Enhancing Load Balancing With In-Network Recirculation to Prevent Packet Reordering in Lossless Data Centers

Jinbin Hu, *Member, IEEE*, Yi He, Wangqing Luo, Jiawei Huang, and Jin Wang, *Senior Member, IEEE*

***Abstract*— Many existing load balancing mechanisms work effectively in lossy datacenter networks (DCNs), but they suffer from serious packet reordering in lossless Ethernet DCNs deployed with the hop-by-hop Priority-based Flow Control (PFC). The key reason is that the prior solutions are not able to perceive PFC triggering correctly and in a timely manner when making load balancing decisions. Once the forwarding path pauses transmission due to PFC triggering, the packets allocated on it are blocked, inevitably leading to out-of-order packets and retransmission. In this paper, we present an Reordering-robust Load Balancing (RLB) scheme with PFC prediction in lossless DCNs. At its heart, RLB leverages the derivative of ingress queue length to predict PFC triggering and proactively notifies the upstream switches to choose an appropriate rerouting path or perform packet recirculation to avoid reordering. Furthermore, under switch failure scenarios, RLB adjusts the recirculation threshold adaptively to mitigate the risk of packets over-recirculation. We have implemented RLB in the hardware programmable switch. As a building block for existing load balancing mechanisms, we have integrated RLB into Presto, LetFlow, Hermes and DRILL. The evaluation results show that the RLB-enhanced solutions deliver significant performance by avoiding packet reordering. For example, it reduces the 99^{th} percentile flow completion time (FCT) by up to 72%, 67%, 58% and 54% over DRILL, Presto, LetFlow and Hermes, respectively.**

***Index Terms*— Data center, lossless networks, load balancing, reordering.**

I. Introduction

Modern datacenter applications such as Online Data Intensive (OLDI) services [2], Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE) [3] and Non-Volatile Memory Express (NVMe) over Fabrics [4] require low-latency and lossless transmission to meet the increasing demands from customers. Even a single packet loss can greatly increase flow completion time (FCT) [4], [5]. To prevent buffer overflow, PFC mechanism is widely deployed in the converged enhanced Ethernet of datacenter networks [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. PFC pauses the related upstream ports when the ingress queue length reaches a specified PFC threshold, and resumes transmission after the buffer occupancy decreases to the PFC threshold [10].

Datacenter networks enable rich parallel paths between host pairs and balance traffic among them to deliver high throughput. However, packets transmitted through multiple paths with different delay may arrive at the receiver out of order. Furthermore, limited by the small on-chip memory at network interface controller (NIC), lossless datacenter networks employ simple go-back-N retransmission scheme to deal with packet reordering. In the go-back-N scheme, the receiver's NIC discards the out-of-order packet and asks the sender to retransmit all packets that are sent after the last acknowledged packet, resulting in significant performance degradation.

In recent years, many load balancing schemes have been proposed to address the packet reordering problem [14], [15], [16], [17], [18]. Despite much progress in lossy datacenter networks, prior solutions cannot effectively avoid out-of-order packets in lossless datacenter networks. The key reason is that the existing load balancing schemes cannot perceive hop-by-hop PFC pausing correctly and in a timely manner when they make rerouting decisions in PFC-enabled datacenter networks. They have no consideration of the possibility that the selected forwarding path may be paused by PFC mechanism. Once the selected path is paused by PFC due to transient congestion, the later-sent packets may arrive before the earlier-sent ones at the receiver, inevitably leading to packet reordering.

Given the above inefficiencies, we ask the following question: can we design a building block for the existing load balancing mechanisms to avoid packet reordering in lossless DCN? In this paper, we present RLB to answer this question affirmatively.

The key contribution of this work is to make the existing load balancing schemes still perform effective rerouting without out-of-order packets in lossless DCNs. We present RLB, a building block for existing load balancing schemes to eliminate packet reordering. RLB realizes its design goal by

Manuscript received 12 October 2023; revised 21 March 2024 and 6 May 2024; accepted 16 May 2024; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor P. P. C. Lee. This work was supported in part by the National Natural Science Foundation of China under Grant 62102046, Grant 62132022, and Grant 62072056; in part by the Natural Science Foundation of Hunan Province under Grant 2024JJ3017, Grant 2022JJ30618, and Grant 2021JJ30867; in part by the Scientific Research Fund of Hunan Provincial Education Department under Grant 22B0300; and in part by Hunan Provincial Key Research and Development Program under Grant 2022GK2019. A preliminary version of this paper appears in ACM ICPP [1], Salt Lake City, Utah, USA, August, 2023 [DOI: 10.1145/3605573.3605617]. *(Corresponding author: Jin Wang.)*

Jinbin Hu, Yi He, and Wangqing Luo are with the School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410004, China (e-mail: jinbinhu@csust.edu.cn; heyi@stu.csust.edu.cn; luowangqing@stu.csust.edu.cn).

Jiawei Huang is with the School of Computer and Communication Engineering, Central South University, Changsha 410004, China (e-mail: jiaweihuang@csu.edu.cn).

Jin Wang is with the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China (e-mail: jinwang@hnust.edu.cn).

Digital Object Identifier 10.1109/TNET.2024.3403671

rerouting or recirculation. Packet recirculation is a mechanism that sends the deparsed packet at the end of egress pipeline to the ingress pipeline to repeat ingress processing without needing to replicate packet. The Portable Switch Architecture (PSA) implementation supports packet recirculation for further processing [19]. Before making the final forwarding decision, RLB allows the existing load balancing mechanisms to deliberately consider whether the initial path selected by its own rerouting algorithm is potentially paused by PFC.

Specifically, RLB predicts PFC for each path by measuring the derivative of ingress queue length. Once PFC is predicted to be triggered on a path, a congestion notification message (CNM) is generated as a warning message and directly sent to the upstream switch (§ III-B.1). Then, the upstream switch considers how to choose appropriate forwarding paths for the arriving packets to avoid packet reordering (§ III-B.2). If the difference in delays between the optimal and suboptimal paths selected by the load balancing mechanism is large, the upstream switch prefers to recirculate the packets from its egress port to its ingress port to let the packets stay on the switch for a little while to avoid reordering. Then the packets reconsider whether to choose the initial optimal path. On the contrary, if the difference in delays between the optimal and suboptimal paths is small, the arriving packets are rerouted to the suboptimal path with no PFC warning. Furthermore, RLB adjusts the recirculation threshold dynamically by considering the recirculation delay and buffer occupancy to avoid packet over-recirculation under switch failure scenarios.

We have implemented an RLB prototype in a hardware programmable switch (§ V), and built a small-scale testbed to evaluate the effectiveness of RLB (§ VI-A). RLB is architecturally compatible with all existing load balancing schemes. We have integrated RLB into four existing load balancing schemes (i.e., DRILL, Presto, LetFlow and Hermes). The testbed evaluation and large-scale NS-3 simulation results indicate that RLB-enhanced solutions achieve significantly better performance than vanilla load balancing schemes. For example, under the realistic Web Search workload [21], [22], RLB+DRILL, RLB+Presto, RLB+LetFlow and RLB+Hermes reduce tail FCT by up to 72%, 67%, 58% and 54% compared to DRILL, Presto, LetFlow and Hermes, respectively.

The rest of this paper is organized as follows. Section II introduces background and motivation. Section III and IV describes the design of our solution in detail. Section VI evaluates the performance of RLB. Section VII presents the related work, and Section VIII concludes this paper.

II. Background and Motivation

A. Background

1) PFC Mechanism: PFC is a hop-by-hop link-layer flow control mechanism, which is intended to eliminate packet loss due to congestion [3], [11], [23]. Fig. 1 shows the architecture of a typical PFC-enabled switch with shared memory. All packets are buffered in the shared memory pool. Each packet is counted in both ingress and egress queues. Once the ingress queue length exceeds the specified PFC threshold checked by memory management unit (MMU), the switch sends PFC

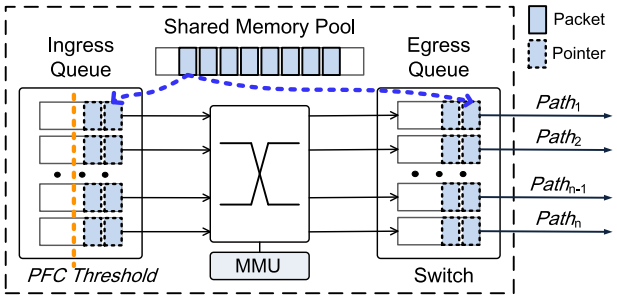


Fig. 1. Architecture of shared memory switch.

PAUSE message to the upstream switch. Then the related upstream ports (or priorities) stop data transmission. When the given pause duration specified in the PAUSE message expires or a RESUME message is received once the queue length decreases to the PFC threshold, the upstream ports resume data transmission.

However, PFC is a coarse-grained mechanism, which operates at port level and cannot distinguish between flows. This can potentially cause head-of-line (HoL) blocking and congestion spreading, etc., leading to serious performance damage for individual flows [3]. Specifically, suppose multiple flows from the same ingress at a switch, if one flow's egress port is paused by PFC from downstream switches, the other flows targeting different egress ports are also blocked, leading to HoL blocking. In addition, this back-pressure behavior potentially causes congestion spreading. Even if end-to-end congestion control protocols such as DCQCN [3], TIMELY [12], MP-RDMA [10], Swift [13] and PCN [4] are employed, PFC is inevitably triggered especially under the transient congestion due to uncontrollable bursty traffic.

2) Impact of Out-of-Order Packets in Lossless DCN: Although PFC mechanism can prevent packet loss due to buffer overflow, retransmission mechanism is still needed to recover the lost packets due to other reasons such as switch configuration errors, link failures or frame check sequence errors [20]. Since the memory of NIC is small, RoCEv2 protocol widely deployed in lossless DCN simply supports go-back-N algorithm. The retransmission starts from the first dropped packet, resulting in a waste of bandwidth. When an out-of-order packet arrives at the receiver, the receiver assumes that the next expected packet has been lost. Then, the receiver generates a negative acknowledgement (NAK) and sends it back to the sender to trigger retransmission. The work [3] indicates that the network throughput will decrease to nearly zero once the packet loss rate exceeds 1%, which seriously degrades the application performance.

3) Existing Load Balancing Schemes in Lossy DCN: A rich body of work [14], [15], [16], [17], [18], [24], [25] has emerged on better load balancing for lossy DCN. However, path diversity due to congestion and asymmetry can easily cause packet reordering for finer switching granularity schemes. In recent years, many proposals strive to reduce out-of-order packets to improve load balancing. Presto [15] selects paths in round-robin fashion for fixed-sized flowcells and re-assembles out-of-order flowcells back in order by

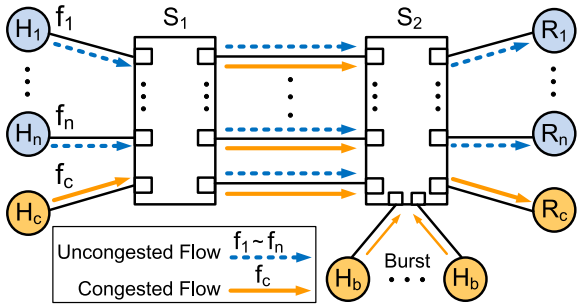


Fig. 2. Typical network scenario in DCN. The uncongested flows $f_1 \sim f_n$ that are not responsible for congestion are sent from the source hosts $H_1 \sim H_n$ to the destination hosts $R_1 \sim R_n$ over multiple parallel paths, respectively. The congested flow f_c that is really responsible for congestion from H_c and the bursty flows from H_b are sent to the same receiver R_c.

utilizing reordering buffer. CONGA [14] and LetFlow [16] balance traffic at a flowlet granularity. If the inactive gap between flowlets is larger than the maximum difference of path delay, flowlets can be rerouted without packet reordering. Hermes [17] is resilient to network uncertainties. It makes deliberate rerouting decisions only if they bring performance gains. DRILL [18] performs micro load balancing at packet granularity in the divided symmetrical area to reduce disorder packets.

Although the above load balancing solutions work effectively in lossy DCN, they cannot achieve good performance in lossless DCN due to packet reordering. The reason is that they cannot perceive PFC pausing correctly and in a timely manner when choosing the optimal forwarding path. This motivates us to design a new building block to guarantee no out-of-order packets for the existing load balancing schemes.

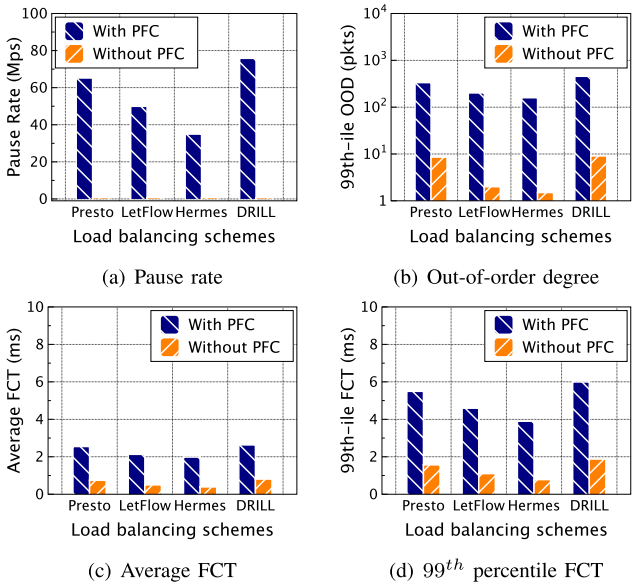
B. Motivation

Through an empirical analysis on the representative load balancing schemes with different switching granularities, we demonstrate how PFC mechanism affects the existing load balancing schemes.

1) Why PFC Leads to Packet Reordering?:

Since the existing rerouting algorithms cannot perceive PFC pausing correctly and in a timely manner, they pick paths in a PFC-oblivious manner for fine-grained granularities such as packets, flowcells or flowlets, resulting in poor performance in the lossless DCN. To concretely show the impact of PFC on load balancing performance, we investigate how the typical load balancing schemes (i.e., Presto [15], Letflow [16], Hermes [17] and DRILL [18]) work with and without PFC under a common scenario in lossless DCN.

Without loss of generality, as shown in Fig. 2, senders ($H_1 \sim H_n$, H_c) and receivers ($R_1 \sim R_n$, R_c) connect to the corresponding leaf switches (S_1, S_2), respectively. There are 40 equal-cost paths between the two leaf switches. In addition, multiple senders (in set H_b) connect to the leaf switch S_2. The capacity of each link is 40Gbps, and the link delay is 2μs. The switch buffer is set to 9MB. The PFC threshold at each ingress port is set to 256KB. The congestion control protocol DCQCN is enabled and the parameters are set to the default values recommended in [3]. We conduct NS-3 simulations to



(a) Pause rate (b) Out-of-order degree

(c) Average FCT (d) 99^{th} percentile FCT

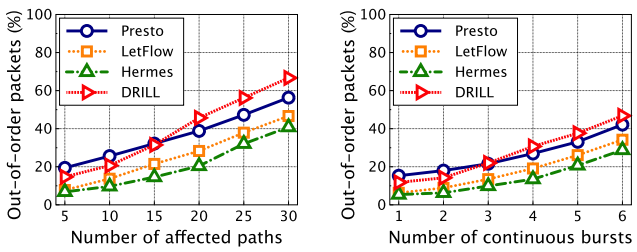
Fig. 3. Performance of four load balancing schemes with and without PFC mechanism.

investigate the impact of PFC on packet reordering for four typical load balancing schemes with finer granularity.

For this test, 100 servers $H_1 \sim H_{100}$ generate dynamic traffic $f_1 \sim f_{100}$ according to the realistic Web Search workload [26], [27], [28], [29] with an average flow size of 1.6MB. Then each server in H_b generates 40 bursty flows with 64KB at line rate and sends them to the receiver R_c. Server H_c starts a long flow with 250MB as a congested flow f_c to R_c. By default, two continuous bursts are generated and f_c is transmitted over 5 parallel paths. In the first case, PFC mechanism is enabled, $f_1 \sim f_{100}$ can choose all of the parallel paths. Thus, they are likely to select the same paths as the congested flow f_c, and these paths have the risk of being paused by PFC due to bursty traffic. In the second case, PFC mechanism is not enabled, even though $f_1 \sim f_{100}$ are transmitted on the same paths as the congested flow f_c, $f_1 \sim f_{100}$ will not be affected by PFC pausing.

We measure the pause rate of PFC, 99^{th} percentile of out-of-order degree (OOD), average FCT and tail FCT. OOD is the difference between the sequence numbers of an out-of-order packet and the expected one. Fig. 3 compares these performances of four load balancing schemes with and without PFC mechanism under two scenarios.

Fig. 3 (a) shows that PFC is triggered under all load balancing schemes. Please note that the PFC pausing rate is zero when the PFC is not enabled. Fig. 3 (b) shows that the 99^{th} percentile of OODs under PFC pausing are larger than that of the second case without PFC pausing under four load balancing schemes. The reason is that the earlier-sent packets are blocked on the paths paused by PFC and then arrive later than the later-sent packets from a same flow, leading to packet reordering and retransmission. Specifically, Presto [15] and LetFlow [16] inevitably choose the rerouting path with PFC PAUSE by randomly choosing rerouting path. The end-to-end Explicit Congestion Notification (ECN) and Round Trip Time (RTT) signals employed in Hermes [17] are difficult to



(a) Increasing affected paths (b) Increasing bursts

Fig. 4. Serious negative impact of PFC on load balancing with the increase number of affected paths and continuous bursts.

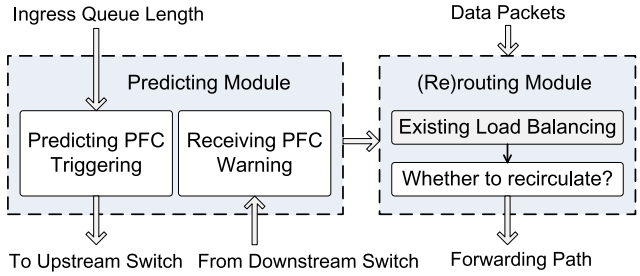


Fig. 5. RLB overview.

feedback hop-by-hop PFC pausing in time. The local queue length used by DRILL [18] cannot sense the PFC pausing in time on the remote downstream switches.

Meanwhile, the out-of-order packets increase the average and tail latency as shown in Fig. 3 (c) and Fig. 3 (d). Compared with the case under PFC Pausing, the average FCT (AFCT) and 99^{th} percentile of FCT for Presto reduces by up to 68% and 72%, respectively. In addition, DRILL with the finest switching granularity suffers from the most serious reordering problem, because more rerouting increases the probability of selecting the paths that are potentially paused by PFC. Therefore, PFC PAUSE mechanism cripples the resilience of load balancing schemes against asymmetric networks.

2) Why Packet Reordering Becomes More Serious?: To show the serious negative impact of PFC on load balancing schemes, we further measure the ratio of out-of-order packets with the increased number of paused paths and continuous bursts. We control the number of affected paths that are paused by PFC through controlling the number of multiple paths that can be chosen by the congested flows. Worse still, on the one hand, as the number of parallel paths paused by PFC increases, the uncongested flows have a higher probability to choose the adversely affected paths. As a result, the existing load balancing rerouting algorithms are not able to guarantee no packet reordering. On the other hand, when the number of continuous bursts increases, the numbers of PFC triggering and paused paths increase correspondingly, resulting in more serious packet reordering. As shown in Fig. 4 (a) and Fig. 4 (b), the ratios of reordering packets dramatically increase with increasing affected paths and bursts, respectively. Therefore, four load balancing schemes perform inferior facing serious packet reordering. For DRILL, the ratio of out-of-order packets under the case of 30 affected paths is 76% higher than that of 5 affected paths.

III. The RLB Design

In this section, we first present the design overview of RLB, which aims to avoid packet reordering for the existing load balancing schemes in lossless DCN. Then we describe the design of RLB in detail, including predicting module and rerouting module.

A. Design Overview

The key point of RLB is to determine whether to recirculate or reroute packets based on the PFC warning and path delay. Specifically, on the one hand, if the initial optimal path chosen by the existing load balancing scheme will not be paused by PFC, RLB forwards packets to this path directly. On the other hand, the initial selected best path is likely to be paused by PFC. If the packets are forwarded on this initial path aggressively, they may be blocked and suffer from large queueing delay, resulting in out-of-order packets. On the contrary, if the packets give up this best path conservatively, they may waste an opportunity to transmit quickly from an uncongested path, because data transmission may be resumed soon by PFC RESUME message in case of transient congestion. Fig. 5 overviews RLB, which mainly contains two modules: predicting module and rerouting module.

- **Predicting Module (§ III-B.1):** RLB predicts whether PFC will be triggered according to the increase rate of ingress queue length rather than the whole buffer occupancy. Once there is a risk of triggering PFC, it directly sends a congestion notification message (CNM) to the upstream switch as PFC warning, which indicates that the corresponding path is likely to be paused by PFC. At the same time, the PFC warning message received from the downstream switch will be sent to the rerouting module for making load balancing decision.
- **Rerouting Module (§ III-B.2):** The key for RLB to avoid packet reordering is to consider whether the initial best path selected by the existing load balancing schemes will be paused by PFC. If so, RLB decides whether to recirculate or reroute packets according to path delay. After recirculating, the packets have a new chance to decide whether the initial best path is still an appropriate forwarding path. For directly rerouting, the packets are protected from blocking. With such a scheme, RLB can effectively split traffic among multiple paths, without causing out-of-order packets. In addition, RLB, as a building block, is compatible with existing load balancing algorithms.

B. Design Details

1) Predicting PFC Triggering: RLB first checks whether the ingress queue length exceeds a certain threshold at the current increasing rate and only performs prediction when there is congestion. Under congestion, RLB repeats the following two steps: (1) predicts whether PFC will be triggered, if it is true then (2) sends PFC triggering warning to upstream switches.

Predicting congestion: To predict PFC triggering, RLB first monitors the increasing speed of ingress queue length. This is

done by calculating the derivative of the buffer occupancy for each ingress queue within a certain time interval Δt (default link delay 2μs [11]). At this rate, we can calculate the time required for reaching the PFC threshold. If the remaining time is smaller than a given threshold, RLB sends PFC warning to upstream switches. Note that PFC warning is based on the prediction of future ingress queue length, rather than the current ingress queue size. As long as the packet passes through the potentially dangerous ingress port before PFC is actually triggered, or chooses other safe path after receiving PFC warning, it will not be blocked due to PFC PAUSE. Therefore, RLB allows packets to reconsider how to choose forwarding path before PFC is triggered. To eliminate out-of-order packets and avoid link utilization degradation, RLB needs to carefully calculate the dynamic threshold for sending PFC warning (§ III-B.3).

Sending PFC warning: After RLB predicts PFC will be triggered in a near future with high probability, the next step is to send PFC triggering warning to upstream switches. To quickly react to the ephemeral congestion before PFC is actually triggered, switches send PFC warning in advance through direct signal CNM of existing QCN mechanism, which is available in commodity switches [30], [31]. However, QCN forwards packets based on link layer addresses and cannot directly send the CNM to upstream switches in IP-routed networks. To address this problem, RLB records the source MAC address of the incoming packets in the flow table and then propagates CNM to upstream switches hop-by-hop. Meanwhile, the identification number of ingress port that is predicted to trigger PFC is filled in the QCN field of CNM.

2) Rerouting Without Packet Reordering: On the one hand, if there is no predicted PFC warning, RLB forwards packets directly to the initial optimal path selected by the existing load balancing, ensuring in-order transmission. On the other hand, with the predicted PFC warning, RLB employs a deliberate rerouting based on existing load balancing to avoid packet reordering. Instead of making a hasty decision to choose the initial optimal path calculated by existing load balancing schemes, RLB makes a careful consideration based on whether PFC will be triggered.

Specifically, if a PFC warning message is received, RLB will not give up the initial optimal path immediately, but give packets more opportunities to choose an appropriate forwarding path without reordering through packet recirculation. RLB decides whether to recirculate or reroute packets to ensure the packets will not arrive at the receiver later than the subsequent packets in the same flow due to being blocked by PFC pausing. If the delay of initial optimal path is much smaller than other paths, packet recirculation has a new opportunity to decide whether the initial optimal path is still an appropriate one and can still make the packets reach the receiver faster than the subsequent packets in the same flow. Otherwise, the packets are rerouted to the other path without PFC warning and then they reach the receiver orderly.

Algorithm 1 shows the rerouting logic of RLB. For each arriving packet, RLB chooses the new routing path based on the initial optimal path selected by existing load balancing schemes (line 2). If the initial selected path has PFC PAUSE warning, there are two cases. In the first one, the difference in delay between the initial optimal path and the suboptimal path is large enough to be worth recirculating packets to decide whether the best path still should be chosen (line 5-6). In the second one, the difference in delay between the initial optimal path and the suboptimal path is small, RLB takes the suboptimal path as the optimal one, and then repeats the above steps from line 3 to make decisions until choosing an appropriate path (line 8). Since RLB only needs to compare the relative delay difference between the optimal and sub-optimal paths with the recirculation delay to make forward decisions, we measure the RTT of the path as referenced in Hermes [17]. RLB finally selects the path without PFC warning to forward packets (line 10).

Algorithm 1 Rerouting Without Packet Reordering

```
Input:
    h_PFC: PFC warning;
    t_rc: Measured delay of packet recirculation;
    t_RTT: Measured RTT of a path;
1   for every packet do
2       Select the initial optimal path p;
3       if receiving p.h_PFC then
4           Select the suboptimal path p_s;
5           if (p_s.t_RTT - p.t_RTT) > t_rc; then
6               Recirculate packet; go to Line 3;
7           else
8               Replace p with p_s; go to Line 3;
9       else
10          p* = p;
11      return p* /* New routing path */
```

In brief, for the packet recirculation mechanism, only when the predicted PFC warning for the optimal path selected by the existing load balancing is received and the difference in delay between the optimal path and the suboptimal path is larger than the delay of packet recirculation, RLB will perform packet recirculation. After each recirculation, the packet will judge whether the above two conditions are met again. If yes, the packet continues to be recirculated, indicating that it will reach the receiver faster than rerouting. Otherwise, recirculation will stop to avoid the endless loop. Unlike prior load balancing routing mechanisms, which do not consider the negative impact of PFC on packet reordering in lossless DCN, RLB chooses routing paths after thoughtful consideration based on both PFC warning and path delay.

3) Calculating PFC Warning Threshold: To predict PFC triggering, we theoretically analyze the value range of PFC warning threshold Q_{th}. Without loss of generality, we model the incast network scenario with the oversubscribe ratio of n:1, where n flows are simultaneously sent to one destination host through a single PFC-enabled switch. The link capacity is C, and the link delay between the source edge switch and the destination edge switch is d. The PFC threshold is Q_{PFC}, and the instantaneous queue length of the ingress port is $Q(t)$ at

time t. The sending rate of each source host is $v_i(t)$ at time t, and the receiving rate of the destination host is $v_r(t)$. Thus, the difference between the sum of sending rate and the receiving rate can be defined as the queue length varying rate, which is calculated as $\sum_{i=1}^{n} v_i(t) - v_r(t)$. The change of queue length during the time interval t is calculated as $\sum_{i=1}^{n} \int_0^t v_i(t)dt - v_r(t)dt$.

We assume RLB triggers PFC prediction mechanism at time t_w when the ingress queue length increases to $Q(t_w)$, the PFC warning message is generated and sent to the upstream switch. During the PFC warning message transmission, the packets can still choose this related forwarding port, and the ingress queue length continues to increase for a short time d.

Thus, to ensure that PFC warning is sent before PFC triggering, the queue length $Q(t_w)$ should be satisfied the following condition

$$Q(t_w) < Q_{PFC} - \sum_{i=1}^{n} \int_{t_w}^{t_w+d} v_i(t)dt + d \times v_r(t). \quad (1)$$

The worst case is that all packets are rerouted to other paths once receiving PFC warning. To avoid the throughput loss due to queue emptying when the PFC warning is lifted at time t_r, the queue length $Q(t_r)$ also should be satisfied the following condition

$$Q(t_r) \geq d \times v_r(t) - \sum_{i=1}^{n} \int_{t_r}^{t_r+d} v_i(t)dt. \quad (2)$$

Thus, the PFC warning threshold Q_{th} is set in the range

$$Q_{th} \in [d \times v_r(t) - \sum_{i=1}^{n} \int_{t_w}^{t_w+d} v_i(t)dt, \\ Q_{PFC} - \sum_{i=1}^{n} \int_{t_w}^{t_w+d} v_i(t)dt + d \times v_r(t)]. \quad (3)$$

Since the future sending rate of other nodes is unknown, we set a conservative PFC warning threshold for RLB. In Equation 1 and Equation 2, the value of $v_i(t)$ is set to the maximum value of link capacity C. Thus, the conservative threshold Q_{th} is set in the range of $[\lfloor d \times C \rfloor, \lfloor Q_{PFC} - d \times C \times (n-1) \rfloor)$. The experimental results show that such PFC warning threshold is effective and robust to a wide range of traffic variations (§ VI).

IV. Over-Recirculation Prevention Under Switch Failures

In Section III, we presented the design principles of RLB aimed at addressing packet reordering issues for existing load balancing schemes in lossless DCNs. This mechanism shows effectiveness in balancing traffic within the normal and healthy topologies. However, it may fall short in addressing the more prevalent challenge of switch malfunctions in PFC-enabled lossless DCNs. In this section, we examine the potential issues under switch failures and propose the corresponding solution to mitigate the risk of packet over-recirculation.

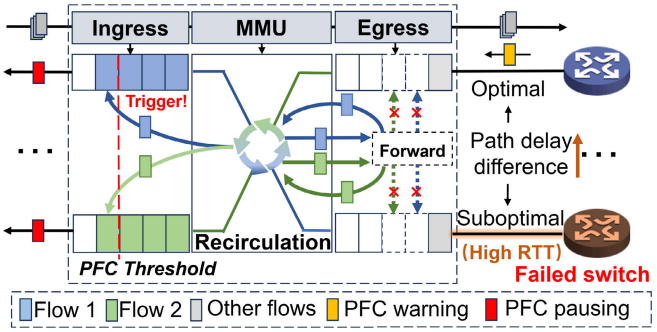


Fig. 6. Over-recirculation and congestion spreading due to switch failures.

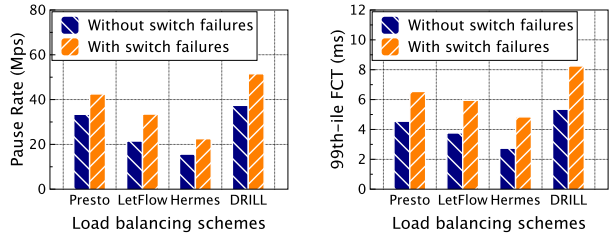
A. Problem Analysis

In modern DCNs, switch failures including packet blackholes and silent random packet drops are occurring widely [17]. Packet blackholes result from specific switch patterns, causing complete packet discards and can be attributed to factors like insufficient TCAM or TCAM table damage [20]. Silent random packet drops may be related to CRC checksum errors, insufficient switching ASICs, or improper installation of line cards, resulting in approximately 1%-2% packet loss.

When switch failures occur, RLB potentially suffers from over-recirculation and congestion spreading issues, even though it can predict PFC pausing. As shown in Fig. 6, there are two flows named as Flow1 and Flow2, which arrive at the different ingress ports. Now, when RLB detects a PFC warning on the selected optimal path and the switch on the suboptimal path also fails, potentially resulting in over-recirculation problem. Specifically, due to RLB lacking awareness of switch failures, it cannot anticipate that a sudden switch malfunction on the suboptimal path could cause increased delays, leading to higher RTT. Consequently, packets recirculate within the switch by looping back to the ingress port. Once the packet queue length at the ingress port exceeds the PFC pause threshold, PFC pause frames are transmitted to upstream switches, leading to an increase in the number of PFC-paused paths and aggravating the negative impact of PFC. In addition, due to malfunctioning switches, the increased recirculation packets exacerbate the reordering problem.

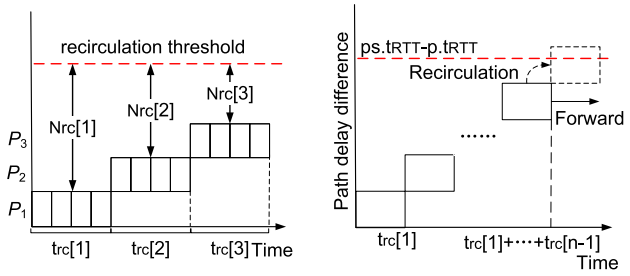
We conduct NS-3 simulations under two scenarios including packet black-holes and silent random packet drops. In the first scenario, we deliberately dropped half of the packets for specific source-destination IP pairs by randomly selecting a switch. In the second scenario, we randomly selected a core switch and imposed a 0.35% packet drop rate. We explored a comparative analysis between four load balancing schemes enhanced with RLB, i.e., Presto+RLB, Hermes+RLB, LetFlow+RLB, and DRILL+RLB.

We measured the performance in terms of PFC pausing rates and the 99^{th} percentile of FCT. In Fig. 7(a), although all four load balancing schemes with RLB effectively predict PFC pausing, the PFC pausing rate remains high during switch failures, resulting in more recirculation packets and serious congestion. Fig. 7(b) demonstrates a significant increase in the 99^{th} percentile of tail FCT, mainly due to PFC diffusion issues and an increase in PFC pausing paths. It's noteworthy



(a) Pause rate (b) 99^{th} percentile FCT

Fig. 7. The RLB performance deteriorates under switch failure (packets black-holes and silent random packet drops).



(a) Calculating threshold (b) Determine path delay difference

Fig. 8. RLB prevents packet over-recirculation by dynamically setting a recirculation threshold for each packet and conducting an over-recirculation check before each recirculation.

that switch failures significantly diminish the performance improvements achieved by RLB.

B. How to Prevent Over-Recirculation ?

Next, we demonstrate how RLB prevents packet over-recirculation. RLB dynamically computes the packet recirculation threshold by considering the available capacity of the current buffer. Simultaneously, RLB combines packet recirculation count and delay to detect switch failures.

1) Calculating Packet Recirculation Threshold: One simple solution to prevent packet over-recirculation is to set a fixed recirculation threshold for all packets. However, this approach raises fairness concerns. On the one hand, when there's an effective buffer capacity, packets have more recirculation opportunities. On the other hand, when buffer capacity is limited, packets may only recirculate a few times or need to be forwarded immediately to the downstream switch to prevent the occupied buffer from reaching the threshold, resulting in PFC warning or pausing.

RLB determines the packet recirculation threshold based on the available buffer capacity. Specifically, as shown in Fig. 8(a), RLB periodically measures the recirculation delay $t_{rc}[i]$ for each round i at the switch. During each t_{rc} interval, it calculates the number of recirculations $N_{rc}[i]$ for the packets presently being processed by the switch. Different packet clusters are assigned distinct number of recirculations. By adapting the maximum number of packets recirculations within each t_{rc} interval based on the remaining available buffer capacity, it can effectively work well even under switch failures.

We conduct a theoretical analysis to determine the value range of recirculations N_{rc}. Without loss of generality, we still use an oversubscription ratio of n:1 to emulate network congestion scenarios. We designate the upper limit of the number of packet recirculation as $N_{rc}[i]$, the other parameters retain their definitions in Section III-B.3.

The remaining capacity of buffer is denoted as $Q(t_w)$. In order to ensure that the queue length does not exceed the PFC pausing threshold due to over-recirculation issue before PFC is triggered, the buffer occupancy should satisfy the following relationship (4):

$$Q(t_w) < Q_{PFC} - \sum_{i=1}^{n} \int_{t_w}^{t_w+t_{rc}} v_i(t)dt + t_{rc} \times v_r(t). \quad (4)$$

Therefore, for each time interval t_{rc}, the upper limit of the number of packet recirculation N_{rc} is calculated as follows:

$$N_{rc} = \lfloor \frac{Q_{PFC} - \sum_{i=1}^{n} \int_{t_w}^{t_w+t_{rc}} v_i(t)dt + t_{rc} \times v_r(t)}{t_{rc}} \rfloor. \quad (5)$$

Moreover, recirculation latency is another critical factor that determines the limit on the number of packet recirculation. As depicted in Fig. 8(b), another issue arises related to packet recirculation. When a packet undergoes multiple recirculations, it can lead to the accumulation of recirculation delays, represented as $\sum_{i=1}^{n} t_{rc}[i]$, exceeding the original latency difference of the path $(p_s.t_{RTT} - p.t_{RTT})$. To ensure that packets do not experience delay larger than the original latency difference due to recirculation, the cumulative recirculation delay should satisfy the following relationship (6).

$$\sum_{i=1}^{n-1} t_{rc}[i] + t_{rc}[n-1] \leq (p_s.t_{RTT} - p.t_{RTT}). \quad (6)$$

To address the above problem, we design a new recirculation criterion. When determining whether a packet should be performed recirculation, if the accumulated recirculation delay exceeds the original path's latency difference, the packet is directed to the optimal path rather than recirculating. Since the forthcoming recirculation delay denoted as $t_{rc}[n]$ is uncertain, we estimate it using the most recent recirculation delay $t_{rc}[n-1]$ to forecast the cumulative recirculation delay. Experimental results show that packet recirculation threshold is resilient to the switch failure scenarios.

2) Detecting Switch Failures: We detect two typical switch failures including packet blackholes and silent random packet drops, in which over-recirculation issue is likely to occur. Compared to packet loss causing network congestion and link Frame Check Sequence (FCS) errors, detecting these two types of switch failures is notably more challenging. The previous work Hermes [17] deduces switch failures through packet retransmission and timeout events on each path. Pingmesh [20] continually assesses latency between host pairs and scrutinizes data to identify any unusual latency patterns or packet loss. However, continuous monitoring TCP sequence numbers and timeout events on each link undeniably introduces additional system overhead for the load balancing solutions.

Furthermore, relying on feedback delay longer than base RTT to detect switch failures may not handle bursty congestion effectively. In such cases, PFC pausing can spread to upstream switches before detecting the switch failures. Our experiment

Algorithm 2 Switch Failure Detection Algorithm

Input:
Rc_avg: Average recirculation count;
$t_{rc}[i]$: Recirculation delay at time i;
N_{rc}: Threshold for detecting switch failure;

1 **for** *every* $t_{rc}[i]$ **do**
2 Record and calculate Rc_avg;
3 **if** $Rc\_avg > N_{rc}$ **then**
4 Identify port with the highest recirculation count;
5 $potential\_port \leftarrow$ identify_port($port$);
6 count($potential\_port$)++;
7 **if** *count(potential_port) == 2* **then**
8 $potential\_port \leftarrow faulty\_port$
$faulty\_switch \leftarrow$ identify_switch($faulty\_port$);
9 trigger_alert($faulty\_switch$);
10 **else**
11 continue; go to Line 3;
12 **else**
13 **for** *every port* **do**
14 count($port$) $\leftarrow$ 0;
15 go to Line 3;

results show that switch failures often lead to a significant increase in the average number of packet recirculation. This happens because when downstream switches fail, high latency forces packets to experience multiple recirculations, even if there's a recirculation threshold in place.

Thus, RLB introduces an algorithm that combines packet recirculation count and recirculation delay for the purpose of switch failure detection through continuous monitoring of local information. Specifically, as depicted in Algorithm 2, RLB maintains records of the average recirculation count (Rc_avg) and recirculation delay ($t_{rc}[i]$) for packets during each recirculation delay time. When Rc_avg exceeds a predefined threshold (N_{rc}), RLB scrutinizes all egress ports to pinpoint the port responsible for the highest packet recirculation count. Once this port is identified as a potentially faulty one, RLB continues to monitor the subsequent recirculation delay time unit ($t_{rc}[i+1]$). If this port remains the one with the highest packet recirculation count, it is marked as a faulty port. Upon identification of a faulty port, RLB designates the downstream switch as experiencing a fault and issues an alert. This alert can serve as a notification to network administrators or automated systems to initiate appropriate actions, such as traffic path redirection, investigation of switch failures, or switch restarts. Meanwhile, the algorithm will eliminate the faulty path, and packets will be rerouted using the predeployed original load balancing algorithm. This algorithm facilitates swift responses based on local information to prevent congestion propagation resulting from switch failures, and avoid the unnecessary overhead introduced by cumbersome monitoring mechanisms.

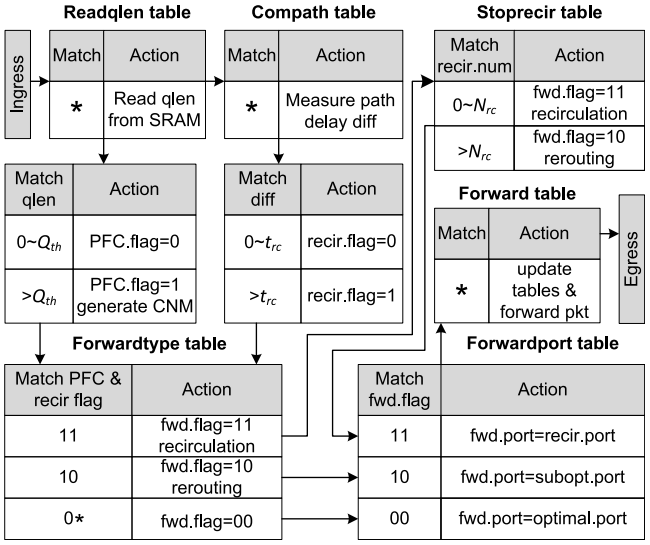


Fig. 9. The packet processing pipeline for RLB implementation.

V. Implementation

RLB dataplane: We have implemented an RLB prototype on a programmable switch with the type of Wedge 100BF-32X and use Programming Protocol-independent Packet Processors (P4) language to specify the packet processing pipeline. As shown in Fig.9, the pipeline consists of several match-action tables, where interdependent ones are arranged to different stages for serial execution, while non dependent ones can be allocated to the same stage for parallel execution. The primary difficulty of implementing RLB lies in guaranteeing that resources are accessed exclusively during the sequential stages of packet processing.

Specifically, in the $Readqlen$ table, the metadata $qlen$ representing the ingress queue length is read from the static random access memory (SRAM) by using stateful ALUs. If $qlen$ exceeds the PFC warning threshold Q_{th}, the warning flag $PFC.flag$ is set to 1 and the warning scheme is triggered. To reduce measurement overhead, RLB measures the RTT of the path at the edge switch over a certain period (e.g., one base RTT). The source leaf switch periodically picks up a packet and records the sending time into the packet header. At the destination leaf switch, the sending time is piggybacked by any packet going through the same pair of ports. Upon receiving the returned packet, the source leaf switch calculates the RTT for the corresponding path by subtracting the sending time from the arriving time of this packet and records the path delay information. In addition, the recirculation delay t_{rc} is the queueing delay measured based on the recirculation queue length. In the implementation of RLB based on programmable switches, the egress queue length cannot be directly read from the ingress intrinsic metadata on P4 hardware platform. Therefore, we implement a queue-size SRAM that performs the similar function by tracking the number of packets being enqueued in and dequeued from the recirculation queue. Then the recirculation queue length is read from the queue-size SRAM by using stateful ALUs at the switch, and then the recirculation delay is obtained by multiplying it with the forwarding rate.

TABLE I
RESOURCE CONSUMPTION

Resource	Presto	DRILL	LetFlow	RLB (additional)
Match Crossbar	2.19%	4.24%	4.82%	5.31%
Hash Bits	3.28%	4.05%	5.67%	3.65%
Gateway	2.92%	3.13%	2.96%	3.22%
SRAM	2.16%	3.74%	3.33%	3.59%
VLIW Actions	1.74%	2.52%	2.34%	2.88%
ALU Instruction	3.21%	5.16%	5.2%	6.19%

In the $Compath$ table, if the measured delay difference between the optimal and suboptimal paths exceeds the packet recirculation delay t_{rc}, the recirculation flag is set to 1. Then matching PFC warning and recirculation flags in the $Forwardtype$ table and comparing the recirculation number in the $Stoprecir$ table, $fwd.flag$ is marked with different values representing recirculation, rerouting, and direct forwarding, respectively. Finally, packets are forwarded in the $Forward$ table.

Resource consumption: As shown in table I, we compare the hardware resource consumption of RLB in the packet processing pipeline at the programmable switch with three switch-based typical load balancing mechanisms. It should be noted that RLB, as an incremental building block for existing load balancing mechanisms, brings additional overhead on top of the existing load balancing algorithms' resource consumption. RLB uses more pipeline stages to detect PFC triggering and switch failures, resulting in more overhead such as match tables and gateway. Meanwhile, RLB needs more blackbox stateful ALUs and SRAM to store and update stateful information such as path delay and recirculation number. In addition, since RLB is only triggered after receiving PFC warning message, in other cases, besides the resource consumption of monitoring path delay and queue length, it will not increase the resource consumption of recirculation and rerouting operations. In brief, the limited resource consumption of RLB is acceptable compared with its benefits.

VI. EVALUATION AND ANALYSIS

We evaluate RLB in conjunction with four typical load balancing schemes (i.e., Presto [15], LetFlow [16], Hermes [17] and DRILL [18]) by conducting a combination of testbed experiments and NS-3 simulations. Our evaluation seeks to answer the following questions:

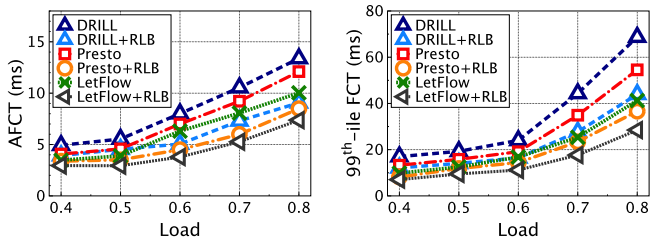
- **How does RLB perform in practice?** Testbed experiments (§ VI-A) show that the RLB-enhanced schemes reduce tail latency by up to 40% compared to the schemes without RLB under varying traffic loads and link speeds.
- **How does RLB perform under a symmetric and an asymmetric topology?** The experiments (§ VI-B.1 and § VI-B.2) demonstrate the superior performance of RLB-enhanced solutions with realistic workloads. Specifically, RLB reduces the average flow completion times by 56%, 49%, 49% and 32% compared to Presto, LetFlow, Hermes and DRILL, respectively.
- **How sensitive is RLB to traffic patterns and conditions?** By varying the incast degree and response size in the bursty scenarios (§ VI-B.3), we show that RLB effectively predicts congestion, reduces out-of-order packets, and achieves persistent good performance under different traffic intensities.
- **How does RLB perform under switch failure scenarios?** The experiments (§ VI-B.4) show that RLB performs well under switch failures. RLB effectively prevents excessive packet recirculation and continues to reduce PFC pause rate and minimize tail latency even in case of packet blackholes and silent random packet drops.
- **How robust is RLB under different parameter settings?** Deep-dive experiments (§ VI-B.5) validate the effectiveness of RLB. The test results show that RLB's performance is stable under a variety of parameter settings, and RLB performs better than vanilla load balancing schemes.

Baseline: We integrated RLB with four aforementioned load balancing schemes described in § II-A.3, including DRILL [18], Presto [15], LetFlow [16] and Hermes [17], and compare the performance of RLB-enhanced solutions with vanilla schemes, respectively. We use and modify the open source codes provided by the Hermes's authors [32] that includes the above four mechanisms and set the parameters as recommended in the corresponding paper. Specifically, DRILL compares queue lengths of two randomly selected ports with the previously least-loaded port and forwards the packet to the least load port. Presto reroutes each fixed-sized flowcell (i.e., 64KB) among parallel paths in a round-robin fashion. LetFlow randomly assigns flowlets over multiple paths. Since Hermes is an edge-based solution with no switch modification, we only compare with it in the simulation experiments.

Realistic workloads: We use four typical realistic workloads observed from production data centers, i.e., Web Server, Cache Follower, Web Search and Data Mining [27]. The average flow sizes range from 64KB to more than 7.41MB, and the distribution of flow sizes is scattered. Particularly, the Data Mining workload is more heavy-tailed with 83% of flows that are smaller than 100KB and 95% of all data bytes from around 3.6% of flows that are larger than 35MB [17]. All flows in Web Server workload are less than 1MB. The flows are generated between random pair of end-hosts according to Poisson processes.

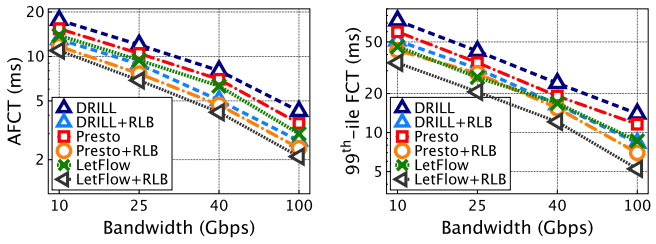
A. Testbed Experiments

Testbed setup: We built a small testbed with a 2×8 leaf-spine topology as shown in Fig. 2 in Section II. Each server (Dell PRECISION TOWER 5820 desktop) is equipped with 10 cores Intel Xeon W-2255 CPU and Mellanox ConnectX-5 EN 100GbE dual-port QSFP28 NICs, runs Ubuntu 20.04.1 with Linux version 5.4.0-42-generic, and is connected to the corresponding hardware leaf switch. Each switch (Wedge 100BF-32X) has 32 full duplex 100Gbps ports and 22MB shared buffer. There are 8 parallel equal-cost paths with a default 40Gbps link capacity. We compare the performance of three switch-based typical load balancing mechanisms with and without RLB under varying traffic loads and link speeds.



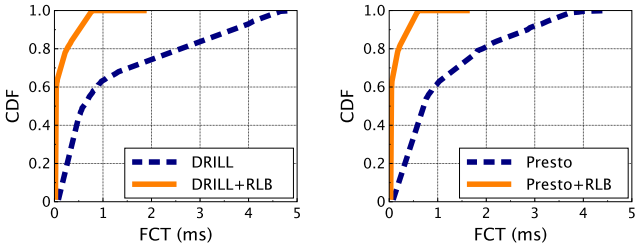
(a) Average FCT (b) 99^{th} percentile FCT

Fig. 10. AFCT and 99^{th}-ile FCT under web search workload.

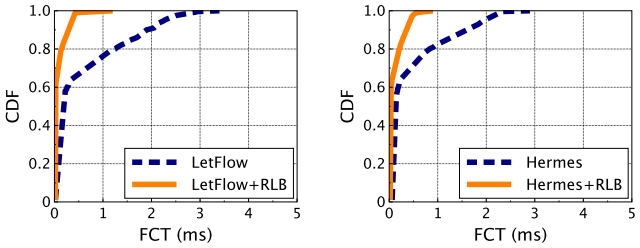


(a) Average FCT (b) 99^{th} percentile FCT

Fig. 11. AFCT and 99^{th}-ile FCT with varying link speeds.



(a) FCT of DRILL (b) FCT of Presto



(c) FCT of LetFlow (d) FCT of Hermes

Fig. 12. FCT of all flows in the symmetric topology under Web Search workload. The average load of the network core is 60%.

1) Performance Under Varying Traffic Load: We first conduct testbed experiments under the realistic Web Search workload with varying load from 0.4 to 0.8 to test the basic effective performance of RLB. Fig.10 shows that the integration of RLB into existing load balancing mechanisms has effectively reduced the average and 99^{th} percentile flow completion time. Specifically, compared to load balancing without RLB, the RLB-enhanced schemes significantly minimize the out-of-order packets due to PFC pausing. Especially under high traffic loads, the possibility of PFC triggering is greater, resulting in more PFC warnings, and more opportunities for RLB to take in. Compared with low traffic load, RLB has a higher improvement in latency performance. For example, at 0.5 and 0.8 loads, the integrated RLB mechanisms reduce AFCT by up to 18%, 17%, 15% and 32%, 30%, 27%, and achieves up to 27%, 25%, 24% and 36%, 33%, 31% lower 99^{th} percentile FCT compared to DRILL, Presto and LetFlow, respectively. This is as expected because the existing load balancing schemes cannot predict PFC triggering and potentially suffer from frequent pausing, which inevitably lead to severe reordering and increase latency, especially under high traffic load.

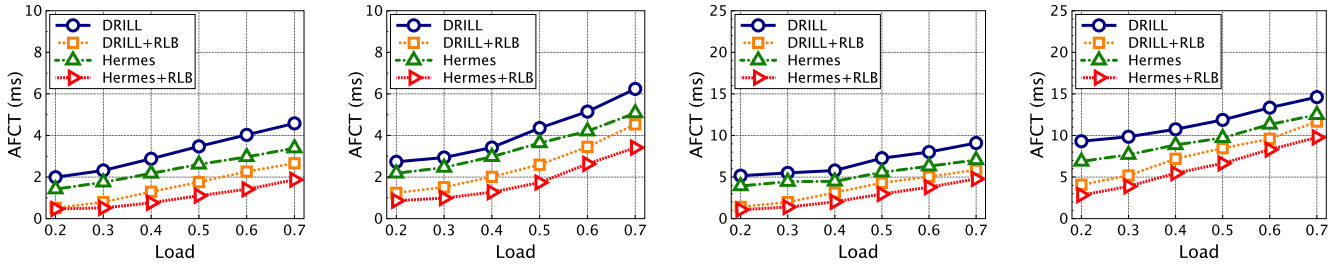
2) Performance Under Varying Link Speed: We further conduct the above experiments at 0.7 load under varying link speeds of 10Gbps, 25Gbps, 40Gbps, and 100Gbps. As shown in Fig.11, the existing load balancing mechanisms that integrate RLB significantly reduce AFCT and 99^{th} percentile FCT across all tested link speeds. The reason is that the conventional load balancing mechanisms without RLB do not account for dynamic rerouting or recirculation based on real-time congestion status of paths and PFC triggering. The improvement of RLB is particularly noteworthy under higher link speeds, where the potential for out-of-order packets can escalate due to the increased variation in bursty congestion and the consequent more PFC pausing. Specifically, we observe that the RLB-enhanced schemes have a higher performance improvement at higher link speeds such as 40Gbps and 100Gbps. For example, the integrated RLB schemes reduce AFCT by up to 27%, 25%, 22% and 37%, 32%, 30%, and reduce 99^{th} percentile FCT by up to 30%, 28%, 26% and 40%, 39%, 35% under 10Gbps and 100Gbps compared to DRILL, Presto and LetFlow, respectively. As the link speed increases, the benefit of integrating RLB became more pronounced.

B. Large-Scale NS-3 Simulations

Simulation setup: Unless otherwise specified, we conduct simulations on a symmetric leaf-spine topology with 12 leaf switches and 12 spine switches. There are 12 equal-cost parallel paths between any pair of leaf switches. Each leaf switch is connected to 24 hosts and 12 spine switches with 40Gbps links. Each link delay is set to 2μs [11]. Each switch enables PFC and the shared buffer size is 9MB. We use DCQCN [3] as the default transport protocol and set the related parameters as suggested in [3].

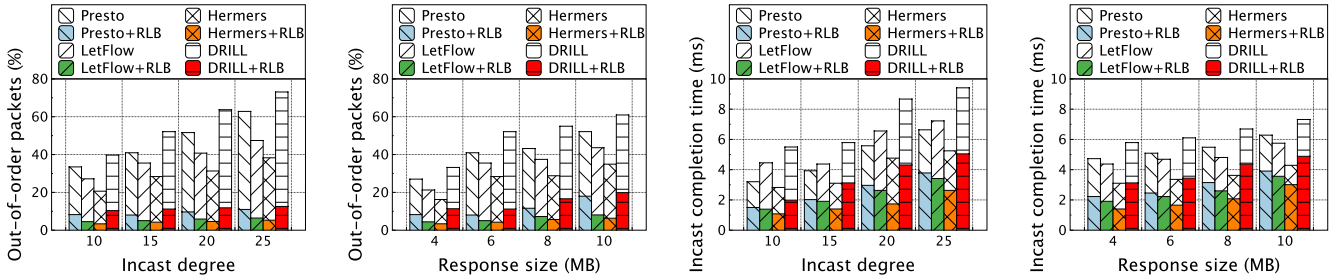
1) Performance Under Symmetric Topology: We evaluated RLB's performance with four load balancing schemes in a symmetric topology. Fig. 12 displays the cumulative distribution function (CDF) of FCT for the Web Search workload. Results show that RLB significantly reduces FCT and tail FCT compared to using the load balancing schemes alone. For instance, RLB-enhanced solutions cut the 99^{th} percentile FCT by up to 72%, 67%, 58%, and 54% over DRILL, Presto, LetFlow and Hermes, respectively. These load balancing schemes benefit from RLB's PFC triggering predictions, reducing out-of-order packets through recirculation or timely rerouting.

DRILL without RLB, however, experiences longer tail delays than other load balancing schemes. This occurs because DRILL reroutes at a finer-grained packet level, potentially affecting more paths with PFC pausing when congested flows are spread across multiple parallel paths. RLB addresses this by predicting PFC triggering and incorporating PFC pausing considerations into load balancing decisions.



(a) Web Server (b) Cache Follower (c) Web Search (d) Data Mining

Fig. 13. AFCT of all flows under realistic workloads in the asymmetric topology with varying load.



(a) Reordering with varying degree (b) Reordering with varying response (c) FCT with varying incast degree (d) FCT with varying response size

Fig. 14. Varying incast degrees and response sizes.

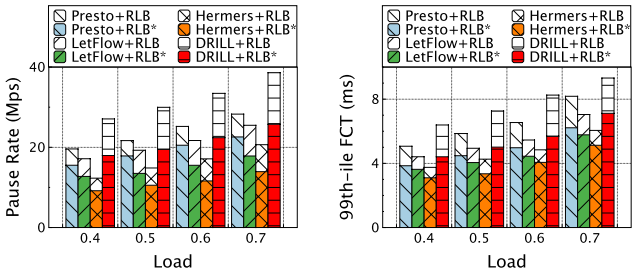
Upon receiving a PFC warning, arriving packets can avoid reaching the destination host later than subsequent packets with larger sequence numbers in two scenarios. In the first, packets can be flexibly rerouted to suboptimal equal-cost paths with minimal delay differences to prevent PFC pausing. In the second, packets can be recirculated rather than radically rerouted to suboptimal paths with significant delay differences. However, when the number of predicted PFC-triggered paths increases, RLB's available effective paths decrease, limiting its performance improvement. In summary, RLB effectively reduces out-of-order packets.

2) Performance Under Asymmetric Topology: We further integrated RLB into DRILL and Hermes under four realistic workloads in an asymmetric topology with the varying load. We adopt the default symmetric topology and reduce the link capacity from 40Gbps to 10Gbps for 20% of randomly chosen leaf-to-spine links [17]. Fig. 13 shows the improvement of RLB in terms of average FCT as the load varies from 20% to 70% of the network capacity. We can see that DRILL and Hermes benefit from RLB across a wide range of loads. For example, DRILL+RLB outperforms DRILL by up to 42% and 28% at 0.6 load for Web Server and Data Mining workloads, respectively. For Cache Follower workload, Hermes+RLB is 58% and 34% better than Hermes at 0.2 and 0.6 load, respectively.

Specifically, RLB-enhanced solutions always outperform the vanilla DRILL and Hermes as the load increases. Firstly, RLB performs better for the Web Server and Cache Follower workloads than Web Search and Data Mining. To explain this, note that the Web Server and Cache Follower workloads contain more small flows that cannot be controlled by the end-to-end transport protocol. Furthermore, the Data Mining workload has larger inter-flow arrival time and much more long flows that can be controlled by the transport protocol. Hence, PFC is more likely to be triggered in Web Server and Cache Follower workloads, and RLB has more chances to take effect. RLB is able to predict PFC triggering and make the rerouting or recirculation decisions cautiously based on the difference in path delay to avoid out-of-order packets. Secondly, we find that as the load increases, the room for improvement by RLB reduces slightly, which is a result of less available parallel paths for rerouting. Last but not least, the performance improvement of RLB on the existing load balancing mechanisms in asymmetric network is greater than that in symmetric network.

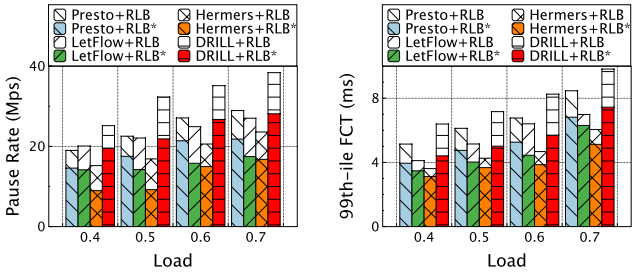
3) Performance Under Incast Scenario: We next evaluate the effectiveness of RLB with different intensities of bursty traffic by varying incast degrees. In this test, a client makes simultaneous requests to fetch responses from multiple servers. By default, the number of involved responders is 15 and the total response traffic is 4MB in each incast initiation. We vary the incast degree from 10 to 25 and change the response size from 4MB to 10MB. We measure the ratio of out-of-order packets and the completion time of the last flow, as shown in Fig. 14.

The results indicate that RLB can help the load balancing schemes to significantly reduce the ratio of out-of-order packets and speed up flows even under stressed incast scenarios. For example, the ratio of out-of-order packets under the scenario where incast degree is 15 and the response size is 10MB is reduced by up to 75%, 66%, 51% and 47% over DRILL, Presto, LetFlow and Hermes, respectively. The incast completion time is improved by 22%∼46% across different response sizes. The key reason is that RLB assists the load balancing schemes to avoid triggering spurious retransmissions due to disorder packets. It is also worth noting that, although RLB greatly alleviates the packet reordering problem, the tail FCT has not reduced in the same proportion due to



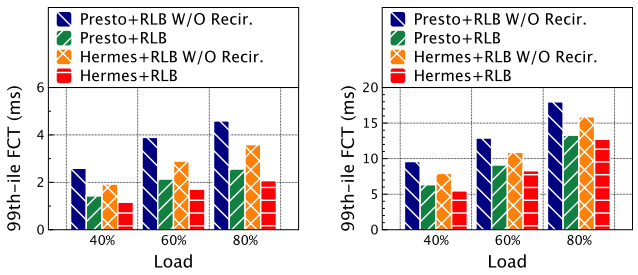
(a) PFC pause rate (b) 99^{th} percentile FCT

Fig. 15. Performance under random packet drops.



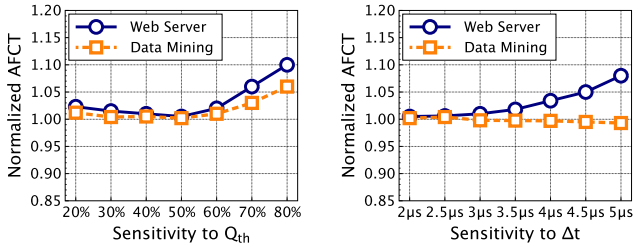
(a) PFC pause rate (b) 99^{th} percentile FCT

Fig. 16. Performance under packet black-holes.



(a) Web Server (b) Data Mining

Fig. 17. RLB deep dive for effectiveness of packet recirculation under realistic workloads. Here "Recir." refers to recirculation.



(a) Sensitivity to Q_{th} (b) Sensitivity to Δt

Fig. 18. Sensitivity to PFC predicting threshold Q_{th} and the calculating time interval Δt.

packet recirculation delay. In addition, the end-to-end transport protocol is able to control the large flows to alleviate congestion. Therefore, the bursty traffic can be controlled by the transport protocol as the response size increases, the performance improved by RLB is smaller than that with the increase of incast degree.

4) Performance Under Switch Failures: We further evaluate RLB performance under switch failure scenarios in the default symmetric topology. We maintain generality by using the default simulation settings described in Section IV. Fig. 15 and Fig. 16 show the PFC pause rate and 99th percentile tail FCT for the four load balancing schemes with web search workload under the silent random packet drops and packet black-holes scenarios, respectively. The traffic load varies from 0.4 to 0.7. RLB* denotes that RLB enables the dynamic recirculation threshold.

The results reveal that RLB* enhanced load balancing schemes continue to reduce both PFC pause rates and 99^{th} percentile tail FCT significantly compared to the RLB enhanced solutions. For instance, in the packet black-holes scenario, compared to Presto, LetFlow, Hermes, and DRILL, RLB* enhanced schemes decrease the PFC pause rate and 99^{th} percentile FCT by up to 23%, 21%, 17%, 35% and 27%, 25%, 23%, 32%, respectively. This is because RLB* dynamically adjusts the packet's recirculation threshold based on real-time network conditions, enabling it to respond more effectively to switch failures and prevent endless packet recirculation at switches.

We also observe that Hermes outperforms other schemes by up to 20% because it can detect switch failures through packet retransmissions and timeout events. DRILL shows that the tail latency improvement reaches up to 47%, mainly because it makes routing decisions based on the local queue lengths in time. While DRILL is more affected by switch failures, RLB effectively compensates for its deficiencies by predicting PFC and detecting faulty ports. LetFlow is less affected because random packet loss creates more opportunities for rerouting on affected paths. However, without PFC prediction and faulty port detection, the performance of LetFlow+RLB is still 1.2 times worse than LetFlow+RLB*.

5) Performance Robustness: **Effectiveness of recirculation:** To avoid the side effects of PFC pausing, a simple method is to reroute packets directly when receiving the warning of PFC triggering. However, this is not always a good choice. Considering the different path congestion and PFC pausing duration, the benefit of packet recirculation on the original path is higher than that of aggressive rerouting. Now we investigate the benefits of packet recirculation using Web Server and Data Mining workloads. Fig. 17 (a) shows that packet recirculation bring about 37% and 28% improvement to the 99^{th} percentile of FCT under 80% load for Presto and Hermes, respectively. A similar trend is observed in Fig. 17 (b) as well.

Sensitivity of parameter settings: We further study how different parameter settings affect the performance of RLB. Fig. 18 shows the normalized AFCT for the optimal parameters under Web Server and Data Mining workloads with varying Q_{th} and Δt. First, we observe that the AFCT is relatively stable when these two parameters are set close to the suggested values. Another observation is that the AFCT is increased under both workloads as the increase of Q_{th}, because PFC prediction is late and PFC may have been triggered when RLB takes effect. We also observe that the two workloads experience different trend as the calculating time interval increases. Since the Web Server workload is

more bursty, PFC warning is generated more frequently, thus an aggressive parameter setting brings better performance due to rapid adaptation to network congestion. In contrast, the performance of Data Mining is less sensitive to Δt.

VII. Related Work

We classify previous work on balancing traffic in DCNs into two categories: load balancing schemes and multi-path transport solutions.

Load balancing schemes: To make good use of multiple paths, a wide range of mechanisms perform rerouting at finer granularity. These include, but are not limited to: 1) using packet-level switching granularity to split traffic flexibly across parallel paths [17], [18], [33], [36], [37], but potentially suffering from packet reordering problem. Specifically, DRILL [18] focuses on mitigating transient congestion caused by micro-burst traffic through microsecond-level load balancing. Hermes [17] employs proactive congestion detection and cautious routing decisions to balance traffic; RPS [33] randomly sends packets across all available paths to improve link utilization; TLB [36] adjusts the switching granularity of long flow based on the intensity of short flow to reduce FCT. ConWeave [37] performs fine-grained rerouting based on priority queues at the RTT level to avoid out-of-order packets. 2) using fixed flowcell-level switching granularity to reduce out-of-order packets [15]. Presto [15] sends each packet cluster (64KB) in a round-robin manner to all available paths. 3) using flowlet-level switching granularity to dynamically adapt to network congestion [14], [16], [24], [25]. Specifically, CONGA [14] dynamically allocates the best path for flowlets based on real-time network congestion. LetFlow [16] employs a fixed flowlet time interval and randomly selects a forwarding path for each flowlet. HULA [24] forwards flowlets to the best next-hop to the destination switch. CLOVE [25] detects path congestion and switches flowlets to the best path by altering the packet header five-tuple. 4) using flow-level coarse granularity to avoid out-of-order delivery at the cost of low link utilization [34], [35]. Hedera [34] uses a central controller to estimate bandwidth demand and reroutes congested long flows to avoid hash collisions. MicroTE [35] also uses a central controller to track long flows and prioritize forwarding paths allocation. However, these schemes are originally designed for lossy DCN and cannot work well in lossless DCN due to the adverse impact of PFC.

Multi-path transport solutions: Multi-path transmission schemes split flows into multiple subflows and assign them on the equal-cost paths to improve throughput. MP-RDMA [10], [38] distributes packets among parallel paths in a congestion-aware manner to achieve high throughput. IRN [39] explores the effective loss recovery schemes for lossy RDMA networks to abandon PFC mechanism. While these schemes effectively alleviate congestion, they still cannot completely avoid PFC triggering, so further efforts are needed to avoid reordering.

VIII. Conclusion

This paper presents RLB, that augments all existing load balancing algorithms to reduce packet reordering in lossless datacenter networks. By predicting PFC triggering, RLB decides whether to choose the initial best path selected by existing load balancing schemes, or reroute to other parallel paths, or recirculate packets to obtain more opportunities to choose appropriate forwarding paths without packet reordering. Moreover, RLB makes timely rerouting decisions by detecting switch failures to prevent packet over-recirculation. We have implemented RLB with programmable switches and evaluated it through testbed experiments and large-scale simulations. The test results indicate that RLB can effectively reduce out-of-order packets and significantly reduce the tail flow completion time by up to 72% compared with the state-of-the-art load balancing schemes.

References

[1] J. Hu, Y. He, and J. Wang, "RLB: Reordering-robust load balancing in lossless datacenter networks," in *Proc. ACM ICPP*, 2023, pp. 576–584.

[2] M. Alizadeh, A. Greenberg, and D. A. Maltz, "Data center TCP (DCTCP)," in *Proc. ACM SIGCOMM*, 2010, pp. 63–74.

[3] Y. Zhu et al., "Congestion control for large-scale RDMA deployments," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 532–536.

[4] W. Cheng, K. Qian, W. Jiang, T. Zhang, and F. Ren, "Re-architecting congestion management in lossless Ethernet," in *Proc. USENIX NSDI*, 2020, pp. 19–36.

[5] K. Qian, W. Cheng, T. Zhang, and F. Ren, "Gentle flow control: Avoiding deadlock in lossless networks," in *Proc. ACM Special Interest Group Data Commun.*, Aug. 2019, pp. 75–89.

[6] Y. Zhu, M. Ghobadi, V. Misra, and J. Padhye, "ECN or delay: Lessons learnt from analysis of DCQCN and TIMELY," in *Proc. 12th Int. Conf. Emerg. Netw. Exp. Technol.*, Dec. 2016, pp. 313–327.

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

[8] W. Bai, A. Agrawal, A. Bhagat, M. Elhaddad, N. John, and J. Padhye, "Empowering Azure storage with 100 ×100 RDMA," in *Proc. USENIX NSDI*, 2023, pp. 49–67.

[9] J. Xue, M. U. Chaudhry, B. Vamanan, T. N. Vijaykumar, and M. Thottethodi, "Dart: Divide and specialize for fast response to congestion in RDMA-based datacenter networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 1, pp. 322–335, Feb. 2020.

[10] Y. Lu et al., "Multipath transport for RDMA in datacenters," in *Proc. USENIX NSDI*, 2018, pp. 357–371.

[11] 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.

[12] R. Mittal et al., "TIMELY: RTT-based congestion control for the datacenter," in *Proc. ACM SIGCOMM*, 2015, pp. 537–550.

[13] G. Kumar, N. Dukkipati, and K. Jang, "Swift: Delay is simple and effective for congestion control in the datacenter," in *Proc. ACM SIGCOMM*, 2020, pp. 514–528.

[14] M. Alizadeh et al., "CONGA: Distributed congestion-aware load balancing for datacenters," in *Proc. ACM Conf. SIGCOMM*, Aug. 2014, pp. 503–514.

[15] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akellay, "Presto: Edge-based load balancing for fast datacenter networks," in *Proc. ACM SIGCOMM*, 2015, pp. 465–478.

[16] E. Vanini, R. Pan, M. Alizadeh, P. Taheri, and T. Edsall, "Let it flow: Resilient asymmetric load balancing with flowlet switching," in *Proc. 14th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2017, pp. 407–420.

[17] H. Zhang, J. Zhang, W. Bai, K. Chen, and M. Chowdhury, "Resilient datacenter load balancing in the wild," in *Proc. ACM SIGCOMM*, 2017, pp. 253–266.

[18] S. Ghorbani, Z. Yang, P. Godfrey, Y. Ganjali, and A. Firoozshahian, "DRILL: Micro load balancing for low-latency data center networks," in *Proc. ACM SIGCOMM*, 2017, pp. 225–238.

[19] The P4.org Architecture Working Group. *$P4_{16}$ Portable Switch Architecture (PSA)*. Accessed: Nov. 11, 2017. [Online]. Available: https://p4.org/p4-spec/docs/PSA-v0.9.0-draft.html#sec-recirculate

[20] C. Guo et al., "Pingmesh: A large-scale system for data center network latency measurement and analysis," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 139–152.

[21] Z. Liu et al., "Enabling work-conserving bandwidth guarantees for multi-tenant datacenters via dynamic tenant-queue binding," in *Proc. IEEE INFOCOM IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 1–9.

[22] W. Bai, S. Hu, K. Chen, K. Tan, and Y. Xiong, "One more config is enough: Saving (DC)TCP for high-speed extremely shallow-buffered datacenters," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 489–502, Apr. 2021.

[23] *IEEE 802.1 Qbb—Priority-Based Flow Control*. Accessed: Feb. 21, 2024. [Online]. Available: https://1.ieee802.org/dcb/802-1qbb/

[24] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "HULA: Scalable load balancing using programmable data planes," in *Proc. Symp. SDN Res. (SOSR)*, 2016, pp. 1–12.

[25] N. Katta et al., "Clove: congestion-aware load balancing at the virtual edge," in *Proc. 13th Int. Conf. Emerg. Netw. EXperiments Technol.*, Nov. 2017, pp. 323–335.

[26] W. Bai, S. Hu, K. Chen, K. Tan, and Y. Xiong, "One more config is enough: Saving (DC)TCP for high-speed extremely shallow-buffered datacenters," in *Proc. IEEE INFOCOM IEEE Conf. Comput. Commun.*, Jul. 2020, pp. 2007–2016.

[27] J. Hu et al., "RPO: Receiver-driven transport protocol using opportunistic transmission in data center," in *Proc. IEEE 29th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2021, pp. 1–11.

[28] J. Hu, J. Huang, Z. Li, J. Wang, and T. He, "A receiver-driven transport protocol with high link utilization using anti-ECN marking in data center networks," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 2, pp. 1898–1912, May 2023.

[29] J. Zhang, W. Bai, and K. Chen, "Enabling ECN for datacenter networks with RTT variations," in *Proc. 15th Int. Conf. Emerg. Netw. Exp. Technol.*, Dec. 2019, pp. 233–245.

[30] *IEEE 802.1Qau—Congestion Notification*. Accessed: Dec. 15, 2009. [Online]. Available: http://www.ieee802.org/1/pages/802.1au.html

[31] A. Saeed et al., "Annulus: A dual congestion control loop for datacenter and WAN traffic aggregates," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Architectures, Protocols Comput. Commun.*, Jul. 2020, pp. 735–749.

[32] *Ns3-Load-Balance*. Accessed: Aug. 16, 2023. [Online]. Available: https://github.com/snowzjx/ns3-load-balance?tab=readme-ov-file

[33] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella, "On the impact of packet spraying in data center networks," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2130–2138.

[34] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. 7th USENIX Conf. Networked Syst. Design Implement.*, 2010, pp. 89–92.

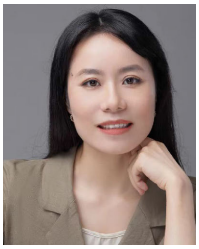
[35] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," in *Proc. 7th Conf. Emerg. Netw. Exp. Technol.*, Dec. 2011, pp. 1–12.

[36] J. Hu, J. Huang, W. Lv, W. Li, J. Wang, and T. He, "TLB: Traffic-aware load balancing with adaptive granularity in data center networks," in *Proc. 48th Int. Conf. Parallel Process.*, Aug. 2019, pp. 1–10.

[37] C. H. Song, X. Z. Khooi, R. Joshi, I. Choi, J. Li, and M. C. Chan, "Network load balancing with in-network reordering support for RDMA," in *Proc. ACM SIGCOMM Conf.*, Sep. 2023, pp. 816–831.

[38] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," in *Proc. ACM SIGCOMM Conf.*, Aug. 2011, pp. 266–277.

[39] R. Mittal et al., "Revisiting network support for RDMA," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2018, pp. 313–326.



Jinbin Hu (Member, IEEE) received the B.E. and M.E. degrees from Beijing Jiao Tong University, China, in 2008 and 2011, respectively, and the Ph.D. degree in computer science from Central South University, China, in 2020. She is currently a Post-Doctoral Researcher with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, and the School of Computer and Communication Engineering, Changsha University of Science and Technology, China. Her current research interests include data center networks, RDMA networking, and learning-based network systems.

Yi He is currently pursuing the M.E. degree with the School of Computer and Communication Engineering, Changsha University of Science and Technology, China. His research interests include data center networks.

Wangqing Luo is currently pursuing the M.E. degree with the School of Computer and Communication Engineering, Changsha University of Science and Technology, China. His research interests include data center networks.

Jiawei Huang received the bachelor's degree from the School of Computer Science, Hunan University, in 1999, and the master's and Ph.D. degrees from the School of Computer Science and Engineering, Central South University, China, in 2004 and 2008, respectively. He is currently a Professor with the School of Computer Science and Engineering, Central South University. His research interests include performance modeling, analysis, and optimization for wireless networks and datacenter networks.

Jin Wang (Senior Member, IEEE) received the M.S. degree from Nanjing University of Posts and Telecommunications, China, in 2005, and the Ph.D. degree from Kyung Hee University, South Korea, in 2010. He is currently a Professor with Hunan University of Science and Technology. He has published more than 400 international journals and conference papers. His research interests include wireless ad hoc and sensor networks, data center networks, and network performance analysis and optimization. He is a Fellow of IET.