RESEARCH-ARTICLE

LAPS: Joint Load Balancing and Congestion Control on Unequal-cost Multi-path Data Center Networks

YING WAN, Southeast University, Nanjing, Jiangsu, China

HAOYU SONG, Futurewei Technologies, Inc., Santa Clara, CA, United States

YU JIA, China Mobile Communications, Beijing, China

YUNHUI YANG, China Mobile Communications, Beijing, China

TAO HUANG, Purple Mountain Laboratory, Nanjing, Jiangsu, China

ZHIKANG CHEN, Tsinghua University, Beijing, China

Open Access Support provided by:

China Mobile Communications

Tsinghua University

Southeast University

Purple Mountain Laboratory

Futurewei Technologies, Inc.

Published: 08 September 2025

SIGCOMM '25: ACM SIGCOMM 2025 Conference
September 8 - 11, 2025
Coimbra, Portugal

Conference Sponsors:
SIGCOMM

https://doi.org/10.1145/3748273.3749198
ISBN: 9798400720826

LAPS: Joint Load Balancing and Congestion Control on Unequal-cost Multi-path Data Center Networks

Ying Wan
Southeast University
Nanjing, China

Haoyu Song*
Futurewei Technologies
Santa Clara, USA

Yu Jia
China Mobile (Suzhou) Software Technology
Suzhou, China

Yunhui Yang
China Mobile (Suzhou) Software Technology
Suzhou, China

Tao Huang
Purple Mountain Laboratories
Nanjing, China

Zhikang Chen
Tsinghua University
Beijing, China

ABSTRACT

The assumption of equal-cost paths no longer holds for newer data center network topologies catering for HPC/AI workloads, challenging both load balancing and congestion control. The existing load-balancing schemes, including random packet spraying, fail to adapt to such networks. In this paper, we propose LAPS, a simple latency-aware packet spraying scheme, to achieve joint load balancing and congestion control regardless of network topology and traffic pattern. As a coherent load-balancing and congestion-control solution, LAPS manages the packet sending rate and distribution simultaneously based on real-time path latency. It adapts to both TCP and RoCE-based transport protocols and can be deployed on SmartNICs at a low implementation cost. Evaluations show that LAPS consistently outperforms the other load-balancing and congestion-control schemes in unequal-cost multi-path topologies for HPC/AI workloads.

CCS CONCEPTS

• **Networks** → **Network protocols**.

KEYWORDS

Data center network, Load balancing, Congestion control, Unequal-cost multi-path, Packet spraying

ACM Reference Format:
Ying Wan, Haoyu Song, Yu Jia, Yunhui Yang, Tao Huang, and Zhikang Chen. 2025. LAPS: Joint Load Balancing and Congestion Control on Unequal-cost Multi-path Data Center Networks. In *2nd Workshop on Networks for AI Computing (NAIC '25), September 8–11, 2025, Coimbra, Portugal*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3748273.3749198

*Haoyu Song is the corresponding author

NAIC '25, September 8–11, 2025, Coimbra, Portugal

ACM ISBN 979-8-4007-2082-6/2025/09
https://doi.org/10.1145/3748273.3749198

1 INTRODUCTION

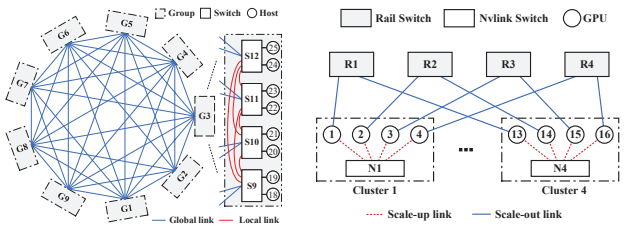
As the core infrastructure, data center supports large AI model training/inference and HPC applications. Data Center Network (DCN) provides the critical communication support for distributed computing nodes. In large model training, huge amounts of intermediate data are exchanged frequently, and a slowdown of any flow hurts the collective communication efficiency. The multi-path opportunity in data center networks must be exploited in conjunction with efficient end-to-end flow congestion control to fully utilize the available network bandwidth and avoid creating hot spots.

Load-Balancing (LB): Many multi-path LB schemes have been proposed and some are widely deployed (e.g., ECMP [1] and Flowlet [2]). Most assume equal-cost paths (e.g., Fat-Tree [3] and Spine-Leaf [4]), aiming to distribute the traffic as evenly as possible with granularity from flow [1] to flowlet [2] to flowcell [5] to packet [6].

The AI workloads exhibit several distinct traffic characteristics [7–9]: while multiple jobs run in parallel, each job produces a relatively small number of huge, concurrent, and intermittent flows. The low flow entropy renders the flow-based LB ineffective and the bursty data hardly generates any flowlets. No wonder Ultra Ethernet Consortium (UEC), dedicated for Ethernet-based AI and HPC optimization, resorts to packet spraying as the default LB scheme [10].

The current packet spraying schemes assume equal-cost paths too. However, the path asymmetry in bandwidth or length can lead to significant load-unbalancing. The static weighted spraying cannot solve the problem due to path/link diversity and traffic dynamics (e.g., a path/link may present a fast-changing weight for each packet).

Data centers for AI and HPC often employ unconventional network topologies that manifest the problem. First, switchless topologies (e.g. Torus [11]) provide multiple paths between each pair of nodes, but the time-variant traffic makes the paths with the same length effectively unequal, let alone the longer detours. Second, high-radix, low-diameter networks (e.g., Dragonfly [12] and UB-Mesh [13]) require non-minimal adaptive routing to access the path diversity. Third, the GPU/TPU clusters often adopt heterogeneous interconnection technologies for intra-cluster scale-up network (e.g., NVLink [14] and UAL [15]) and inter-cluster scale-out network (e.g., IB [16] and UEC). In each case above, LB faces the unequal-cost multi-path problem which is challenging for the packet-spraying-based schemes in particular.



(a) Dragonfly. (b) Rail.

Figure 1: Typical AI DCN topologies.

Congestion-Control (CC): LB and CC are used to be considered orthogonal for network performance optimization. However, in our circumstance, they are entangled and should be considered jointly for the following reasons.

First, conventional DCN CC algorithms (e.g., DCTCP [17] and DCQCN [18]) usually take ECN and packet drop as congestion signals. Multi-path packet spraying renders such signals unreliable and less meaningful. Second, the widely used RoCE transport requires rigid packet delivery ordering in each flow. Any out-of-order (OOO) packet delivery triggers an expensive Go-Back-N retransmission process. Unfortunately, packet spraying makes OOO unavoidable. Although moderate OOO can be tolerated by adopting certain techniques [6, 19–21], packet spraying on unequal-cost paths can lead to an excessive OOO rate. Third, multi-path LB is impotent to deal with incast happening at the last hops. The source node cannot differentiate the ECN generated by incast or a node on the path.

Our goal is to *enable effective packet spraying to achieve global LB and end-to-end flow CC on arbitrary network topologies and traffic patterns.* We believe that the link bandwidth, path length (e.g., hops), and *local* congestion status (e.g., queue depth) are not effective path cost metrics. Instead, the real-time end-to-end one-way path latency should be the deterministic factor to choose a packet's forwarding path, which consists of all the other factors. Intuitively, packets should always be sprayed on one or more paths exhibiting the smallest latencies at the moment. Meanwhile, the path latency can also serve as a more indicative congestion signal: as long as some candidate paths present low latencies, the flow can continue to send with increasing rate; only when all the paths present higher than expected latency, which is likely caused by incast, should the flow reduce its rate.

A NIC-based solution is ideal to realize the above vision for the following reasons: (1) it allows the complete coverage of end-to-end paths and joint consideration of LB and CC; (2) it is applicable to all network topologies including the switchless ones; (3) it can take advantage of the programmable SmartNIC to offload algorithms, avoiding the scalability concern and the reliance on discontinued programmable switches. However, several technical hurdles need to be overcome. First, the host needs to compute and maintain the candidate paths for each flow. Second, the network needs to sense the real-time path latency to make timely packet spraying decisions. Third, we need an efficient method to direct packets to follow specific end-to-end paths. Fourth, we need a new algorithm to handle latency-based congestion signals, OOO, and lost packet retransmission. To this end, we propose Latency-Aware Packet Spraying (LAPS).

The remainder of the paper is organized as follows. Sec. 2 provides the background. Sec. 3 describes the architecture and implementation of LAPS. Sec. 4 presents the performance evaluation. Sec. 5 summarizes the related work. Finally, Sec. 6 concludes the paper.

2 BACKGROUND

When moving packets from A to B with multiple paths, the sensible choice is to take the fastest way if the delay information can be acquired in advance, and only if all the paths are congested, shall the sending rate be slowed down to avoid worsening the situation. LAPS sticks to this first principle to design its LB and CC mechanisms jointly.

Unequal-cost Multi-path. Fig.1 illustrates a few typical network topologies in today's AI DCN which features unequal-cost multi-path. The switches and servers in Fig. 1(a) are divided into nine identical groups. From S5 to S11, the network provides three 3-hop paths, 12 4-hop paths, and many other longer paths. In the mixed scale-up and scale-out network in Fig. 1(b), to reach a node in another cluster, a packet can be routed to any node in the same cluster first before being forwarded to the scale-out network, resulting in multiple unequal-cost paths. Since each link is involved in multiple paths for multiple transient flows, the link load is difficult to predict, and so is the path cost.

Path Finding. Although the choices are plethora, it is unnecessary and uneconomic to use all possible paths for packet forwarding. Usually, it is sufficient to only consider the top-k paths in terms of path cost. The choice of k is subject to the network type and scale. In Fig. 1(a), the 3-hop and 4-hop paths (15 in total) provide a good balance between path diversity and maintenance cost. The choice of k may also depend on the adjacency degree of two nodes. For example, in a Torus network, more paths should be considered when two nodes are further apart. The k-shortest-path algorithm [22] can be used to find the candidate paths.

In-band Network Telemetry (INT). INT [23] adds a custom header to packets to instruct network nodes to collect specific data for network performance monitoring. INT has been enabled on commercial switches (e.g., Broadcom [24]). In contrast to out-of-band active measurements, INT has low overhead (the measurement data is carried in application packets), low feedback latency (the measurements can be refreshed in one RTT), and high accuracy (it reflects the real experience of the application traffic). LAPS only needs the one-way path latency, so we use INT to collect only the packet timestamp at sender nodes.

Source Routing (SR). LAPS balances the traffic load at path level, requiring the candidate paths to be explicitly maintained at a source node and each packet pinned on a path. SR is thus a natural choice to support non-minimal adaptive routing in unequal-cost multi-path networks. To forward a packet to another node, the head node first figures out the candidate paths based on the packet's destination address, and then inserts an SR header according to the path selection result.

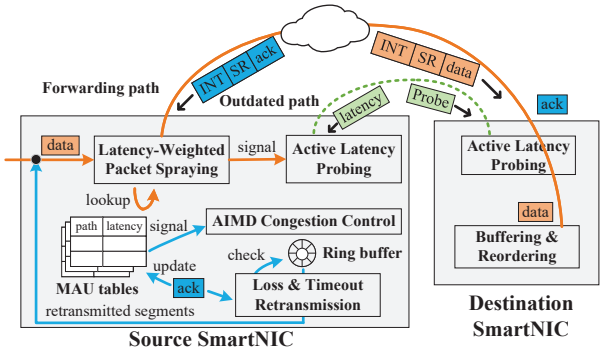


Figure 2: The architecture of LAPS.

Multi-path Packet Spraying. We consider the path latency as the dynamic path cost and use the general Softmax function to model the packet spraying strategy in Equation 1.

$$P(d_i) = \frac{e^{-\beta d_i}}{\sum_{j=1}^{k} e^{-\beta d_j}}, \forall i \in [1, ..., k] \quad (1)$$

In the equation, d_i is path i's measured latency, and $P(d_i)$ is the probability that i is selected to send a packet. The non-negative real-value parameter β is used to adjust the probability distribution. When $\beta = 0$, the strategy degenerates into Random Packet Spraying (RPS) [6]: the packets are evenly distributed to all candidate paths. Clearly, this will lead to serious load imbalance. On the other hand, increasing β will make packets concentrate more toward the paths of the smallest latencies. When β is large enough, the Softmax function degenerates into an ArgMax function and only the path(s) with the minimum latency will be chosen. It is interesting to find the optimal β for the best overall performance.

CC & Lost Recovery. Since packets are always sprayed on the fastest path(s), low latencies on these paths imply that the flow sending rate can be increased. Only when all the paths exhibit an abnormal latency should the flow rate be reduced. Latency as the congestion signal is more expressive and stable than ECN for multi-path packet spraying. However, we must differentiate OOO from packet drop events for efficient retransmission. Inherently, packets distributed on the same path are delivered in order if not lost, and their ACKs should be received in order if they also follow the same path. Hence, if the ACKs received for a path skip an expected sequence number, it indicates that the corresponding packet is lost, and a retransmission can be issued immediately.

3 ARCHITECTURE & IMPLEMENTATION

3.1 Architecture Overview

Fig. 2 illustrates the architecture of LAPS. The source node sprays packets across selected paths with a higher probability towards paths with lower latency. Intermediate switches forward the packets according to the SR header. The destination node acknowledges the received data segment by sending ACKs along the reversed path acquired from the SR header. The ACK also embeds the receiver timestamp for the source to refresh its path latency. At the source node, if the latency of a path is outdated, which may be due to the lack of packets sent on it or returned ACKs for a while, it sends a probe packet along that path to obtain its latency. The source node adjusts a flow's sending rate using the "Additive Increase and Multiplicative Decrease" (AIMD) approach based on the latencies of all its candidate paths for CC. The source node detects packet loss through ACKs and performs selective retransmissions. If no ACK is received within a timeout period, a retransmission of the missing packets is triggered.

3.2 Packet Format

To support the above functions, LAPS embeds a custom header in each packet, which consists of three parts:

- **Type**: The 1-bit `type`, if set, indicates that the corresponding data/ACK packet is a probe rather than a normal one.
- **INT**: The INT fields contain a 16-bit `pid` and a 48-bit `time`. For a data packet, `pid` and `time` record the forwarding path ID and the transmission timestamp[1], respectively. For an ACK packet, `pid` is copied from the corresponding data packet, and `time` carries the calculated one-way path latency.
- **SR**: The field contains a series of 32-bit node IDs for the forwarding path, as well as some other facilitating data such as path length and node pointer. A path can be specified with a few key anchor points if the shortest path between adjacent anchor points is assumed.

A path can usually be specified by up to two or three anchor nodes, so the additional overhead is about 20 bytes per packet, accounting for only 1.3% of 1500-byte IP packets.

3.3 Match-Action Tables

LAPS maintains and uses two tables shown in Fig. 3:

- **Path Search Table (PST)**: `PST` uses the packet's destination address as the key for Exact Matching (EM) to obtain the IDs of candidate forwarding paths.
- **Path Information Table (PIT)**: Indexed by path ID, `PIT` maintains the path information. `idleVal` represents the baseline latency of a path. `time` and `realVal` are the latest measurement time and the measured latency, respectively. `anchors` indicates the anchor points defining the path.

DstIP	Pids
ip1	p1, p2

Pid	valid	time	idleVal	realVal	anchors
p1	1	50	15	15	s_a, s_b, ...
p2	1	42	10	8	s_a, s_c, ...

Figure 3: Two EM match tables in LAPS.

`PST` contains at most n-1 entries for a network of n hosts. In addition, `PIT` consumes $O(nk)$ entries to store path information, where k is the number of paths between a pair of hosts. Modern NICs can easily support these needs.

[1]Although using timestamps on different nodes for latency calculation, LAPS does not require clock synchronization between the nodes, because LAPS only compares path latencies between each pair of nodes, and their offsets to the real time only produce a fixed error which does not affect the relative comparison result.

3.4 Latency Measurement

LAPS measures path latency with two methods: (1) Upon receiving each normal ACK, the sender updates the entry `PIT[pid]`; (2) Active Probing (AP) is used to refresh the entry `PIT[pid]` if it is not updated for more than 2·`realVal`. When waiting for the AP ACK, `PIT[pid].time` is set to Now(), and `PIT[pid].realVal` is doubled to reduce the probing frequency and exclude the path from forwarding packets.

3.5 Latency-Weighted Packet Spraying

To send a data packet, LAPS looks up `PST` and `PIT` to determine the latencies of all candidate paths. Then, based on the probabilities calculated by Equation 1, the packet is distributed to a path[2]. Meanwhile, AP is triggered if necessary.

When the receiver needs to send an ACK for a packet, it simply uses the reversed SR path due to the following rationale: (1) it avoids path lookups at the receiver; (2) the small ACK packets have little impact on load imbalance, and the network nodes can give ACKs higher priority than data packets to avoid queuing delay; (3) it enables efficient loss detection and fast retransmission (see Sec. 3.7).

3.6 Congestion Detection and Control

LAPS adjusts the sending rate based on real-time path latency. Normally, the biased traffic distribution mitigates the congestion on paths with longer latencies without reducing the overall flow rate. When traffic is heavy, or during receiver incast, the latencies of all paths for a flow converge to the largest baseline latency `T`. At this point, it is no longer possible to mitigate congestion by adjusting traffic distribution. Instead, LAPS starts to reduce the flow sending rate.

LAPS uses AIMD to adjust the flow rate. When receiving an ACK, the sender first updates the corresponding `PIT` entry. Then, it detects if all the valid paths have a higher latency than `T`. If true, the sender sets the flow's expected rate `expRate` to the current rate `curRate` and halves the `curRate`; otherwise, `curRate` is linearly increased by $\frac{\texttt{expRate}-\texttt{curRate}}{2}$. The rate adjustment is only allowed once in 2`T` to prevent over-adjustment before the effect is perceived by the sender.

To find the maximum available bandwidth, `expRate` will be doubled after `N` consecutive linear increases. This Hyper Rate Increase (HAI) phase continues as long as any path has a latency lower than `T`. Otherwise, HAI is terminated.

3.7 Loss & Timeout Retransmission

LAPS enforces an ACK packet to use the reverse path of the corresponding data packet. Therefore, the OOO ACKs on a path imply packet loss, so the sender can immediately retransmit the deemed lost packets. To achieve this, LAPS records the sequence numbers of data packets sent along each path that have not yet been acknowledged. Upon receiving a data packet `pkt`, the receiver sends an ACK to selectively acknowledge it. The sender compares the unacknowledged data segments sent along this path until it finds `pkt`. Consequently, all data segments before `pkt` are lost. LAPS will immediately retransmit these lost data segments in sequence before sending new data.

[2]The default value of β is empirically set to be 1. Methods for automatically converging β to the optimal value under varying network conditions is left for future work. The other settings can be found at https://github.com/wany16/Laps-ns3.

3.8 Implementation

To verify the feasibility of LAPS on an FPGA-based NIC, we implement it on the Xilinx Alveo U280 Accelerator Card by leveraging the open-source FastRMT [25] and Corundum [26]. As shown in Fig. 4, LAPS is realized as a pipeline composed of a series of Match-Action Units (MAUs) which fit in the classic RMT model [27, 28]. The design (*i*) decomposes `PIT` into 32 sub-tables and look up the candidate paths in parallel, (*ii*) transforms the path traversal process (e.g., calculating path weights and identifying outdated paths) into parallel circuits, and (*iii*) converts the Softmax calculation into lookup tables, additions, and divisions [29].

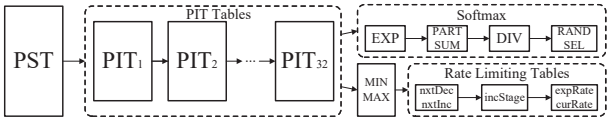


Figure 4: Pipelined implementation of LAPS.

4 EVALUATION

We conduct extensive simulations using the network simulator NS3 [30] to compare LAPS with flow-based LB (ECMP, PLB [31]), Flowlet-based LB (LetFlow, CONGA), and ConWeave [32] by reusing implementations in [32–35], with parameters set to default values as specified in [33–35] unless otherwise stated.

4.1 Testbed

Topologies: We run simulations on two typical DCN topologies: Dragonfly (Fig. 1(a)) and Rail (Fig. 1(b)). Dragonfly comprises nine groups with 36 switches and 144 servers, and Rail comprises eight clusters with eight Rail switches, eight intra-node Nvlink switches, and 64 servers. In Dragonfly, not only the minimal path but also a set of sub-optimal paths are considered: for inter-group traffic, each group is used as a midpoint to bridge the two sections of the shortest paths from the source server and to the destination server; for intra-group traffic, each switch of the same group is used as a midpoint. Thus, there are three and eight candidate paths for intra-group and inter-group communication, respectively. In Rail, GPUs within the same cluster communicate directly through the Nvlink switch that connects them. For inter-cluster GPU communication, the data is routed through all Rail switches. The number of available paths thus equals the number of Rail switches (8). All the links are set to 100Gbps.

Workloads: We adopt four representative real-world DCN workloads to generate the test traffic: Data Mining (DM) [36] from Microsoft, Remote Procedure Call (RPC) [37] from Google, Hadoop (HDP) from Meta, and Cloud Storage (STR) from Alibaba [38]. They are all heavy-tailed, with the top 5% of flows accounting for 97.9%, 90.7%, 81.8%, and 77.1% of the total traffic for DM, RPC, HDP, and STR, respectively. We also test the performance under

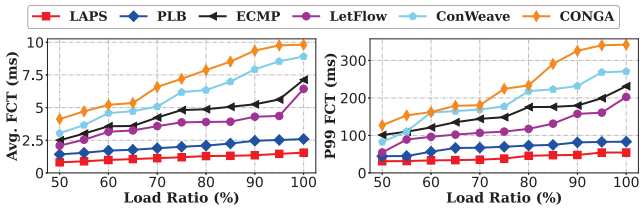


Figure 5: The FCTs on (Rail, DM, All2all).

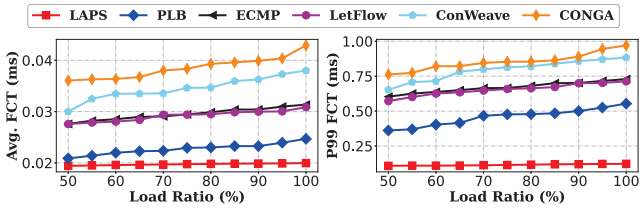


Figure 6: The FCTs on (Rail, STR, All2all).

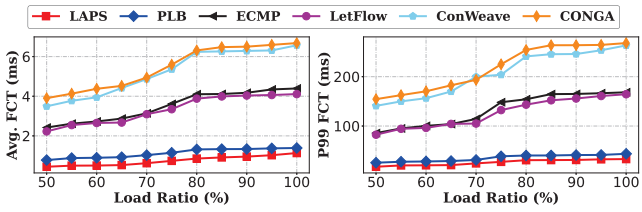


Figure 7: The FCTs on (Dragonfly, RPC, All2all).

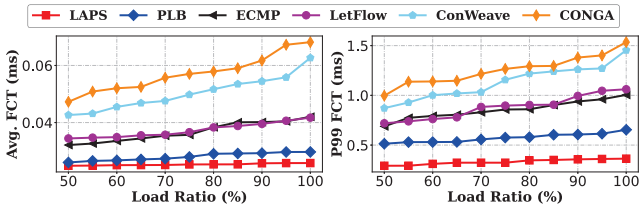


Figure 8: The FCTs on (Dragonfly, HDP, All2all).

the distributed training of the large language model LLAMA-2 with 7B parameters [39].

Patterns: We select three representative communication patterns: All2all, AllReduce, and AllScatter. For All2all, the servers/GPUs send traffic to all the other servers/GPUs; for AllReduce based on parameter server, all the other servers/GPUs send traffic to one server/GPU; for AllScatter, one server/GPU sends traffic to all the other servers/GPUs.

4.2 Simulation Results

Fig. 5~8 illustrate the FCTs of the LB algorithms under different settings, in which (X, Y, Z) indicates the combination of the topology X, the workload Y, and the pattern Z.

Average FCT. The first subfigures in Fig. 5~8 illustrate the average FCT of all flows. LAPS consistently outperforms the other algorithms in all cases. At 80% load in Fig. 5, LAPS improves the average FCT by 3.3×, 2.8×, 1.6×, 4.8×, and 6.1× compared to ECMP, LetFlow, PLB, ConWeave, and CONGA, respectively. Besides, LAPS exhibits the slowest growth rate in average FCT as the load ratio increases. As shown in Fig. 5, when the load ratio increases from 50% to 100%, LAPS only increases the average FCT by 0.64ms, whereas ECMP, LetFlow, PLB, ConWeave, and CONGA increase it by 4.6, 4.3, 1.1, 5.2, and 5.6ms, respectively.

P99 FCT. The second subfigures in Fig. 5~8 illustrate the FCT of the 99th percentile flow, which indicates fair treatment of all flows to meet the QoS of latency-sensitive applications. LAPS achieves the best and most stable performance across all scenarios. As shown in Fig. 6, at 80% load in (Rail, STR, All2all) , the P99 FCT of LAPS is only 0.12ms, while ECMP, LetFlow, PLB, ConWeave, and CONGA reach 0.67, 0.70, 0.48, 0.80, and 0.85ms, respectively.

Insight. ECMP randomly selects a path for each flow without considering the inequality of path costs or the significant flow size differences, which can easily lead to path congestion. ConWeave uses the destination ToR switch to buffer all packets on the newly selected path until all packets on the original path are passed, which puts pressure on the switch's buffer and delays the arrival of packets on the new path. CONGA sends flowlets to the least congested path. However, flowlets are hard to identify in RDMA scenarios. Consequently, most flows stick to the same path. Similarly, LetFlow transmits each flow almost entirely along the initially randomly selected path until completion, resulting in a performance very close to ECMP. Meanwhile, CONGA presents a severe "herd effect" that flows tend to rush to the same least congested path to quickly congest it. The performance of PLB is the closest to that of LAPS. When congestion is detected and there are no in-flight packets, PLB switches it to a randomly selected path. Moreover, if a flow detects congestion for 12 consecutive times without finding a suitable switching opportunity, it will force the path switching. Therefore, PLB often requires multiple attempts to find a suitable path.

LLM Training: Table 1 shows the average and P99 FCTs for different LB schemes during the gradient aggregation (64MB) in one round of LLAMA distributed training with and without random packet drops, which involves incast congestion. LAPS outperforms other algorithms in both average FCT and P99 FCT. For instance, given no packet drop, LAPS's P99 FCT is improved by 3.5×, 3.4×, 3.6×, 3.8×, and 1.4× compared to ECMP, LetFlow, PLB, ConWeave, and CONGA, respectively. Table 1 not only demonstrates LAPS's ability to transmit packets along the minimal-cost path but also highlights its superior rate control capabilities. Table 1 also shows that only LAPS's average FCT and P99 FCT are almost unaffected by random packet loss, which demonstrates that LAPS can distinguish packet loss from OOO, and enables rapid selective retransmission.

Buffer Size. Fig. 9 shows the statistics of the buffer size in LAPS due to OOO packets at the receiver for the LLM training on Rail networks. Although LAPS employs packet spraying, it achieves a low overall packet reordering ratio. For the AllScatter operations when no congestion is experienced, only 0.2% of packets are found to be reordered and the buffer size is only 0.003KB on average. Even for the AllReduce operations that trigger severe congestion, LAPS requires buffering fewer than 16 1KB packets on average, with only less than 4% probability that the buffer size exceeds 30KB. This indicates that LAPS has low pressure on the receiver's OOO packet

Table 1: FCTs on (Rail, LLM, AllReduce)

Drop ratio	0						0.0001%					
Scheme	ECMP	LetFlow	CONGA	ConWeave	PLB	LAPS	ECMP	LetFlow	CONGA	ConWeave	PLB	LAPS
Avg. FCT (ms)	564	553	540	563	211	162	580	578	547	610	219	162
P99 FCT(ms)	617	589	628	659	235	175	645	633	658	757	240	175

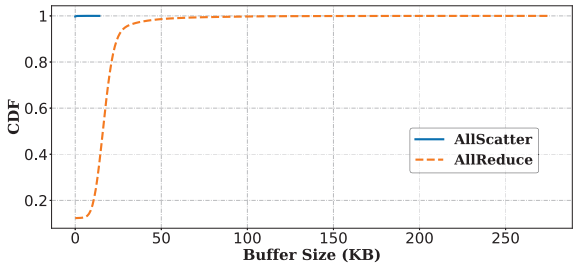


Figure 9: The buffer size in LLM training on Rail.

Table 2: Hardware consumption on FPGA

	LUT	FF	CARRY8	BRAM
PST Table	74	177	0	8
PIT Table	2950	2976	0	112
Rate Limiting Tables	3012	3866	119	8
Softmax Function	26949	7938	1752	0
Total	33808 (2.51%)	16711 (0.64%)	1909 (1.17%)	135.5 (6.72%)

buffer and reordering process. Considering the scenarios such as LLM training for which the number of flows is relatively small, the current commercial RDMA NICs are sufficient to store and reorder the OOO packets.

4.3 Hardware Prototype Analysis

We implement a prototype of LAPS using 1,129 lines of code in Chisel 5.0 (which maps to 5,992 lines of code in SystemVerilog) on an Alveo U280. The table `PST` has a size of 1,024 to support up to 1,024 servers, with each `PST` entry containing 8 valid `pids`. Correspondingly, `PIT` has a size of 1,024×8=8,192, with each `PIT` entry allowing 4 anchors. The rate-limiting module can support up to 1,024 flows. When evaluating the resource utilization, we excluded the NIC functions themselves (Xilinx 100G CMAC IP, schedulers, etc.) and only included the LAPS parsing/deparsing modules and the processing pipeline. We separately evaluated the resources consumed by the `PST`, `PIT`, rate limiting module, and Softmax, as shown in Table 2. The synthesis results indicate that the resource consumption of LAPS on U280 is acceptable, and it is feasible to deploy LAPS on an FPGA-based SmartNIC.

5 RELATED WORK

Previous works on multi-path LB are mainly for topologies with equal-cost paths. The approaches can be classified using the primary deployment location as the first dimension and the scheduling granularity as the second dimension.

Switch-based: ECMP and WCMP [40] distribute flows to different links. Flare [2], LetFlow [41], LocalFlow [42], and BurstBalancer [43] spray segments of flows (e.g., flowlet) across links. RPS and Drill [44] spray packets to different links. CONGA [45], Hula [46], and Proteus [33, 47] distribute flowlets or flows according to the RTT or link utilization.

Controller-based: Hedera [48], Mahout [49], Freeway [50], DRLet [51], and Fastpass [52] determine the paths for flows or packets based on exposed network and application information. Relying on a central controller, such solutions cannot scale to large networks and react to micro-bursts [53].

Host-based: MPTCP [54], Flowbender [55], Presto [5], and Clove [56] modify the transport layer to split a single flow into subflows for transmission on different paths. PLB [31] changes the flow label of any congested flow in the hope that the in-network ECMP may change its path.

To mitigate the packet reordering issue for packet spraying [6], SRED selectively drops packets that would lead to unequal queue lengths [57], and QDAPS and QDAPS* [58] ensure a packet has a longer queuing delay than the previous packet of the same flow. With added complexity, CAPS [59], HTPC [60], and Corrective [61] introduce a coding layer and spray the coded packets to address the reorder problem.

Adaptive routing can be applied to utilize the unequal-cost paths. UGAL-L [12] always sprays packets to the port with the smallest queue length. UGAL-G [12] uses the queue length of other switches and hop count to estimate the path latency. PAR [62] only uses the minimal path but allows intermediate switches to reroute to avoid congestion. Q-adaptive [63] adopts reinforcement learning technique to predict global path conditions based on local information.

DCTCP and DCQCN are widely deployed CC algorithms in DCN. Many others are used in wide-area networks [64, 65]. CC can be achieved with or without the assistance of network switches, based on ECN, RTT, or other signals, and using window or credit-based mechanisms. In recent years, many improvements to DCTCP and DCQCN are proposed [38, 66]. However, all these works consider CC independent of LB, making them ill-suited for unequal-cost multi-path networks. STrack [67] is a joint LB/CC algorithm for packet spraying on equal-cost multi-paths using ECN and RTT.

6 CONCLUSION

Designed for AI/HPC workload, LAPS is a joint load-balancing and congestion-control scheme to support *unequal-cost multi-path packet spraying* on arbitrary network topologies and for any traffic patterns. Although simple, it occupies a unique niche in the wide spectrum of load-balancing algorithms and reveals the need for congestion-control adaptation by providing a practical solution with synergistic benefits.

REFERENCES

[1] C Hopps. 2000. *RFC2992: Analysis of an Equal-Cost Multi-Path Algorithm.* Technical Report. RFC Editor. https://dl.acm.org/doi/pdf/10.17487/RFC2992

[2] Srikanth Kandula, Dina Katabi, Shantanu Sinha, and Arthur Berger. 2007. Dynamic load balancing without packet reordering. *ACM SIGCOMM Computer Communication Review* 37, 2 (3 2007), 51–62. https://doi.org/10.1145/1232919.1232925

[3] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication (SIGCOMM '08)*. ACM, New York, NY, USA, 63–74. https://doi.org/10.1145/1402958.1402967

[4] Mohammad Alizadeh and Tom Edsall. 2013. On the Data Path Performance of Leaf-Spine Datacenter Fabrics. In *2013 IEEE 21st Annual Symposium on High-Performance Interconnects*. IEEE, San Jose, CA, USA, 71–74. https://doi.org/10.1109/HOTI.2013.23

[5] Keqiang He, Eric Rozner, Kanak Agarwal, Wes Felter, John Carter, and Aditya Akella. 2015. Presto: Edge-based Load Balancing for Fast Datacenter Networks. *ACM SIGCOMM Computer Communication Review* 45, 4 (9 2015), 465–478. https://doi.org/10.1145/2829988.2787507

[6] Advait Dixit, Pawan Prakash, Y. Charlie Hu, and Ramana Rao Kompella. 2013. On the Impact of Packet Spraying in Data Center Networks. In *2013 Proceedings IEEE INFOCOM*. IEEE, Turin, Italy, 2130–2138. https://doi.org/10.1109/INFCOM.2013.6567015

[7] Wenxue Li, Xiangzhou Liu, Yuxuan Li, Yilun Jin, Han Tian, Zhizhen Zhong, Guyue Liu, Ying Zhang, and Kai Chen. 2024. Understanding Communication Characteristics of Distributed Training. In *Proceedings of the 8th Asia-Pacific Workshop on Networking (APNet '24)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3663408.3663409

[8] Sudarsanan Rajasekaran, Manya Ghobadi, and Aditya Akella. 2024. CASSINI: Network-Aware Job Scheduling in Machine Learning Clusters. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Santa Clara, CA, 1403–1420. https://www.usenix.org/conference/nsdi24/presentation/rajasekaran

[9] Weiyang Wang, Manya Ghobadi, Kayvon Shakeri, Ying Zhang, and Naader Hasani. 2024. Rail-only: A Low-Cost High-Performance Network for Training LLMs with Trillion Parameters. (2024). https://arxiv.org/abs/2307.12169

[10] UEC. 2024. UEC Progresses Towards v1.0 Set of Specifications. https://ultraethernet.org/uec-progresses-towards-v1-0-set-of-specifications/

[11] Norman P. Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Cliff Young, Xiang Zhou, Zongwei Zhou, and David Patterson. 2023. TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings. https://arxiv.org/abs/2304.01433

[12] John Kim, Wiliam J. Dally, Steve Scott, and Dennis Abts. 2008. Technology-Driven, Highly-Scalable Dragonfly Topology. *ACM SIGARCH Computer Architecture News* 36, 3 (6 2008), 77–88. https://doi.org/10.1145/1394608.1382129

[13] Heng Liao, Bingyang Liu, Xianping Chen, Zhigang Guo, Chuanning Cheng, Jianbing Wang, Xiangyu Chen, Peng Dong, Rui Meng, Wenjie Liu, Zhe Zhou, Ziyang Zhang, Yuhang Gai, Cunle Qian, Yi Xiong, Zhongwu Cheng, Jing Xia, Yuli Ma, Xi Chen, Wenhua Du, Shizhong Xiao, Chungang Li, Yong Qin, Liudong Xiong, Zhou Yu, Lv Chen, Lei Chen, Buyun Wang, Pei Wu, Junen Gao, Xiaochu Li, Jian He, Shizhuan Yan, and Bill McColl. 2025. UB-Mesh: a Hierarchically Localized nD-FullMesh Datacenter Network Architecture. arXiv:2503.20377 [cs.AR] https://arxiv.org/abs/2503.20377

[14] Nvidia. 2024. NVLink and NVLink Switch. https://www.nvidia.com/en-us/data-center/nvlink/

[15] UALink Consortium. 2024. Ultra Accelerator Link. https://www.ualinkconsortium.org/

[16] IBTA. 2024. InfiniBand Trade Association. https://www.infinibandta.org/

[17] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data center TCP (DCTCP). In *Proceedings of the ACM SIGCOMM 2010 Conference*.

[18] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. 2015. Congestion Control for Large-Scale RDMA Deployments. In *Proceedings of the 2015 ACM SIGCOMM Conference*.

[19] Yuanwei Lu, Guo Chen, Bojie Li, Kun Tan, Yongqiang Xiong, Peng Cheng, Jiansong Zhang, Enhong Chen, and Thomas Moscibroda. 2018. Multi-Path Transport for RDMA in Datacenters. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. USENIX Association, Renton, WA, 357–371. https://www.usenix.org/conference/nsdi18/presentation/lu

[20] Guo Chen, Yuanwei Lu, Bojie Li, Kun Tan, Yongqiang Xiong, Peng Cheng, Jiansong Zhang, and Thomas Moscibroda. 2019. MP-RDMA: Enabling RDMA With Multi-Path Transport in Datacenters. *IEEE/ACM Transactions on Networking* 27, 6 (12 2019), 2308–2323. https://doi.org/10.1109/TNET.2019.2948917

[21] Radhika Mittal, Alexander Shpiner, Aurojit Panda, Eitan Zahavi, Arvind Krishnamurthy, Sylvia Ratnasamy, and Scott Shenker. 2018. Revisiting Network Support for RDMA. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, New York, NY, USA, 313–326. https://doi.org/10.1145/3230543.3230557

[22] Jin Yen. 1971. Finding the K Shortest Loopless Paths in a Network. *Management Science 17(11)* (1971).

[23] Lizhuang Tan, Wei Su, Wei Zhang, Jianhui Lv, Zhenyi Zhang, Jingying Miao, Xiaoxi Liu, and Na Li. 2021. In-band Network Telemetry: A Survey. *Computer Networks* 186 (2021), 107763. https://doi.org/10.1016/j.comnet.2020.107763

[24] Broadcom. 2017. In-band Telemetry. https://docs.broadcom.com/doc/IBT-PB100

[25] Xiangrui Yang, Lingbin Zeng, Zhongpei Liu, Yingwen Chen, Gaofeng Lv, Cheng Yang, and Jinshu Su. 2024. FastRMT: A High-Speed Data Plane Programmable System for Micro-Architecture Innovation. *Chinese Journal of Computers* 47, 2 (2 2024), 473–490.

[26] Alex Forencich, Alex C. Snoeren, George Porter, and George Papen. 2020. Corundum: An Open-Source 100-Gbps NIC. In *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 38–46. https://doi.org/10.1109/FCCM48280.2020.00015

[27] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. 2013. Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN. *ACM SIGCOMM Computer Communication Review* 43, 4 (2013), 99–110.

[28] Anirudh Sivaraman, Alvin Cheung, Mihai Budiu, Changhoon Kim, Mohammad Alizadeh, Hari Balakrishnan, George Varghese, Nick McKeown, and Steve Licking. 2016. Packet transactions: High-level programming for line-rate switches. In *Proceedings of the 2016 ACM SIGCOMM Conference*. 15–28.

[29] Qiwei Sun, Zhixiong Di, Zhengyang Lv, Fengli Song, Qianyin Xiang, Quanyuan Feng, Yibo Fan, Xulin Yu, and Wenqiang Wang. 2018. A High Speed SoftMax VLSI Architecture Based on Basic-Split. In *2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*. 1–3. https://doi.org/10.1109/ICSICT.2018.8565706

[30] Thomas R Henderson, Mathieu Lacage, George F Riley, Craig Dowell, and Joseph Kopena. 2008. Network simulations with the ns-3 simulator. *SIGCOMM demonstration* 14, 14 (2008), 527.

[31] Mubashir Adnan Qureshi, Yuchung Cheng, Qianwen Yin, Qiaobin Fu, Gautam Kumar, Masoud Moshref, Junhua Yan, Van Jacobson, David Wetherall, and Abdul Kabbani. 2022. PLB: congestion signals are simple and effective for network load balancing. In *Proceedings of the ACM SIGCOMM 2022 Conference (SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 207–218. https://doi.org/10.1145/3544216.3544226

[32] Cha Hwan Song, Xin Zhe Khooi, Raj Joshi, Inho Choi, Jialin Li, and Mun Choon Chan. 2023. Conweave: Network Load Balancing with In-network Reordering Support for RDMA. In *Proceedings of the ACM SIGCOMM 2023 Conference* (New York, NY, USA). ACM, 816–831. https://doi.org/10.1145/3603269.3604849

[33] Junxue Zhang, Wei Bai, and Kai Chen. 2019. Enabling ECN for datacenter networks with RTT variations. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies (CoNEXT '19)*. Association for Computing Machinery, New York, NY, USA, 233–245. https://doi.org/10.1145/3359989.3365426

[34] Hong Zhang, Junxue Zhang, Wei Bai, Kai Chen, and Mosharaf Chowdhury. 2017. Resilient Datacenter Load Balancing in the Wild. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 253–266. https://doi.org/10.1145/3098822.3098841

[35] Google. 2024. TCP-PLB Open Source. https://github.com/google/plb?tab=readme-ov-file

[36] Albert Greenberg, James R Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A Maltz, Parveen Patel, and Sudipta Sengupta. 2009. VL2: a scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication (SIGCOMM '09)*. ACM, New York, NY, USA, 51–62. https://doi.org/10.1145/1592568.1592576

[37] Behnam Montazeri, Yilong Li, Mohammad Alizadeh, and John Ousterhout. 2018. Homa: A Receiver-Driven Low-Latency Transport Protocol Using Network Priorities. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 221–235. https://doi.org/10.1145/3230543.3230564

[38] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, and Minlan Yu. 2019. HPCC: High Precision Congestion Control. In *Proceedings of the ACM Special Interest Group on Data Communication* (New York, NY, USA). ACM, 44–58. https://doi.org/10.1145/3341302.3342085

[39] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. https://arxiv.org/abs/2302.13971. *ArXiv* (2023).

[40] Junlan Zhou, Malveeka Tewari, Min Zhu, Abdul Kabbani, Leon Poutievski, Arjun Singh, and Amin Vahdat. 2014. WCMP: weighted cost multipathing for improved fairness in data centers. In *Proceedings of the Ninth European Conference on Computer Systems (EuroSys '14)*. Association for Computing Machinery, New

York, NY, USA, 1–4. https://doi.org/10.1145/2592798.2592803

[41] Erico Vanini, Rong Pan, Mohammad Alizadeh, Parvin Taheri, and Tom Edsall. 2017. Let It Flow: Resilient Asymmetric Load Balancing with Flowlet Switching. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. USENIX Association, Boston, MA, 407–420. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/vanini

[42] Siddhartha Sen, David Shue, Sunghwan Ihm, and Michael J Freedman. 2013. Scalable, Optimal Flow Routing in Datacenters via Local Link Balancing. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT '13)*. Association for Computing Machinery, New York, NY, USA, 151–162. https://doi.org/10.1145/2535372.2535397

[43] Zirui Liu, Yikai Zhao, Zhuochen Fan, Tong Yang, Xiaodong Li, Ruwen Zhang, Kaicheng Yang, Zihan Jiang, Zheng Zhong, Yi Huang, Cong Liu, Jing Hu, Gaogang Xie, and Bin Cui. 2024. BurstBalancer: Do Less, Better Balance for Large-Scale Data Center Traffic. *IEEE Transactions on Parallel and Distributed Systems* 35, 6 (2024), 932–949. https://doi.org/10.1109/TPDS.2023.3295454

[44] Soudeh Ghorbani, Zibin Yang, P Brighten Godfrey, Yashar Ganjali, and Amin Firoozshahian. 2017. DRILL: Micro Load Balancing for Low-latency Data Center Networks. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. ACM, New York, NY, USA, 225–238. https://doi.org/10.1145/3098822.3098839

[45] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, and George Varghese. 2014. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters. In *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM, New York, NY, USA, 503–514. https://doi.org/10.1145/2619239.2626316

[46] Naga Katta, Mukesh Hira, Changhoon Kim, Anirudh Sivaraman, and Jennifer Rexford. 2016. HULA: Scalable Load Balancing Using Programmable Data Planes. In *Proceedings of the Symposium on SDN Research (SOSR '16)*. Association for Computing Machinery, New York, NY, USA, 1. https://doi.org/10.1145/2890955.2890968

[47] Jinbin Hu, Chaoliang Zeng, Zilong Wang, Junxue Zhang, Kun Guo, Hong Xu, Jiawei Huang, and Kai Chen. 2024. Load Balancing With Multi-Level Signals for Lossless Datacenter Networks. *IEEE/ACM Transactions on Networking* 32, 3 (2024), 2736–2748. https://doi.org/10.1109/TNET.2024.3366336

[48] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. 2010. Hedera: Dynamic Flow Scheduling for Data Center Networks. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI'10)*. USENIX Association, USA, 19. https://www.usenix.org/legacy/events/nsdi10/tech/full_papers/al-fares.pdf

[49] Andrew R Curtis, Wonho Kim, and Praveen Yalagandula. 2011. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. In *2011 Proceedings IEEE INFOCOM*. IEEE, Shanghai, China, 1629–1637. https://doi.org/10.1109/INFCOM.2011.5934956

[50] Wei Wang, Yi Sun, Kai Zheng, Mohamed Ali Kaafar, Dan Li, and Zhongcheng Li. 2014. Freeway: Adaptively Isolating the Elephant and Mice Flows on Different Transmission Paths. In *2014 IEEE 22nd International Conference on Network Protocols*. IEEE, Raleigh, NC, USA, 362–367. https://doi.org/10.1109/ICNP.2014.59

[51] Xinglong Diao, Huaxi Gu, Wenting Wei, Guoyong Jiang, and Baochun Li. 2024. Deep Reinforcement Learning Based Dynamic Flowlet Switching for DCN. *IEEE Transactions on Cloud Computing* 12, 2 (2024), 580–593. https://doi.org/10.1109/TCC.2024.3382132

[52] Jonathan Perry, Amy Ousterhout, Hari Balakrishnan, Devavrat Shah, and Hans Fugal. 2014. Fastpass: a centralized "zero-queue" datacenter network. In *Proceedings of the 2014 ACM Conference on SIGCOMM (SIGCOMM '14)*. Association for Computing Machinery, New York, NY, USA, 307–318. https://doi.org/10.1145/2619239.2626309

[53] Yanshu Wang, Dan Li, Yuanwei Lu, Jianping Wu, Hua Shao, and Yutian Wang. 2022. Elixir: A High-performance and Low-cost Approach to Managing Hardware/Software Hybrid Flow Tables Considering Flow Burstiness. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. USENIX Association, Renton, WA, 535–550. https://www.usenix.org/conference/nsdi22/presentation/wang-yanshu

[54] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. 2011. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*. USENIX Association, Boston, MA, 1–14. https://www.usenix.org/conference/nsdi11/design-implementation-and-evaluation-congestion-control-multipath-tcp

[55] Abdul Kabbani, Balajee Vamanan, Jahangir Hasan, and Fabien Duchene. 2014. FlowBender: Flow-level Adaptive Routing for Improved Latency and Throughput in Datacenter Networks. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*. ACM, New York, NY, USA, 149–160. https://doi.org/10.1145/2674005.2674985

[56] Naga Katta, Aditi Ghag, Mukesh Hira, Isaac Keslassy, Aran Bergman, Changhoon Kim, and Jennifer Rexford. 2017. Clove: Congestion-Aware Load Balancing at the Virtual Edge. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '17)*. Association for Computing Machinery, New York, NY, USA, 323–335. https://doi.org/10.1145/3143361.3143401

[57] S Floyd and V Jacobson. 1993. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking* 1, 4 (1993), 397–413. https://doi.org/10.1109/90.251892

[58] Jiawei Huang, Wenjun Lyu, Weihe Li, Jianxin Wang, and Tian He. 2021. Mitigating Packet Reordering for Random Packet Spraying in Data Center Networks. *IEEE/ACM Transactions on Networking* 29, 3 (2021), 1183–1196. https://doi.org/10.1109/TNET.2021.3056601

[59] Jinbin Hu, Jiawei Huang, Wenjun Lv, Yutao Zhou, Jianxin Wang, and Tian He. 2019. CAPS: Coding-Based Adaptive Packet Spraying to Reduce Flow Completion Time in Data Center. *IEEE/ACM Transactions on Networking* 27, 6 (2019), 2338–2353. https://doi.org/10.1109/TNET.2019.2945863

[60] Jiawei Huang, Shiqi Wang, Shuping Li, Shaojun Zou, Jinbin Hu, and Jianxin Wang. 2021. HTPC: heterogeneous traffic-aware partition coding for random packet spraying in data center networks. *Journal of Cloud Computing* 10, 1 (2021), 31. https://doi.org/10.1186/s13677-021-00248-4

[61] Tobias Flach, Nandita Dukkipati, Andreas Terzis, Barath Raghavan, Neal Cardwell, Yuchung Cheng, Ankur Jain, Shuai Hao, Ethan Katz-Bassett, and Ramesh Govindan. 2013. Reducing Web Latency: the Virtue of Gentle Aggression. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*. ACM, New York, NY, USA, 159–170. https://doi.org/10.1145/2486001.2486014

[62] Nan Jiang, John Kim, and William J Dally. 2009. Indirect adaptive routing on large scale interconnection networks. In *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA '09)*. Association for Computing Machinery, New York, NY, USA, 220–231. https://doi.org/10.1145/1555754.1555783

[63] Yao Kang, Xin Wang, and Zhiling Lan. 2021. Q-adaptive: A Multi-Agent Reinforcement Learning Based Routing on Dragonfly Network. In *Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '21)*. Association for Computing Machinery, New York, NY, USA, 189–200. https://doi.org/10.1145/3431379.3460650

[64] Venkat Arun and Hari Balakrishnan. 2018. Copa: Practical Delay-Based Congestion Control for the Internet. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. USENIX Association, Renton, WA, 329–342. https://www.usenix.org/conference/nsdi18/presentation/arun

[65] Gaoxiong Zeng, Wei Bai, Ge Chen, Kai Chen, Dongsu Han, Yibo Zhu, and Lei Cui. 2022. Congestion Control for Cross-Datacenter Networks. *IEEE/ACM Transactions on Networking* 30, 5 (2022), 2074–2089. https://doi.org/10.1109/TNET.2022.3161580

[66] Gautam Kumar, Nandita Dukkipati, Keon Jang, Hassan M. G. Wassel, Xian Wu, Behnam Montazeri, Yaogong Wang, Kevin Springborn, Christopher Alfeld, Michael Ryan, David Wetherall, and Amin Vahdat. 2020. Swift: Delay is Simple and Effective for Congestion Control in the Datacenter *(SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 514–528. https://doi.org/10.1145/3387514.3406591

[67] Yanfang Le, Rong Pan, Peter Newman, Jeremias Blendin, Abdul Kabbani, Vipin Jain, Raghava Sivaramu, and Francis Matus. 2024. STrack: A Reliable Multipath Transport for AI/ML Clusters. arXiv:2407.15266 [cs.NI] https://arxiv.org/abs/2407.15266