LBoDSN: An In-network Load Balancing Mechanism for Lossless Data Center Networks Based on Direct Switch Notification

Qingyu Shi[1,2(✉)], Fangxue Jiang[1], Huang Huang[2], Xiaocui Li[1,2], Chuang Li[1,2(✉)], Wenzhi Cao[1,2], and Limei Liu[1,2]

[1] School of Advanced Interdisciplinary Studies, Hunan University of Technology and Business, Changsha, China
qingyushi@hust.edu.cn, chuangli@hutb.edu.cn
[2] Xiangjiang Laboratory, Changsha, China

Abstract. With the extensive deployment of Remote Direct Memory Access (RDMA) in lossless data center networks (DCNs), enhancing the RDMA load balancing performance for distributed AI training and HPC applications becomes particularly critical. However, existing load balancing schemes either suffer difficulty in responding to congestion in sub-RTT time, or cannot accurately detect and reroute flows that are on the verge of creating path congestion for Priority-based Flow Control (PFC) enabled lossless DCNs. In this paper, we propose LBoDSN, an in-network load balancing mechanism for lossless DCNs based on direct swtich notification, to address above challenges. LBoDSN monitors ingress queue length evolution at destination switches to predict the triggering time of PFC pause, and accurately identifies congested flows based on the congestion contribution level before PFC pause, then further proactively sends the flow congestion notification (FCN) to source switches for fast rerouting. The FCN contains a flow ID that identifies the congested flow, and a path ID that identifies the target path to which the congested flow will be rerouted. And after rerouting, the Congestion Notification Packet (CNP) of the old path is selectively discarded at source switches to improve transmission performance, while out-of-order packets are reordered at the destination switch. The experimental results show that under realistic workloads, LBoDSN achieves 9-64% and 20-80% better than CONGA for the average and tail Flow Completion Times (FCTs), respectively. Compared with ConWeave, LBoDSN achieves around 8% better performance for the average and tail FCTs while significantly reducing switch queue usage for reordering.

Keywords: Data center networks · Lossless networks · Network load balancing.

1 Introduction

Modern data centers are typically using RoCEv2[23] or InfiniBand[1] to construct RDMA lossless networks to support distributed storage, high-performance computing (HPC), and distributed AI training[5, 7, 20, 4, 12]. To further improve the

transmission performance of lossless networks in data centers, researchers have proposed a series of transmission control and load balancing schemes to optimize the RDMA communication between servers[16, 15, 13, 19]. Among them, the goal of load balancing schemes is to evenly distribute the network traffic between nodes across multiple paths, as there are multiple end-to-end paths between any two server racks in data center network topologies (e.g., leaf-spine). However, current studies have shown that Equal Cost Multiple Path (ECMP) forwarding[8], as the standard load balancing strategy in DCNs, falls short due to hash collisions and its inability to adapt to dynamic traffic[3, 21, 19]. Therefore, numerous schemes have proposed new load balancing mechanisms to optimize network performance, such as congestion sensing, packet spraying, flowlet scheduling, etc. Moreover, it is often necessary to consider the impact of PFC mechanism on the load balancing in lossless networks, where downstream network devices notify upstream network devices to pause/send packets based on the local queuing length through PFC pause/resume frames[10, 19, 9].

In recent years, though many schemes utilized in lossy networks in data centers, such as CONGA[3] and LetFlow[21], can improve the multi-path transmission efficiency in lossless networks, they easily aggravate PFC pausing and PFC congestion spreading without considering the PFC mechanism, leading to suboptimal performance. And thus some load balancing mechanisms for lossless networks have been implemented in programmable switches, greatly improving transmission performance by considering the PFC mechanism. Several schemes implemented in source switches perform load balancing based on probe information such as link utilization and delay (e.g., ConWeave[19]). Besides, some other schemes at destination switches monitor ingress queue length to predict the PFC pause time and actively notify the source switch to perform load balancing before PFC pause (e.g., RLB[9]).

However, current load balancing schemes for lossless network still have two drawbacks that lead to performance losses: 1) the schemes implemented in source switches usually takes one round-trip time (RTT) at least to perceive congestion signals and cannot timely respond to congestion; 2) the schemes monitoring congestion at destination switches do not accurately identify the congested flow causing path congestion and the congestion notification information sent to the source switch is not precise enough, which may reroute uncongested flows and cause suboptimal performance.

Therefore, we proposes a novel in-network load balancing mechanism for lossless DCNs using direct switch notification based on programable switches, denoted as LBoDSN, to address above challenges. LBoDSN monitors ingress queue length evolution at destination switches to predict the triggering time of PFC pause, and identifies congested flows based on the congestion contribution level before PFC pause, then further proactively sends the flow congestion notification (FCN) to source switches for fast rerouting. The FCN contains a flow ID that identifies which flow should be rerouted, and a path ID that identifies which path should be rerouted to. And after rerouting, the CNP (Congestion Notification Packet)[23] of the old path is selectively discarded at source switches to

improve transmission performance, while out-of-order packets are reordered at the destination switch. In summary, we have made three principal contributions:

- According to our analysis, we meticulously analyze the limitations of existing load balancing schemes for lossless networks in data centers and reveal the need for faster and more precise load balancing mechanism.
- We propose LBoDSN, a lossless network load balancing mechanism for data centers based on DSN. LBoDSN enables quick sensing of the congested flow and the target path to which the congested flow will be rerouted before PFC pause. LBoDSN provides sub-RTT rerouting time and more precise rerouting for RDMA flows. Besides, after rerouting, LBoDSN prevents the impact of CNPs of the old path on the transmission rate and guarantees packet reordering.
- We implemented LBoDSN using NS3[2], a widely-used network simulation platform, and conducted a performance comparison against existing lossless network load balancing schemes including CONGA, LetFlow, DRILL, and ConWeave. The experimental results demonstrate the effectiveness of LBoDSN in reducing the FCT and the switch queue usage.

2 Related Work and Motivation

Existing lossless networks in PFC-enabled data centers usually employ the hop-by-hop flow control mechanism. When the switch ingress port queue length exceeds the PFC pause threshold, it sends a PFC pause message to the upstream switch's relevant port (or priority queue) to pause data transmission, and when the ingress port queue length decreases to the PFC resume threshold, it sends a PFC resume message to the upstream switch to resume data transmission. However, as PFC pause is based on ports or queues, the coarse-grained feedback on congestion can lead to issues such as congestion spreading[19, 9]. Although existing load balancing schemes are capable of sensing PFC pause events, they still have negative impacts on performance due to their untimely and imprecise in sensing congestion and making load balancing decisions.

Link utilization-based load balancing schemes (e.g., CONGA[3], HULA[11]) cannot sense the PFC pause/resume message, where a path with low link utilization due to PFC pausing maybe classified as a good path without congestion. Delay-based load balancing schemes (e.g., Hermes[22], ConWeave[19]) need one RTT time at least to receive congestion signals and the probe reply can be blocked by PFC pause, which causes slow responds to congestion. Some solutions (e.g., Proteus[10]) sense path states using RTT and link utilization at the source switch to guide initial path selection and makes more fine-grained rerouting decisions using sub-RTT level signals (e.g., cumulative sojourn time) when encountering PFC pause events. Moreover, some schemes implemented in destination switches (e.g., RLB[9]) can predict the trigger time of PFC pause to sense congestion faster than the above described schemes. However, current mechanisms cannot distinguish which flows cause path congestion before PFC

pause, and may schedule uncongested flows to other suboptimal paths, resulting in performance loss.

Furthermore, the existing schemes for lossless DCNs do not pay attention to the impact of the congestion feedback signal CNP (Congestion Notification Packet)[23] from multiple paths on the transmission rate after rerouting. Because the sender with DCQCN enabled adjusts sending rate according to CNPs from the receiver, the flow sending rate after rerouting can be affected by CNPs generated before rerouting.

Therefore, it is very necessary to study how to sense congestion within one RTT time and reroute flows leading to path congestion before PFC pause, as well as to avoid the effect of CNPs on transmission rate of old path. In this paper, we proposes LBoDSN to solve above problems. The design of LBoDSN is detailed in the next section.

3 Design

3.1 Overview of LBoDSN

LBoDSN is an in-network load balancing mechanism based on programable switches for lossless DCNs. LBoDSN incorporates congestion sensing at the destination switch to identify congested flows and directly notifies the source switch to perform load balancing, as shown in Fig. 1. The algorithm of this scheme is specifically deployed on both source and destination switches. It consists of two main components, namely congestion perception and rerouting decision module at the destination switch and rerouting execution module at the source switch.

The Congestion perception and rerouting decision module encompasses several key functions: predicting PFC pauses, identifying congested flows, selecting rerouting path, and sending flow congestion notification (FCN). The FCN contains a flow ID that identifies the congested flow, and a path ID that identifies the target path to which the congested flow will be rerouted. The functions of rerouting execution module include the allocation of new flows, management of FCNs, and handling of CNPs. Besides, LBoDSN reroutes congested flows to specified paths according to the information of FCNs received at the source switch.

3.2 Congestion Perception and Rerouting Decision

This module initially checks whether the ingress queue length exceeds a given threshold at the current rate and only performs predictions when network congestion occurs. Upon detecting congestion, LBoDSN follows a streamlined process: (1) it predicts if the ingress queue length exceeds a predefined threshold below the PFC pause level; (2) activates the congestion flow identification mechanism; and (3) dispatches a direct notification to the upstream source switch.

Predicting PFC pauses. This function predicts the onset of PFC pause by monitoring the rate of increase in the ingress queue length at the destination

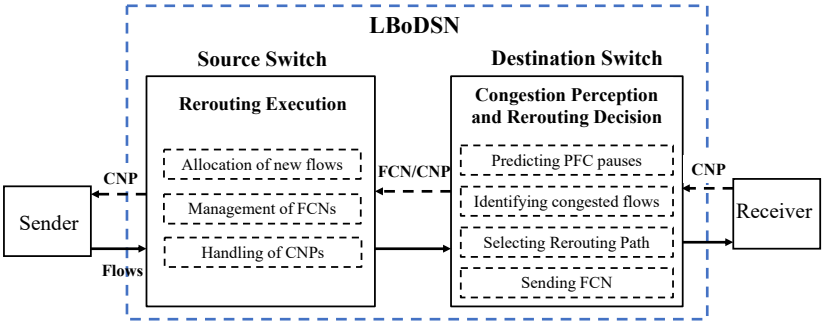


Fig. 1: LBoDSN overview.

switch. Specifically, the scheme leverages flow monitoring techniques on programmable switches to obtain data such as link latency and queue length. We assume that an $n:1$ ratio is used to simulate the incast scenario and n flows are forwarded simultaneously from the source switch to the same destination switch. LBoDSN triggers PFC pause prediction mechanism after t_d that is the link delay for transmitting FCNs from the destination switch to the source switch when the ingress queue length increases to Q_{th}. The packet arrival rate at the destination switch ingress port can be denoted as λ_t and the forwarding rate as μ_t. The queue length threshold Q_{th} is calculated as Eq. (1) shows:

$$Q_{th} = (Q_{PFC} - (\lambda_t - \mu_t) \cdot t_d) \cdot \varepsilon \tag{1}$$

where Q_{PFC} is the threshold for triggering a PFC pause in the ingress port queue of the destination switch, ε is a multiplicative factor between 0 and 1. When C is the link bandwidth, we can predict the range of the Q_{th} value, which can be expressed in Eq. (2):

$$Q_{th} \in [\lfloor t_d \cdot C \rfloor, \lfloor (Q_{PFC} - C \cdot t_d) \cdot \varepsilon \rfloor] \tag{2}$$

In our experimental results, the above Q_{th} threshold is effective and robust in our simulations when we set the multiplicative variable ε around 0.8.

Identifying congested flows. When the ingress queue length of the destination switch increases to Q_{th}, the congested flow causing a large accumulation of the queue length are immediately identified. Specifically, inspired by FlowSail[14], LBoDSN calculates the flow congestion contribution threshold F_{cc} at the destination switch to identify the congested flow as Eq. (3) shows, where each flow is assigned the same weight, i.e:

$$F_{cc} = Q \gg \lceil \log_2 Q_T[qInd].fNum \rceil \tag{3}$$

where Q represents the transient ingress queue length of the destination switch, Q_T is the queue table maintained with the switch, $qInd$ is the index of the indicated queue, and $fNum$ is the number of flows in this queue. When the queued

size of a flow in current ingress queue exceeds the flow congestion contribution threshold F_{cc}, this flow is identified as a congested flow with a high contribution to the imminent triggering of the PFC pause, and its $flowID$ is recorded as one of the carry information for the next FCN. Besides, since it takes at least one RTT time for the FCN to arrive at the source switch and the packet after rerouting to arrive at the destination switch, LBoDSN generates at most one new FCN in one RTT time for the same ingress port at the destination switch to ensure stable load balancing performance.

Selecting rerouting path. In the next step, LBoDSN selects the rerouting path for the congested flow at the destination switch. Concretely, LBoDSN leverages the well-known power-of-two-choices technique[17, 22] to avoid the herd behavior, and LBoDSN defines the new overall path delay T_d to sense the least load path, which is expressed in Eq. (4):

$$T_d = T_{path} + T_{PFC} \tag{4}$$

where T_{path} represents the original path delay monitored by the destination switch and T_{PFC} represents the delay required for PFC resume. When the congested flow is identified, calculating the overall path delay T_d for each path. For the ingress queue that is in PFC pause, based on the queue length trend and the PFC resume threshold, we can predict the time required for PFC resume, that is:

$$T_{PFC} = \frac{Q - Q^*_{PFC}}{-\Delta L} + T_{re\_flight} \tag{5}$$

Where Q is the transient queue length of ingress port of destination switch, Q^*_{PFC} is the PFC resume threshold of ingress queue at the destination switch, and T_{re_flight} measured by INT[15] is the time for the PFC resume frame to arrive at the source switch. ΔL denotes the growth gradient of the ingress queue length of the destination switch. For flows that are not in the PFC pause path, T_{PFC} is 0.

Sending congestion notification. When LBoDSN has predicted the impending PFC pause and identifies the congested flow and rerouting path at the destination switch, the next step is to send the FCN to the source switch, which can be generated by packet mirroring replication of the programmable switch. The FCN packet contains a flow ID that identifies the congested flow, a path ID that identifies the target path to which the congested flow will be rerouted, and a one-bit congestion tag with value of 1. If the ingress queue length drops below the Q_{th} threshold and the queue growth gradient is small, LBoDSN sends the FCN containing the current path ID and the congestion tag with value of 0 to the source switch to inform this path is in an uncongested state.

3.3 Rerouting Execution

Rerouting execution implemented at the source switch needs to mange new flows and the congestion feedback information of FCNs and CNPs. The new flow selects its transmission path according to a random algorithm based on path classification. Besides, the congested flow is identified by the FCN, which contains its

Table 1: Outcome of path classification using FCN and link utilization.

FCN	Link Utilizaiton	Reason	Characterization
No FCN or FCN with a congestion tag of 0	Low/Moderate	No PFC pause and under-utilized link	Uncongested path
	High	Fully-utilized link with no congested flow	
FCN with a congestion tag of 1	Low/Moderate	Prone to cause PFC pause and congestion is easing	Undetermined path
FCN with a congestion tag of 1	High	Prone to cause PFC pause and congestion is happening	Congested path

flow ID and a path ID that identifies the target path to which it will be rerouted. Moreover, LBoDSN selectively discards CNPs of old path at the source switch to prevent the CNP generated by the old path affecting the transmission efficiency of flows in the new path after rerouting.

Allocation of new flows. When the source switch selects paths for new flows arriving at the switch, LBoDSN leverages a random selection method based on path classification to evenly allocate new flows to different paths as much as possible. Firstly, we classify parallel paths into the following three types based on the congestion feedback of FCNs and link utilization: (1) uncongested path that has not received an FCN or has received an FCN with a congestion tag of 0; (2) undetermined path that has received an FCN with a congestion tag of 1 and has low or moderate link utilization; (3) congested path that has received an FCN with a congestion tag of 1 and has full link utilization. Table 1 summarizes the outcome of path classification and the reasons behind them.

For each new flow, LBoDSN first tries to randomly select a rerouting path from the set of uncongested paths. If the set is empty, LBoDSN further checks the set of undetermined paths. If that fails as well, LBoDSN makes a random selection of a path from among those possessing the three least loaded local queues.

Management of FCNs. When the network is congested, the source switch generally receives FCNs from the destination switch. The rerouting execution module will manage FCNs according to the relative congestion tag value. If the congestion tag value is 1, the path will be recorded as having received the FCN with a congestion tag of 1 as Table 1 shows. Besides, LBoDSN extracts flow id fx and path id Px from the FCN, modifies the flow fx routing path to Px, and the subsequent packets of flow fx will be forwarded from path Px. If the congestion tag value is 0, the flow is recorded as a state of cleared congestion, and when all the congested flows in this path have entered the state of cleared congestion, the path state will be recorded as having received the FCN with a congestion tag of 0.

Handling of CNPs. After rerouting, LBoDSN selectively discards CNPs of old paths (denoted as stale CNPs in the following sections) at source switches to improve transmission performance, while out-of-order packets are reordered at the destination switch like ConWeave[19].

First, we distinguish stale CNPs through a direct notification from destination switches. Specifically, after rerouting, the source switch adds a one-bit special tag to the tail of the first packet on the new path as a notification message based on the INT technique[15]. When the notification message reaches the destination switch, the destination switch firstly replicates the message by using

the ingress mirroring technique and mapping its source-destination IP address, while removing the payload and adding the same tag with the source packet to generate a notification reply message to the source switch. When the next CNP of this flow arrives at the source switch, verify whether its notification reply message has been returned. If not, the CNP is regarded as a stale CNP and will be selectively discarded at the source switch. Otherwise, it will be forwarded to the sender.

The above selectively discarding stale CNPs is applied to avoid that congestion control algorithms over-adjust the flow rate in new paths after rerouting. Considering that the sender under the DCQCN protocol reduces the flow rate based on CNPs and increases the flow rate according to timers and byte counters, discarding all stale CNPs will result in excessive growth of the flow rate, which causes congestion on the new path. And not discarding stale CNPs at all will result in too low flow rate, which causes under-utilization of bandwidth on the new path. Therefore, we implement a model that discards stale CNPs based on probability to smooth the flow rate control at the sender side. Firstly, calculate the average $avgCNP$ of stale CNPs, as Eq. (6) shows:

$$avgCNP = (1 - w_q) \cdot avgCNP + w_q \cdot W \tag{6}$$

Where W is actual number of stale CNPs for this flow, $w_q (0 \leqslant w_q \leqslant 1)$ is a weight value. Meanwhile, we set a threshold T and calculate the discarding probability p of stale CNPs, as Eq. (7) shows:

$$p = \frac{\max P \cdot avgCNP}{T} \tag{7}$$

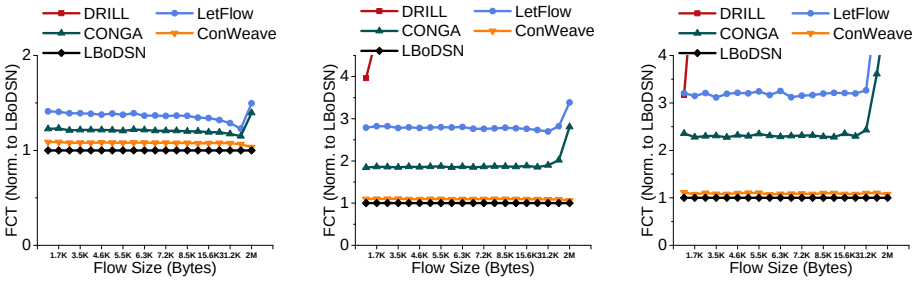
Where $maxP$ is the maximum discarding probability of stale CNPs. When the number of stale CNPs does not exceed the threshold T, the stale CNP is discarded with the discarding probability p, and the discarding probability p increases linearly with the number of arriving stale CNPs. When the number of stale CNPs reaches the threshold T, stale CNPs will be all discarded.

4 Evaluation

For performance evaluation, we compare LBoDSN with the state-of-the-art load balancing mechanisms to investigate the enhancement in performance with NS3 large-scale simulations[2].

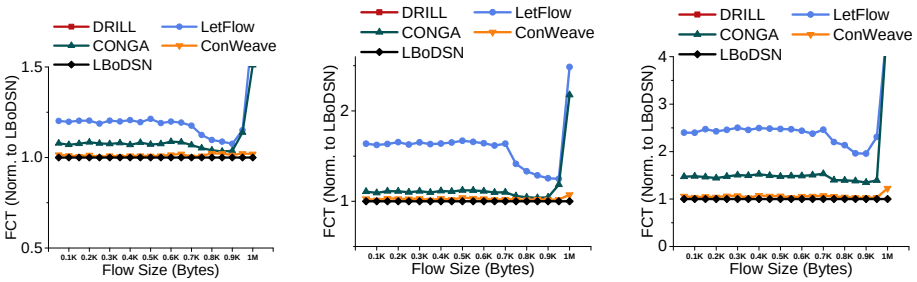
Topology: We construct an 8×8 leaf-spine network topology in NS3, featuring 100 Gbps links and a server count of 128 (16 servers for each leaf switch). This design ensures eight distinct equal-cost paths between any host pair, interconnected through diverse switches. Consequently, we implement a 2:1 oversubscription ratio at the leaf level to meet standard configurations in common DCNs[3].

Workloads: We utilize two realistic workloads (AliCloud storage[19, 15] and Meta Hadoop[19, 18]) derived from operational data centers to simulate traffic



(a) 50% Load (avg) (b) 80% Load (avg) (c) 80% Load (99%-ile)

Fig. 2: FCT for the AliCloud storage workload (normalized to LBoDSN).



(a) 50% Load (avg) (b) 80% Load (avg) (c) 80% Load (99%-ile)

Fig. 3: FCT for the Meta Hadoop workload (normalized to LBoDSN).

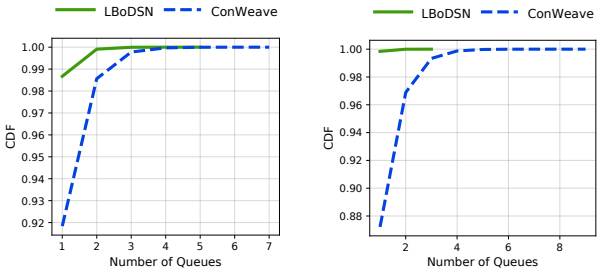
dynamics for our evaluations. To replicate these workloads, we generate flows between randomly selected senders and receivers within various leaf switches, following Poisson processes.

Schemes compared: We compare LBoDSN with ConWeave[19], CONGA[3], Letflow[21], and DRILL[6]. ConWeave performs fine-grained load balancing for RDMA flows in the switch and reorders the out-of-order packets through the in-network reordering mechanism. CONGA and LetFlow implements flowlet switching in the switches, where we choose a flowlet time gap of 100μs. DRILL employs per-packet load balancing based on local queue utilization in switches. In all schemes, we use DCQCN[23] as the standard congestion control scheme and PFC mechanism for lossless DCNs.

Metrics: Similar to previous work, we adopt the average and tail Flow Completion Time (FCT) as the principal metric for evaluating performance, where we normalize the FCT to LBoDSN in order to better visualize the results.

4.1 Reduction in FCT

We conducted experiments under average traffic loads of 50% and 80% in different workloads, corresponding to a moderately and highly congested network,



(a) AliCloud storage (b) Meta Hadoop

Fig. 4: Number of queues usage per egress port (80% Avg.Load).

respectively. We show the average and tail FCTs in Fig. 2 and Fig. 3. In some of the test results, the performance metrics for DRILL are omitted, as incorporating their FCTs would necessitate a large adjustment to the scale for proper representation.

As Fig. 2 and Fig. 3 show, LBoDSN outperforms all other schemes. LBoDSN achieves around 8% better performance than ConWeave for the average and tail FCTs. LBoDSN predicts the triggering time of PFC pause and identifies congested flows based on the congestion contribution level before PFC pause at destinations, then further proactively sends FCNs to source switches for fast rerouting. ConWeave detects path congestion based on continuous RTT monitoring at source switches and does not distinguish the congestion state of different flows on the congested path to make differentiated load balancing decisions. Therefore, LBoDSN can sense congested flows and select rerouting path faster at the destination switch than ConWeave. And LBoDSN solves the performance damage caused by stale CNPs, thereby improving the transmission performance. Besides, LBoDSN performs 9-64% and 20-80% better than CONGA for the average and tail FCTs, respectively. Compared with LetFlow, LBoDSN improves the average and tail FCTs by 18-70% and 29-83%, respectively. The results show that LBoDSN can effectively improve the performance by sensing the trend of PFC pause and rerouting congested flows in advance. CONGA and LetFlow distribute flowlets at source switches but cannot sense PFC pause/resume events, which degrades performance for lossless DCNs. For example, a path with low link utilization because of PFC pausing is mistaken for a good path in CONGA. Therefore, LBoDSN can outperform CONGA and LetFlow in our experiments.

4.2 Reduction in Queues Usage

Rerouting at the source switch will produce out-of-order packets. As RDMA is highly sensitive to out-of-order packets in lossless DCNs[19], LBoDSN employs the same mechanism as ConWeave to reorder out-of-order packets at the destination switch, which requires switch queues for reordering. Queues are a valuable hardware resource for switches and using too many queues may affect other ser-

vices. Therefore, we monitor the number of queues used per switch egress port for schemes in our experiments to check the feasibility of LBoDSN.

As Fig. 4 shows, under both workloads, LBoDSN uses fewer queues than ConWeave and up to five additional queues. This is because LBoDSN only reroutes congested flows which have a large probability of causing path congestion or PFC pause, and ConWeave reroutes flows more frequently. Compared with ConWeave, LBoDSN achieves more fine-grained congestion sensing and more cautious rerouting decisions. Therefore, LBoDSN can reduce the queue usage of switches to achieve less hardware resource consumption.

5 Conclusion

In this work, we propose LBoDSN, which realizes in-network load balancing for lossless DCNs using direct switch notification based on programable switches. LBoDSN monitors ingress queue length evolution at destination switches to predict the triggering time of PFC pause, and identifies congested flows based on the congestion contribution level before PFC pause, then further proactively sends the FCN to source switches for fast rerouting. Besides, after rerouting, the CNP of the old path is selectively discarded at source switches to improve transmission performance, while out-of-order packets are reordered at destination switchs. Simulation experiments under realistic workloads strongly confirm the effectiveness of LBoDSN in reducing the FCT and the switch queue usage.

6 Acknowledgments

This work was supported by the National Key Research and Development Program of China (2021YFC3300603, 2023YFC3306204); the National Natural Science Foundation of China (62376092); the MOE (Ministry of Education in China) Project of Humanities and Social Sciences (23YJCZH183); the Natural Science Foundation of Hunan Province of China (2022JJ40129, 2023JJ40236); the Open Project of Xiangjiang Laboratory (23XJ01012, 22XJ03014); and the Interdisciplinary Research Project of Hunan University of Technology and Business (2023SZJ16).

References

1. Infiniband trade association. infinibandtm architecture specification volume 1 release 1.3. March 2015
2. NS3. Accessed August 13, 2024. [Online]. Available: https://www.nsnam.org/
3. Alizadeh, M., et al.: CONGA: Distributed congestion-aware load balancing for datacenters. In: Proceedings of the ACM SIGCOMM. pp. 503–514 (2014)
4. Bai, W., et al.: Empowering azure storage with RDMA. In: Proceedings of the USENIX NSDI. pp. 49–67 (2023)
5. Gao, Y., et al.: When cloud storage meets RDMA. In: Proceedings of the USENIX NSDI. pp. 519–533 (2021)

6. Ghorbani, S., Yang, Z., Godfrey, P.B., Ganjali, Y., Firoozshahian, A.: DRILL: Micro load balancing for low-latency data center networks. In: Proceedings of the ACM SIGCOMM. pp. 225–238 (2017)
7. Guo, C., et al.: Rdma over commodity ethernet at scale. In: Proceedings of the ACM SIGCOMM. p. 202–215 (2016)
8. Hopps, C.: Analysis of an equal-cost multi-path algorithm. RFC 2992 (2000)
9. Hu, J., He, Y., Wang, J., Luo, W., Huang, J.: Rlb: Reordering-robust load balancing in lossless datacenter networks. In: Proceedings of the ACM ICPP. p. 576–584 (2023)
10. Hu, J., et al.: Enabling load balancing for lossless datacenters. In: Proceedings of the IEEE ICNP. pp. 1–11 (2023)
11. Katta, N., Hira, M., Kim, C., Sivaraman, A., Rexford, J.: HULA: Scalable load balancing using programmable data planes. In: Proceedings of the ACM SOSR. p. 10 (2016)
12. Lao, C., et al.: ATP: In-network aggregation for multi-tenant learning. In: Proceedings of the USENIX NSDI. pp. 741–761 (2021)
13. Li, W., et al.: Flow scheduling with imprecise knowledge. In: Proceedings of the USENIX NSDI. pp. 95–111 (2024)
14. Li, W., Zeng, C., Hu, J., Chen, K.: Towards fine-grained and practical flow control for datacenter networks. In: Proceedings of the IEEE ICNP. pp. 1–11 (2023)
15. Li, Y., et al.: Hpcc: high precision congestion control. In: Proceedings of the ACM SIGCOMM. p. 44–58 (2019)
16. Lu, Y., et al.: Multi-Path transport for RDMA in datacenters. In: Proceedings of the USENIX NSDI. pp. 357–371 (2018)
17. Mitzenmacher, M.: The power of two choices in randomized load balancing. IEEE Transactions on Parallel and Distributed Systems (TPDS) **12**(10), 1094–1104 (2001)
18. Roy, A., Zeng, H., Bagga, J., Porter, G., Snoeren, A.C.: Inside the social network's (datacenter) network. In: Proceedings of the ACM SIGCOMM. p. 123–137 (2015)
19. Song, C.H., Khooi, X.Z., Joshi, R., Choi, I., Li, J., Chan, M.C.: Network load balancing with in-network reordering support for rdma. In: Proceedings of the ACM SIGCOMM. p. 816–831 (2023)
20. Taranov, K., Byan, S., Marathe, V., Hoefler, T.: Kafkadirect: Zero-copy data access for apache kafka over rdma networks. In: Proceedings of the ACM SIGMOD. p. 2191–2204 (2022)
21. Vanini, E., Pan, R., Alizadeh, M., Taheri, P., Edsall, T.: Let it flow: Resilient asymmetric load balancing with flowlet switching. In: Proceedings of the USENIX NSDI. pp. 407–420 (2017)
22. Zhang, H., Zhang, J., Bai, W., Chen, K., Chowdhury, M.: Resilient datacenter load balancing in the wild. In: Proceedings of the ACM SIGCOMM. pp. 253–266 (2017)
23. Zhu, Y., et al.: Congestion control for large-scale rdma deployments. In: Proceedings of the ACM SIGCOMM. p. 523–536 (2015)