Congestion Control for AI Workloads with Message-Level Signaling

Yuxuan Li
Hong Kong University of Science and Technology
Hong Kong, China
ylisn@connect.ust.hk

Zhenghang Ren
Hong Kong University of Science and Technology
Hong Kong, China
zrenak@cse.ust.hk

Wenxue Li
Hong Kong University of Science and Technology
Hong Kong, China
wlicv@connect.ust.hk

Xiangzhou Liu
Hong Kong University of Science and Technology
Hong Kong, China
xliugg@connect.ust.hk

Kai Chen
Hong Kong University of Science and Technology
Hong Kong, China
kaichen@cse.ust.hk

Abstract

Large-scale AI training is among the most demanding workloads in datacenter networks. The unique characteristics of low flow entropy exacerbate flow collisions under traditional Equal-Cost-Multi-Path (ECMP) Load Balance (LB). To mitigate these issues, per-packet LB mechanisms, such as packet spraying and adaptive routing, emerge as a promising alternative to eliminate flow collisions. However, since per-packet LB inherently rebalances traffic to absorb partial congestion, existing Congestion Control (CC) algorithms overlook this behavior and overreact to per-packet congestion signals which only reflect individual path congestion rather than congestion across multiple paths. Consequently, this overreaction leads to unnecessary rate reductions, resulting in throughput degradation in model training processes.

To tackle the incompatibility between current CC and per-packet LB, we propose using message-level signals as a more accurate alternative to capture congestion across multiple paths. The key idea is to leverage message delays rather than solely relying on per-packet delays to better capture the global congestion state. Based on these signals, we build MCC, a congestion control mechanism that is compatible with per-packet LB for AI workloads. Specifically, MCC adopts a window-based approach and reduces in-flight data only when the global pipe is congested. Furthermore, considering the challenges related to hardware programmability and compatibility, MCC adopts a software-based approach to control traffic in collective communication libraries (CCLs), achieving both the efficiency and independence of the NIC hardware. Our simulation results demonstrate that MCC effectively alleviates congestion without overreaction in collective communication operations.

CCS Concepts

• **Networks** → **Transport protocols**.

APNET 2025, Shang Hai, China

ACM ISBN 979-8-4007-1401-6/25/08
https://doi.org/10.1145/3735358.3735378

Keywords

Datacenter Networks, Congestion Control

ACM Reference Format:
Yuxuan Li, Zhenghang Ren, Wenxue Li, Xiangzhou Liu, and Kai Chen. 2025. Congestion Control for AI Workloads with Message-Level Signaling. In *9th Asia-Pacific Workshop on Networking (APNET 2025), August 07–08, 2025, Shang Hai, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3735358.3735378

1 Introduction

The growing scale of distributed AI training demands high-throughput collective communication across tens of thousands of GPUs, which has imposed significant pressure on the network infrastructure. However, traditional per-flow LB mechanisms, such as ECMP, have been found insufficient for distributing AI workloads due to flow collision problems [10]. These collisions not only reduce link utilization, but also impair AI communication throughput, highlighting the need for finer-grained LB to fully utilize multiple network paths for each flow. Therefore, per-packet LB techniques, such as packet spraying [5] and Adaptive Routing (AR) [26], provide promising solutions and effectively mitigate flow collision problems. They dynamically route packets across all available paths, inherently has the ability to absorb internal congestion imbalances.

Although per-packet LB builds a promising foundation for AI network transport, widely adopted CC mechanisms, such as DCQCN [28] are *incompatible* with per-packet LB and perform poorly due to the following reasons:

- Congestion signals, such as Explicit Congestion Notification (ECN) and packet Round-Trip Time (RTT), only represent congestion on individual paths.
- The growing packet processing rate enforces coalescing ACKs, preventing delay measurement on all packets and identification of congestion across all paths.
- Traditional CCs overlook the dynamic rebalancing ability of per-packet LB, thus often overreact to partial congestion on individual paths.

In per-packet LB, packets within a network flow are dynamically routed to multiple paths, and packets can be considered to traverse a *network pipe*. However, the per-packet congestion signal only reflects the congestion status of an individual path, failing to capture

the overall congestion of the network pipe. Moreover, per-packet signals highly fluctuate as the degrees of congestion across paths are different, making it difficult for CC algorithms to distinguish transient local congestion on individual paths from real congestion on multiple routes. As a result, when congestion arises on one path while others remain uncongested, CC algorithms misinterpret it as global congestion and slow down, leading to unnecessary rate reductions and ultimately degrading overall throughput.

Moreover, the real-world network bandwidths across links are intrinsically asymmetric and have imbalanced topology [2, 22, 23]. For instance, in networks with heterogeneous switch components, imbalanced striping [27] occurs when the uplink number on a switch is not an integer multiple of the total number of upper-layer switches, thus inherently resulting in network asymmetry. Additionally, link failures [11] are well-documented issues, and any link between two switches that goes down can lead to uneven bandwidth distribution, exacerbating network imbalance. In such asymmetric topologies, CC overreaction is unavoidable, forcing existing AI training practices to disable CC [10, 17] and rely on priority-based flow control (PFC) to avoid network buffer overflow, which further brings issues such as network deadlock [12] and head-of-line blocking [9, 19].

Facing invalid packet-level congestion signals, we pose a key question: *How can we properly interpret congestion signals and control congestion under per-packet LB for AI training workloads?* In this paper, we answer this question by proposing message-level signaling for congestion control in AI workloads. Our key finding is that messages are more accurate units when acting as congestion signals in a multipath network with per-packet load balancing. As messages are split into multiple paths, they are inherently resilient to single path congestion. Additionally, with message-level signals properly interpreting congestion states across paths, we can prevent unnecessary slowdowns and overreactions, thus maintaining high throughput for AI training processes.

Achieving a message-level congestion control, however, has a key challenge that message-level congestion control contradicts with existing hardware offloaded transport in Network Interface Cards (NICs). Existing CC algorithms, such as DCQCN, control at packet-level and are implemented in NICs to ensure high throughput. While commercial NICs are mostly non-programmable, extending current CC to support message-level signals is impracticable, and implementing a new CC approach at those NICs can be quite challenging. Moreover, with heterogeneous NIC environments becoming inevitable, industry experience shows that incompatibilities in CC algorithms are a major cause of performance degradation [4, 15]. These limitations motivate us to build a NIC-independent congestion control that does not rely on the programmability of NICs.

We propose MCC, a message-level congestion control for collective communication in AI workloads. AI flows are originally generated through CCLs, which partition data volume into messages and transmit them in a pipeline fashion. As this inherent structure facilitates message-level signaling for coarser-grained congestion control, we implement MCC in CCLs to ensure efficient congestion management.

At its core, MCC measures message delay and expected delay of sampled packets within messages to regulate the number of in-flight messages. For delay measurement, MCC interacts with existing collective communication modules, which decompose the traffic as regular-sized messages. Since each message is composed of multiple packets and travels through multiple paths, message-level signals aggregate congestion information across the network pipe, providing dense and comprehensive sampling of network-wide congestion. By capturing overall congestion rather than transient fluctuations on individual paths, message-level signals prevent misinterpretations caused by localized congestion. To better accommodate AI workloads, MCC leverages concurrent flows and interconnect topologies for fast window adjustment.

This paper makes the following contributions:

- We identify message-level signals as more accurate congestion indicators for multipath networking than packet-level signals.
- We propose MCC as a prototype to handle message-level congestion signals in typical distributed AI training workloads, offering better compatibility with per-packet LB compared to current CC.
- Our preliminary results show that MCC outperforms the two de-facto congestion control standards, DCQCN and TIMELY, by reducing latency by at least 16 percent and improving throughput by 22 percent in collective communication.

2 Background

Per-packet LB has been recognized as a promising solution to overcome drawbacks in traditional flow-based load balancing, such as ECMP. We explain the advantages of per-packet LB over ECMP and analyze how existing CC mechanisms fail to handle congestion signals properly under per-packet LB.

2.1 Necessity of Per-Packet Load Balance

Traditional LB mechanisms, such as ECMP, suffer from flow-collision problems during communication in distributed AI applications. The root cause is that collective communication in AI training workloads features fewer, larger flows and low entropy [10, 13], which downgrades the entire throughput once two flows collide on the same path even when other paths are idle. Existing alternative techniques, such as QP scaling [10, 20] and flowlet routing [6], fail to eliminate the issue and low entropy persists in the communication pattern. Flowlet-based LB relies on flowlet gaps to trigger path switching, but rate shaping utilized by RDMA [18] leads to a continuous flow of packets with minimal and unstable time gaps, resulting in generating uneven packet groups and flow skew, which cannot fully solve low-entropy challenges. Scaling QP balances loads by assigning multiple QPs with different UDP source ports to a single flow, enabling the traffic to traverse through different paths. However, in some cases, the entropy cannot increase as expected [10]. Moreover, QP scaling requires maintaining more queue pairs simultaneously, posing more challenges in resource allocation.

To fully eliminate path collision, per-packet LB is necessary for its ability to spray packets to multiple paths and its flexibility to route packets based on local congestion. Consequently, load balancing at packet granularity is suitable and efficient for AI workloads to overcome the low-entropy challenges and fully leverage all available network paths.

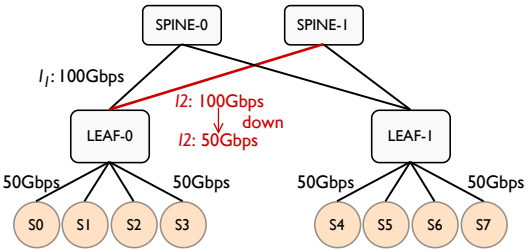


Figure 1: Asymmetric network topology with link speed downgrade.

2.2 CC Inefficiency under Per-packet LB

With per-packet LB fully leveraging paths for each flow, per-packet signaling, such as ECN, falls short of indicating network congestion because it is generated from a single path, leading to overreactions of existing CC mechanisms. To illustrate this, we consider DCQCN, the CC algorithm commonly deployed at scale with per-packet signals, and set up a simulation experiment in NS-3 with adaptive routing and DCQCN. We also consider adaptive routing without CC and only relying on PFC.

Due to the fact that link failures as well as imbalanced striping are common in real deployments, the actual bandwidths across links in the network are not symmetric, but have some links with lower bandwidths. Thus, we conduct the experiments in asymmetric topology as shown in Figure 1, all the links between the leaf and the spine switches have 100Gbps bandwidth, except for l_2 from leaf0 to spine1 obtains only 50Gbps due to link failure (one of the two 50Gbps uplinks fails). We set up 8 servers and each with 8 GPUs, and simulate distributed training with Data Parallelism (DP) degree 2, Tensor Parallelism (TP) degree 8, and Pipeline Parallelism (PP) degree 4, where 8 GPUs in the same server form a TP group, 4 GPUs under the same leaf switch form PP groups, and 2 GPUs across different spine switches form DP groups. Thus, we can observe one DP AllReduce traffic from server S_0, S_1, S_2, S_3 to S_4, S_5, S_6, S_7, respectively, during one iteration.

We experiment with three DP traffic sizes: 0.5 GB, 1 GB, and 2 GB. The corresponding Collective Completion Time (CCT) results of DCQCN+AR, PFC+AR are shown in Figure 2. In each message sending process, AR without CC assistance, i.e. PFC+AR, outperforms DCQCN+AR. The abnormal results provide evidence that DCQCN performs overconservative rate control decisions, thus an even more aggressive sending with frequent PFC can lead to lower CCT. The overreaction of DCQCN can also be observed in Figure 3, where we record the link utilization of l_1 and l_2 when performing DCQCN+AR. The link downgrades are frequently triggered in both links, leading to around 1.51x CCT delays that harm AI training performance, and achieved normalized throughput still below 60 percent.

The main reason for such link downgrades and performance degrades is that packet-level signaling in DCQCN only reflects congestion in single links, thus making senders overreact to temporary congestion without a global understanding of the whole network pipe. To illustrate this, we explore the relationship between ECN marking and link utilizations, as shown in Figure 3.

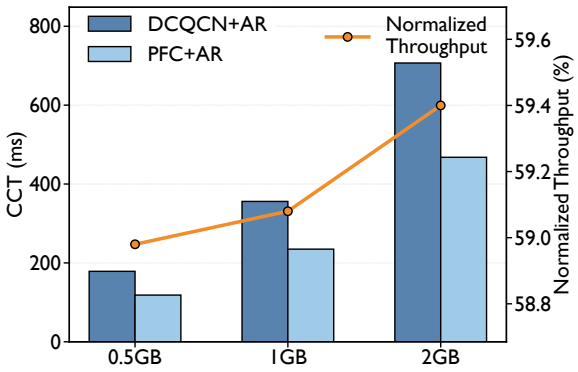


Figure 2: CCT and normalized throughput with three traffic sizes in data parallel processing.

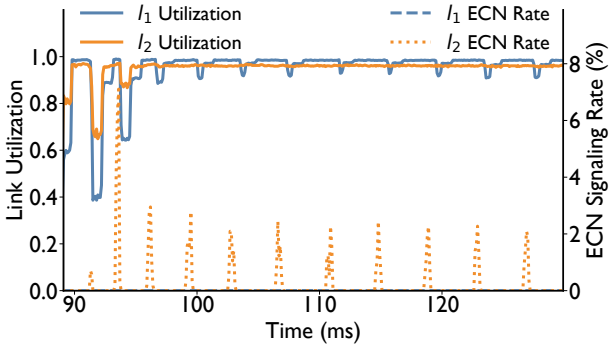


Figure 3: Link utilization and ECN marking rates of l_1 and l_2.

With adaptive routing, traffic from server S_0 to S_3 is distributed to l_1 and l_2 according to local congestion states. While l_2 has lower available bandwidth, queues are quickly built up and reach the ECN-mark threshold. Those per-packet ECN signals further trigger receivers to send CNPs, thus servers S_0 to S_3 react by cutting down sending rates in response to congestion signals. However, such ECN marking only indicates congestion in l_2, rather than that in l_1. As can be seen in Figure 3, it is obvious that the link downgrade in l_1 has high relevance with frequent ECN marking in l_2, rather than the ECN marks from itself. Thus, even though DCQCN rate deduction alleviates congestion in l_2, it also raises overreactions in l_1, where 1 percent ECN signaling rate causes utilization to rapidly drop to nearly 40 percent, causing downgrades in overall performance.

2.3 Opportunities from CCLs

As AI workloads are generally derived from CCLs, we found unique features in CCLs can provide valuable opportunities to generate, interpret, and leverage message-level congestion signals.

Chunk-based transport. In CCLs, initial messages are divided into smaller message chunks, with each chunk associated with its own Work Queue Element (WQE) and will be sequentially posted to Send Queue (SQ) for NIC transmission. Since each chunk is processed and transmitted independently, this mechanism allows for message-level control that decides whether to send the next chunk message. Moreover, as each message chunk typically sized at 64KB that consists of several packets, it further supports the

generation and reception of message-level congestion feedback, thereby facilitating our message-level signaling CC algorithm.

Topology awareness and application demand perception. During initialization, CCLs detect system topology and identify available interconnect bandwidths, which will be used to set up communication channels and further optimize communication. This information can also be used to estimate in-flight capacity and properly interpret congestion signals for better controlling. Moreover, the application-level requirements, for instance, the throughput demands of AI training jobs, can also be perceived and utilized to achieve CC fast recovery. Thus, the ability of CCLs to provide valuable additional information can effectively enhance the CC designs.

Mild incast. AI workloads are inherently resilient to incast problems as collective communication operations from CCLs are carefully designed and scheduled to avoid incast [7, 8, 13, 21]. For instance, in an AlltoAll operation where all participants exchange messages, CCLs schedule sending times to stagger arrivals at each receiver. Moreover, recent work [17] carefully designs new CCL for Mixure-of-Experts (MoE) training, which consists of substantial AlltoAll operations, and significantly alleviates incast problems. Therefore, when incast remains mild and fabric congestion can be well-managed by per-packet LB, the network maintains a more stable level of in-flight bytes. This stability allows CC mechanisms to regulate in-flight data and leverage coarse-grained message-level more effectively, preventing both excessive throttling and underutilization of bandwidth.

3 Message-level Congestion Signal

Existing CC mechanisms overreact to individual path congestion in a network with per-packet LB. In contrast, MCC interprets congestion signals on the message level and directly interacts with AI application messages in collective communication. The regular-sized messages and window-based in-flight message controlling enable MCC to bypass NIC programmability limitations when implementing congestion control.

Network pipe under per-packet LB. In multi-path transport with per-packet LB, each switch node routes every packet to the least congested port with adaptive routing. Thus, the network between end hosts works as a pipe, and packets can take different paths in this pipe from source to destination. However, under the network pipe model, neither senders nor receivers know the actual paths taken by each packet, as shown in Figure 4. Network congestion happens in the following cases:

- Individual path congestion: some paths encounters link reduction or failures, causing asymmetric bandwidth across links and port buffer build up.
- Global congestion: the aggregated ingress rate to a set of ports has surpassed the total rate of the egress. For example, global congestion may happen on a switch with oversubscribed upper links.

The global congestion is considered to be real congestion and needs to slow down. And individual path congestion happens mainly due to link failure and bandwidth cut-down that causes temporary asymmetry in path throughput. Since congestion signals are path-agnostic, i.e., cannot notify end hosts about the path

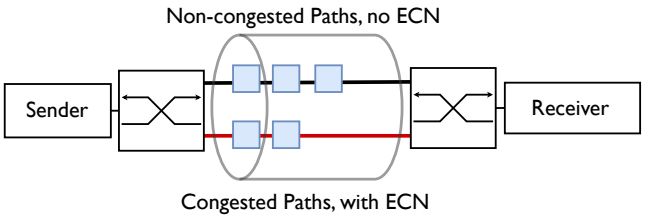


Figure 4: Congestion signals on individual paths.

information, end hosts overreact to individual path congestion and cut down sending rates, leading to underutilized links.

Message-level signals. To prevent overreaction in the congestion of individual paths, we extract congestion information by measuring the delay of messages and the delay of sampled packets in messages. Different from packet-level signals that mislead end hosts with individual path congestion and slow down on all routes, message delays are determined by the delay of a sequence of packets that span across multiple paths, which can effectively reduce the impact of individual path fluctuation. Moreover, we also capture and identify individual path congestion by sampling delays of packets, and accordingly estimate the expected packet delay.

To support this design, MCC collects timestamps from NICs to sample delays of packets in messages, which allows recording the time when the packet is put into the wire or received from the wire. With message-level signal carrying network pipe congestion information, MCC smooths transient fluctuations caused by individual path congestion and captures the global status of the entire network pipe.

Signal interpretation. MCC interprets message delay and sampled packet delay with *target_msg_delay* as well as *target_max_delay*. The congestion is classified into three levels:

- Non-Congestion. If the message delay is less than *target_msg_delay*, and the sampled delays are less than *target_max_delay*, it indicates that the network pipe has sufficient capacity to accommodate additional messages, allowing senders to increase the congestion window and let AR to switch route.
- Partial Congestion. When the message delay is lower than *target_msg_delay* but the maximum sampled delay is larger than *target_max_delay*, it means some paths exhibit congestion while others remain underutilized. In this case, MCC maintains the congestion window, leveraging AR's inherent ability to rebalance congestion by directing packets toward less congested paths, thereby fully utilizing network capacity without wasting bandwidth on non-congested paths.
- Global Congestion. In the case where the message delay is larger than *target_msg_delay*, it indicates severe congestion where all paths are saturated, and the entire network pipe is congested.

4 Efficient Message-level Control

Message-level congestion controls are incompatible with the existing packet-level approach mainly because of the hardware offload implementation and message-agnostic context on existing NIC packet processing. Moreover, as NIC speed increases, processing

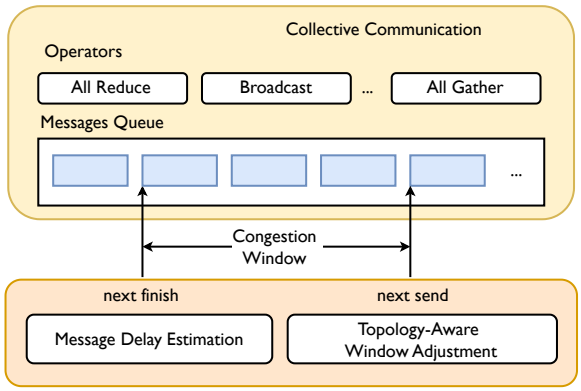


Figure 5: MCC operates in message granularity and directly interacts with collective communication.

messages efficiently becomes critical. We build MCC to leverage the message processing pipeline in existing collective communication, which is as efficient as packet-based congestion control in NICs.

Figure 5 shows the key components and workflow of MCC. Compared to existing general-purpose congestion controls, MCC distinguishes itself by cooperating with collective communications in AI applications and limiting on-the-fly data through message-level controlling. With the message delay estimation module, MCC actively samples the delays of packets when transmitting messages to reflect individual path congestion, and records averaged message delay as an indication of global network congestion. The topology-aware window adjustment module cooperates with collective communication to adjust the congestion window, achieving fast convergence in congested networks. Moreover, the hardware-independent implementation enables MCC to be readily deployable to existing infrastructures, especially commercial NICs without programmability and heterogeneous network environments.

Interaction with CCLs. As discussed in § 2.3, CCLs divide large messages into smaller chunks (typically 64 KB) for transmission, making it natural to treat these chunks as the basic unit for MCC's message-level signaling. Additionally, in AI training workloads, incast is mild and fabric congestion is well-managed, and leveraging message-level signals to regulate in-flight data can be enough and efficient for congestion control. To achieve this, MCC adopts a window-based congestion control approach to limit in-flight data, but takes a different approach by shifting the window control granularity from per-packet to per-message, and making transmission decisions at the message level.

Implementing in CCLs is easier and can be more effective compared with other alternatives. Since congestion signals are collected from messages which are directly posted by applications, packet-level congestion control struggles to process them effectively. The presence of multiple concurrent messages within the application necessitates signal aggregation, which packet-level mechanisms are not designed to handle. The CCL implementation approach is as efficient as hardware-offloaded transport because it controls in message granularity, allowing end hosts to get high throughput with little CPU usage.

During runtime, CCLs post message chunks to the Send Queue (SQ) and poll for completion events upon successful transmission.

Parameters	Values
initial_target_msg_delay	msg_size/bw + target_max_delay / 2 = 92 us
initial_target_max_delay	net_base_rtt + queue_delay =64 us

Table 1: Parameter settings in evaluation.

MCC intercepts these events and aggregates feedback from multiple paths to evaluate the attainable bandwidth for the entire network pipe. Based on this aggregated feedback, MCC dynamically adjusts the sending rate by modifying the inflight window size in message granularity. This window-based approach ensures that inflight data remains proportional to the network pipe's capacity, avoiding overreaction while maintaining high throughput.

Fast convergence. Existing CCLs, such as NCCL [1], inherently detect communication topology and interconnect bandwidth to optimize communication patterns for collective operations. MCC leverages this capability by incorporating topology-aware feedback into its congestion control decisions. Specifically, MCC uses topology and bandwidth information to estimate the optimal window size, accounting for variations in link capacities and congestion hotspots.

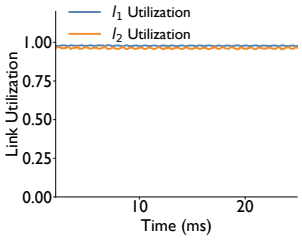
In the beginning, MCC finds the congestion hotspots with the highest oversubscription ratio and decides the window increase unit with the ratio. For example, with an 8:1 oversubscription, we add 1/8 Bandwidth-Delay-Product (BDP) to the window. Thus, a new message will be issued once the available window size is larger than the message size.

5 Evaluation

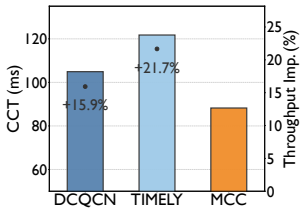
Evaluation setup. To validate the effectiveness of MCC, we conduct simulations in NS-3 in TACC platform [24, 25], and simulate representative micro-benchmarks in AI training workloads. In particular, we focus on AllReduce which is the most widely used operation in both TP and DP. The simulation uses the topology shown in Figure 1, which consists of 8 servers and 4 switches and serves as the communication fabric for collective messages. We configure 4 AllReduce groups, identical to those described in Section 2.2, with each transmitting a 250MB message. Since flows in collective communication are typically highly synchronized, we measure the CCT to better reflect the end-to-end efficiency of the operation. We further report the throughput improvement achieved by MCC to demonstrate its performance gains. In addition, we monitor the link utilization of l_1 and l_2 over the window where partial congestion occurs to highlight MCC's ability to mitigate overreaction to partial congestion. The initial congestion control parameters are configured as summarized in Table 1.

Baselines. To illustrate the efficiency of our CC algorithms, we utilize DCQCN, TIMELY as our baseline approaches for comparison with MCC under AR load balancing. We tune DCQCN mainly based on the parameters recommended by HPCC [16]. As DCQCN and TIMELY represent distinct congestion control algorithms that rely on different per-packet congestion signals, our results highlight the inefficiencies of per-packet signaling and demonstrate that MCC outperforms both.

Results. The results are depicted in Figure 6. We can find in Figure 6a that the link utilization in the non-congested path l_1 remains nearly 100 percent without significantly downgrading in partial



(a) Link utilizations of l_1 and l_2.



(b) CCT and throughput improvement.

Figure 6: Simulation Results for AllReduce.

congestion. Though with different congestion signals, per-packet RTT as well as ECN both react regressively to congestion in l_2, ignoring the capacity of l_1 and underutilizing the corresponding link. Thus, as can be seen in Figure 6b, by avoiding the unnecessary slowdown, MCC achieves up to 22 percent higher throughput, and the CCT of both DCQCN and TIMELY are significantly longer than that of MCC, and MCC achieves 16 percent ~27 percent lower latency. With AllReduce being the most common operation in modern AI distributed training, improving control efficiency in AllReduce is of great necessity.

6 Related Work

Multi-Path transport with per-packet LB. There have been multi-path transports leveraging available paths in networks to achieve efficient transmission. MP-RDMA [18] utilizes the 5-tuple hash feature in ECMP and indirectly controls packets to traverse multiple paths by modifying UDP source ports at endpoints. STrack [13] is designed specifically for AI workloads, and carefully devises an adaptive load balancing scheme to address low-entropy challenges. Unlike these works, which focus on designing transport-layer protocols to support multi-path transmissions and depend on endpoints for path selection, MCC leverages switches to perform per-packet LB and make path choices. Additionally, MCC focuses on designing a CC mechanism that is compatible with per-packet LB, emphasizing congestion management rather than the creation of new protocols.

CC with application-level participation. Researchers have proposed congestion control methods that leverage application-level information and avoid complexities associated with hardware modifications. RoGUE [14], Flor [15] design protocols that take message chunks as congestion control units to alleviate CPU costs for traditional workloads. Meta [10] leverages Clear-To-Send (CTS) signals and the buffer recycling scheme in CCLs for AI workload CC and necessitates modifications in network switches. DeepSeek v3 [3] eliminates reliance on CC algorithms altogether by focusing on NCCL optimizations and other techniques for networks that combine computation, communication, and storage traffic. MCC differs from them as it is specifically tailored for AI workloads, and fully integrates CCL information to achieve hardware-independent congestion control.

7 Conclusion

In this paper, we present MCC that offers compatibility with per-packet LB for collective communication in AI workloads. MCC exploits the effectiveness of message-level signals under per-packet LB thus accurately interpreting global congestion states in the high-bandwidth network pipe. Additionally, MCC fully leverages the opportunities provided by CCLs and is integrated into CCLs. Such hardware-independent implementation enables more flexible deployment of MCC, especially for non-programmable commercial NICs and heterogeneous network environments. Our preliminary results verify that MCC is effective with per-packet LB in AI training workloads, providing higher throughput, lower CCT without overreaction.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. This work is supported in part by the Hong Kong RGC TRS T41-603/20R, the GRF 16213621, and the ITC ACCESS. Kai Chen is the corresponding author.

References

[1] 2025. NVIDIA Collective Communications Library (NCCL) | NVIDIA Developer. https://developer.nvidia.com/nccl. (2025).

[2] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, et al. 2014. CONGA: Distributed congestion-aware load balancing for datacenters. In *Proceedings of the 2014 ACM conference on SIGCOMM*. 503–514.

[3] Wei An, Xiao Bi, Guanting Chen, Shanhuang Chen, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Wenjun Gao, Kang Guan, Jianzhong Guo, Yongqiang Guo, Zhe Fu, Ying He, Panpan Huang, Jiashi Li, Wenfeng Liang, Xiaodong Liu, Xin Liu, Yiyuan Liu, Yuxuan Liu, Shanghao Lu, Xuan Lu, Xiaotao Nie, Tian Pei, Junjie Qiu, Hui Qu, Zehui Ren, Zhangli Sha, Xuecheng Su, Xiaowen Sun, Yixuan Tan, Minghui Tang, Shiyu Wang, Yaohui Wang, Yongji Wang, Ziwei Xie, Yiliang Xiong, Yanhong Xu, Shengfeng Ye, Shuiping Yu, Yukun Zha, Liyue Zhang, Haowei Zhang, Mingchuan Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, and Yuheng Zou. 2024. Fire-Flyer AI-HPC: A Cost-Effective Software-Hardware Co-Design for Deep Learning. (2024). arXiv:cs.DC/2408.14158 https://arxiv.org/abs/2408.14158

[4] Wei Bai, Shanim Sainul Abdeen, Ankit Agrawal, Krishan Kumar Attre, Paramvir Bahl, Ameya Bhagat, Gowri Bhaskara, Tanya Brokhman, Lei Cao, Ahmad Cheema, et al. 2023. Empowering azure storage with {RDMA}. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 49–67.

[5] Jiaxin Cao, Rui Xia, Pengkun Yang, Chuanxiong Guo, Guohan Lu, Lihua Yuan, Yixin Zheng, Haitao Wu, Yongqiang Xiong, and Dave Maltz. 2013. Per-packet load-balanced, low-latency routing for clos-based data center networks. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. 49–60.

[6] Peirui Cao, Wenxue Cheng, Shizhen Zhao, and Yongqiang Xiong. 2024. Network Load Balancing with Parallel Flowlets for AI Training Clusters. In *Proceedings of the 2024 SIGCOMM Workshop on Networks for AI Computing*. 18–25.

[7] Vinton Cerf and Robert Kahn. 1974. A protocol for packet network intercommunication. *IEEE Transactions on communications* 22, 5 (1974), 637–648.

[8] Peng Cheng, Fengyuan Ren, Ran Shu, and Chuang Lin. 2014. Catch the whole lot in an action: Rapid precise packet loss notification in data center. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. 17–28.

[9] Wenxue Cheng, Kun Qian, Wanchun Jiang, Tong Zhang, and Fengyuan Ren. 2020. Re-architecting congestion management in lossless ethernet. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 19–36.

[10] Adithya Gangidi, Rui Miao, Shengbao Zheng, Sai Jayesh Bondu, Guilherme Goes, Hany Morsy, Rohit Puri, Mohammad Riftadi, Ashmitha Jeevaraj Shetty, Jingyi Yang, Shuqiang Zhang, Mikel Jimenez Fernandez, Shashidhar Gandham, and Hongyi Zeng. 2024. RDMA over Ethernet for Distributed Training at Meta Scale. In *Proceedings of the ACM SIGCOMM 2024 Conference (Acm Sigcomm '24)*. Association for Computing Machinery, New York, NY, USA, 57–70. https://doi.org/10.1145/3651890.3672233

[11] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. 2011. Understanding network failures in data centers: measurement, analysis, and implications. In *Proceedings of the ACM SIGCOMM 2011 Conference*. 350–361.

[12] Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni, Jianxi Ye, Jitu Padhye, and Marina Lipshteyn. 2016. RDMA over commodity ethernet at scale. In *Proceedings of the 2016 ACM SIGCOMM Conference*. 202–215.

[13] 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 preprint arXiv:2407.15266* (2024).

[14] Yanfang Le, Brent Stephens, Arjun Singhvi, Aditya Akella, and Michael M Swift. 2018. Rogue: Rdma over generic unconverged ethernet. In *Proceedings of the ACM symposium on cloud computing*. 225–236.

[15] Qiang Li, Yixiao Gao, Xiaoliang Wang, Haonan Qiu, Yanfang Le, Derui Liu, Qiao Xiang, Fei Feng, Peng Zhang, Bo Li, et al. 2023. Flor: An open high performance {RDMA} framework over heterogeneous {RNICs}. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*. 931–948.

[16] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, et al. 2019. HPCC: High precision congestion control. In *Proceedings of the ACM special interest group on data communication*. 44–58.

[17] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437* (2024).

[18] 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)*. 357–371.

[19] 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*. 313–326.

[20] Kun Qian, Yongqing Xi, Jiamin Cao, Jiaqi Gao, Yichi Xu, Yu Guan, Binzhang Fu, Xuemei Shi, Fangbo Zhu, Rui Miao, et al. 2024. Alibaba hpn: A data center network for large language model training. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 691–706.

[21] 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*. 207–218.

[22] 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*. 151–162.

[23] 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)*. 407–420.

[24] Kaiqiang Xu, Decang Sun, Han Tian, Junxue Zhang, and Kai Chen. 2025. {GREEN}: Carbon-efficient Resource Scheduling for Machine Learning Clusters. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. 999–1014.

[25] Kaiqiang Xu, Decang Sun, Hao Wang, Zhenghang Ren, Xinchen Wan, Xudong Liao, Zilong Wang, Junxue Zhang, and Kai Chen. 2025. Design and Operation of Shared Machine Learning Clusters on Campus. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*. 295–310.

[26] David Zats, Tathagata Das, Prashanth Mohan, Dhruba Borthakur, and Randy Katz. 2012. DeTail: Reducing the flow completion time tail in datacenter networks. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*. 139–150.

[27] 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*. 1–14.

[28] 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. *ACM SIGCOMM Computer Communication Review* 45, 4 (2015), 523–536.