Bifrost: Extending RoCE for Long Distance Inter-DC Links

Peiwen Yu*, Feiyang Xue*, Chen Tian*, Xiaoliang Wang*, Yanqing Chen*, Tao Wu‡,
Lei Han†, Zifa Han‡, Bingquan Wang‡, Xiangyu Gong‡, Wanchun Dou* and Guihai Chen*
*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
†Nanjing University of Posts and Telecommunications, Nanjing, China
‡Huawei Technologies Co., Ltd

2023 IEEE 31st International Conference on Network Protocols (ICNP) | 979-8-3503-0322-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICNP59255.2023.10355634

***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. 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.**

***Index Terms*—datacenter, flow control**

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]–[6]. Besides, many data-intensive applications, *e.g.*, data analytics [7]–[9], machine learning [10], graph processing [11], [12] and super computing [13]–[17] 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 [18]–[21]. Infiniband (IB) [22] and RDMA over Converged Ethernet (RoCEv2) [23] 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) [14]. 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 [24]–[27].

979-8-3503-0322-3/23/$31.00 ©2023 IEEE

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 [28] 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.

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).

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]–[6], [29]. 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 [17], [30]. 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 [28], [31]–[34], data analytics [7]–[9], machine learning [10] and graph processing [11], [12]. Besides, the Energy Sciences Network [13], InfiniCortex [14], [15] and InfiniCloud [16] are built to connect supercomputing centers across the nation and continents for scientific research.

Third, federated cloud computing is proposed for data privacy management and international digital sovereignty requirements [35], [36]. 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 [37]–[39].

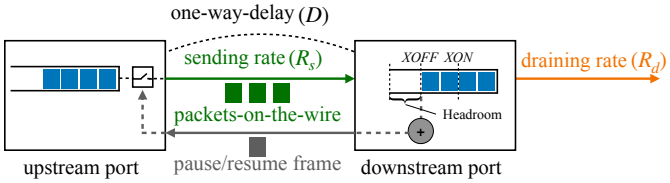


Fig. 1: PFC controls flow rate according to the queue length.

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]–[3], [40], [41]. 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) [22] and RDMA over Converged Ethernet (RoCEv2) [23] are the state-of-the-art RDMA protocols. IB is designed as an independent dedicated protocol and has been deployed 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 [24]–[28], [42].

RoCEv2 requires a lossless network (*i.e.* no switch buffer overflow) to guarantee the performance [43]–[45], which is supported by Priority-based Flow Control (PFC) [46] 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 threshold (*XOFF*) and sends it to the upstream port to pause the flow. When the ingress queue length decreases below a 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 [47] 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 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 [22] 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 [48]. 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.

Besides the lossless flow controls, there are also lossy solutions for RoCEv2. Improved RoCE NIC (IRN) [49] is an approach designed to be compatible with RoCEv2 and performs better than RoCEv2 with PFC within DC. IRN uses an end-to-end flow control with selective retransmission to deal with packet loss. The IRN sender maintains a sliding window implemented in a bitmap for the selective retransmission and also to bound the in-flight packets.

III. Motivation

A. Extending RoCEv2 to cross-DC

Nowadays, with the increasing demand for cross-DC communication, network for DC interconnection should be a new area of concern in the future. The trend of the emerging high performance cross-DC applications implies the underlying support of 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 [14]. 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 [50]. 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 to support high throughput and low latency data transmission. However, the long-haul links with large round-trip-time (RTT) introduce several issues

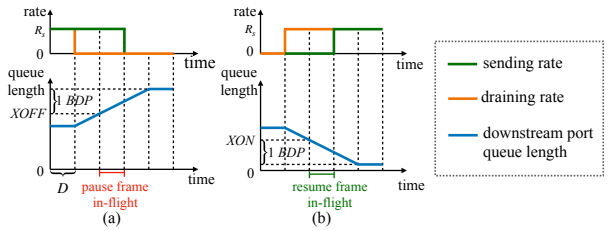


Fig. 2: PFC downstream port queue length in the worst case.

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. The worst case is when the upstream port is sending packets at its line rate while the downstream switch draining rate drops to zero and lasts for a long time, as shown in Figure 2(a). It takes a one-way-delay D for the pause frame to arrive at the upstream and takes 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. 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, which is 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, as shown in Figure 2(b). The packets buffered at downstream queue should be at least $2R_dD$, otherwise, the downstream link will be drained and lead to throughput-loss.

If we configure *XOFF* and *XON* to be the same value, the buffer size of downstream port should be at least $2D(R_s + R_d)$. In fact, R_d could be assumed to be equal to R_s as the typical case. If $R_d > R_s$, it makes no sense to guarantee no throughput-loss. When $R_d < R_s$, the bandwidth of the expensive long-haul link is wasted, which is not the case in actual production. Thus the typical buffer reserved for PFC is 2 BDP. The large buffer space increases the flow latency due

to the large queueing delay.

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 [28] tackles the problem by simply adding off-chip DRAMs and leveraging shared memory between switch ports. 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 [47] 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

Lossy solutions allows packets dropped by switches and the sender retransimits lost packet when particular event is triggered. IRN for RoCEv2 and traditional TCP are the typical lossy solutions.

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 [47], 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.

Furthermore, lossy solutions like IRN and TCP are not suitable for cross-DC transmission by its 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 the cross-DC applications.

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.

A. Insight and challenges

Packets transition through four states when transmitted from the upstream to the downstream, as shown in Figure 3: 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.

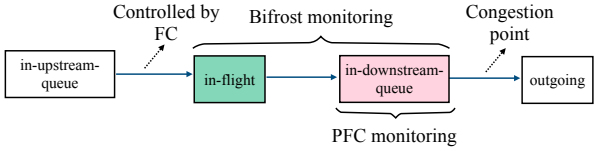


Fig. 3: Packets transition through 4 states when transmitting from upstream to downstream.

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.

We address these challenges as follows. 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 4 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, because there are 0.25 BDP of packets on-the-wire and the pause frames on-the-wire will be transitioned to 0.5 BDP packets D time later. 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

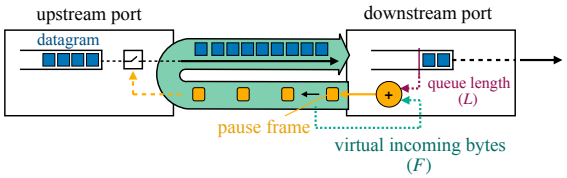


Fig. 4: Bifrost leverages virtual incoming bytes and queue length to make a pause decision to control the flow rate.

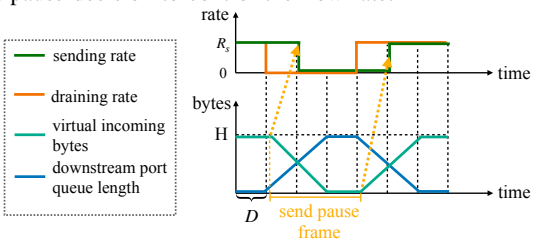


Fig. 5: Bifrost downstream port queue length in the worst case.

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 and then notify the upstream but notifies it in advance. In this way, 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 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 4 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 I summarizes the notations used in the paper.

We use an example to demonstrate the basic workflow as shown in Figure 5. 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. Then congestion occurs, and the draining rate drops to zero, leading to an immediate increase in queue length. Bifrost infers that there could be one BDP of bytes incoming when congestion occurs, so it starts to send a pause frame to the upstream every time slot to stop the sending. 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

TABLE I
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 Δ, 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

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.

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.

At last, 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

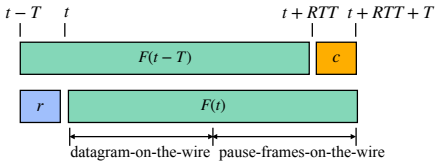


Fig. 6: Relationship between variables in Bifrost at adjacent time slots.

received from time t to time $t + RTT + T$. So here we have the problem of how to update from $F(t-T)$ to $F(t)$. The logic is illustrated in Figure 6. 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: The condition for Bifrost to work is that H should be larger than $\Delta + 2R_sT$. Due to the limited space, we give a brief analysis here and put the full mathematical proof in the online Appendix [51].

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.

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 5 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 in Section III-B. 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 only 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, it will only cost at most 0.05% of the total bandwidth for the pause frame.

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 than the traditional network technology.

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 [52]. 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

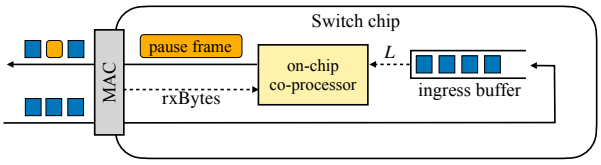


Fig. 7: Bifrost prototype implementation.

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.

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 [53]. 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 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 $\sim$1 μs. Second, it is limited by the buffer available for Bifrost, the one-way-delay, and 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. Evaluation shows that the performance of Bifrost is not sensitive to T (Section VI-B1).

Notice that the PFC pause frame includes eight pause time fields, each of which is two bytes [46]. 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 [46]. 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

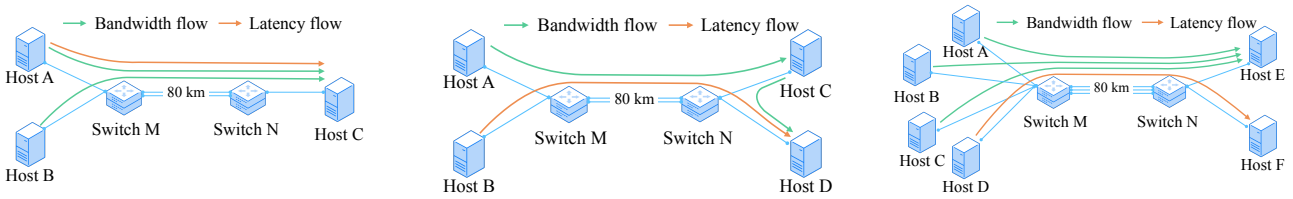
We evaluate the performance of Bifrost with real-world deployment and simulations. We deploy the Bifrost prototype switch we have implemented to a long distance transmission scenario to verify that Bifrost requires less buffer and provides lower latency than PFC. Simulations are conducted to evaluate that Bifrost outperforms PFC, IRN, and IB in cross-DC environments under the production workload.

A. Real deployment

Figure 8 shows the topologies of the deployment. In topology A shown in Figure 8a and topology B shown in Figure 8b, 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 8c, 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.

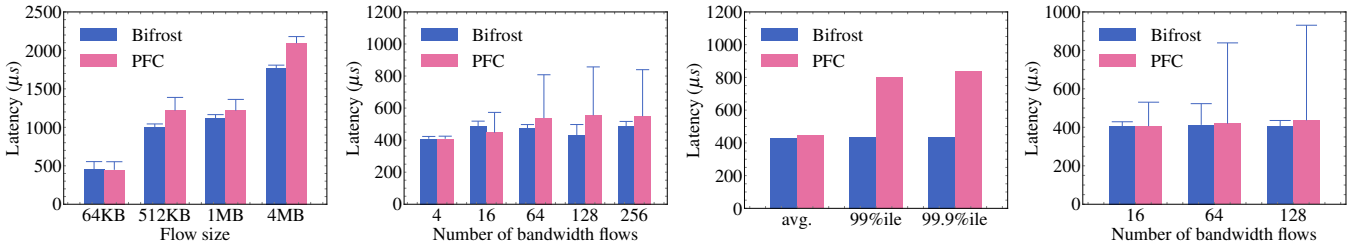
We use perftest [54] to generate various RDMA flows, which includes *bandwidth flows* and *latency flows*. We combine the two traffics to evaluate the performance of PFC and Bifrost. The bandwidth-sensitive flows serve as background traffic, and we evaluate the latency of flows which serve as the interactive 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 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 8a, 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



(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. 8: 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. 9: Avg. latency and 99%ile latency of flows when using PFC and Bifrost in topology A, B, and C

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 8a 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 9a. 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 9b, 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 8b 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 9c, 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 8c 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 9d 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.

B. Simulations

We use ns-3 simulations to evaluate the performance of Bifrost in the cross-DC environment and compare it with PFC, IRN, 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 [55] in DC as shown in Figure 10. Each Top-of-Rack (ToR) switch

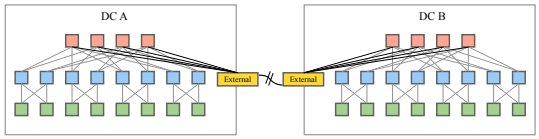


Fig. 10: Topology in the simulation where the external is one DCI switch. Hosts are omitted.

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 of IRN are based on the recommended settings in [49].

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

Traffic loads. The evaluations adopt commonly used DC traffic trace, WebSearch [56] and FB_Hadoop [57]. The WebSearch workload is characterized by small requests and large responses, with 95% of the flows exceeding 1 MB [58]. 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

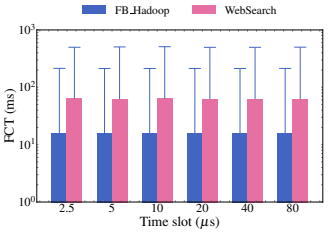


Fig. 11: FCT of flows using Bifrost with different time slots.

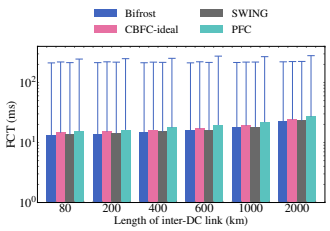
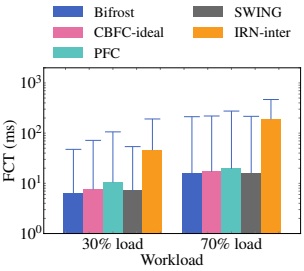
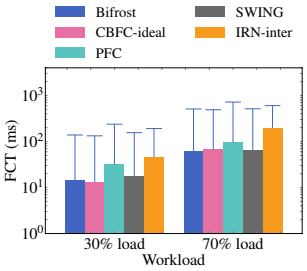


Fig. 12: FCT of flows using Bifrost over different distances.



(a) 16 to 16 FB_Hadoop



(b) 16 to 16 WebSearch

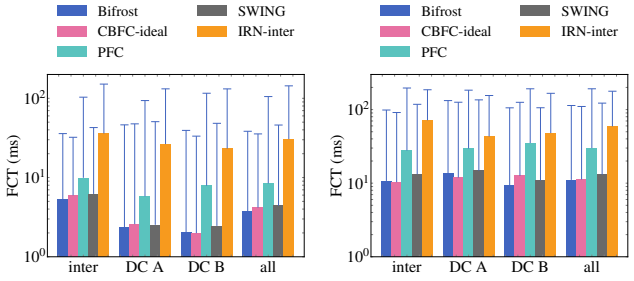
Fig. 13: FCT of flows in 16 to 16 inter-DC traffic pattern under 30% and 70% workloads.

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 10 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 11 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.

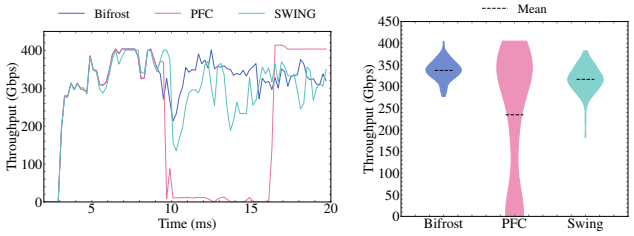
2) FCT under different distances: Figure 12 shows the average FCT and 99%ile FCT with different lengths of the inter-DC link using Bifrost, CBFC-ideal, SWING, and PFC. We omit IRN in the figure because it leads to extremely large FCTs. Flows are generated from 16 servers of DC A to 16 servers in DC B 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 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 13 shows the average and 99%ile FCT of these inter-DC flows. In the 30% load FB_Hadoop traffic in Figure 13a, 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



(a) all-to-all FB_Hadoop (b) all-to-all WebSearch

Fig. 14: FCT of flows in 50% load of all-to-all traffic.



(a) Throughput variation over time (b) Throughput distribution

Fig. 15: Comparison of the long distance link throughput.

CBFC, 40.1% and 55.2% better than PFC, and 86.4% and 75.2% better than IRN.

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 14. Compared with PFC under the 50% workload of FB_Hadoop (Figure 14a), 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%.

5) 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 15a, the throughput of Bifrost is steadier than PFC and SWING because it controls the flow with foresight and in a fine-grained manner. Figure 15b 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.

VII. Related Work

Flow controls for DC network. Prior works mainly focused on the side effects caused by PFC and proposed solutions. CaPFC [59] modifies PFC by using ingress queue and egress queue statistics to react earlier. P-PFC [60] leverages the change rate of queue length to trigger PFC pause instead of using a fixed threshold. BFC [61] proposes a per-flow, hop-by-hop flow control and Floodgate [62] 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 [63], 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 [64] 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 [65] strategically integrates both ECN and delay signals for cross-DC congestion control. Annulus [66] leverages the increase of queueing delay at the ToR switch to early-detect the congestion over the WAN and controls the sending rate. FlashPass [67] 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 [68] 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 actual deployment 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.

Acknowledgment

The authors gratefully acknowledge the shepherd Yang Zhang and the anonymous reviewers for their constructive comments. The project is partially supported by the National Key Research and Development Program of China under Grant Number 2022YFB2702803 and the National Natural Science Foundation of China under Grant Numbers 62325205, 62072228, and 62172204.

REFERENCES

[1] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, 2013.

[2] H. Zhang, K. Chen, W. Bai, D. Han, C. Tian, H. Wang, H. Guan, and M. Zhang, "Guaranteeing deadlines for inter-data center transfers," *IEEE/ACM transactions on networking*, 2016.

[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 *Proceedings of the 21st ACM Internet Measurement Conference*, 2021.

[4] V. Dukic, G. Khanna, C. Gkantsidis, T. Karagiannis, F. Parmigiani, A. Singla, M. Filer, J. L. Cox, A. Ptasznik, N. Harland, W. Saunders, and C. Belady, "Beyond the mega-data center: Networking multi-data center regions," in *Proceedings of the ACM SIGCOMM 2020 Conference on Data Communication*, 2020.

[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] (2016) Aws re:invent 2016: Enterprise fundamentals: Design your account and vpc architecture for enterprise operating models (ent203). [Online]. Available: https://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-enterprise-fundamentals-design-your-account-and-vpc-architecture-for-enterprise-operating-models-ent203

[7] C.-C. Hung, L. Golubchik, and M. Yu, "Scheduling jobs across geo-distributed datacenters," in *Proceedings of the Sixth ACM Symposium on Cloud Computing*, 2015.

[8] R. Viswanathan, G. Ananthanarayanan, and A. Akella, "CLARINET: WAN-Aware optimization for analytics queries," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016.

[9] S. Liu, L. Chen, and B. Li, "Siphon: Expediting Inter-Datacenter coflows in Wide-Area data analytics," in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, 2018.

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

[11] 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 Transactions on Parallel and Distributed Systems*, 2020.

[12] 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 *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2022.

[13] ESnet. (2018) A nationwide platform for science discovery. [Online]. Available: https://www.es.net/engineering-services/the-network

[14] M. T. Michalewicz, T. G. Lian, L. Seng, J. Low, D. Southwell, J. Gunthorpe, G. Noaje, D. Chien, Y. Poppe, J. Chrzundefinedszczyk, A. Howard, T. T. Wee, and L. Sing-Wu, "Infinicortex: Present and future invited paper," in *Proceedings of the ACM International Conference on Computing Frontiers*, 2016.

[15] G. Noaje, A. Davis, J. Low, S. Lim, G. L. Tan, L. Orlowski, D. Chien, S.-W. Liou, T. W. Tan, Y. Poppe, K. Ban Hon Kim, A. Howard, D. Southwell, J. Gunthorpe, and M. Michalewicz, "Infinicortex - from proof-of-concept to production," *Supercomputing Frontiers and Innovations*, 2017.

[16] J. Chrzeszczyk, A. Howard, A. Chrzeszczyk, B. Swift, P. Davis, J. Low, T. W. Tan, and K. Ban, "Infinicloud 2.0: distributing high performance computing across continents." *Supercomputing Frontiers and Innovations*, 2016. [Online]. Available: https://superfri.org/index.php/superfri/article/view/95

[17] 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

[18] J. Lee, Z. Tong, K. Achalkar, X. Yuan, and M. Lang, "Enhancing infiniband with openflow-style sdn capability," in *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2016.

[19] Y. Chen, Y. Lu, and J. Shu, "Scalable rdma rpc on reliable connection with efficient resource sharing," in *Proceedings of the Fourteenth EuroSys Conference 2019*, 2019.

[20] H. Shi and X. Lu, "Inec: Fast and coherent in-network erasure coding," in *SC '20: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020.

[21] T. Li, H. Shi, and X. Lu, "Hatrpc: Hint-accelerated thrift rpc over rdma," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021.

[22] I. T. Association *et al.* (2004) Infinibandtm architecture specification. [Online]. Available: http://www.infinibandta.org

[23] (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

[24] A. Dragojević, D. Narayanan, M. Castro, and O. Hodson, "FaRM: Fast remote memory," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, 2014.

[25] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, "Rdma over commodity ethernet at scale," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016.

[26] Z. He, D. Wang, B. Fu, K. Tan, B. Hua, Z.-L. Zhang, and K. Zheng, "Masq: Rdma for virtual private cloud," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2020.

[27] Y. Gao, Q. Li, L. Tang, Y. Xi, P. Zhang, W. Peng, B. Li, Y. Wu, S. Liu, L. Yan, F. Feng, Y. Zhuang, F. Liu, P. Liu, X. Liu, Z. Wu, J. Wu, Z. Cao, C. Tian, J. Wu, J. Zhu, H. Wang, D. Cai, and J. Wu, "When cloud storage meets RDMA," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, 2021.

[28] W. Bai, S. S. Abdeen, A. Agrawal, K. K. Attre, P. Bahl, A. Bhagat, G. Bhaskara, T. Brokhman, L. Cao, A. Cheema, R. Chow, J. Cohen, M. Elhaddad, V. Ette, I. Figlin, D. Firestone, M. George, I. German, L. Ghai, E. Green, A. Greenberg, M. Gupta, R. Haagens, M. Hendel, R. Howlader, N. John, J. Johnstone, T. Jolly, G. Kramer, D. Kruse, A. Kumar, E. Lan, I. Lee, A. Levy, M. Lipshteyn, X. Liu, C. Liu, G. Lu, Y. Lu, X. Lu, V. Makhervaks, U. Malashanka, D. A. Maltz, I. Marinos, R. Mehta, S. Murthi, A. Namdhari, A. Ogus, J. Padhye, M. Pandya, D. Phillips, A. Power, S. Puri, S. Raindel, J. Rhee, A. Russo, M. Sah, A. Sheriff, C. Sparacino, A. Srivastava, W. Sun, N. Swanson, F. Tian, L. Tomczyk, V. Vadlamuri, A. Wolman, Y. Xie, J. Yom, L. Yuan, Y. Zhang, and B. Zill, "Empowering azure storage with RDMA," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023.

[29] X. Zhou and H. Liu, "Pluggable dwdm: Considerations for campus and metro dci applications," in *Proc. Eur. Conf. Opt. Commun*, 2016.

[30] Z. Fu. (2022) Understanding china's "eastern data western" calculation project. PINGWEST. [Online]. Available: https://en.pingwest.com/a/9940

[31] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, 2013.

[32] A. Kumar, S. Jain, U. Naik, A. Raghuraman, N. Kasinadhuni, E. C. Zermeno, C. S. Gunn, J. Ai, B. Carlin, M. Amarandei-Stavila, M. Robin, A. Siganporia, S. Stuart, and A. Vahdat, "BwE: Flexible, hierarchical bandwidth allocation for WAN distributed computing," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015.

[33] C.-Y. Hong, S. Mandal, M. Al-Fares, M. Zhu, R. Alimi, K. N. B., C. Bhagat, S. Jain, J. Kaimal, S. Liang, K. Mendelev, S. Padgett, F. Rabe, S. Ray, M. Tewari, M. Tierney, M. Zahn, J. Zolla, J. Ong, and A. Vahdat, "B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018.

[34] Y. Zhang, J. Jiang, K. Xu, X. Nie, M. J. Reed, H. Wang, G. Yao, M. Zhang, and K. Chen, "BDS: A centralized near-optimal overlay network for inter-datacenter data replication," in *Proceedings of the Thirteenth EuroSys Conference 2018*, 2018.

[35] K. Niedzielewski, M. Semeniuk, J. Skomiał, J. Proficz, P. Sumioka, B. Pliszka, and M. Michalewicz, "Long distance geographically distributed infiniband based computing," *Supercomputing Frontiers and Innovations*, 2020.

[36] (2022) What is a federated network? VMware. [Online]. Available: https://www.vmware.com/topics/glossary/content/federated-network.html

[37] C. of Justice of the European Union. (2015) The court of justice declares that the commissions us safe harbour decision is invalid. CURIA. [Online]. Available: https://curia.europa.eu/jcms/upload/docs/application/pdf/2015-10/cp150117en.pdf

[38] 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 Transactions on Parallel and Distributed Systems*, 2019.

[39] Y. Xiao, A. C. Zhou, X. Yang, and B. He, "Privacy-preserving workflow scheduling in geo-distributed data centers," *Future Generation Computer Systems*, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X21004854

[40] L. Luo, H. Yu, K.-T. Foerster, M. Noormohammadpour, and S. Schmid, "Inter-datacenter bulk transfers: Trends and challenges," *IEEE Network*, 2020.

[41] S.-H. Tseng, S. Agarwal, R. Agarwal, H. Ballani, and A. Tang, "CodedBulk: Inter-Datacenter bulk transfers using network coding," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, 2021.

[42] Q. Li, Q. Xiang, Y. Wang, H. Song, R. Wen, W. Yao, Y. Dong, S. Zhao, S. Huang, Z. Zhu, H. Wang, S. Liu, L. Chen, Z. Wu, H. Qiu, D. Liu, G. Tian, C. Han, S. Liu, Y. Wu, Z. Luo, Y. Shao, J. Wu, Z. Cao, Z. Wu, J. Zhu, J. Wu, J. Shu, and J. Wu, "More than capacity: Performance-oriented evolution of pangu in alibaba," in *21st USENIX Conference on File and Storage Technologies (FAST 23)*, 2023.

[43] 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

[44] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion control for large-scale rdma deployments," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015.

[45] Y. Zhu, N. Kang, J. Cao, A. Greenberg, G. Lu, R. Mahajan, D. Maltz, L. Yuan, M. Zhang, B. Y. Zhao, and H. Zheng, "Packet-level telemetry in large datacenter networks," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015.

[46] "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," *IEEE Std 802.1Qbb-2011 (Amendment to IEEE Std 802.1Q-2011 as amended by IEEE Std 802.1Qbe-2011 and IEEE Std 802.1Qbc-2011)*, 2011.

[47] Y. Chen, C. Tian, J. Dong, S. Feng, X. Zhang, C. Liu, P. Yu, N. Xia, W. Dou, and G. Chen, "Swing: Providing long-range lossless rdma via pfc-relay," *IEEE Transactions on Parallel and Distributed Systems*, 2023.

[48] I. T. Association. (2007) Infiniband architecture specification volume 1. InfiniBand Trade Association. [Online]. Available: https://www.afs.enea.it/asantoro/V1r1_2_1.Release_12062007.pdf

[49] R. Mittal, A. Shpiner, A. Panda, E. Zahavi, A. Krishnamurthy, S. Ratnasamy, and S. Shenker, "Revisiting network support for rdma," in *Proceedings of the ACM SIGCOMM 2018 Conference on Data Communication*, 2018.

[50] (2022) Infiniband range limitation.

[51] P. Yu. (2022) Bifrost online appendix. [Online]. Available: https://github.com/Pavinberg/Bifrost/blob/main/bifrost-appendix.pdf

[52] 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 *Photonics*, 2022.

[53] Huawei cloudengine 8851 switch datasheet. https://e.huawei.com/en/material/networking/data-center-network/1d1aa7be70054ca9b0f2e165fb98d36b.

[54] linux rdma. (2022) linux-rdma/perftest: Infiniband verbs performance tests. [Online]. Available: https://github.com/linux-rdma/perftest

[55] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, 2008.

[56] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," in *Proceedings of the ACM SIGCOMM 2010 Conference*, 2010.

[57] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015.

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

[59] S. N. Avci, Z. Li, and F. Liu, "Congestion aware priority flow control in data center networks," in *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, 2016.

[60] C. Tian, B. Li, L. Qin, J. Zheng, J. Yang, W. Wang, G. Chen, and W. Dou, "P-pfc: Reducing tail latency with predictive pfc in lossless data center networks," *IEEE Transactions on Parallel and Distributed Systems*, 2020.

[61] P. Goyal, P. Shah, K. Zhao, G. Nikolaidis, M. Alizadeh, and T. E. Anderson, "Backpressure flow control," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022.

[62] K. Liu, C. Tian, Q. Wang, H. Zheng, P. Yu, W. Sun, Y. Xu, K. Meng, L. Han, J. Fu, W. Dou, and G. Chen, "Floodgate: Taming incast in datacenter networks," in *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, 2021.

[63] K. Qian, W. Cheng, T. Zhang, and F. Ren, "Gentle flow control: Avoiding deadlock in lossless networks," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019.

[64] H. Lim, W. Bai, Y. Zhu, Y. Jung, and D. Han, "Towards timeout-less transport in commodity datacenter networks," in *Proceedings of the Fourteenth EuroSys Conference 2021*, 2021.

[65] G. Zeng, W. Bai, G. Chen, K. Chen, D. Han, Y. Zhu, and L. Cui, "Congestion control for cross-datacenter networks," *IEEE/ACM Transactions on Networking*, 2022.

[66] A. Saeed, V. Gupta, P. Goyal, M. Sharif, R. Pan, M. Ammar, E. Zegura, K. Jang, M. Alizadeh, A. Kabbani, and A. Vahdat, "Annulus: A dual congestion control loop for datacenter and wan traffic aggregates," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2020.

[67] G. Zeng, J. Qiu, Y. Yuan, H. Liu, and K. Chen, "Flashpass: Proactive congestion control for shallow-buffered wan," in *2021 IEEE 29th International Conference on Network Protocols (ICNP)*, 2021.

[68] S. Zou, J. Huang, J. Liu, T. Zhang, N. Jiang, and J. Wang, "Gtcp: Hybrid congestion control for cross-datacenter networks," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, 2021.