtain the in-flight packets on the link, Bifrost downstream port maintains an upper bound of the incoming packets for the next RTT. Figure 6 describes the idea. The upstream port sending rate is fully controlled by the downstream with continuous pause messages. If the downstream port records the pause messages it has sent in the last RTT, it could infer the incoming packets of the next RTT from the history. For example, suppose that there is a long flow sending across the link. During the first D time, the downstream port sends pause frames every 10 μs each with a pause time of 5 μs. In the next D time, the downstream port stops sending pause frames. Then at the moment of $2D$, the downstream expects to receive 0.75 BDP in the next RTT. This is because there are 0.25 BDP of packets on-the-wire controlled by the PFC messages in the first D period and the other 0.5 BDP packets without control in the second D period. However, the upstream does not always have packets to send, so the pause history of the downstream is an upper bound of the actual incoming packets. We name the upper bound of incoming packets indicated by the pause history *virtual incoming* packets.

By leveraging the *virtual incoming* packets and queue length together, Bifrost predicts the buffer occupation in the near future and issues a forethoughtful pause message. In other words, the downstream port does not have to wait for the queue length to grow or shrink but notifies the upstream in advance. Thus, the downstream port is *farseeing* rather than *myopic*, leading to less buffer usage and higher performance.

B. Bifrost Design

We design Bifrost to work similarly to PFC, but in a fine-grained manner. Bifrost downstream port sends a pause frame carrying an elaborately calculated pause time periodically to fully control the sending rate of the upstream port. The length of time slot is denoted as T, which is configured to be far smaller than the one-way-delay D. The calculation of pause

TABLE III
NOTATIONS IN THE DESIGN

Notation	Meaning
T	Time slot
Δ	BDP
F	*virtual incoming* bytes
L	Current ingress queue length
H	Total queue buffer
R_s	Sending rate
D	One-way-delay

Algorithm 1 Bifrost Downstream Port Algorithm (blue Text Is for Pause Frame Leaking handling)

```
Input: BDP $\Delta$, time slot $T$, sending rate $R_s$, queue
       buffer reserved $H$ and $H \geq \Delta + R_sT$, pause
       frame leaking handling parameter $k$
Output: pause frame with specified pause time
1  $F \leftarrow \Delta + R_sT$        // Initiate the port
2  for every time slot $T$ do
3      $L \leftarrow$ current queue length
4      $r \leftarrow$ bytes received during last time slot
5      $n \leftarrow$ current time slot number
6      $p \leftarrow$ Bifrost_update$(L,\ r,\ n)$
7      if $p > 0$ then
8          generate_frame_and_send$(p)$
   /* calculate pause time $p$ and update
      $F$                                     */
9  Function Bifrost_update($L$, $r$, $n$):
10     $c \leftarrow \min(R_sT,\ H{-}L-F)$
11     $\hat{c} \leftarrow c$
       /* for every $k$ time slots           */
12     if $n\%k == 0$ then
           /* deduct over-granted bytes      */
13         $\hat{c} \leftarrow \max(0, c - (L + F - H))$
14     $p \leftarrow T - \hat{c}/R_s$
15     $F \leftarrow \min(\Delta + R_sT,\ F{-}r + c)$
16     return $p$
```

time takes both the current ingress queue length L and F into account, where F is the *virtual incoming* bytes in the duration of $RTT + T$. Since the control message only contains pause time, which is irrelevant to the distance of the link, Bifrost does not have distance limitation as to CBFC. Figure 6 describes the workflow of Bifrost and the algorithm is described in Algorithm 1. The blue text in Algorithm 1 is for special case handling, which is illustrated in Section IV-D. Table III summarizes the notations used in the paper.

We use an example to demonstrate the basic workflow as shown in Figure 7. R_s is the bandwidth between the upstream and downstream port, and Δ is the BDP along the link. H is the total queue buffer reserved for this queue. Suppose the link is sending a long flow at its line rate initially. The queue length of the downstream is close to zero, and the *virtual incoming* bytes is one BDP. Consider the worst case, when congestion occurs at downstream, at time T, the draining rate drops to zero, leading to an immediate increase in queue length. At the same time, Bifrost infers that there could be one BDP data incoming, which equals the buffer size, from Time T to T+2D. Thus, it sends a pause frame to the upstream immediately, stopping the sending during the second D period. Then, the upstream stops data transmission during the second D period. After the downstream stops emitting pause messages after the third D period, the upstream resumes data transmission after the fourth D period. If the congestion persists, the queue length continues to grow so that Bifrost persistently pauses the upstream and the virtual incoming bytes decrease commensurately. At last, it will reach a state where the sending rate equals the draining rate because of the periodic pause frames sent by the downstream. This indicates that Bifrost starts to decrease the sending rate as soon as congestion occurs, which is one D earlier than PFC.

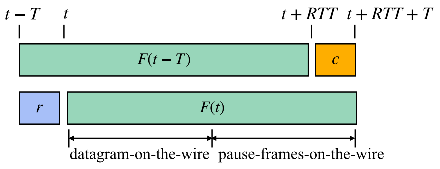


Fig. 8. Relationship between variables in Bifrost at adjacent time slots.

Next, we need to decide how the pause time is calculated precisely. If the draining rate drops, fewer packets will be sent out from downstream during the time slot, causing an increase in the queue length, which indicates the difference between received bytes and drained bytes during this time slot. So the upstream should send fewer bytes accordingly. However, instead of monitoring the change in queue length, we use the buffer remaining to reflect this change. The calculation is shown in Algorithm 1 line 10-14. Bifrost first calculates a granted bytes c to indicate how many bytes the upstream is allowed to send in the time slot one RTT later, and the granted bytes c is no more than R_sT. $H{-}L-F$ indicates the available buffer space at this moment with consideration of virtual incoming bytes. As the queue length grows, granted bytes c would decrease. Granted bytes c will be transformed to pause time p and sent out through the pause frame.

Finally, Bifrost maintains the virtual incoming bytes with counter r and the granted bytes c calculated. To illustrate the underlying logic, we define $F(t)$ as the downstream port virtual incoming bytes at time t, which is equal to bytes received from time t to time $t + RTT + T$. Here we have the problem of how to update from $F(t-T)$ to $F(t)$. The logic is illustrated in Figure 8. The granted bytes, in the form of pause time, will arrive at the upstream port after half of the RTT and take effect after another half of the RTT. That is to say, the granted bytes c calculated at time t indicates bytes received during time $t + RTT$ to $t + RTT + T$. Meanwhile, r is the bytes received during time $t - T$ to t. As a result, we have $F(t) = F(t-T){-}r + c$, corresponding to line 15 in Algorithm 1. We bound F to be less than $\Delta + R_sT$ because of the definition of F.

C. Bifrost Analysis

1) Why Does Bifrost Fit the Restrictions: Since Bifrost divides time into slots and updates variables at the end of every time slot, we denote the variables at the n-th time slot with a subscript n. Specifically, we use L_n, F_n, and c_n to denote the downstream port queue length, the virtual incoming bytes, and the granted calculated at the end of the n-th time slot. We use r_n to denote the bytes received during the n-th time slot and $\tilde{R}_n$ to denote the downstream port average draining rate during the n-th time slot. So we have $\forall n \in \mathbb{N}^+, \tilde{R}_n \leq R_s$. In addition, Δ is the BDP of the long distance link.

The design goals of Bifrost fall into two folds. (1) Bifrost is lossless means that $\forall n \in \mathbb{N}^+, L_n \leq H$. (2) Bifrost has no throughput loss means that L_n will not be negative, i.e., $\forall n \in \mathbb{N}^+, L_n \geq 0$. *With the Bifrost algorithm, if we configure H to a value larger than $\Delta + 2R_sT$, we can have the Theorem 1, and achieves both design goals.* We will give the full mathematical proof of Theorem 1 in the Appendix.

Theorem 1: $\forall n \in \mathbb{N}^+$, $\Delta + R_sT \leq L_n + F_n \leq H$.

Theorem 1 is consistent with our analysis in Section IV-A. According to Algorithm 1, if the queue length L increases because congestion occurs, the granted bytes will decrease and further decrease the virtual incoming bytes F correspondingly, and vice versa. L and F are a pair of variables that if one increases, then the other one would decrease. Finally, the sum of L and F will always be between $\Delta + R_sT$ and H.

Since F_n is defined as the virtual incoming bytes during time $RTT + T$, there should be $0 \leq F_n \leq BDP + R_sT$. With the Theorem 1, $L_n \leq H - F_n \leq H$ so that the buffer never overflows, and $L_n \geq \Delta + R_sT - F_n \geq 0$ so that Bifrost will not have throughput loss. To sum up, Bifrost fits the restriction of providing a lossless network without throughput loss.

2) Difference Between Bifrost and PFC: Figure 7 describes how the Bifrost downstream port queue length varies in the worst case, where we configured $H = \Delta + 3R_sT$. PFC requires a buffer of 2Δ as discussed before. Contrarily, Bifrost requires almost half the buffer reserved compared to PFC because R_sT is far smaller than Δ. Besides, the downstream port can quickly react to congestion (i.e., at most after time $2T$) and send out the pause frame, which is earlier than PFC by $1D$.

3) Bandwidth Cost by Pause Frames: The cost of the pause frames created by Bifrost is negligible. The pause frame is sent when the pause time calculated is non-zero at each time slot. The time slot is configured to tens of microseconds, *e.g.*, 10 μs, and one pause frame is 64 B. For a 100 Gbps link, the pause frame will cost at most 0.05% of the total bandwidth. Bifrost provides a near-optimal per-hop lossless flow control which can be a replacement for PFC. It is feasible to apply Bifrost to traditional Ethernet and even in the intra-DC environment. However, since Bifrost focuses on the inter-DC environment and targets high performance networks, cross-DC RDMA is better suited for Bifrost.

D. Special Cases Handling

Bifrost algorithm above is demonstrated without considering special cases. First, the pause frame may not be sent immediately after being generated because there could be a packet just in the process of transmission along the same link so that the MAC of the switch has to wait for the packet transmission to finish and then send out the pause frame. Second, the optical fiber's propagation delay can fluctuate in the order of nanoseconds for a link of hundreds of kilometers [47]. The pause time of one frame is bounded to one time slot T, and is scheduled to be sent with period T. If the pause frame is blocked by a packet-being-sent or delayed by the optical fiber, it will arrive late at the upstream port, when the upstream port has already started sending new data packets. This leads to the upstream port pausing less and sending one more packet than the downstream port expected. Besides, any unexpected delay inside the downstream switch, *e.g.*, computing delay, or data movement delay, may also cause the same issue in the long run. We name this issue as *pause frame leaking*.

Theorem 1 points out that $\forall n \in \mathbb{N}^+, L + F \leq H$ when no pause frame leaking occurs. If one pause frame leaks l bytes, then the downstream port updates the F as usual, but the upstream port sends more bytes (i.e., l) than the downstream port expects. Those bytes will finally lead to L larger than expected by l and cause $L + F > H$. As a result, Bifrost can check the $L + F$ periodically to see if it exceeds H to infer whether there is a pause frame leaking. The value $L + F - H$ indicates the "excessively granted" bytes before, and thus Bifrost can deduct them in the future to recover. The blue text in Algorithm 1 handles the pause frame leaking.

Besides, there needs to be another small headroom reserved for the queue to avoid buffer overflow in the worst case. If Bifrost checks the $L + F$ every k time slot, the worst case is that each of the k pause frames waits until a packet with maximum transmission unit (MTU) to send. The upstream will overly send $k \times \mathrm{MTU}$ bytes data leading to $L + F - H = k \times \mathrm{MTU}$. Therefore, an extra headroom of $k \times \mathrm{MTU}$ is required to prevent buffer overflow in the worst case.

The value of k is chosen according to the buffer size available. If the buffer is sufficient, we can configure a larger k. However, for ordinary cases, buffer overflow caused by pause frame leaking merely occurs when L actually exceeds the buffer reserved (i.e., the draining rate drops to zero and lasts a long time), which happens infrequently in production. As a result, neither performance nor correctness of Bifrost is sensitive to the parameter k.

E. BifrostX: Multi-Priority Version

To generalize to the multi-priority case, we extend the single-queue Bifrost and propose BifrostX, which is based on the virtual incoming bytes computed by Bifrost. In this case, granted bytes c is sent from downstream port to upstream port as a token, along with another token c_i for each priority.

The downstream port in BifrostX calculates c exactly the same as Algorithm 1. This token ensures that BifrostX is lossless and has no throughput loss. We use $c_i = R_sT - \max(\Delta L_i, 0)$ as the other set of tokens, where ΔL_i is the length change of the priority i. $\Delta L_i \geq 0$ means that the i-th queue causes the increase of the buffer occupation, which indicates the upstream port should reduce the number of packets in the future. $\Delta L_i \leq 0$ means the queue should not be paused. The calculation of the downstream port is shown in Algorithm 2 line 3-5. Note that the sum of all c_i is always greater than c_{max}, which results in no throughput loss. We can prove it as follows: according to line 5 in Algorithm 2, $\sum_i^N c_i$ is equal to $N \times R_sT - \sum_i^N max(\Delta L_i, 0)$. Because $\sum_i^N \Delta L_i \leq R_sT$, we get $\sum_i^N c_i \geq (N-1) \times R_sT \geq R_sT \geq c_{max}$, which demonstrates no throughput loss.

Once the upstream port receives a feedback frame, it uses the two sets of tokens to calculate the cumulative token of each

Algorithm 2 BifrostX Downstream Port Algorithm

```
Input: Total priority $P$, time slot $T$, sending rate $R_s$
Output: Feedback frame with 2 sets of tokens
1 for every time slot $T$ do
2     $c_{max} \leftarrow \hat{c}$ from Algorithm 1
3     for $i$ in $1 \cdots P$ do
4         $\Delta L_i \leftarrow$ length change of the priority $i$
5         $c_i \leftarrow R_s T - \max(\Delta L_i, 0)$
6     generate_frame_and_send($c_{max}, c_1, \ldots, c_P$)
```

Algorithm 3 BifrostX Upstream Port Algorithm

```
Input: Total token $c_{max}$, token of each priority
       $c_1 \cdots c_P$
Output: The readjusted tokens $c'_1 \cdots c'_P$
1 $c_{left} \leftarrow c_{max}$
2 for $i$ in $1 \cdots P$ do
3     $q_i \leftarrow$ current length of the priority $i$
4     $c'_i \leftarrow \min(q_i, c_i, c_{left})$
5     $c_{left} \leftarrow \max(c_{left} - c'_i, 0)$
6 return $c'_1, \ldots, c'_P$
```

priority. From high to low priority, token will be adjusted to the minimum of the current queue length, c_i, and the remaining token in c_{max}, as c'_i shown in line 3-5 of Algorithm 3. If the current queue length is the minimum, it means that this priority would not have enough data to send. Lower priorities will benefit from the remaining token. $c'_i \leq c_i$ and $c'_i \leq c_{left}$ ensures that, (1) the token of each priority is less than c_i, and (2) the sum of tokens for all priorities is less than c_{max}.

V. Implementation

Bifrost reuses the PFC pause frame format. The upstream port of Bifrost behaves the same as PFC, and the downstream port integrates the strategy described in Algorithm 1. The main difference is that Bifrost is triggered by time slot instead of by queue length threshold. Besides, Bifrost can reutilize the priority queues management of PFC. As the modification is moderate, Bifrost is compatible with the current Ethernet/IP network and is easy to implement and deploy.

We prototype Bifrost on a commodity switch provided by Huawei [48]. The switch integrates an on-chip co-processor that provides the capability to perform complex computing on the data plane. The co-processor reads the queue length from the ingress buffer and the RX bytes from MAC every time slot T. It maintains the virtual incoming bytes F and stores the RX bytes of the last time slot, which is used to calculate r. It then performs the Bifrost downstream algorithm and generates the PFC pause frame. The implementation is illustrated in Figure 9.

The configuration of time slot T has two constraints. First, it has to be larger than the internal delay of the switch, which includes the computation time, pause frame generation time, pause frame moving delay through the internal bus, and the delay in MAC. The delay is ~ 1 μs. Second, it is limited by the buffer available for Bifrost, the one-way-delay, and

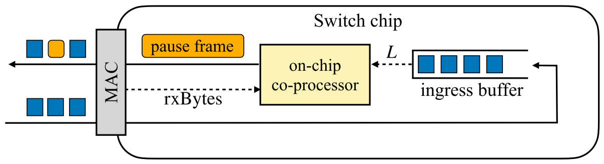


Fig. 9. Bifrost prototype implementation.

the PFC frame pause time field. (1) Based on our analysis in Section IV-C, Bifrost requires a buffer size of at least $\Delta + 2R_sT$, *i.e.*, a large T leads to a large buffer reserved. (2) T should be smaller than the one-way-delay; otherwise, it could not react in time. (3) The PFC frame pause time 16 bits field represents the time it costs to transmit data at line rate, which indicates at most 65535 quanta, equaling 4 MB. CT should be less than 4 MB consequently. To meet both the constraints, we configure T to be 10 μs.

Notice that the PFC pause frame includes eight pause time fields, each of which is two bytes [42]. The field indicates a time measured in the units of pause quanta, equal to the time required to transmit 512 bits of a frame at the data rate of the MAC [42]. Each field can assume a value of up to 65535 quanta. The pause time in Bifrost is bounded to one time slot. For a 400 Gbps link and a time slot 10 μs, the maximum pause time in Bifrost would be 7813 quanta, which would not provoke PFC pause frame field overflow.

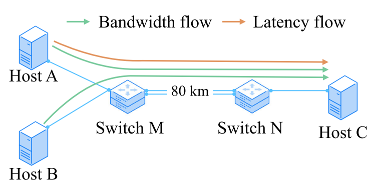
VI. Evaluation

Bifrost and BifrostX are evaluated with simulations and Bifrost is further tested in the real-world experiments. We deploy the Bifrost prototype switch to a long distance transmission scenario to verify that Bifrost requires less buffer and provides lower latency than PFC. Simulations are conducted to evaluate that Bifrost outperforms PFC, IRN, and IB in cross-DC environments under the production workload.

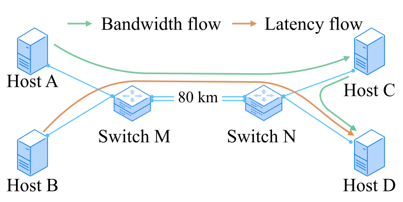
A. Real Deployment

Figure 10 shows the topologies of the deployment. In topology A shown in Figure 10a and topology B shown in Figure 10b, switch M and switch N are Bifrost DCI prototype switches connected by two optical cables of 80 kilometers with 100 Gbps bandwidth, respectively. DCI switch balances the traffic over the two ports by equal-cost multi-path (ECMP) algorithm, so the long distance link can have a maximum throughput of 200 Gbps. In topology C shown in Figure 10c, we deploy three 100 Gbps links instead. These configurations are based on typical production deployment. The one-way-delay of one long distance link is 400 μs and the BDP is 9.5 MB. The DCI switch is equipped with a 64 MB buffer. The bandwidth between every host and switch is 100 Gbps.

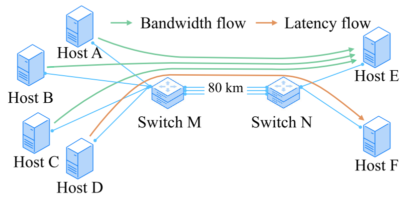
We use perftest [49] to generate various RDMA flows, which includes *bandwidth flows* and *latency flows*. We combine the two traffics to evaluate the performance of PFC and Bifrost. The bandwidth-sensitive flows serve as background traffic, and we evaluate the latency of flows which serve as the interactive traffic that is sensitive to delay. Perftest can generate multiple flows that can be considered to start simultaneously. We use perftest to generate various flows in different sizes to simulate real-world traffic. We disable the congestion control



(a) Case A: 2 hosts to 1 host

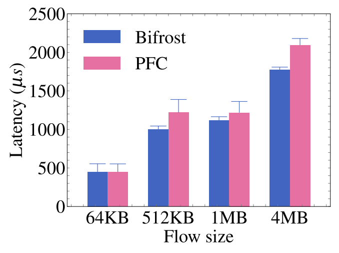


(b) Case B: mix of inter-DC and intra-DC flows

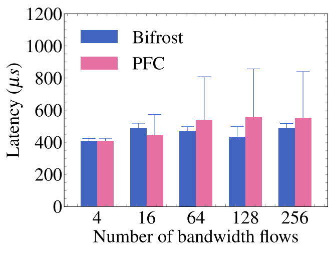


(c) Case C: incast flows and victim flows

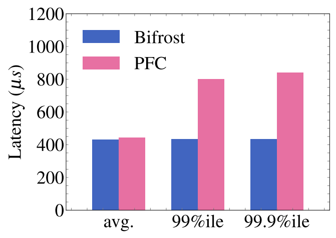
Fig. 10. Topology and traffic pattern in testbed experiments.



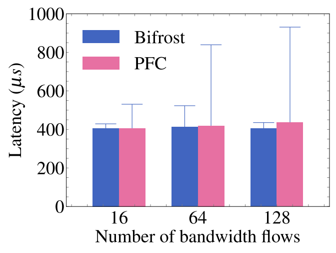
(a) Single flow in topology A



(b) Multiple flows in topology A



(c) latency of flows in topology B



(d) Flows in topology C

Fig. 11. Avg. latency and 99%ile latency of flows when using PFC and Bifrost in topology A, B, and C.

to avoid interference. Since the pause frames will never be paused and have a higher priority than normal data packets, the flow control can be seen as a full duplex. As a result, flows in the experiments are mostly in one direction, which is sufficient and concise for the evaluation.

1) Buffer Reserved for PFC and Bifrost: PFC guaranteeing no throughput loss requires a large buffer that cannot be applied to our switch. We conducted some experiments to explore the appropriate value of *XON* and *XOFF*. PFC *XON* and *XOFF* are set to the same in the experiments. We use the topology shown in Figure 10a, and launch two bandwidth flows with perftest from host A to host C and host B to host C. Congestion occurs at the egress port of switch N and thus the ingress port of switch N will trigger PFC pause to switch M during the transmission. In this topology, the expected total bandwidth of flows is 100 Gbps.

We conducted a series of experiments with different PFC thresholds configured and found that the total bandwidth of flows achieves close to the expected 100 Gbps when the *XOFF* is set to a value larger than 6 MB. According to Section III-B, the *XOFF* should be set to the BDP, which is 9.5 MB, to guarantee no throughput loss in the worst case. Here 6 MB is enough to achieve the expected bandwidth because the traffic pattern is simple, and the worst case is less likely to occur. With this exploration, we set the long distance link port's PFC *XOFF* to 6 MB in the following experiments. Thus, the total buffer reserved for one long distance link is 15.5 MB because we need to reserve a headroom of one BDP (*i.e.*, 9.5 MB) to accommodate in-flight packets.

On the other hand, Bifrost needs to reserve at least a buffer of $\Delta+2R_sT$ according to the design. We configured the H to be $\Delta+3R_sT$ which is 9.72 MB, 37% less than PFC. Bifrost parameter k is set to 1. Notice that PFC may cause throughput loss with these configurations while Bifrost will not.

2) Latency Evaluation: In this section, we evaluate the latency of flows when using Bifrost and PFC under different topologies and traffic patterns.

Case A in Figure 10a is a scenario when two hosts send flows to one host at the same time. We first launch a single bandwidth flow A-to-C and B-to-C, and launch latency flows from host A to host C. The bandwidth flow size varies from 64 KB to 4 MB. With the appropriate PFC configuration explored in Section VI-A, both PFC and Bifrost achieve the same bandwidth. However, latency flow with PFC enabled has higher average latency and 99%ile latency, as shown in Figure 11a. As to bandwidth flow size 512 KB, Bifrost reduces average latency by 18% and 99%ile latency by 25%. We then launch multiple bandwidth flows instead to increase the background workload and launch the latency test flow. The size of latency flows is set to 8 KB and the same below. As shown in Figure 11b, with a different number of bandwidth flows, Bifrost achieves lower 99%ile latency than PFC. With 128 bandwidth flows, Bifrost reduces average latency by 22.5% and 99%ile latency by 42.0%.

Case B in Figure 10b evaluates the performance of inter-DC traffic when it conflicts with intra-DC traffic. We launch inter-DC background traffic with bandwidth test from host A to host C and intra-DC background traffic from host C to host D. We evaluate the latency of flows from host B to host C. As shown in Figure 11c, Bifrost has lower average, 99%ile and 99.9%ile latency than PFC. Bifrost reduces the 99%ile latency by 46% than PFC.

Case C in Figure 10c evaluates the performance of victim flow when incast occurs. Host A, B, and C send bandwidth flows to host E simultaneously to generate incast background traffic, while host D launches latency flows to host F as victim flows. Figure 11d shows the average and 99%ile latency of the flows from host D to host F. Bifrost outperforms PFC when the number of bandwidth flows varies, especially the 99%ile latency. When 128 bandwidth flows are launched, Bifrost reduces 99%ile latency by 53%. The average latency of PFC is close to Bifrost when PFC has more buffer to deal with the severe congestion caused by incast and pauses less.

To sum up, Bifrost achieves lower latency, especially tail latency, than PFC in different testbed scenarios while using 37% less buffer. This latency improvement can be attributed to the lower queueing delay of Bifrost.

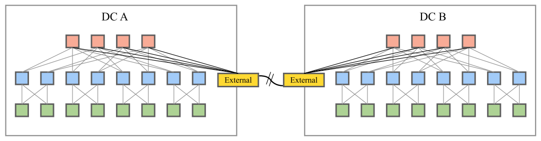


Fig. 12. Topology in the simulation where the external is one DCI switch. Hosts are omitted.

B. Simulations

We use ns-3 simulations to evaluate the performance of Bifrost in the cross-DC environment and compare it with PFC, IRN, SWING, and CBFC. Simulations are conducted under the production traffic workloads using public traffic traces to simulate the actual production scenario.

Topology settings. We use the fat-tree topology [50] in DC as shown in Figure 12. Each Top-of-Rack (ToR) switch connects to 2 servers, so there are 16 servers in one DC. Since we mainly focus on inter-DC traffic, 16 servers are enough to generate flows that fill up the link bandwidth. Each link in the fat-tree topology is 100 Gbps. The external is one DCI switch, with every core switch connecting to it. The bandwidth of the link between two DCI switches is 400 Gbps and is configured to be 600 kilometers long if not specified, which means the one-way-delay of the long link is 3 ms.

PFC settings. For all the switches in DC, PFC threshold *XOFF* and *XON* are configured to be 288 KB. The headroom for each queue is 30 KB, which is the one-hop BDP in DC. We also configured a 10 MB shared buffer and a dynamic threshold of PFC by setting α to 4, which is a typical setting in production. For DCI switch, the PFC threshold of ports connecting to intra-DC links is configured the same as the intra-DC switch. The PFC threshold of ports connecting to the long-haul link is configured to 1 MB and a headroom of 286 MB. Shared buffer is disabled in DCI switch.

IRN settings. As discussed in Section III-C, we choose the IRN configuration optimized for the inter-DC flows, which has better performance. The bitmap size of IRN is set to 50,134 packets, and RTO_{high} and RTO_{low} are 7,075 μs and 6,017 μs respectively. PFC is disabled when evaluating IRN. These configurations are based on the recommendation in [46].

Bifrost settings. We only deploy Bifrost on the long-haul ports on DCI switches, and all the switches inside the DC use PFC as the flow control, configured as the same thresholds as above. The time slot T of Bifrost is configured to 10 μs if not specified below. The BDP of the long distance link is 286 MB. H is set to 287 MB, and k is set to 1.

CBFC-ideal settings. Infiniband is deployed to all hosts and switches in the topology. As discussed in Section III-A, CBFC cannot be deployed in long links due to the format limitation. We break the 12 bits limitation in the simulation to evaluate the best possible performance of CBFC, named *CBFC-ideal*. The buffer space of switches is set to the same as Bifrost.

SWING settings. All switches in SWING are configured the same as the intra-DC switches in the PFC experiment. Besides, SWING relay device is configured with a 287 MB buffer and 198 KB *XON*/*XOFF*.

Congestion control. DCQCN [40] is enabled in the experiments since RoCEv2 and IB uses DCQCN by default.

Traffic loads. The evaluations adopt commonly used DC traffic trace, WebSearch [51] and FB_Hadoop [52]. The

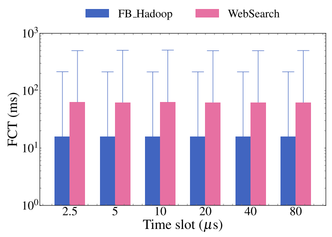


Fig. 13. FCT of flows using Bifrost with different time slots.

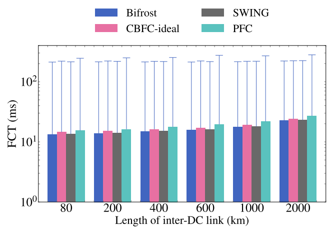
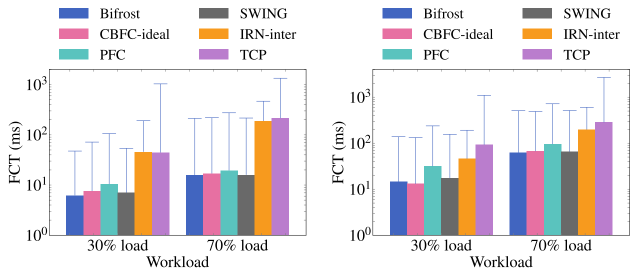


Fig. 14. FCT of flows using Bifrost over different distances.



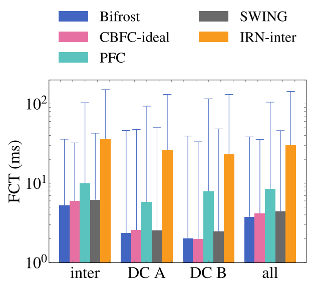
(a) 16 to 16 FB_Hadoop (b) 16 to 16 WebSearch

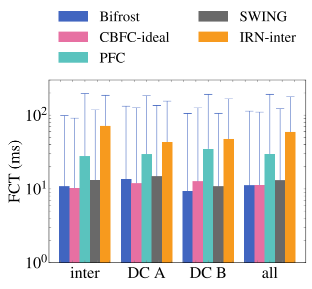
Fig. 15. FCT of flows in 16 to 16 inter-DC traffic pattern under 30% and 70% workloads.

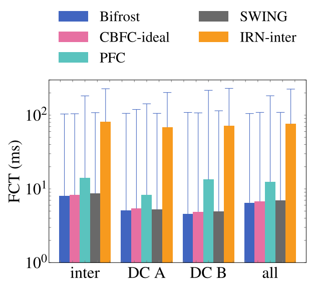
WebSearch workload is characterized by small requests and large responses, with 95% of the flows exceeding 1 MB [53]. 70% of the flows in the FB_Hadoop traffic are smaller than 10 KB, but 90% of the total traffic is contributed by flows larger than 100 KB. Though they are originally intra-DC workloads, the distribution of the flows corresponds to the characteristics of inter-DC traffic (§ II-A), so we use them in the simulation. We set different workloads to evaluate the performance of Bifrost, PFC, IRN, and CBFC-ideal.

1) FCT With Different Time Slots: We launch traffic from DC A to DC B in Figure 12 to evaluate the average FCT and 99%ile FCT of inter-DC flows. Flows are generated from 16 servers of DC A to 16 servers in DC B with a 70% workload of FB_Hadoop and WebSearch traffic. Since there are only inter-DC flows, flow control is mainly triggered by the ECMP routing collision within DC B. We use this case to gain insight into Bifrost performance with different time slots. Figure 13 shows that Bifrost is not sensitive to time slot settings as long as it satisfies the restrictions discussed in Section V. We choose $T = 10$ μs in the following experiments.

2) FCT Under Different Distances: Figure 14 shows the average FCT and 99%ile FCT with different lengths of the inter-DC link using Bifrost, CBFC-ideal, SWING, and PFC. We omit IRN in the figure because it leads to extremely large FCTs. Flows are generated from 16 servers of DC A to 16 servers in DC B with a 70% workload of FB_Hadoop traffic. The comparison results show that the length of the inter-DC link does not affect the relative performance of


(a) 50% all-to-all FB_Hadoop


(b) 50% all-to-all WebSearch


(c) 80% load FB_Hadoop

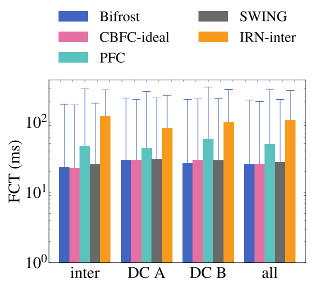
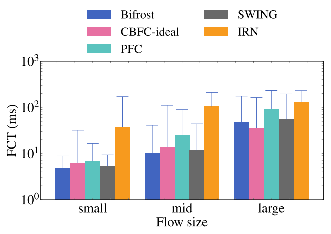

(d) 80% load WebSearch

Fig. 16. FCT of flows in 50/80% load of all-to-all traffic.


(a) FCT of inter-DC flows in different sizes in 30% load 16 to 16 inter-DC WebSearch traffic.

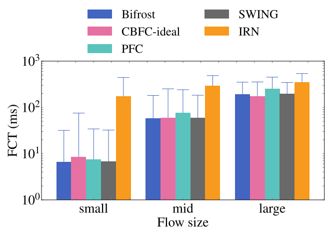
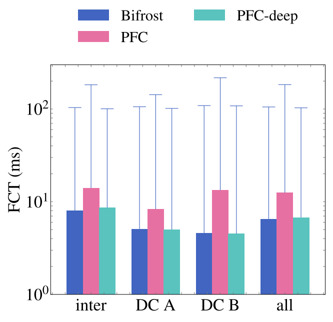

(b) FCT of inter-DC flows in different sizes in 70% load 16 to 16 inter-DC WebSearch traffic.

Fig. 17. FCT of inter-DC flows in different sizes in 30/70% load 16 to 16 inter-DC WebSearch traffic.


(a) FCT of all-to-all flows compared with PFC-deep under 80% load.

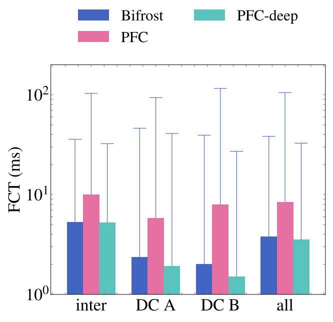
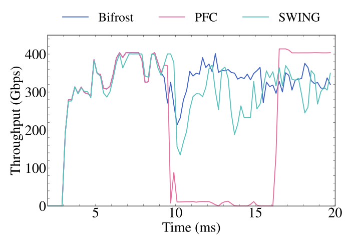

(b) FCT of all-to-all flows compared with PFC-deep under 50% load.

Fig. 18. FCT of all-to-all flows compared with PFC-deep under 80/50% load.


(a) Throughput variation over time

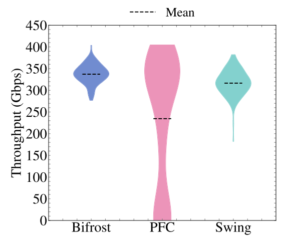

(b) Throughput distribution

Fig. 19. Comparison of the long-distance link throughput.

Bifrost, CBFC-ideal, SWING, PFC, and IRN. We choose 600 kilometers in other experiments as a typical case.

3) Inter-DC Traffic: We launch flows generated from 16 servers of DC A to 16 servers in DC B following the FB_Hadoop and WebSearch with workloads of 30% and 70%. Figure 15 shows the average and 99%ile FCT of these inter-DC flows. In the 30% load FB_Hadoop traffic in Figure 15a, the average FCT and 99%ile FCT of Bifrost is 11.5% and 11.8% better than SWING, 17.4% and 33.7% better than CBFC, 40.1% and 55.2% better than PFC, and 86.4% and 75.2% better than IRN.

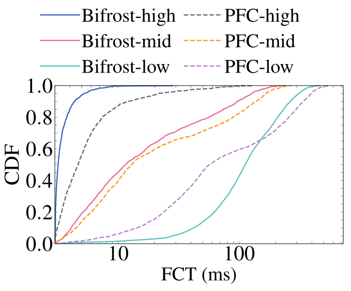
We also evaluate the inter-DC flows using traditional TCP NewReno and results show that TCP has worse performance than the RDMA solutions, because of its slow convergence and unsophisticated loss recovery with the high BDP environment.

4) All-to-All Traffic: We evaluate the performance of Bifrost with all-to-all traffic, which contains inter-DC traffic together with intra-DC traffic in both DC A and DC B. We generate random flows with FB_Hadoop and WebSearch workload, with some flows staying within DC and others sent across two DCs. The results are shown in Figure 16. Compared with PFC under the 50% workload of FB_Hadoop (Figure 16a), Bifrost reduces average FCT and 99%ile FCT of inter-DC flows by 46.8% and 56.3%, while reducing average FCT and 99%ile FCT of the overall flows by 55.2% and 63.5%. Compared with SWING, Bifrost reduces the average FCT and 99%-tile FCT of inter-DC flows by 14.9% and 16.1%, while reducing the overall average FCT and 99%-tile FCT by 14.2% and 16.6%. When the network load grows to 80%, the performance advantage of Bifrost becomes smaller, while it still gains one of the best performance.

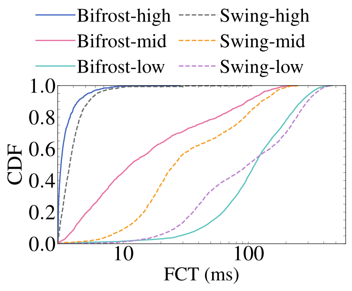
5) Different Flow Sizes: Figure 17 shows the average and 99%ile FCT of inter-DC flows in different sizes. A flow with a size smaller than 100 KB is a small flow, and a size larger than 1 MB is a large flow. The remaining flows are classified into middle flows. As shown in Figure 17a, Bifrost improves the performance of small and middle flows compared to CBFC-ideal. The reason is that since CBFC is used for IB and all switches are implemented with CBFC, the time slot of CBFC is too large for an intra-DC environment, and it reacts slowly to small flow congestion within DC. Compared with CBFC, Bifrost is friendly to small flows so that it improves the performance of interactive and elastic traffic in the applications. In all three types of flows, PFC and SWING perform worse because of their *myopic* feature as mentioned in Section III-B. IRN is significantly inferior with small and middle flows because packet loss is more damaging to small flows and loss recovery causes higher overhead. When we increase the network load from 30% to 70%, the FCT of different-sized flows enlarges, while the comparison of different schemes remains the same, shown in Figure 17b.

6) Compare to PFC With Deep Buffer: We compare the performance of PFC with enough buffer size to Bifrost. As discussed in Section III-B, PFC needs 2 BDP of buffer size to perform well. We configured the PFC threshold and headroom both to be 1 BDP (286 MB), denoted as PFC-deep. We use the 80/50% load of all-to-all FB_Hadoop traffic.

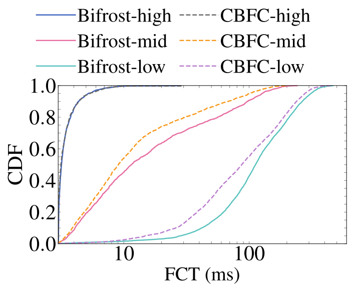
As shown in Figure 18a, PFC-deep has a better performance than the original PFC. The reason is that when PFC back



(a) BifrostX vs. PFC



(b) BifrostX vs. SWING



(c) BifrostX vs. CBFC

Fig. 20. FCT of 3 priority flows in 70% load of inter-DC Websearch traffic.

pressures flows to the upstream by pausing, PFC with deep buffer accommodates most of those packets in the DCI downstream queue and pauses less. Bifrost achieves a comparable performance to PFC-deep but with half the buffer utilized, which not only decreases the queue delay but results in lower costs on switches. When we decrease the load from 80% to 50%, PFC is also inferior to other schemes, shown in Figure 18b. PFC-deep performs similarly to Bifrost for the inter-DC traffic and overall traffic, with both 50% and 80% network loads. While for the intra-DC traffic, PFC-deep achieves shorter FCT, under light load (50%).

7) Throughput: We monitor the throughput of the long distance link when using Bifrost, PFC, and SWING in the all-to-all traffic experiment. As shown in Figure 19a, the throughput of Bifrost is steadier than PFC and SWING because it controls the flow with foresight and in a fine-grained manner. Note that PFC stays close to 0 throughput for periods of 10 to 16 milliseconds, because it doesn't get enough 2BDP buffer and it is *myopic* that only considers queue length. SWING also has this problem, but its strategy allows for faster response time, resulting in higher throughput than PFC.

Figure 19b shows the throughput distribution in 0.1 s, which illustrates the stability of Bifrost in comparison with PFC and SWING. The result shows that Bifrost improves the average throughput by 43.67% over PFC and by 6.51% over SWING. A steady throughput is friendly to congestion control since the congestion control can converge faster and better. Besides, Bifrost decreases the queueing delay at the DCI switch and avoids confusing congestion control to overreact. Bifrost is expected to be compatible with most congestion control that leverages queue length or delay as the congestion signal.

C. BifrostX Simulations

We use ns-3 simulations to evaluate the performance of BifrostX in the cross-DC environment and compare it with PFC, SWING, and CBFC. According to Section VI-B, Bifrost significantly outperforms IRN and TCP, so this section does not provide a detailed comparison with them.

Priority Settings. Recent studies [3], [52] have shown that cross-DC traffic accounts for 30% of the total volume. About 30% of the intra-DC traffic and 16% of the inter-DC traffic are high-priority. Based on this, we set three priorities in this section. For the intra-DC traffic, each priority accounts for 28%, 32%, and 40% respectively; For the inter-DC traffic, each priority accounts for 16%, 25%, and 59% respectively.

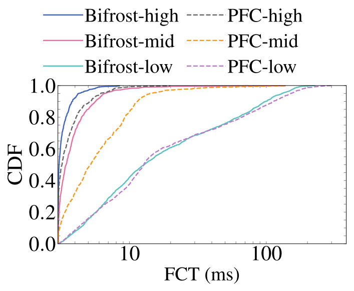
Topology and traffic load settings. The settings are identical to Section VI-B.

PFC, SWING, and CBFC settings. For each priority, settings are the same as Section VI-B. To avoid priority inversion, we use the static threshold in PFC.

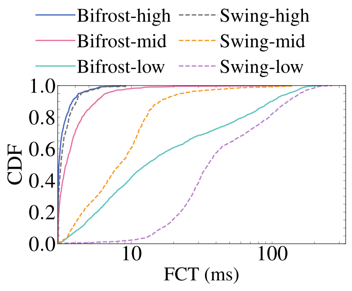
BifrostX settings. There is no need to set additional parameters for BifrostX, making the setting still the same as Section VI-B. Note that this means BifrostX has only a third of the buffer of PFC, SWING, and CBFC.

1) Inter-DC Traffic: We launch inter-DC flows with a workload of Websearch and 70% network load. As shown in Figure 20a and Figure 20b, BifrostX has lower latency than PFC and SWING. Compared to PFC, BifrostX reduces the average FCT by 54.5% and the 99%ile FCT by 88.1% for high-priority traffic and reduces the average FCT by 27.4% and the 99%ile FCT by 19.7% for middle-priority traffic. For low-priority traffic, BifrostX reduces the 99%ile FCT by 20.5% but increases the average FCT by 5.0%. Compared to Swing, BifrostX reduces the average FCT by 16.5% and the 99%ile FCT by 16.5% for high-priority traffic and reduces the average FCT by 37.2% and the 99%ile FCT by 12.8% for middle-priority traffic. For low-priority traffic, BifrostX slightly increases he average FCT by 0.3% and the 99%ile FCT by 6.9%. BifrostX reduces high-priority latency by increasing a small amount of latency for low priority traffic, which normally is the latency-insensitive background traffic in data center networks. BifrostX has a slight performance degradation for middle and low priorities compared to CBFC. Since CBFC uses three times the buffer, leading to relatively fewer pauses, this small performance loss is acceptable. To evaluate the performance of different schemes under a light network load, we decrease the load from 70% to 30% and rerun the simulation. The results depicted in Figure 21 are similar to the above 70% results, i.e., BifrostX outperforms PFC and SWING, with high and middle-priority traffic, while its performance degrades with the low-priority traffic slightly. Meanwhile, BifrostX achieves comparable performance to CBFC, with all kinds of traffic.

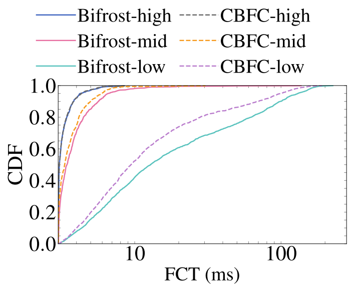
2) All-to-All Traffic: Unlike traffic in Section VI-B.4, we evaluate traffic that only 30% of flows are the inter-DC flows, and we monitor the FCT of this kind of flows. The results are shown in Figure 22. Compared to PFC, BifrostX reduces the average FCT by 4.0% and the 99%ile FCT by 13.5% for high-priority traffic and reduces the average FCT by 51.5% and the 99%ile FCT by 28.5% for middle-priority traffic. For low-priority traffic, BifrostX reduces the



(a) BifrostX vs. PFC

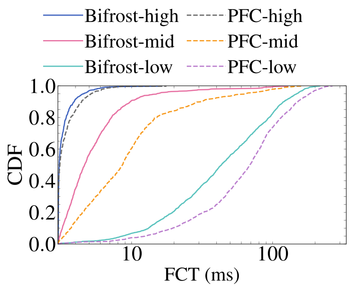


(b) BifrostX vs. SWING

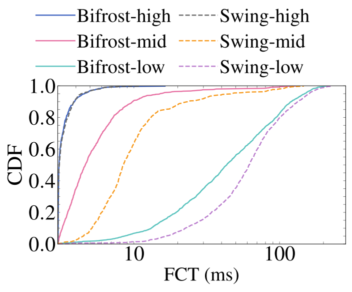


(c) BifrostX vs. CBFC

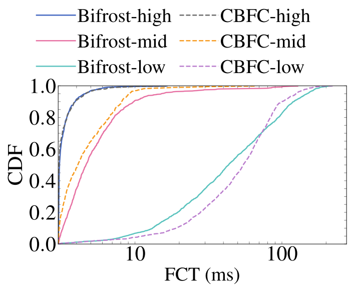
Fig. 21. FCT of 3 priority flows in 30% load of inter-DC Websearch traffic.



(a) BifrostX vs. PFC

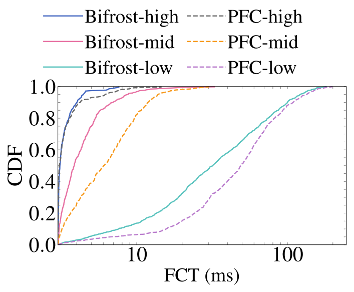


(b) BifrostX vs. SWING

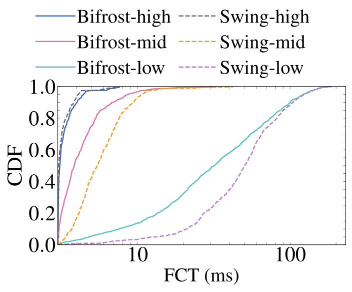


(c) BifrostX vs. CBFC

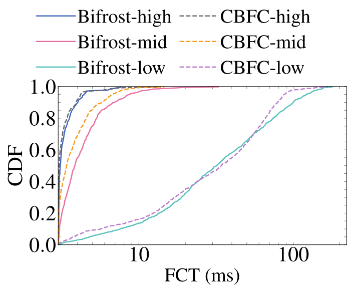
Fig. 22. FCT of 3 priority flows in 90% load of all-to-all Websearch traffic.



(a) BifrostX vs. PFC



(b) BifrostX vs. SWING

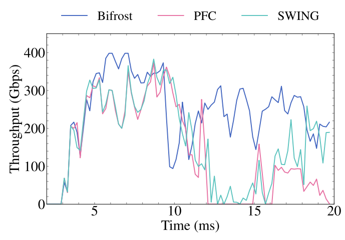


(c) BifrostX vs. CBFC

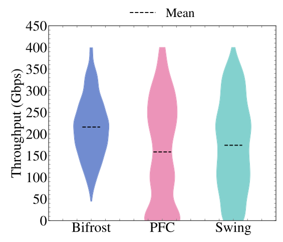
Fig. 23. FCT of 3 priority flows in 60% load of all-to-all Websearch traffic.

average FCT by 26.6% and the 99%ile FCT by 21.0%. Compared to SWING, the FCT of high-priority traffic of BifrostX is generally the same, with a difference of not more than 1%. BifrostX reduces the average FCT by 47.6% and the 99%ile FCT by 22.7% for middle-priority traffic and reduces the average FCT by 19.1% and the 99%ile FCT by 9.9% for low-priority traffic. In summary, with this more complicated traffic pattern, BifrostX still gains latency advantage over PFC and SWING, and obtains comparable performance with CBFC in the high priority. To evaluate the performance of different schemes under a light network load, we decrease the load from 90% to 60% and rerun the simulation. The results depicted in Figure 23 are similar to the above 90% results, i.e., BifrostX consistently outperforms PFC and SWING, with all kinds of traffic, especially for high and middle-priority traffic. Meanwhile, BifrostX achieves comparable performance to CBFC, with all kinds of traffic.

3) Throughput: In the all-to-all traffic pattern, we measure the throughput with BifrostX, PFC, and SWING by the same method as Section VI-B.7. Figure 24a shows that none of the three protocols can maintain high throughput for a long period of time under heavy load, but BifrostX has a higher and more stable throughput than PFC and SWING. In the range of 12 to 15ms, *myopic* PFC and SWING can only maintain a very low throughput because they only consider each priority's queue length, resulting in long-term congestion in both DC A and DC B. Meanwhile, BifrostX maintains a steadily high throughput by planning the credits of each priority in advance. Figure 24b illustrates that by using credit and significantly shallower buffer, BifrostX does not impair throughput, but improves it by 36.2% and 24.1% compared to PFC and SWING.



(a) Throughput variation over time



(b) Throughput distribution

Fig. 24. Throughput comparison in the multi-priority queue scenario.

VII. Related Work

Flow controls for DC network. Prior works mainly focused on the side effects caused by PFC and proposed solutions. CaPFC [54] modifies PFC by using ingress queue and egress queue statistics to react earlier. P-PFC [55] leverages the change rate of queue length to trigger PFC pause instead of using a fixed threshold. BFC [56] proposes a per-flow, hop-by-hop flow control and Floodgate [57] proposes a per-destination, hop-by-hop flow control, aiming to mitigate incast in DC. Both provide a fine-grained flow control that switches need to maintain more states. Another work, GFC [58], solves the deadlock problem in lossless networks by avoiding the hold-and-wait condition of deadlock. However, these flow controls are not designed for cross-DC applications. TLT [59] proposes an extension to existing transport to provide a lossy network by proactively dropping less important packets. It requires modifications to all hosts and switches in DC, which does not meet our goals.

Congestion control over WAN. There are congestion controls proposed for cross-DC communication over WAN that aim to improve the performance of inter-DC traffic. GEMINI [60] strategically integrates both ECN and delay signals for cross-DC congestion control. Annulus [61] leverages the increase of queueing delay at the ToR switch to early-detect the congestion over the WAN and controls the sending rate. FlashPass [62] proposes a proactive congestion control that adopts a sender-driven emulation process with send time calibration and early data transmission at starting phase to mitigate the buffer usage of switches. GTCP [63] switches between sender-driven and receiver-driven congestion control to adapt to intra-DC and inter-DC congestion. These solutions are proposed for WAN, which is different from DCI, where we can have full control of the network devices and control the flows precisely.

VIII. Conclusion

This paper extends RoCEv2 to long distance that enables a rethinking of cross-DC applications design. It proposes Bifrost, a downstream-driven lossless flow control for long distance DCI transmission. With Bifrost, RoCEv2 achieves low latency and high throughput while using a minimum buffer space. The paper evaluates the performance of Bifrost in inter-DC scenarios with experiments and simulations against existing solutions under various scenarios. The results show that Bifrost outperforms other flow controls by significantly reducing the average and 99%ile latency of flows. Bifrost is compatible with Ethernet/IP protocols and can support a link of thousands of kilometers. Moreover, the multi-priority queue version, BifrostX, is proposed and evaluated extensively. The results show that BifrostX can achieve the relatively shortest FCT for high, middle-priority traffic, while inducing moderate performance degradation for low-priority traffic.

References

[1] C.-Y. Hong et al., "Achieving high utilization with software-driven WAN," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, 2013, pp. 15–26.

[2] H. Zhang et al., "Guaranteeing deadlines for inter-data center transfers," *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 579–595, Feb. 2017.

[3] Z. Wang, Z. Li, G. Liu, Y. Chen, Q. Wu, and G. Cheng, "Examination of wan traffic characteristics in a large-scale data center network," in *Proc. ACM IMC*, 2021, pp. 1–14.

[4] V. Dukic et al., "Beyond the mega-data center: Networking multi-data center regions," in *Proc. ACM SIGCOMM*, 2020, pp. 765–781.

[5] Y. Sverdlik. (2018). *Facebook Rethinks In-region Data Center Interconnection. DataCenter Knowledge*. [Online]. Available: https://www.datacenterknowledge.com/networks/facebook-rethinks-region-data-center-interconnection

[6] C.-C. Hung, L. Golubchik, and M. Yu, "Scheduling jobs across geo-distributed datacenters," in *Proc. ACM SoCC*, 2015, pp. 111–124.

[7] R. Viswanathan, G. Ananthanarayanan, and A. Akella, "CLARINET: WAN-aware optimization for analytics queries," in *Proc. USENIX OSDI*, 2016, pp. 435–450.

[8] S. Liu, L. Chen, and B. Li, "Siphon: Expediting inter-datacenter coflows in wide-area data analytics," in *Proc. USENIX ATC*, 2018, pp. 507–518.

[9] I. Cano, M. Weimer, D. Mahajan, C. Curino, and G. Matteo Fumarola, "Towards geo-distributed machine learning," 2016, *arXiv:1603.09035*.

[10] A. C. Zhou, B. Shen, Y. Xiao, S. Ibrahim, and B. He, "Cost-aware partitioning for efficient large graph processing in geo-distributed datacenters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 7, pp. 1707–1723, Jul. 2020.

[11] A. C. Zhou, J. Luo, R. Qiu, H. Tan, B. He, and R. Mao, "Adaptive partitioning for large-scale graph analytics in geo-distributed data centers," in *Proc. IEEE 38th Int. Conf. Data Eng. (ICDE)*, May 2022, pp. 2818–2830.

[12] ESnet. (2018). *A Nationwide Platform for Science Discovery*. [Online]. Available: https://www.es.net/engineering-services/the-network

[13] M. T. Michalewicz et al., "InfiniCortex: Present and future invited paper," in *Proc. ACM Int. Conf. Comput. Frontiers*, May 2016, pp. 267–273.

[14] G. Noaje et al., "Infinicortex—From proof-of-concept to production," *Supercomputing Frontiers Innov.*, vol. 4, no. 2, pp. 87–102, 2017.

[15] J. Chrzeszczyk et al., "InfiniCloud 2.0: Distributing high performance computing across continents," *ACM Supercomputing Frontiers Innov.*, vol. 3, no. 2, pp. 54–71, 2016.

[16] Z. Dongfang. (2022). *China Initiates East-Data-West-Computing Project*. ECNS. [Online]. Available: https://www.ecns.cn/news/cns-wire/2022-02-22/detail-ihavwrts3794804.shtml

[17] J. Lee, Z. Tong, K. Achalkar, X. Yuan, and M. Lang, "Enhancing InfiniBand with OpenFlow-style SDN capability," in *Proc. IEEE SC*, 2016, pp. 421–432.

[18] Y. Chen, Y. Lu, and J. Shu, "Scalable RDMA RPC on reliable connection with efficient resource sharing," in *Proc. ACM EuroSys*, Mar. 2019, pp. 1–14.

[19] H. Shi and X. Lu, "INEC: Fast and coherent in-network erasure coding," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2020, pp. 1–17.

[20] T. Li, H. Shi, and X. Lu, "HatRPC: Hint-accelerated thrift RPC over RDMA," in *Proc. Int. Conf. for High Perform. Comput., Netw., Storage Anal.*, Nov. 2021, pp. 1–15.

[21] IT Association. (2004). *Infinibandtm Architecture Specification*. [Online]. Available: http://www.infinibandta.org

[22] (2014). *Infiniband Architecture Specification Volume 1 Release 1.2.1 Annex A17: RoCEv2. InfiniBand Trade Association*. [Online]. Available: https://cw.infinibandta.org/document/dl/7781

[23] A. Dragojević, D. Narayanan, M. Castro, and O. Hodson, "FaRM: Fast remote memory," in *Proc. USENIX NSDI*, 2014, pp. 401–414.

[24] C. Guo et al., "RDMA over commodity Ethernet at scale," in *Proc. ACM SIGCOMM*, Aug. 2016, pp. 202–215.

[25] Z. He et al., "MasQ: RDMA for virtual private cloud," in *Proc. ACM SIGCOMM*, Jul. 2020, pp. 1–14.

[26] Y. Gao et al., "When cloud storage meets RDMA," in *Proc. USENIX NSDI*, 2021, pp. 519–533.

[27] W. Bai et al., "Empowering Azure storage with RDMA," in *Proc. USENIX NSDI*, 2023, pp. 49–67.

[28] X. Zhou and H. Liu, "Pluggable DWDM: Considerations for campus and metro DCI applications," in *Proc. Eur. Conf. Opt. Commun.*, 2016, pp. 1–13.

[29] Z. Fu. (2022). *Understanding China's Eastern Data Western*. Calculation Project. PINGWEST. [Online]. Available: https://en.pingwest.com/a/9940

[30] S. Jain et al., "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.

[31] C.-Y. Hong et al., "B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in Google's software-defined WAN," in *Proc. ACM SIGCOMM*, Aug. 2018, pp. 74–87.

[32] Y. Zhang et al., "BDS: A centralized near-optimal overlay network for inter-datacenter data replication," in *Proc. ACM EuroSys*, Apr. 2018, pp. 1–14.

[33] K. Niedzielewski et al., "Long distance geographically distributed infiniband based computing," *ACM Supercomputing Frontiers Innov.*, vol. 7, no. 2, pp. 24–34, 2020.

[34] (2022). *What is a Federated Network? VMware*. [Online]. Available: https://www.vmware.com/topics/glossary/content/federated-network.html

[35] C-Justice of the European Union. (2015). *The Court of Justice Declares That the Commissions Us Safe Harbour Decision is Invalid*. CURAI. [Online]. Available: https://curia.europa.eu/jcms/upload/docs/application/pdf/2015-10/cp150117en.pdf

[36] A. C. Zhou, Y. Xiao, Y. Gong, B. He, J. Zhai, and R. Mao, "Privacy regulation aware process mapping in geo-distributed cloud data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 8, pp. 1872–1888, Aug. 2019.

[37] Y. Xiao, A. C. Zhou, X. Yang, and B. He, "Privacy-preserving workflow scheduling in geo-distributed data centers," *Future Gener. Comput. Syst.*, vol. 130, pp. 46–58, May 2022.

[38] S.-H. Tseng, S. Agarwal, R. Agarwal, H. Ballani, and A. Tang, "CodedBulk: Inter-datacenter bulk transfers using network coding," in *Proc. USENIX NSDI*, 2021, pp. 15–28.

[39] CES. (2015). *Monitoring, Managing and Troubleshooting Large Scale Networks*. Obsidian Strategics. [Online]. Available: http://ces-nanog64.blogspot.com/2015/06/monitoring-managing-and-troubleshooting.html

[40] Y. Zhu et al., "Congestion control for large-scale RDMA deployments," in *Proc. ACM SIGCOMM*, 2015, pp. 523–536.

[41] Y. Zhu et al., "Packet-level telemetry in large datacenter networks," in *Proc. ACM SIGCOMM*, Aug. 2015, pp. 479–491.

[42] *IEEE Standard for Local and Metropolitan Area Networks—Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks—Amendment 17: Priority-Based Flow Control*, Standard IEEE Standard 802.1Qbb-2011 (Amendment to IEEE Standard 802.1Q-2011 as amended by IEEE Standard 802.1Qbe-2011 and IEEE Standard 802.1Qbc-2011), 2011.

[43] Y. Chen et al., "Swing: Providing long-range lossless RDMA via PFC-relay," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 1, pp. 63–75, Jan. 2023.

[44] IT Association. (2007). *Infiniband Architecture Specification Volume 1*. [Online]. Available: https://www.afs.enea.it/asantoro/V1r1_2_1.Release_12062007.pdf

[45] *Infiniband Range Limitation*, IEEE, USA, 2022.

[46] R. Mittal et al., "Revisiting network support for RDMA," in *Proc. ACM SIGCOMM*, Aug. 2018, pp. 313–326.

[47] B. Liu, X. Guo, W. Kong, T. Liu, R. Dong, and S. Zhang, "Stabilized time transfer via a 1000-km optical fiber link using high-precision delay compensation system," in *Proc. MDPI*, 2022, p. 522.

[48] *Huawei Cloudengine 8851 Switch Datasheet*. Accessed: 2024. [Online]. Available: https://e.huawei.com/en/material/networking/data-center-network/1d1aa7be70054ca9b0f2e165fb98d36b

[49] Linux Rdma. (2022). *Linux-RDMA/Perftest: Infiniband Verbs Performance Tests*. [Online]. Available: https://github.com/linux-rdma/perftest

[50] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM*, 2008, pp. 63–74.

[51] M. Alizadeh et al., "Data center TCP (DCTCP)," in *Proc. ACM SIGCOMM Conf.*, 2010, pp. 63–74.

[52] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proc. ACM SIGCOMM*, 2015, pp. 123–137.

[53] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout, "Homa: A receiver-driven low-latency transport protocol using network priorities," in *Proc. ACM SIGCOMM*, 2018, pp. 221–235.

[54] S. N. Avci, Z. Li, and F. Liu, "Congestion aware priority flow control in data center networks," in *Proc. IFIP Netw. Conf. (IFIP Networking) Workshops*, May 2016, pp. 126–134.

[55] C. Tian et al., "P-PFC: Reducing tail latency with predictive PFC in lossless data center networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1447–1459, Jun. 2020.

[56] P. Goyal, P. Shah, K. Zhao, G. Nikolaidis, M. Alizadeh, and T. E. Anderson, "Backpressure flow control," in *Proc. USENIX NSDI*, 2022, pp. 1–3.

[57] K. Liu et al., "Floodgate: Taming incast in datacenter networks," in *Proc. ACM CoNEXT*, 2021, pp. 30–44.

[58] K. Qian, W. Cheng, T. Zhang, and F. Ren, "Gentle flow control: Avoiding deadlock in lossless networks," in *Proc. ACM SIGCOMM*, 2019, pp. 75–89.

[59] H. Lim, W. Bai, Y. Zhu, Y. Jung, and D. Han, "Towards timeout-less transport in commodity datacenter networks," in *Proc. ACM EuroSys*, 2021, pp. 33–48.

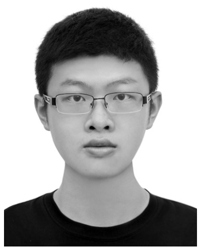
[60] G. Zeng et al., "Congestion control for cross-datacenter networks," *IEEE/ACM Trans. Netw.*, vol. 30, no. 5, pp. 2074–2089, Oct. 2022.

[61] A. Saeed et al., "Annulus: A dual congestion control loop for datacenter and wan traffic aggregates," in *Proc. ACM SIGCOMM*, 2020, pp. 735–749.

[62] G. Zeng, J. Qiu, Y. Yuan, H. Liu, and K. Chen, "FlashPass: Proactive congestion control for shallow-buffered WAN," in *Proc. IEEE 29th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2021, pp. 1–12.

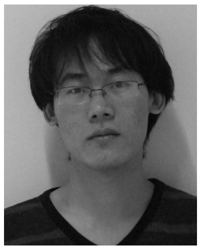
[63] S. Zou, J. Huang, J. Liu, T. Zhang, N. Jiang, and J. Wang, "GTCP: Hybrid congestion control for cross-datacenter networks," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2021, pp. 932–942.

Chengyuan Huang (Member, IEEE) received the B.Eng. and Ph.D. degrees from Beijing University of Posts and Telecommunications in 2015 and 2021, respectively. From 2021 to 2023, he was a Post-Doctoral Researcher with Purple Mountain Laboratories, Nanjing, China. He is currently an Assistant Researcher with the Department of Computer Science and Technology, Nanjing University. His research interests include data center networks, software-defined networking, and distributed systems.



Feiyang Xue (Member, IEEE) received the B.S. degree from the Department of Computer Science and Technology, Nanjing University, China, in 2022, where he is currently pursuing the M.E. degree. His research interests include data center networks and network architecture.

Peiwen Yu received the B.E. degree from the School of the Environment, Nanjing University, in 2020, and the M.E. degree from the Department of Computer Science and Technology, Nanjing University, in 2023. His research interests include data center networks and network architecture.



Xiaoliang Wang received the Ph.D. degree from the Graduate School of Information Sciences, Tohoku University, Japan. From 2010 to 2014, he was an Assistant Professor with the Department of Computer Science and Technology, Nanjing University, China, where he is currently an Associate Professor. His research interests include network systems and optical switching networks.

Yanqing Chen received the B.S. degree from the Department of Computer Science and Engineering, Southeast University, China, in 2019. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Nanjing University, China. His research interests include programmable switches and data center networks.

Tao Wu is currently a Research Engineer with Huawei Technologies Company Ltd. He has created patents that are authorized by China and the USA. His research outcome has been productized and commercialized. His research interests include data center networks and high-performance computing.

Lei Han is currently a Professor with the School of Computer, Nanjing University of Posts and Telecommunications. He has created patents that are authorized by China, the USA, the U.K., Japan, South Korea, and Germany. He has authored papers in research-related international conferences and journals, such as IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, GLOBECOM, MOBISYS, and ICCN. His research interests include computer networks and data center systems.

Zifa Han received the Ph.D. degree in electronic engineering from the City University of Hong Kong in 2016. He is currently a Research Engineer with Huawei Technologies. His research interests include machine learning and computer networks.

Bingquan Wang received the B.S. degree from the Department of Computer Science and Technology, Southeast University, Nanjing, China, and the M.S. degree from the Department of Computer Science and Technology, Nanjing University, Nanjing. He is currently a Software Engineer with Huawei. His research interests include data center networks, distributed systems, and computing networks.

Xiangyu Gong received the master's degree from Southeast University in 2008. He joined Huawei Technologies Company Ltd., in 2008. His research interests include campus networking, data center networks, and distributed systems.

Chen Tian (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the Department of Electronics and Information Engineering, Huazhong University of Science and Technology, China, in 2000, 2003, and 2008, respectively. He is currently a Professor with the State Key Laboratory for Novel Software Technology, Nanjing University, China. His research interests include data center networks, network function virtualization, distributed systems, internet streaming, and urban computing.

Wanchun Dou received the Ph.D. degree in mechanical and electronic engineering from Nanjing University of Science and Technology, China, in 2001. He is currently a Full Professor with the State Key Laboratory for Novel Software Technology, Nanjing University. He has chaired the three National Natural Science Foundation of China projects and published more than 60 research papers in international journals and international conferences. His research interests include workflow, cloud computing, and service computing.

Guihai Chen (Fellow, IEEE) received the B.S. degree in computer software from Nanjing University in 1984, the M.E. degree in computer applications from Southeast University in 1987, and the Ph.D. degree in computer science from The University of Hong Kong, in 1997. He has published more than 350 peer-reviewed papers and more than 200 of them are in well-archived international journals, such as IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE/ACM TRANSACTIONS ON NETWORKING, and *ACM Transactions on Sensor Networks*; and also in well-known conference proceedings, such as HPCA, MOBIHOC, INFOCOM, ICNP, ICDCS, CoNext, and AAAI. His research interests include parallel computing, wireless networks, data centers, peer-to-peer computing, high-performance computer architecture, and data engineering.

Hao Yin is currently a Changjiang Distinguished Professor with Tsinghua University. Previously, he was the Chief Scientist at China's largest content distribution network service provider, ChinaCache (NASDAQ: CCIH), and the Deputy Director of the Ministry of Education-Microsoft Joint Key Laboratory. His research interests include computer networks, big data, and blockchain. He has been honored with awards, including the Second Prize of the National Technology Invention Award and the Second Prize of the National Natural Science Award.