DLBF: A Dynamic Load Balancing Framework Resilient to Packet Reordering for RDMA

Xingjian Zhang[1], Jiaxue Liu[1], Yiran Zhang[1,2], Hu Zhang[2,3], Shangguang Wang[1]
[1]Beijing University of Posts and Telecommunications
[2]Key Laboratory of Computing Power Network and Information Security, Ministry of Education,
Shandong Computer Science Center (National Supercomputer Center in Jinan),
Qilu University of Technology (Shandong Academy of Sciences)
[3]Shandong Provincial Key Laboratory of Computing Power Internet and Service Computing,
Shandong Fundamental Research Center for Computer Science

***Abstract*—To accommodate the diverse requirements of co-existing AI applications and background workloads, datacenter networks often employ hybrid load balancing strategies: Equal-Cost Multi-Path (ECMP) for background flows and Random Packet Spraying (RPS) for AI traffic. However, in this paper, we show that ECMP-induced hotspots create severe path-delay asymmetry, leading to excessive packet reordering for RPS flows. This reordering is misinterpreted by the RoCEv2 transport layer as congestion signals, causing unnecessary rate reduction and significant throughput degradation. To address these problems, we propose ynamic Load Balancing Framework (DLBF), an end-host load-balancing mechanism. DLBF dynamically migrates traffic away from hotspots based on per-path congestion signals. Further, by transmitting duplicate packets based on the delay gap between old and new paths, DLBF eliminates reordering during path switching without violating Remote Direct Memory Access (RDMA) semantics. Last, DLBF utilizes lightweight notifications to probe path states without aggravating congestion. Large-scale NS-3 simulations demonstrate that DLBF reduces the average Flow Completion Time (FCT) by $1.1\times$–$1.7\times$ and lowers P99 tail latency by $1.2\times$–$3.4\times$ compared to existing schemes.**

***Index Terms*—Load Balancing, Packet Reordering, RDMA**

I. Introduction

As large-scale AI models continue to scale toward trillion-parameter regimes, datacenter infrastructures are evolving from general-purpose computing clusters into highly specialized AI super-clusters [3], [4]. The network has become a critical determinant of performance, as distributed training workloads impose stringent demands on both bandwidth and latency [5]. In production environments, AI datacenters commonly operate under a regime of heterogeneous traffic coexistence, where diverse application types share the same fabric. Specifically, AI training workloads require high throughput to maintain computational efficiency, whereas critical background workloads—such as storage backup, control signaling, and legacy applications—require stable, in-order delivery to maintain system reliability.

Given the distinct traffic characteristics mentioned above, datacenter networks are deploying differentiated load balancing mechanisms within the same physical architecture. For instance, ECMP remains the mainstream choice for in-order delivery of background storage workload. For bandwidth-sensitive AI training traffic, which suffers from link under-utilization caused by ECMP hash collisions [6], [8], packet-level load balancing mechanisms such as RPS are widely adopted on end hosts/RDMA NICs to leverage multipath diversity [1]. Nowadays, Nvidia Spectrum-X supports packet spraying [2], and the Ultra Ethernet Consortium (UEC) has formally adopted packet spraying as a core architectural feature of the Ultra Ethernet Transport (UET) protocol [12].

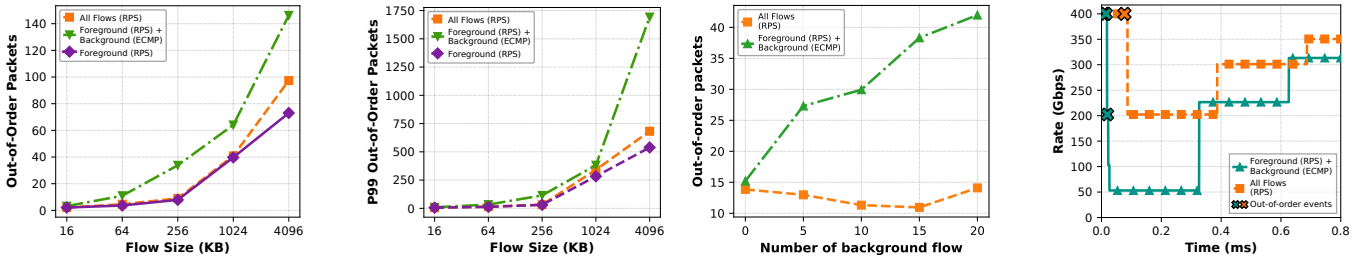
However, the existing RPS scheme usually assumes a single, homogeneous workload in the network. In this paper, we experimentally show that RPS can indeed perform well (i.e., with little packet reordering) in the absence of background traffic but degrades significantly when coexisting with background ECMP traffic. Background traffic routed via ECMP can lead to persistent congestion hotspots on certain paths, resulting in highly asymmetric delays across paths. Thus, when foreground AI traffic uses RPS to spray packets across these unevenly delayed paths, the order of packet arrivals at the receiver is disrupted. However, the most detrimental consequence is the collapse of transmission rate: the RoCEv2 protocol interprets the reordering as a congestion signal [9], [10]. Such "false" congestion signals triggered by frequent out-of-order events cause the sender to reduce its rate frequently, directly leading to a decline in AI training throughput (Section II).

In light of this, we propose DLBF, an end-host load balancing framework resilient to packet reordering caused by asymmetric path delays in RDMA networks. DLBF adopts an adaptive path switching mechanism to dynamically migrate traffic away from hotspot paths based on fine-grained per-path congestion signals. What's more, leveraging the property that RoCEv2 silently discards duplicate packets while not affecting the transmission rate, the DLBF sender selectively transmits duplicate packets that are likely to arrive out-of-order based on the delay gap between the old and new path. Thus, the receiver side only sees an in-order arrival sequence, masking reordering caused by path switching. Finally, to avoid exacerbating congestion caused by blindly sending duplicate packets to unknown-state paths, DLBF introduces an optimization that transmits only a notification packet on the new path.

We evaluate DLBF via large-scale simulations based on NS-3. Results demonstrate that, compared with ECMP and original RPS schemes, under typical foreground AI collective communication workload, DLBF reduces the average FCT by 1.1-1.7$\times$ and lowers the P99 tail latency by 1.2-3.4$\times$.

II. Motivation

In this section, to quantify the impact of realistic background ECMP traffic on foreground random spraying traffic, we conduct an experimental study using NS3 simulations. The



(a) Average number of out-of-order packets per QP

(b) P99 number of out-of-order packets per QP

(c) Impact of the number of background flows

(d) Impact on sending rate

Fig. 1: Packet reordering and rate degradation issue.

topology adopts FatTree (k=8), consisting of 16 core switches, 32 aggregation switches, 32 Top-of-Rack (ToR) switches, and 320 servers. All links operate at 400 Gbps with a propagation delay of 100 ns. DCQCN [10] is enabled by default. Selective Repeat (SR) is also enabled on RDMA NICs.

Foreground and background workloads: For the foreground random-spraying traffic, among the 320 servers, 256 are randomly selected to act as Mixture-of-Experts (MoE) sender nodes, and 8 servers are randomly selected as receiver nodes. This forms 8 parallel aggregation groups, where each receiver collects data from a subset of the 256 senders. The foreground traffic uses message sizes ranging from 16 KB to 4 MB, generating a total of 256 point-to-point flows. For the background ECMP traffic, the remaining 64 servers generate 20 ECMP background flows of 100 MB each using a random pairwise exchange pattern.

Performance metrics: We measure the extent of reordering by counting packets that arrive out of order. Specifically, we periodically sample the out-of-order buffer of each Queue Pair (QP) and record the number of packets stored due to out-of-order arrivals. Second, we measure the sending rate of the foreground flows at the sender RNIC to observe how reordering affects the sender-side transmission behavior.

ECMP background flows introduce packet reordering for all message sizes. As shown in Figure 1a, when all flows use RPS, the reordering level remains low and stable across all message sizes. A similar trend appears when only the MoE foreground uses RPS. In contrast, when RPS foreground flows coexist with ECMP background flows, the number of out-of-order packets increases substantially for every message size, with larger messages showing a more pronounced rise. This experiment demonstrates that ECMP-induced path-delay imbalance directly increases the reordering severity experienced by random-spraying flows.

Reordering issue deteriorates as the number of ECMP background flows increases. Next, we fix the MoE foreground message size at 512 KB and vary the number of ECMP background flows to quantify how background-load intensity affects reordering. As shown in Figure 1c, reordering remains consistently low for all-RPS traffic across all background-flow counts. When ECMP background flows are introduced, however, the number of out-of-order packets steadily increases as the number of background flows grows. This experiment confirms that a heavier background ECMP load amplifies the delay variability across paths and progressively worsens packet reordering for spraying flows.

Out-of-order packets lead to undesirable sending rate decline of the foreground traffic. To further quantify the impact of packet reordering on the RDMA transport protocol, we randomly select an MoE foreground flow and monitor its rate evolution. As shown in Figure 1d, when there is RPS foreground traffic with ECMP background traffic, the MoE flow exhibits a significantly higher frequency of out-of-order events, i.e., six reordering events. With all-RPS enabled, the MoE flow experiences only one reordering event. Because background flows also use RPS, the overall network load remains evenly distributed, indicating that the observed rate degradation is not caused by bandwidth contention. The substantially higher reordering frequency under the mixed RPS and ECMP traffic directly results in a sharper decline in sending rate. The underlying cause is that DCQCN (as well as other similar congestion control mechanisms) cannot distinguish packet reordering from actual packet loss. Therefore, the RNIC treats reordered packets as losses and reduces the sending rate, making reordering the direct driver of the observed degradation. Notably, even with SR enabled, out-of-order arrivals still trigger sender-side rate reduction events.

In conclusion, our experimental observations reveal that path-delay asymmetry induced by background ECMP hotspots can lead to severe packet reordering for foreground RPS traffic. This reordering directly triggers a significant decline in the sending rate. To address these problems, we argue that it is necessary to enhance existing random packet spraying by masking packet reordering in RDMA networks.

III. Design

A. Overview

The core idea of DLBF is to adaptively switch paths for each packet to avoid congestion, while selectively transmitting duplicate packets during path switching to eliminate potential reordering. As shown in Figure 2, there are three main mechanisms:

- **Adaptive path switching:** DLBF actively maintains the congestion state of each path. Once congestion is detected, the sender dynamically selects a new transmission path based on the path states to avoid persistent hotspots (Section III-B).
- **Selective duplicate transmission:** Based on the observation that duplicate packets do not violate the semantics of RDMA transport but can benefit in-order preserving, after a

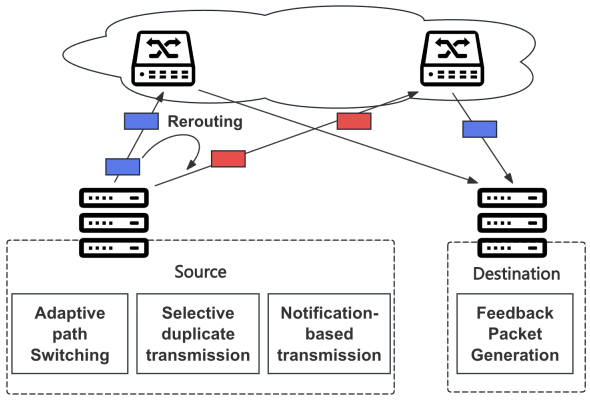


Fig. 2: DLBF Overview

path switching, the sender selectively transmits packets that are likely to be reordered based on the delay gap between the old and new path, maintaining high link utilization while preserving packet order (Section III-C).

• **Notification-based transmission optimization:** To reduce the overhead of transmitting duplicate packets, DLBF supports transmitting lightweight notifications instead of entire packets. Besides, DLBFadaptively determines whether to resume transmitting entire duplicate packets based on the receiver buffer occupancy (Section III-D).

B. Adaptive Path Switching

DLBF performs adaptive path switching to promptly migrate packets away from persistently congested paths and improve overall load distribution. The paths are represented as five-tuples of packets. Then, path switching is achieved by modifying the source port at the sender. DLBFcontinuously monitors path congestion signals and makes routing decisions as soon as persistent congestion is detected.

Concretely, DLBF employs Explicit Congestion Notification (ECN) to detect congestion and assist in path selection [10]. When the queue length at a switch exceeds a predefined threshold, packets are marked with the ECN. The receiver echoes these CE marks in ACK packets, allowing the sender to evaluate congestion status during each round trip.

The DLBF sender maintains a state machine with three states for paths: ***unprobed***, ***available***, and ***unavailable***. All paths are initially marked as *unprobed*, indicating the congestion state is unknown. Once a congestion signal is received on a path, the path is immediately marked as *unavailable*. When a path is selected for transmission and no congestion signal is observed, its state is updated to *available*. To avoid stale congestion information, an available path expires after T_{avail}, while an unavailable path expires after T_{unavail}. After expiration, the path transits to the unprobed state.

When a new active flow arrives or an existing path reports a congestion event, the DLBF sender triggers the path selection procedure in Algorithm 1. DLBF evaluates candidate paths in a strict priority order. The DLBF sender first checks whether the previously selected active path is not marked as *unavailable* and simply reuses it to avoid unnecessary switching (lines 1–2). Then, the DLBF sender randomly chooses one path from the *available* set (lines 3–4). If no *available* path exists, the sender considers *unprobed* paths and randomly selects one to explore newly recovered or untested links (lines 5–6). Finally, when all paths are either *congested* or *unavailable*, it falls back to the old path, indicating that the network is currently saturated and no beneficial rerouting path exists (lines 7–8).

Algorithm 1 Path Selection Algorithm

Input:
$path\_state[]$: The state of each path;
old_port: The previously used path;

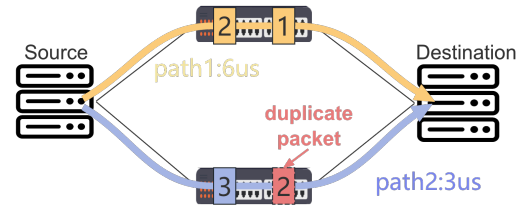
1: **if** old_port exists $\wedge$ path_state[old_port] $\neq$ *unavailable* **then**
2: **return** old_port /* Reuse active path */
3: **else if** $\exists$ port : path_state[port] = *available* **then**
4: **return** random(*available* set)
5: **else if** $\exists$ port : path_state[port] = *unprobed* **then**
6: **return** random(*unprobed* set)
/* Explore unknown path */
7: **else**
8: **return** old_port /* Fallback to old path */
9: **end if**

C. Selective Duplicate Transmission

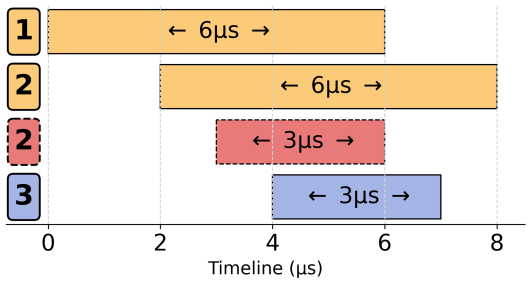
Key observation. Recent research reveals that RDMA RNICs deterministically classify any packet whose PSN is smaller than the locally maintained expected PSN (ePSN) as a duplicate packet [11]. Once detected, the RNIC silently discards the packet without advancing the PSN state, without notifying the transport or upper layers, and while still acknowledging the sender. This behavior indicates that duplicate packets do not violate the correctness of reliable transport or disrupt ongoing flows. We leverage this property to enable selective packet transmitting without additional protocol impact.

Example. As illustrated in Fig. 3, specifically, when a packet P_2 is in flight on the old path while a subsequent packet P_3 is sent on the new path after path switching, the shorter delay of the new path may cause P_3 to arrive earlier than P_2, resulting in out-of-order delivery. In this case, the receiver observes the out-of-order arrival sequence $P_1 \rightarrow P_3 \rightarrow P_2$. To avoid this case, the sender can duplicate the transmission of P_2 on the new path before sending P_3. With selective duplicate transmission enabled, the arrival sequence becomes $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_2$, where the second arrival of P_2 is a duplicate. Although this may introduce duplicate packets at the receiver, the RNIC silently discards duplicates without affecting protocol correctness. In this way, the receiver preserves in-order reception during path transitions.

Thus, the key is to determine whether a packet on the old path is likely to be reordered after a path switching. The sender needs to obtain the one-way delay of each available path. To ensure accurate delay measurement, we rely on hardware-level timestamp at the RNIC. The sender RNIC records the transmission timestamp t_1 when a packet is placed on the wire and is carried in the packet header. Upon packet arrival, the receiver RNIC records the timestamp t_2 and attaches the



(a) Topology and path delays



(b) Packet arrival timeline at the destination

Fig. 3: Example: out-of-order packet caused by path switching and selective duplicate transmission.

corresponding pair (t_1, t_2) to the ACK. When the sender receives the ACK, it computes the one-way delay of that path simply as $(t_2 - t_1)$.

One-way delay vs. Round-Trip Time (RTT). In the absence of time synchronization, the clocks of the sender and receiver RNICs are not aligned, but the resulting clock offset is added to all one-way delay measurements in the same way. Although the absolute value of the one-way delay includes a specific offset, the offset between the same source and destination is approximately constant over a short time window. Therefore, subtracting the one-way delays of two paths naturally cancels out the offset and yields the true delay difference between the paths. In contrast, RTT includes the delays of both the forward and reverse paths, and the reverse path delay often differs across paths. This discrepancy may introduce errors when computing the delay difference between paths, making RTT unsuitable for estimating path delay gaps. Hence, to accurately measure the relative latency between two paths, we rely on one-way delay rather than RTT.

Reordering detection based on one-way delay gap. After obtaining the one-way delays of the old and new paths (denoted as OWD_{old} and OWD_{new}, respectively), the delay gap between the two paths is defined as:

$$\Delta = OWD_{\text{old}} - OWD_{\text{new}}. \tag{1}$$

At the switching time t_{now}, for each unacknowledged packet k, the sender records its transmission timestamp $t_{\text{sent}}(k)$ and computes the time it has already spent on the old path as:

$$\tau_k = t_{\text{now}} - t_{\text{sent}}(k). \tag{2}$$

If the time a packet has spent on the old path is insufficient to compensate for the delay gap between the two paths, the packet may arrive later than packets sent on the new path after the switch, leading to reordering. Therefore, we determine whether a packet is reordering-prone by the condition:

$$\tau_k < \Delta. \tag{3}$$

When the above condition holds, the in-flight packets are likely to arrive at the receiver later than packets transmitted on the new path after the path switching. To prevent potential packet reordering, the sender therefore duplicates this in-flight packet on the new path before transmitting subsequent packets. Otherwise, we infer that the in-flight packet will arrive earlier than the packets sent on the new path and thus will not cause reordering; in this case, duplicate transmission is unnecessary.

D. Notification-based transmission optimization

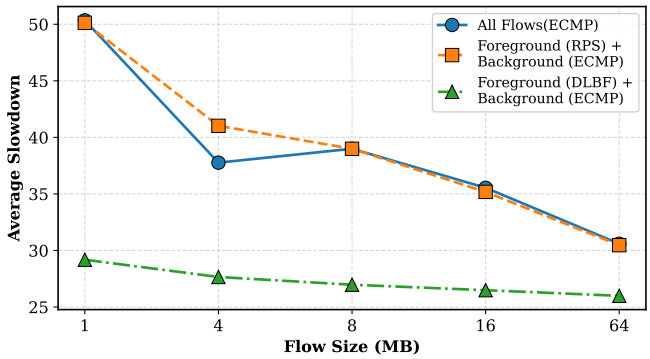
When switching to an unprobed path, the sender cannot accurately determine its congestion state. As a result, the unprobed path may already be congested. If full data packets are transmitted immediately on this path, it may further aggravate congestion, leading to performance degradation. To address this issue, DLBF introduces a notification-based transmission optimization.

Specifically, when the new switching path is an unprobed path, instead of fully transmitting the packets identified by the selective duplicate transmission mechanism (Section III-C), the sender transmits only a notification packet on the new path. This notification packet carries no payload data; it contains only essential control information, including the sender-side port and the range of all packets transmitted over the old path. Likewise, the notification packet can also be marked with ECN. The receiver can infer whether the new path is congested by checking the notification packets.

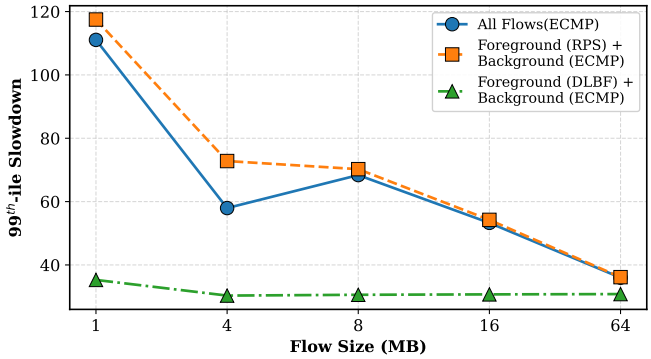
Upon receiving a notification packet, the receiver evaluates the current path state to decide whether to let the sender retransmit the packets that are most likely to cause reordering. The receiver and sender employ SR to conduct retransmission. There are two cases:

Case 1: If the notification packet is marked with ECN, it implies that the new path is congested, and transmitting duplicate packets would further aggravate congestion. The receiver therefore records the corresponding packet sequence and enters a waiting state with a timeout θ. During this interval, any packet reordering that is induced by the incompleteness of the recorded sequence is intentionally ignored. Specifically, while packets within the recorded sequence are still pending, the arrival of subsequent packets with larger packet sequence numbers (PSNs) (e.g., receiving packet 4 when packets 1–3 are not yet fully received) does not trigger Selective Acknowledgment (SACK) generation. Instead, the receiver uses a bitmap indexed by PSNs to track packet reception progress and waits for the remaining in-flight packets on the old path. To prevent indefinite waiting caused by packet loss on the old path, we set an expiration time θ.Only if packets in the recorded sequence are still missing upon the expiration of θ does the receiver treat the situation as a packet-loss event.

Case 2: If the notification packet does not carry any ECN mark, the new path is considered available with sufficient bandwidth. Thus, the receiver immediately sends SACKs for



(a) Average FCT slowdown



(b) P99 FCT slowdown

Fig. 4: FCT performance. Foreground workload: Allreduce.

the out-of-order packets, prompting the sender to transmit only the packets that are most likely to cause reordering.

Although sending notification packets can save network bandwidth by avoiding sending full data packets, in *Case* 1, the receiver must temporarily store packets that arrive out of order. Thus, notification packets can introduce additional receiver-side buffer occupancy. To prevent excessive buffer usage, the receiver employs a receiver buffer threshold-based switching mechanism. When the size of the out-of-order buffer exceeds the threshold, the receiver instructs the sender to suspend transmission of notification packets. The sender then falls back to the original selective duplicate transmission. Once buffer pressure is relieved, the receiver issues a recovery notification and the sender recovers to send notification packets when switching to an unprobed path.

IV. Evaluation

A. Experiment Setup

Network Setup. We use NS-3 simulator to simulate a 2-tier Fat-Tree (Clos) topology with 320 accelerators. The topology follows a $k = 8$ Fat-Tree structure with an over-subscription ratio of 2.5:1, comprising 32 ToR switches, 32 aggregation switches, and 16 spine switches. Each ToR switch connects 10 accelerators. All links operate at 400 Gbps with 100 ns propagation latency. We enable DCQCN as the congestion-control mechanism and turn on SR at the RNIC to support efficient loss recovery during all experiments. We use ASTRA-sim [13] to generate the Allreduce workload as the foreground traffic, configured with 64 accelerators. For background traffic, we randomly instantiate five pairs of flows among the remaining endpoints, and these flows persist throughout the lifetime of the foreground workload.

Compared schemes. For all background traffic, we uniformly apply ECMP as the load-balancing mechanism. Because our design is deployed at the RNIC, the comparison focuses on NIC-side load-balancing schemes. Accordingly, the foreground traffic is evaluated under three schemes: RPS, ECMP, and our proposed DLBF. RPS is implemented by varying the source port in the five-tuple to realize per-packet random path selection.

Metrics. We evaluate performance improvement using flow completion time as the primary metric. In addition, we periodically sample the sending rate at the foreground sender and compute the average rate across the collected samples. This average rate reflects the extent to which packet reordering affects the sender-side transmission behavior. We also record the buffer occupancy caused by out-of-order arrivals.

B. Results

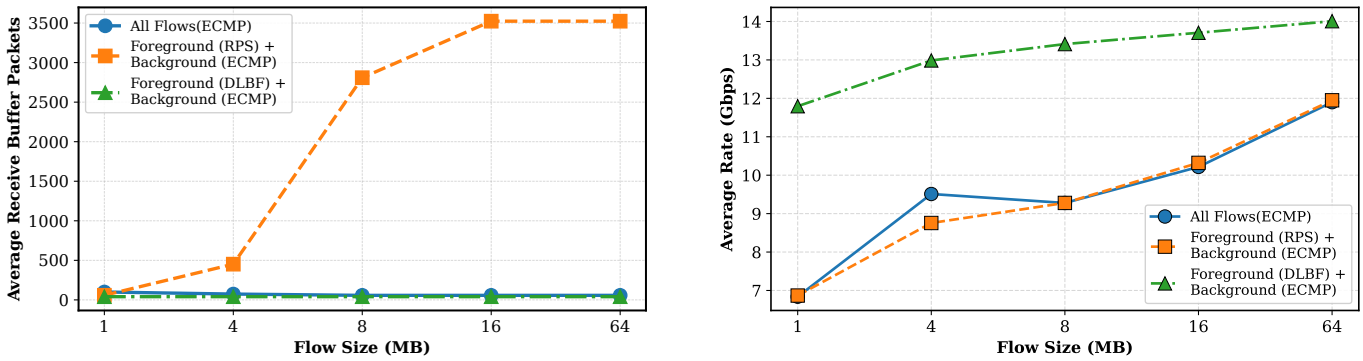
1) FCT performance: As shown in Fig. 4, DLBF provides consistent improvements across the evaluated Allreduce message sizes, yielding overall gains of 1.1–1.7× in average FCT and 1.2–3.4× in P99 FCT when compared with ECMP and RPS. Among these message sizes, the improvement is most pronounced for small flows. In particular, for 1 MB messages, DLBF reduces average FCT by up to 1.7× and lowers the P99 tail latency by as much as 3.3×.

In theory, RPS should outperform ECMP for AI workloads by more fully utilizing multipath bandwidth. However, in our scenario, the background traffic uses ECMP, which may create persistent hotspots due to static hashing. As a result, RPS foreground flows are more likely to traverse these congested links, increasing path-delay imbalance and substantially amplifying packet reordering. Because reordered packets are interpreted as losses by RoCEv2, RPS frequently triggers sender-side rate reductions, degrading FCT performance despite its intended load-spreading benefits.

In contrast, DLBF avoids these issues by quickly migrating foreground flows away from ECMP-induced hotspots while preventing the delay-gap–driven reordering that typically occurs during path transitions. By jointly reducing exposure to congested paths and suppressing reordering during switching, DLBFmaintains higher sending rates and achieves consistently lower average FCT and P99 FCT across message sizes.

2) Buffer occupancy: We periodically observed the receiver buffer occupancy of each QP and calculated the average value for all QPs. Fig. 5a evaluates the receiver-side buffer occupancy under the Allreduce workload. RPS incurs the largest buffer footprint because frequent out-of-order arrivals accumulate in the receiver's reordering buffer, increasing receiver-side buffer pressure.

In contrast, DLBF significantly reduces receiver buffer occupancy across all message sizes. Notably, its buffer usage closely matches that of ECMP and consistently remains at a



(a) Receiver buffer occupancy

(b) Sending rate

Fig. 5: Receiver-side buffer occupancy and sender-side rate. Foreground workload: Allreduce.

comparably low level, effectively eliminating excessive buffering caused by packet reordering. This behavior is enabled by DLBF's threshold-controlled notification-based switching mechanism. During path switching, the receiver enforces an explicit threshold on outstanding notification packets to regulate whether the sender transmits full data packets or lightweight notification packets, thereby keeping the reordering buffer occupancy at a consistently low level.

3) Rate performance: As shown in Figure 5b, the performance gap among the three schemes becomes numerically evident when examining their sending rates. Across all flow sizes, RPS delivers the lowest throughput, operating consistently below ECMP. Quantitatively, RPS shows a rate deficit of roughly 15–25% relative to ECMP, reflecting the impact of hotspot-induced reordering and sender-side rate degradation.

In contrast, DLBF sustains the highest throughput across all message sizes. Compared with ECMP, DLBF improves the average sending rate by approximately 10–20%. When compared with RPS, the improvement is even larger, amounting to a 30–40% throughput gain. This numerical advantage aligns with DLBF's ability to avoid ECMP hotspots and prevent reordering during path transitions, allowing the sender to maintain a substantially higher and more stable transmission rate.

V. Conclusion

This paper presented DLBF, an end-host dynamic load balancing framework designed to mitigate packet reordering in RDMA networks. The core of DLBF contains: 1) an adaptive path switching mechanism to dynamically migrate traffic away from hotspots based on fine-grained congestion signals; 2) a selective duplicate transmission strategy to eliminate reordering during path transitions by leveraging delay gaps; and 3) a notification-based transmission optimization to prevent aggravating congestion on unknown paths. Large-scale NS-3 simulations show that DLBF is a viable solution that significantly reduces both average and tail FCT while minimizing receiver buffer occupancy.

VI. Acknowledgment

This work is supported in part by the National Key Research and Development Program of China (No. 2024YFB2907000), and by National Science Foundation of China (NSFC) under Grant No. 62302055, and in part by the Fundamental Research Funds for the Central Universities, and in part by the Shandong Provincial Natural Science Foundation under Grant No.ZR2024LZH006. Corresponding author: Yiran Zhang.

References

[1] 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*, 2013, pp. 2130–2138.

[2] I. Bukspan and G. Shainer, "Optimize Large-Scale AI Workloads with NVIDIA Spectrum-X," *NVIDIA Developer Blog*, May 23, 2023. [Online]. Available: https://developer.nvidia.com/blog/optimize-large-scale-ai-workloads-with-nvidia-spectrum-x/.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.

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

[5] W. Wang, M. Khazraee, Z. Zhong, M. Ghobadi, Z. Jia, D. Mudigere, Y. Zhang, and A. Kewitsch, "TopoOpt: Co-optimizing network topology and parallelization strategy for distributed training jobs," in *Proc. 20th USENIX Symp. Networked Systems Design and Implementation (NSDI '23)*, 2023, pp. 739–767.

[6] 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. Netw. Syst. Des. Implement. (NSDI '10)*, USENIX Association, 2010.

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

[8] A. Gangidi, R. Miao, S. Zheng, S. J. Bondu, G. Goes, H. Morsy, R. Puri, M. Riftadi, A. J. Shetty, J. Yang, S. Zhang, M. J. Fernandez, S. Gandham, and H. Zeng, "RDMA over Ethernet for Distributed Training at Meta Scale," in *Proc. ACM SIGCOMM 2024*, 2024, pp. 57–70.

[9] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, "RDMA over commodity ethernet at scale," in *Proc. ACM SIGCOMM*, 2016, pp. 202–215.

[10] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. Haj Yahia, and M. Zhang, "Congestion control for large-scale RDMA deployments," *ACM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 523–536, 2015.

[11] B. Rothenberger, K. Taranov, A. Perrig, and T. Hoefler, "ReDMArk: Bypassing RDMA Security Mechanisms," in *Proc. 30th USENIX Security Symp. (USENIX Security '21)*, 2021, pp. 4277–4292.

[12] Ultra Ethernet Consortium, "The Ultra Ethernet Consortium White Paper v1.1," *Ultra Ethernet Consortium (UEC)*, September 2024. [Online]. Available: https://ultraethernet.org/.

[13] W. Won, T. Heo, S. Rashidi, S. Sridharan, S. Srinivasan, and T. Krishna, "Astra-sim2.0: Modeling Hierarchical Networks and Disaggregated Systems for Large-Model Training at Scale," in *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, 2023, pp. 283–294.