Revisiting RDMA Reliability for Lossy Fabrics

Wenxue Li*
Hong Kong University of Science and Technology, Huawei
Hong Kong, China

Xiangzhou Liu
Hong Kong University of Science and Technology
Hong Kong, China

Yunxuan Zhang
Hong Kong University of Science and Technology
Hong Kong, China

Zihao Wang
Hong Kong University of Science and Technology, Hong Kong, China

Wei Gu
Huawei
Nanjing, China

Tao Qian
Huawei
Beijing, China

Gaoxiong Zeng
Huawei
Shenzhen, China

Shoushou Ren
Huawei
Beijing, China

Xinyang Huang
Hong Kong University of Science and Technology, Hong Kong, China

Zhenghang Ren
Hong Kong University of Science and Technology, Hong Kong, China

Bowen Liu
Hong Kong University of Science and Technology, Hong Kong, China

Junxue Zhang
Hong Kong University of Science and Technology, Hong Kong, China

Kai Chen
Hong Kong University of Science and Technology, Hong Kong, China

Bingyang Liu
Huawei
Shenzhen, China

Abstract

Due to the high operational complexity and limited deployment scale of lossless RDMA networks, the community has been exploring efficient RDMA communication over lossy fabrics. State-of-the-art (SOTA) lossy RDMA solutions implement a simplified selective repeat mechanism in RDMA NICs (RNICs) to enhance loss recovery efficiency. However, these solutions still face performance challenges, such as unavoidable ECMP hash collisions and excessive retransmission timeouts (RTOs). In this paper, we revisit RDMA reliability with the goals of being independent of PFC, compatible with packet-level load balancing, free from RTO, and friendly to hardware offloading. To this end, we propose DCP, a transport architecture that co-designs both the switch and RNICs, fully meeting the design goals. At its core, DCP-Switch introduces a simple yet effective lossless control plane, which is leveraged by DCP-RNIC to enhance reliability support for high-speed lossy fabrics, primarily including header-only-based retransmission and bitmap-free packet tracking. We prototype DCP-Switch using P4 switch and DCP-RNIC using FPGA. Extensive experiments demonstrate that DCP achieves 1.6× and 2.1× performance improvements, compared to SOTA lossless and lossy RDMA solutions, respectively.

*This work was done while Wenxue Li was an intern at Huawei.

SIGCOMM '25, Coimbra, Portugal

ACM ISBN 979-8-4007-1524-2/25/09
https://doi.org/10.1145/3718958.3750480

CCS Concepts

• **Networks** → **Transport protocols**; **Data center networks**.

Keywords

RDMA NICs, Reliability, Lossy Fabrics, Lossless Control Plane

ACM Reference Format:
Wenxue Li, Xiangzhou Liu, Yunxuan Zhang, Zihao Wang, Wei Gu, Tao Qian, Gaoxiong Zeng, Shoushou Ren, Xinyang Huang, Zhenghang Ren, Bowen Liu, Junxue Zhang, Kai Chen, and Bingyang Liu. 2025. Revisiting RDMA Reliability for Lossy Fabrics. In *ACM SIGCOMM 2025 Conference (SIGCOMM '25), September 8–11, 2025, Coimbra, Portugal.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3718958.3750480

1 Introduction

Remote Direct Memory Access (RDMA) is a widely adopted high-speed networking technique in modern datacenters (DCs), driven by the demanding performance requirements of applications [16, 22, 23, 41, 42, 53]. RDMA delivers high performance by offloading the entire network stack to RDMA NICs (RNICs). Traditional RNICs (RNIC-GBN) adopt a Go-Back-N (GBN) retransmission mechanism to handle packet loss, which significantly degrades performance in lossy Ethernet fabrics [42, 53, 56], prompting operators to rely on Priority Flow Control (PFC) [1] to ensure lossless transmission [16, 22, 23, 41, 56]. However, PFC is a coarse-grained mechanism that introduces several issues (§2.1), such as head-of-line (HoL) blocking, congestion spreading, deadlock, and restrictions on deployment distance [16, 23, 25, 29, 52].

Recently, both industry and academia have been exploring efficient RDMA communication over lossy fabrics (without PFC enabled) to avoid the limitations and drawbacks of PFC [5, 11, 40, 42, 49]. The primary goal of lossy RDMA transport is to maintain efficiency in lossy fabrics while preserving the offloading capabilities

of RNICs. Many previous works have focused on software-based transports, which do not benefit from RNIC offloading [18, 26, 47]. SOTA lossy RDMA solutions (RNIC-SR) implement a simplified selective repeat (SR) mechanism in RNICs to enhance loss recovery efficiency and avoid significant performance degradation due to packet loss [8, 9, 42, 53]. However, even with RNIC-SR, performance issues persist in lossy fabrics (§2.2).

First, RNIC-SR assumes flow-level single-path transmission, with Equal-Cost Multi-Path (ECMP) as the default load balancing (LB) scheme in the network. However, ECMP hashing collisions are inevitable and their impact on throughput degradation is increasingly significant in today's datacenter workloads [38, 50]. Second, certain lost packets (such as retransmitted and tail packets) cannot be recovered through fast retransmission in RNIC-SR; instead, they rely on retransmission timeouts (RTOs). This leads to an increase in RTO events, significantly prolonging tail latency.

Packet-level LB techniques (such as packet spraying [21] and adaptive routing (AR) [3]) are promising alternatives to ECMP and are increasingly gaining attention from various scenarios, such as large-scale LLM training [22]. However, RNIC-SR is natively incompatible with packet-level LB. The root cause of this incompatibility is that packet-level LB can cause out-of-order (OOO) packet arrivals even in the absence of packet loss, whereas RNIC-SR assumes that all OOO packets are due to packet loss. As a result, combining packet-level LB with RNIC-SR leads to excessive spurious retransmissions, significantly degrading throughput.

Based on these issues, we pose the question: *Can we revisit RDMA reliability to meet the following requirements?*

- (R1) Independence from PFC to avoid its drawbacks, extending network scale and communication length.
- (R2) Compatibility with packet-level LB, eliminating hotspots and increasing network throughput.
- (R3) Ability to quickly retransmit any lost packet without relying on RTO, reducing network latency.
- (R4) A hardware-oriented design, with the feasibility of low memory and processing overhead.

We answer this question optimistically with DCP, a transport architecture consisting of DCP-RNIC, which revisits the reliability support for high-speed lossy fabrics, and DCP-Switch, which introduces a simple yet effective innovation to enhance DCP-RNIC. DCP conceptually defines the data plane (DP) for payload transfer and the control plane (CP) for header transfer. While lossless RDMA ensures that both DP and CP are lossless via PFC, DCP-Switch ensures a lossless CP while allowing the DP to operate in a lossy manner. The key design point of DCP is to leverage the lossless CP feature to enhance the RNICs' reliability, enabling compatibility with packet-level LB, precise retransmission, and minimal memory and processing overhead.

DCP-Switch implements a packet trimming mechanism similar to [18] ((R1)). Specifically, when there is no congestion, the data packets are queued in the normal queue (a.k.a., data queue) and forwarded to the receiver. When the data queue becomes congested, the switch trims the payload from packets, modifies flags in the remaining header, and enqueues the header-only (HO) packet into another queue (a.k.a., control queue). To ensure *lossless CP transfer* (§4.2) while avoiding starvation of DP delivery, the switch uses a weighted round-robin (WRR) scheduler to prioritize the control queue and ensure lossless delivery of HO packets[1].

Leveraging the lossless CP transfer, DCP-RNIC employs a precise and fast *HO-based retransmission* (§4.3). Specifically, upon receiving an HO packet, the receiver swaps the source and destination fields in the HO packet and forwards it to the sender. The sender then retransmits the lost packet precisely, based on the PSN carried by the HO packet ((R3)). This mechanism is unaffected by OOO packets caused by packet-level LB ((R2)). DCP-RNIC incorporates microarchitecture innovations to minimize PCIe transactions, enhancing retransmission efficiency. Furthermore, since HO packets are stateless, the retransmission rate is inherently tied to their arrival rate. To address it, DCP-RNIC adds a retransmission queue to enable a controllable retransmission rate.

DCP-RNIC supports *order-tolerant packet reception* at the receiver, enabling compatibility with packet-level LB ((R2)) (§4.4). This is achieved through a thoughtfully designed RDMA header extension that allows each packet to specify its corresponding memory address. Consequently, receiver-side RNIC can directly write any packet, whether in-order or out-of-order, to its appropriate application memory location, eliminating the need for reordering buffers.

Traditionally, RNICs maintain bitmaps to track which packets have been received or lost. This bitmap-based approach introduces trade-offs between RNIC memory overhead and packet processing efficiency, which limits connection scalability [39, 53] and packet rate (packet-per-second) [30, 42], respectively. DCP-RNIC leverages the "exactly once" feature of the lossless CP to eliminate the need for packet-level bitmaps. Specifically, it employs a *bitmap-free packet tracking* scheme (§4.5), using packet counting to track aggregated message-level information, significantly reducing both memory overhead and packet processing cycles ((R4)). Furthermore, DCP-RNIC incorporates a coarse-grained timeout mechanism as a fallback to ensure reliability if the lossless control plane assumption is violated (e.g., link/switch crashes).

We prototype DCP-Switch using P4 programmable switch [2] and DCP-RNIC using FPGA [4] (§5). The DCP-RNIC prototype consumes only 1.7% and 1.1% more computation and memory resources compared to RNIC-GBN. We evaluate DCP through extensive testbed experiments and simulations (§6). The testbed experiments show that DCP maintains consistent throughput when combined with AR[2] and achieves 1.6×∼72× higher loss recovery efficiency and 42% lower completion time for AI workloads, compared to Mellanox CX5 RNIC. Large-scale simulations demonstrate that DCP achieves 2.1× and 1.6× improvements in realistic workloads, compared to IRN [42] and MP-RDMA [38], respectively. Moreover, DCP achieves greater improvement in cross-DC scenarios, and its lossless CP remains robust under severe incast congestion.

2 Background and Motivation

2.1 Lossless RDMA Network

RDMA was originally designed for lossless InfiniBand networks. Thus, RNICs were initially equipped with a Go-Back-N retransmission mechanism to simplify their processing logic. RDMA over

[1] More precisely, "HO packet loss is very rare," as there is no mechanism that can prevent losses under heavy load.

[2] We implement an in-network adaptive routing mechanism in this work.

ASIC	Tomahawk 3	Tomahawk 5	Tofino 1	Tofino 2	Spectrum	Spectrum-4
Capacity (ports × bandwidth per port)	32 × 400 Gbps	64 × 800 Gbps	32 × 100 Gbps	32 × 400 Gbps	32 × 100 Gbps	64 × 800 Gbps
Total buffer	64 MB	165 MB	20 MB	64 MB	16 MB	160 MB
Buffer per port per 100Gbps	0.5 MB	0.32 MB	0.62 MB	0.5 MB	0.5 MB	0.31 MB
Max. lossless length (1 queue)	4.1 km	2.62 km	5.08 km	4.1 km	4.1 km	2.56 km
Max. lossless length (8 queues)	512 m	327 m	634 m	512 m	512 m	320 m

Table 1: The maximum lossless communication distance with PFC enabled of various commodity switching ASICs.

Converged Ethernet version 2 (RoCEv2) encapsulates the InfiniBand header within a UDP layer, enabling RDMA to operate over Ethernet. Traditional RoCEv2 RNICs (i.e., RNIC-GBN) inherit the Go-Back-N mechanism and depend on PFC [1] to transform Ethernet into a lossless fabric [42, 53, 56]. Despite its utility, PFC is a coarse-grained mechanism that introduces several challenges in performance degradation, operational complexity, and scalability limitations, inhibiting its broader deployment.

PFC operates as follows: when the ingress port/queue length exceeds a specified threshold, a PAUSE signal is sent to all related upstream egress ports/queues, instructing them to halt data transmission until a RESUME signal is received. As a result, PFC causes issues such as HoL blocking, PFC storms, and deadlocks, significantly degrading overall network performance [23, 25, 29, 56].

Moreover, because PFC spreads hop-by-hop, a single failure can have a cascading effect, potentially impacting the entire network. To mitigate this, operators employ various monitoring schemes to limit the scope of the failure. For example, a PFC watchdog detects if a queue has remained in the paused state for an abnormally long duration; if this occurs, the system disables PFC and drops all packets [16, 25]. However, this inevitably complicates network operations and cannot fully avoid all accidents.

Additionally, PFC requires switches to reserve sufficient buffer space (i.e., headroom) for in-flight packets between hops, making RoCEv2 suitable primarily for short-distance communication but impractical for long-distance scenarios (e.g., cross-datacenter) [16, 52]. Meanwhile, switch buffers are scarce resources, further exacerbating this situation. We list the capacity and buffer information of several commodity datacenter switching chips in Table 1. We calculate the maximum lossless communication distance (L) that these switches can support with PFC enabled as follows:

$$L = \frac{buffer}{bandwidth \times one\text{-}hop\text{-}delay \times 2} \quad (1)$$

where for example the one-hop-delay of a 1 km fiber is approximately $5\mu s$[3]. The results show that commodity switches face challenges in scaling the distance to tens of kilometers. Some works propose adopting off-chip DRAM to store in-flight packets, successfully enabling adaptation to 100 km distances [16]. However, this approach inevitably complicates network operations and degrades communication performance, as DRAM bandwidth is significantly lower than on-chip switch SRAM.

2.2 Existing Solutions over Lossy Fabrics

Due to the inherent limitations of PFC, both industry and academia have been exploring efficient RDMA communication over lossy fabrics (without PFC enabled) to bypass its drawbacks [5, 11, 12, 40, 42, 49, 53]. On one hand, since traditional RNICs (i.e., RNIC-GBN) suffer significant throughput degradation upon packet loss due to their GBN retransmission logic, a class of works aims to develop efficient congestion control (CC) schemes that minimize overall switch queueing to reduce packet loss [12, 34, 37]. However, they do not address the root issue, as packet loss remains inevitable in lossy fabrics. On the other hand, many works focus on software-based efficient loss recovery mechanisms, which unfortunately cannot leverage RNIC offloading capabilities [18, 26, 47].

Therefore, one of the primary goals of lossy RDMA transport should be achieving efficient loss recovery while preserving the offloading capabilities of RNICs. SOTA lossy RDMA solutions, such as IRN [42] and new-generation ConnectX RNICs [8, 9] implement a simplified selective repeat (SR) mechanism in RNICs (RNIC-SR) to improve loss recovery efficiency. However, even with RNIC-SR, performance issues persist in lossy fabrics (IRN as an example[4]).

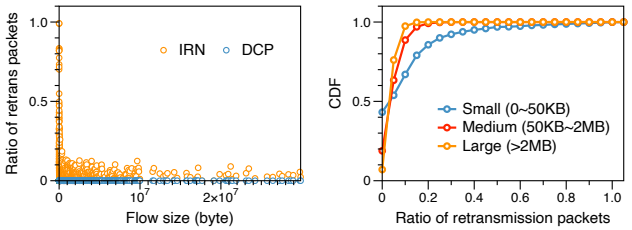
Issue #1: Incompatibility with Packet-level LB. The loss recovery mechanism of IRN operates as follows: Upon every out-of-order (OOO) packet arrival, the IRN receiver sends a selective acknowledgment (SACK), which carries both the cumulative acknowledgment (ePSN) and the PSN of the OOO packets. When a SACK is received or when a timeout occurs, the IRN sender enters loss recovery mode. It maintains a bitmap to track which packets have been cumulatively and selectively acknowledged. When in the loss recovery mode, the sender begins retransmitting lost packets as indicated by the bitmap. A packet is considered lost only if another packet with a higher PSN has been SACKed.

This mechanism functions correctly in single-path transmission, thus often used with ECMP in the network. However, ECMP collisions are unavoidable, and their impact on throughput degradation has become increasingly significant [38, 50]. Packet-level LB, such as adaptive routing [3], is gaining increasing interest from the community because it offers substantial advantages, such as avoiding ECMP collisions, eliminating network hotspots, and dynamically adapting to path failures. These benefits are particularly crucial in emerging scenarios such as LLM training and clouds [22, 27, 35, 49]. Although non-packet-level LBs, such as flowlet-based LB [14, 32, 51] and congestion-aware path switching [31, 46, 55], exist, they represent compromises that trade load balancing granularity for lower OOO degrees. Consequently, these approaches are generally less efficient than packet-level LB schemes.

IRN suffers from spurious retransmissions when combined with packet-level LBs. The root cause is that packet-level LB inherently causes OOO packet arrivals even in the absence of actual packet

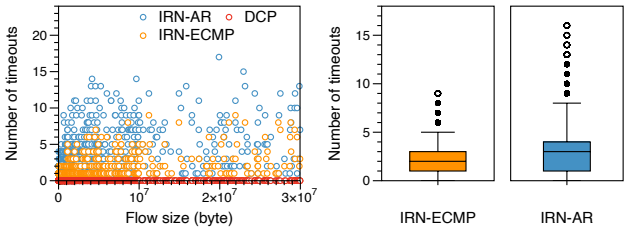
[3] $5\mu s = \frac{10^3 m}{2\times10^8 m/s}$, where 2×10^8 m/s is the transmission speed of light in fiber.

[4] We adopt IRN as a representative example of RNIC-SR solutions and primarily analyze IRN in this paper.



(a) Retransmission ratio. (b) CDF of IRN's retrans ratio.

Figure 1: IRN generates significant spurious retransmissions, whereas DCP avoids them.



(a) Background flow. (b) Incast flow.

Figure 2: IRN experiences excessive RTOs in both background and incast flows, whereas DCP avoids them.

loss. These OOO arrivals prompt the receiver to send SACKs, which in turn trigger the sender to incorrectly enter loss recovery mode and unnecessarily retransmit a large number of packets. Thus, integrating them leads to excessive spurious retransmissions.

We conducted an experiment using NS3 to validate this observation. The topology is a two-layer CLOS network with 256 endhosts and 32 switches. We used a WebSearch [15] workload with an average load of 0.3. IRN and DCP are deployed at the endhosts, while adaptive routing is applied in switches. During the experiment, we monitored loss events and found no packet loss in either setup. However, IRN generated a significant number of spurious retransmissions. Fig. 1a illustrates the ratio of retransmission packets across various flow sizes. The results reveal that retransmissions occur across all flow sizes, with a ratio up to 100%. We further categorized the flow size into three classes: small, medium, and large. The cumulative distribution function (CDF) of their retransmission ratios is shown in Fig. 1b. The results indicate that approximately 50%, 80%, and 90% of small, medium, and large flows, respectively, experience spurious retransmissions. In contrast, all flows in DCP avoid incorrect retransmissions entirely.

Issue #2: Excessive RTOs. Certain lost packets in IRN cannot be recovered through fast retransmission and instead rely on retransmission timeouts (RTOs). Specifically, the IRN sender requires a SACK to trigger the loss recovery mode. Consequently, if the tail packet of a flow is lost, no SACK is generated, preventing recovery through fast retransmission and necessitating reliance on an RTO. Additionally, to avoid retransmission ambiguities, the sender enters the loss recovery mode *only once* and remains in this state until it exits. The IRN sender exits loss recovery mode only when it receives an ePSN greater than the largest PSN of outstanding packets prior to entering the loss recovery mode. As a result, if the retransmitted packets are dropped again, they can only be recovered through an RTO. The increase in RTO events could significantly degrade the performance of IRN.

The loss of tail and retransmitted packets is common in modern datacenters [24, 28, 36]. The reasons are two-fold. First, the average flow size in modern datacenters becomes increasingly "shorter" [23, 33, 48], resulting in a higher frequency of tail packets. Second, packet loss often exhibits locality, as the congestion tends to persist for a period. During this interval, all passing packets, including rapidly retransmitted ones, are likely to be dropped.

To validate this analysis, we conducted an experiment using NS3 with a CLOS topology. The workload consists of a WebSearch workload with an average load of 0.3 (as background traffic) combined with 128-to-1 incast traffic at an average load of 0.1. We evaluated IRN under both ECMP and adaptive routing (AR) and measured the number of timeouts. The results, illustrated in Fig. 2, indicate that IRN experiences excessive timeouts in both background and incast flows. Furthermore, IRN-AR encounters even more timeouts due to the spurious retransmission traffic it generates, which increases network load and exacerbates congestion. In contrast, all flows in DCP experience no timeout.

3 Goals

Based on the above analysis, we aim to revisit RDMA reliability to meet the following requirements.

(R1) Independence from PFC. The proposed solution must eliminate the dependence on PFC to completely avoid its associated drawbacks. This requires the ability to efficiently handle packet loss in Ethernet-based datacenters.

(R2) Compatibility with packet-level LBs. RNIC-SR suffers from significant spurious retransmissions when combined with packet-level LBs. The proposed solution must be inherently compatible with packet-level LBs, which necessitates the accurate distinction of real packet losses from OOO packet arrivals.

(R3) Fast retransmission for lost packets. RNIC-SR often fails to recover certain packet losses via fast retransmission, leading to excessive RTOs. The proposed solution should enable prompt retransmission for any lost packets. This necessitates an explicit loss notification scheme.

(R4) Hardware-oriented design. While software solutions can bypass hardware resource limitations due to the greater resource availability in software, they cannot leverage RNIC offloading capabilities and face inherent performance limitations. Therefore, the proposed solution should adopt a hardware-oriented design, ensuring minimal memory and computational overhead simultaneously.

DCP vs. closely related works. We propose DCP, which satisfies all the above requirements. DCP revisits RDMA reliability, which maintains high goodput under significant packet loss ((R1)), avoids spurious retransmissions when combined with packet-level LBs ((R2)), and ensures fast recovery for any lost packets ((R3)). DCP is hardware-friendly; we implemented a fully functional DCP prototype using FPGA and P4 switch, demonstrating its line-rate efficiency and low memory/computational overhead ((R4)).

Table 2 summarizes the differences between DCP and its closely related works, including RNIC-GBN and RNIC-SR. Among these, MP-RDMA [38] redesigns RNICs to support packet-level multipath

Requirements	R1	R2	R3	R4
RNIC-GBN [7]	×	×	×	✓
RNIC-SR [8, 9, 42, 53]	✓	×	×	✓
MPTCP [47]	✓	✓	×	×
NDP [26]	✓	✓	✓	×
CP [18]	✓	✓	✓	×
MP-RDMA [38]	×	✓	×	✓
DCP	✓	✓	✓	✓

Table 2: Comparison of DCP and closely related works.

transmission. However, MP-RDMA still uses GBN as its loss recovery scheme, which is inefficient in the presence of packet loss and therefore still requires PFC to create a lossless environment. MPTCP [47], NDP [26], and CP [18] are software solutions designed primarily for TCP networks, not for RDMA networks.

Note that DCP focuses primarily on reliability, while specific LB and congestion control (CC) designs are orthogonal to its design goals. Regarding LB, DCP is compatible with any packet-level LB schemes, as it supports order-tolerant packet reception and avoids spurious retransmissions. For CC, although DCP currently integrates DCQCN [56], it is microarchitecturally compatible with any CC scheme [12, 34, 37], as DCP's retransmission and CC modules are designed to operate in a decoupled manner.

4 Design

4.1 DCP Overview

DCP co-designs the switch and RNICs to fully meet the design requirements (§3). It consists of DCP-Switch and DCP-RNIC, with its workflow illustrated in Fig. 3.

Upon receiving a DCP data packet, the switch[5] determines the egress port based on load balancing requirements. It then checks whether the length of the egress *data queue* exceeds a specified threshold. If so, the switch trims the payload from the packet, modifies specific flags in the remaining header, and enqueues this header-only (HO) packet into a separate egress queue, i.e., *control queue*. The switch uses a weighted round-robin (WRR) scheduler to prioritize the control queue over the data queue, ensuring a *lossless control plane (CP)* (§4.2) while preventing starvation of the data plane (DP) (❶). Upon receiving the HO packet, the receiver swaps its source and destination IP and Queue Pair Number (QPN) fields and forwards it back to the sender (❷).

Leveraging the lossless control plane, the sender employs an *HO-based retransmission* (§4.3) triggered by the HO packets (❸). Since the HO packet explicitly carries the PSN of the lost packet, the sender precisely retransmits the lost packets indicated by it. We integrate RNIC micro-architecture innovations to further improve the efficiency of the retransmission phase. Furthermore, based on the ACK packet, the sender implements a coarse-grained timeout mechanism as a fallback to ensure reliability if the lossless control plane assumption is violated (e.g., link/switch crashes).

If the length of the egress *data queue* does not exceed the threshold, the data packet is enqueued in the egress *data queue*. Upon reaching the receiver, the data packet—whether in-order or out-of-order—is directly written to the appropriate location in application

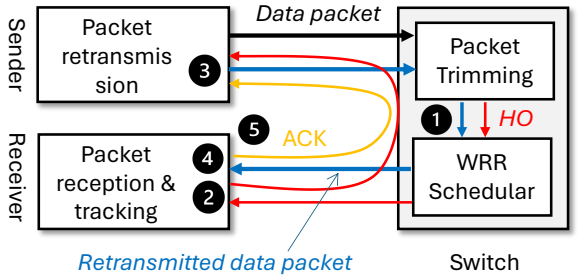


Figure 3: DCP workflow.

memory (❹), based on an extended RDMA header that supports *order-tolerant packet reception* (§4.4). The receiver-side DCP-RNIC maintains the metadata of data packets in a *bitmap-free packet tracking* manner (§4.5) and sends acknowledgments (ACKs) to the sender based on this metadata, if necessary (❺).

4.2 Lossless Control Plane

As shown in Fig. 4, two bits in the Type of Service (ToS) field of the IP header are reserved as the DCP tag to distinguish four types of packets classified by DCP.

- Non-DCP packets (DCP tag = 00): They are dropped by the switch when the buffer exceeds the defined threshold.
- DCP data packets (DCP tag = 10): This category includes both normal and retransmitted data packets. These packets are processed by the Packet Trimming module when the switch buffer exceeds the defined threshold.
- HO packets (DCP tag = 11): A DCP data packet is converted into an HO packet after its payload is trimmed by the Packet Trimming module in the switch.
- DCP ACK packets (DCP tag = 01): These packets contain acknowledgments and are used to expire sender messages.

Packet Trimming Module. Upon receiving a packet, the DCP-Switch first determines its egress port based on specific load-balancing schemes (e.g., AR or ECMP). If the packet is an HO packet, it is enqueued directly into the *control queue*. Otherwise, it is enqueued into the *data queue* when the data queue length is below a given threshold. When the data queue length exceeds the threshold, the switch handles the packet based on its type: if it is a non-DCP or DCP ACK packet, the packet is dropped; if it is a DCP data packet, the switch trims the payload from the packet, modifies the DCP tag in the remaining header to 11, and enqueues the remaining header into the *control queue*. As illustrated in Fig. 4(a), the remaining header in our design is 57 bytes[6].

WRR Scheduling. The DCP-Switch employs WRR scheduling to manage the data and control queues, ensuring a lossless control queue. The weight of WRR (w) depends on the switch radix (N) and the ratio between the HO and data packet sizes ($1 : r$). Assuming the worst-case scenario where there is an $N-1$ to 1 incast burst in a switch and all data packets are trimmed, this will generate $B \times \frac{N-1}{r}$ traffic to the control queue, where B represents the port bandwidth. The draining rate of the control queue is $B \times \frac{w}{1+w}$. To ensure a lossless control queue, the draining rate must be at least equal to the input rate. Therefore, the weight w can be set

[5] In §4, "switch" refers to DCP-Switch, and "receiver" and "sender" refer to the receiver- and sender-side DCP-RNICs, respectively.

[6] 57 bytes = 14 bytes MAC header + 20 bytes IP header + 8 bytes UDP header + 12 bytes BTH header + 3 bytes for the MSN field.

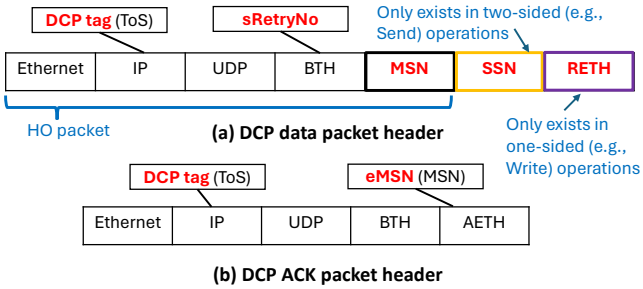


Figure 4: DCP extends the traditional RDMA header with specific fields (indicated with red bold text).

to $\frac{N-1}{r-N+1}$ (theoretical value), meaning that the ratio of scheduled traffic volume between the control and data queues is $\frac{N-1}{r-N+1} : 1$.

Note that this equation is valid only in scenarios where $r > (N-1)$. In some cases where $r < (N-1)$, we cannot theoretically guarantee lossless HO packet transmission by simply configuring the weight. However, we evaluated in §6.3 that a small w can effectively handle extreme incast scales. Note that when an HO packet is accidentally lost, DCP relies on a timeout for loss recovery instead of HO-based retransmission (detailed in §4.5).

4.3 Efficient HO-based Retransmission

We first describe the current packet-sending strategy of the transmit (Tx) path in DCP-RNIC. Then we outline the challenges of implementing an efficient retransmission mechanism based on HO packets. Finally, we present our solution.

DCP-RNIC Tx Path. As shown in Fig. 5, during the Tx path of DCP-RNIC, the Queue Pair (QP) Scheduler first determines which QP will be chosen to send Work Request Elements (WQEs)[7] in the next round. Then, the DMA Engine fetches the data of the selected QP and encapsulates the data into multiple packets. To reduce memory footprint, DCP-RNIC does not cache WQEs for *active QPs* that have unfinished WQEs in the Send Queue (SQ). Instead, it adopts a *fetch-and-drop* strategy for scheduling QPs and WQEs, which is commonly used in RNIC microarchitecture designs [53].

Specifically, the QP Scheduler first selects an active QP with an available window (*awin*), which is determined by the CC module. The DMA Engine then fetches up to n WQEs from its SQ. The RNIC processes the fetched WQEs and fetches up to *min(round_quota, awin)* bytes of data from application memory. If there are unused WQEs left in the RNIC after a scheduling round[8], these unused WQEs are dropped rather than cached in the RNIC and will be fetched again the next time this QP is scheduled. Adjusting the values of n and *round_quota* affects the tradeoff between PCIe bandwidth utilization and scheduling granularity. In our design, we set n to 8 and *round_quota* to 16 KB (≈ the PCIe BDP), to balance performance and scheduling granularity.

Challenges. Unlike transmitting normal data packets in the Tx path, HO-based retransmission faces challenges in efficiency and compatibility. This is because HO packets have a unique feature:

[7]In RNICs, QP and WQE are descriptors for connections and messages. Each QP typically consists of a Send Queue (SQ), a Receive Queue (RQ), and a Completion Queue (CQ) on both sides.

[8]This occurs when the total message size associated with the n WQEs exceeds the *min(round_quota, awin)* bytes.

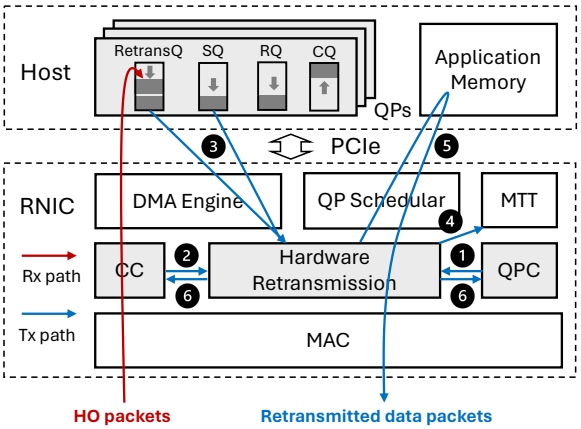


Figure 5: Working steps of HO-based retransmission. The gray modules are that DCP makes modifications.

they are independent and stateless.

- **#1: Inefficient retransmission:** Since HO packets are independent, one way to apply the *fetch-and-drop* strategy during loss recovery phase would be as follows: for each HO, the RNIC first fetches its corresponding WQE, processes the WQE, and fetches the associated data. This approach however results in significantly low throughput because each HO packet requires two PCIe transactions[9].
- **#2: Incompatible with CC module:** Since HO packets are stateless, the retransmission rate is tied to the receiving rate of the HO packets. Consequently, CC cannot regulate the retransmission rate, which may worsen congestion in certain scenarios. For instance, if there exists severe congestion that causes substantial packet loss, the generated HO packets may trigger excessive retransmissions, further aggravating the congestion.

Solution: HO-based Retransmission. We present our HO-based retransmission design, with Fig. 5 illustrating its working steps. As shown, the retransmission process is separated into the receive (Rx) and Tx paths, adopting a batched-fetch strategy to reduce PCIe transactions during retransmissions and incorporating a retransmission queue (RetransQ) in host memory for each QP to enable the CC module to regulate the retransmission rate.

In the Rx path, upon receiving an HO packet, the RNIC extracts metadata from its header and packages it into a *retransmission entry*, which consists of *(MSN, PSN)*. Then the DMA Engine writes this retransmission entry into the corresponding QP's RetransQ located in the host memory. The RetransQ is allocated along with the SQ, RQ, and CQ during QP creation. Once allocated, it is exclusively managed by the RNIC without involving any software manipulation. Therefore, no additional CPU overhead is introduced.

In the Tx path, when a QP is scheduled, the RNIC first checks if its RetransQ is empty by examining the Queue Pair Context (QPC) status (❶). If the RetransQ is not empty, the DMA Engine retrieves the *awin* value from the CC module (❷) and fetches *min(16, len, awin/MTU)*[10] retransmission entries from the RetransQ, where *len* represents the length of the RetransQ, which is maintained in the

[9]Assuming the PCIe round-trip latency between the RNIC and host is 1 μs, the throughput during the loss recovery period is 1KB/2μs = 4Gbps.

[10]16 × 1KB = 16KB, equals the previously configured *round_quata*.

QPC. Simultaneously, the DMA Engine fetches up to n WQEs from the SQ (❸). We configure n to 8, consistent with the previous setting. For each retransmission entry, the RNIC calculates its virtual address based on the fetched WQE information (which contains the starting address) and translates it into a physical address using the Memory Translation Table (MTT) (❹). If the required WQE is not present in the RNIC, it re-fetches the targeted WQE and replaces a randomly selected existing WQE in the RNIC.

Subsequently, the DMA Engine fetches the payload from application memory and encapsulates it into a packet (❺). Once the retransmitted packet is sent, the RNIC updates the length of the RetransQ in the QPC and adjusts the *awin* in the CC module (❻). After processing all fetched retransmission entries, the RNIC begins transmitting normal data packets if there is an available sending quota. Note that throughout the entire process, the retransmission and CC modules operate in a decoupled manner, making DCP microarchitecturally compatible with any CC scheme.

4.4 Order-tolerant Packet Reception

Both packet loss and load balancing can lead to out-of-order (OOO) packet arrivals. The standard RDMA header is designed for in-order packet arrivals and is not well-suited for handling OOO packet reception. A feasible way to handle OOO packets with the standard RDMA header is to allocate a large reorder buffer in the RNIC or host, where OOO packets are temporarily stored. Once all packets are received, the RNIC or host reorders them before pushing them into application memory [30, 39]. However, this approach introduces significant memory and CPU overhead.

To address this issue, we extend the standard RDMA header, allowing the RNIC to write all packets, whether in-order or OOO, directly to the correct locations in application memory. This eliminates the need for a reorder buffer. Here, we focus on the widely-used `Send`, `Write`, and `Write-with-Immediate` operations.

Write. In the standard specifications, the RDMA Extended Transport Header (RETH), which contains the remote memory location, is included only in the first packet of a `Write` message. As a result, during OOO arrivals, if the middle packets arrive first, they cannot determine the remote memory location. To address this, as shown in Fig. 4(a), DCP includes the RETH header in all packets (including first, middle and last) of the `Write` message.

Send and Write-with-Immediate. Two-sided operations require each arriving packet to be matched with a corresponding Receive WQE at the responder. This matching is implicit for in-order packet arrivals but fails in the case of OOO arrivals. For example, if a packet of a latter message arrives before the preceding message completes, it cannot find a matching Receive WQE. Since Receive WQEs must be consumed by `Send` and `Write-with-Immediate` requests in the same order they are posted, we introduce a Send Sequence Number (SSN) for these operations, as shown in Fig. 4(a). The SSN indicates the posting order and is included in all `Send` packets and the last `Write-with-Immediate` packet. It is used to identify the appropriate Receive WQE for processing. Without the SSN, the RNIC would need to buffer packets from out-of-order messages, which incurs significant memory overhead.

Other specific fields for DCP. Besides the above extensions, DCP introduces additional fields to correlate metadata at RNICs, as shown in Fig. 4. The first is the Message Sequence Number (MSN), which specifies the posting order of all requests in the SQ. Furthermore, DCP includes the sRetryNo in data packets and the eMSN in ACK packets. The usage of these fields is detailed in §4.5.

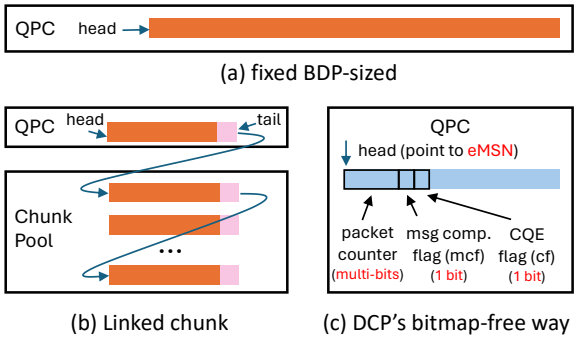


Figure 6: Three approaches of packet tracking.

4.5 Bitmap-free Packet Tracking

DCP-RNIC avoids the need for a reordering buffer via RDMA header extension (§4.4). However, traditionally, the RNIC still needs to maintain bitmaps to track which packets have been received or lost. The bitmap introduces undesired trade-offs between memory overhead and processing efficiency.

One approach for maintaining bitmaps is to pre-allocate fixed-length, typically BDP-sized ($\frac{BDP}{MTU}$ bits), bitmaps for all active QPs [42, 53], as shown in Fig. 6(a). This method exhibits constant packet processing latency, as accessing any slot in the bitmap requires constant steps: 1) calculating the address by adding the bitmap head and PSN offset, and 2) accessing the address. Although promising, this approach leads to significant memory overhead. Table 3 illustrates the memory overhead of bitmaps in typical intra-DC scenarios (400Gbps bandwidth, 10μs RTT). As the number of QPs increases, the footprint of BDP-sized bitmaps can easily exceed the typical RNIC SRAM capacity (usually ~2MB).

Another common approach is to maintain a chunk pool [30, 39], e.g., each chunk with 128 bits. Each QP is pre-allocated with only one chunk and is linked with additional chunks as needed (Fig. 6(b)). This approach reduces bitmap memory overhead when the degree of OOO packets is low (Table 3). However, as the OOO degree increases, not only does the memory overhead eventually reach that of the BDP-sized approach, but this method also introduces another issue: high access latency. Accessing bits in the n_{th} chunk requires $O(n)$ steps: determining whether the PSN is within the current chunk; if not, retrieving the next chunk's address and proceeding to the next chunk. The access latency impacts the packet rate (pps)[11]. Fig. 7 illustrates the theoretical packet rate under various OOO degrees for a clock frequency of 300MHz. As shown, the packet rate of the linked chunk degrades as the OOO degree increases.

Opportunities. DCP's lossless control plane and HO-based retransmission ensure that only truly lost packets are retransmitted. This guarantees that for any given packet, exactly one copy arrives at the receiver. This "exactly-once" property allows us to move away from traditional packet-level tracking strategies. Instead, we

[11]Bandwidth = $pps \times MTU$. 50 Mpps amounts to 400 Gbps with a 1KB MTU.

Schemes	BDP-sized	Linked chunk	DCP
Per-QP, Intra-DC	320B	80B~320B	32B
10k QPs, Intra-DC	3MB	0.76MB~3MB	0.3MB

Table 3: Memory overhead for packet tracking.

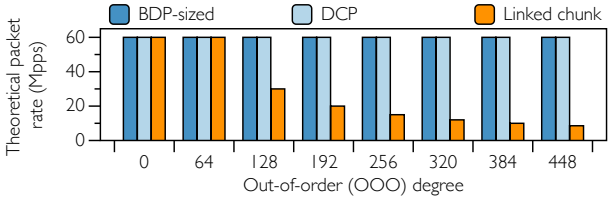


Figure 7: Theoretical packet rate, i.e., packet per second (pps), under various out-of-order degrees.

can simply count the number of packet arrivals for each message and compare this count to the message size to determine message completion. By adopting this approach, we reduce the memory requirement at the receiver from n bits to $\log_2(n)$ bits where n is the number of in-flight packets. Note that the sender-side bitmap is already eliminated in HO-based retransmission.

Solution: Bitmap-free Packet Tracking. The receiver-side DCP-RNIC employs packet counting to track the aggregated message-level information, rather than tracking at the packet level. As shown in Fig. 6(c), it maintains a multi-bit counter for each message. Specifically, upon the arrival of a packet, the corresponding message's counter is incremented by 1. When the packet counter equals the message size, the receiver determines that the message is complete and sets the message completion flag (mcf) to 1. If the message requires a CQ Element (CQE), the CQE flag (Cf) is also set to 1. For each QP, the receiver maintains an expected message sequence number (eMSN) state. In cases where messages are completed out of order, the receiver waits until the eMSN_{th} message finishes, then updates the eMSN and generates one or multiple CQEs for the application. This behavior aligns with the common assumption in upper-level application programming that messages are completed in order [38]. When the eMSN is updated, the receiver generates an ACK that carries the updated eMSN value (as shown in Fig. 4(b)).

The memory overhead of DCP-RNIC is determined by the outstanding message size in upper-layer applications. For example, AI applications typically use NCCL [44] as the communication backend, where the outstanding messages per QP is 8, and the typical message size is several MBs. Therefore, we allow each QP to track 8 messages and set the counter to 14 bits, resulting in a tracking status of 2 bytes per message. Table 3 shows the associated memory overhead. Furthermore, since the processing latency for each packet in DCP is constant, merely increasing the correlated counter by 1, the packet rate remains constant (Fig. 7).

Coarse-grained Timeout as a Fallback. The assumption of a lossless control plane may be violated due to link/switch failures or accidental HO packet losses. In these cases, HO-based retransmission fails to recover the lost packets, so DCP-RNIC falls back to using a coarse-grained timeout mechanism to ensure reliability. Specifically, the sender maintains the smallest unacknowledged MSN (unaMSN) for each QP and keeps a timer associated with it. If the received ACK's eMSN is greater than the unaMSN, the timer is reset and the unaMSN is updated to the received ACK's eMSN.

Schemes	LUT	Registers	BRAM	URAM
RNIC-GBN	66k (5.4%)	102k (3.5%)	408 (20%)	38 (3.9%)
DCP-RNIC	67k (5.5%)	103k (3.6%)	412 (20%)	37 (3.8%)

Table 4: Resource usage of our prototype. The number inside the bracket is the ratio of total FPGA resources.

Otherwise, the timer continues. If the timer expires, the sender retransmits all packets of the unaMSN_{th} message.

However, this approach disrupts the "exactly once" delivery guarantee, which affects the correctness of the receiver's packet counting. To address this, the sender maintains a value called sRetryNo (initialized to 0) for each QP, which tracks the number of timeouts for the unaMSN_{th} message. The data packet includes the sRetryNo value in its header. On the receiver side, the receiver maintains a rRetryNo value for the eMSN_{th} message. When a packet of the eMSN_{th} message is received, the receiver first checks if the sRetryNo field in its header matches rRetryNo. If they are equal, the eMSN_{th} message's packet counter is incremented by 1. If sRetryNo is greater than rRetryNo, the receiver resets the packet counter to 0, updates the rRetryNo to this sRetryNo, and starts recounting for the newest timeout round.

Orthogonality. This bitmap-free design is orthogonal to the rest of DCP-RNIC's architecture. If we choose to maintain traditional packet-level bitmaps at the receiver, the remaining components of DCP-RNIC, such as HO-based retransmission, still function correctly. We believe this bitmap-free approach is an innovative direction that contributes to rethinking next-generation high-speed RNIC design. At the same time, we acknowledge that it faces tremendous challenges in real-world deployment.

5 Implementation

DCP-Switch. We implement the lossless control plane, including packet trimming and WRR scheduling, using P4 switch [2]. When the egress queue length exceeds a given threshold, DCP-Switch generates a mirrored packet and sets its packet length to the length of the header during mirroring, while dropping the original packet; this process utilizes the Mirror function. After the mirrored packet is re-enqueued, we modify its DCP tag and the *packet_len* in the IP header, then push it to the control queue of the egress port. We also implement adaptive routing in the switch, where the ingress pipeline monitors the egress queue length and, based on this information, selects the egress port with the lowest queue length. We enable WRR scheduling in the egress pipeline and assign it an appropriate weight according to §4.2. Discussions with multiple vendors confirm that supporting the lossless control plane in switch ASICs is feasible, as it is simple, stateless, and does not interfere with any existing switch functionalities.

DCP-RNIC. We build a fully functional prototype of DCP-RNIC using an FPGA board [4] with PCIe Gen3 x16 and 100 Gbps Ethernet ports, running at a clock frequency of 300MHz. First, we implement an RNIC-GBN prototype as a baseline. We then implement DCP-RNIC by modifying specific modules of the RNIC-GBN baseline. Table 4 illustrates the resource consumption for the RNIC-GBN and DCP-RNIC prototypes. As shown, DCP-RNIC consumes only 5.5%, 3.6%, 20%, and 3.8% of the FPGA board's total LUT, registers, BRAM, and URAM resources, respectively. Moreover, DCP-

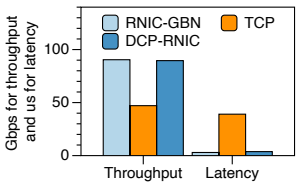


Figure 8: Basic validation of DCP-RNIC prototype.

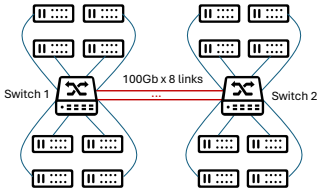


Figure 9: Testbed topology used in this paper.

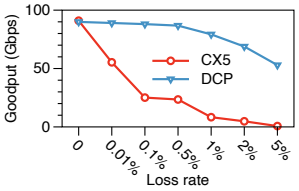


Figure 10: DCP's superior loss recovery efficiency.

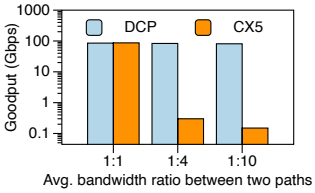


Figure 11: DCP adapts to unequal paths via AR.

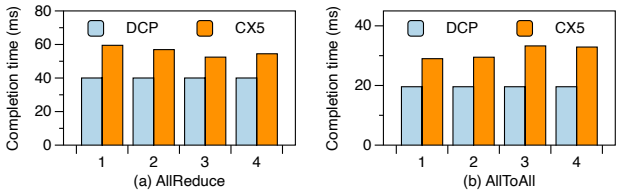


Figure 12: DCP vs. CX5 under AI workloads.

RNIC consumes only 1.7%, 0.4%, and 1.1%, more LUT, registers, and BRAM, respectively, compared to RNIC-GBN.

We evaluate the basic performance of the DCP-RNIC prototype by connecting two DCP-RNICs directly and using the *perftest* benchmark to measure throughput and latency. DCP-RNIC is compared with RNIC-GBN and TCP. For the throughput test, we launch a long-running flow consisting of multiple 512 KB messages, and for the latency test, we launch a 64 B message. The results in Fig. 8 demonstrate that DCP-RNIC successfully maintains hardware offloading capabilities, achieving throughput and latency comparable to RNIC-GBN, both of which outperform TCP significantly.

6 Evaluation

We evaluate DCP through extensive testbed experiments and simulations, which reveal the following key results:

- DCP achieves a 1.6×-72× improvement in loss recovery efficiency and a 42% reduction in completion time for AI workloads, compared to Mellanox CX5. Moreover, DCP maintains consistent throughput, whether with adaptive routing or over a 10 km communication distance (§6.1).
- DCP achieves 16% and 10% lower tail FCT under general workloads, and 38% and 45% lower completion time under AI workloads, compared to MP-RDMA and IRN (i.e., SOTA lossless and lossy solutions), respectively. Additionally, DCP achieves even greater performance improvements in cross-datacenter (cross-DC) scenarios (§6.2).
- DCP+CC outperforms all other comparisons under high-load scenarios, and DCP's lossless control plane remains robust under severe incast congestion (§6.3).

6.1 Testbed Benchmarks

Setup. Fig. 9 illustrates the testbed topology, consisting of two P4 programmable switches ($Switch_1$ and $Switch_2$), each connected to 8 FPGAs. All links operate at 100 Gbps. The two switches are connected via 8 parallel cross-switch links. We compare DCP-RNIC with Mellanox CX5 RNIC (i.e., RNIC-GBN).

Loss recovery efficiency. We select two RNICs as sender and receiver, respectively, where the sender transmits a long-running flow to the receiver. We manipulate the P4 programmable switch to drop packets with a given loss rate. Upon packet loss, the P4 switch executes the packet trimming module for DCP traffic, while it simply drops packets for CX5 traffic. We vary the loss rate from 0.01% to 5% and measure the goodput for both DCP and CX5 traffic. Fig. 10 shows the results. As demonstrated, DCP achieves a 1.6× to 72× improvement in loss recovery efficiency under loss rates ranging from 0.01% to 5%, compared to CX5.

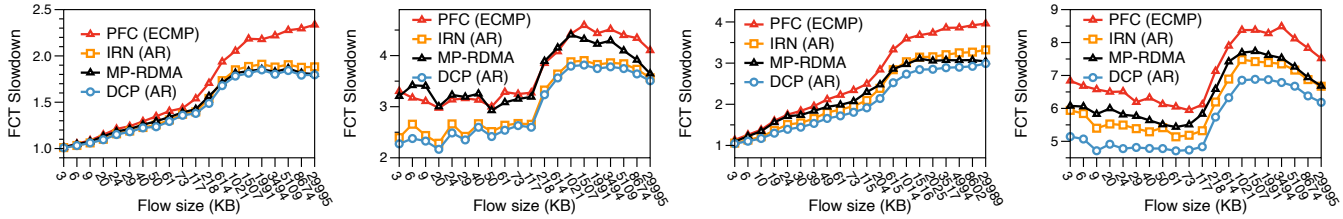
Compatibility with AR. We select two RNICs from $Switch_1$ as senders and two RNICs from $Switch_2$ as receivers. The senders transmit two long-running cross-switch flows. We enable two cross-switch paths and modify their port capacities, setting their capacity ratios to 1:1, 1:4, and 1:10. We implement adaptive routing (AR) on the switches, which forwards traffic according to the capacity ratio of the links. We measure the average goodput of the two flows. Fig. 11 illustrates the results. As shown, DCP maintains stable goodput under all capacity ratios, as it is natively compatible with packet-level LBs. In contrast, the goodput of CX5 significantly decreases under non-equal port capacities.

AI workload. We implement an AllReduce and AllToAll benchmark using the *verbs* API [6] and OpenMPI [10]. The 16 RNICs in the testbed are arranged into four groups, each consisting of 4 RNICs. Each group executes an AllReduce/AllToAll operation, with all groups starting execution simultaneously. DCP and CX5 are integrated with AR and ECMP in the network, respectively. We measure the job completion time (JCT) for each group and present the results of 4 groups in Fig. 12. As shown, DCP reduces the JCT of AllReduce and AllToAll by up to 33% and 42%, respectively.

Long-haul communication. We replace one cross-switch link with a 10 km optical link (one-hop delay is 50 µs) and use 100G-LR transceivers at the endpoints. We select one RNIC from $Switch_1$ as the sender and one RNIC from $Switch_2$ as the receiver. The sender transmits a long-running flow to the receiver. We measure the throughput of this flow and observe that DCP can stably operate at around 85 Gbps. This experiment serves as a first-step validation that DCP can adapt to long-haul communication scenarios.

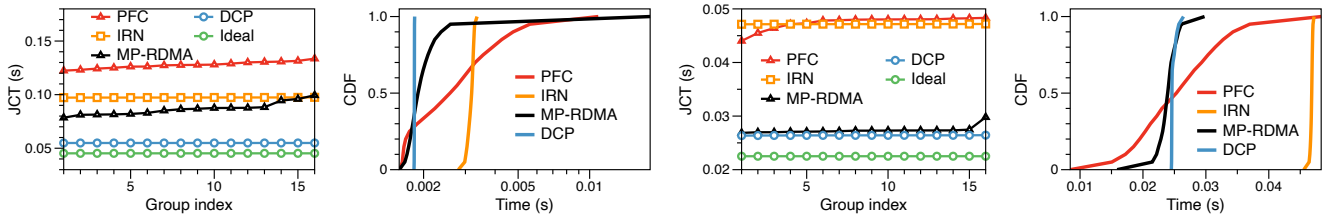
6.2 Large-scale Simulations

Setup. We use NS3 for simulations. The topology consists of a two-layer CLOS network with 16 spine switches, 16 leaf switches, and 256 servers (16 per rack). Each server has a single NIC connected to a single leaf switch. All links operate at 100 Gbps. In all experiments, except for cross-datacenter (DC) scenarios, the propagation delay of all links is set to 1 µs. In cross-DC experiments, the propagation delay of links between servers and leaf switches is 1 µs, while the propagation delay between leaf and spine switches is set to 500 µs and 5 ms. The switch buffer size is 32 MB, and the entire



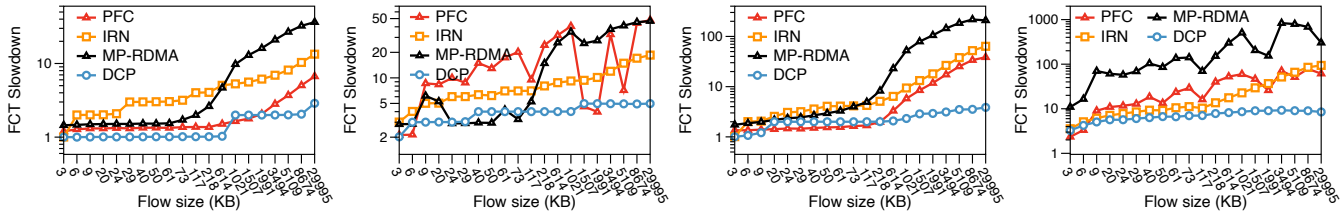
(a) Load 0.3, P50. (b) Load 0.3, P95. (c) Load 0.5, P50. (d) Load 0.5, P95.

Figure 13: Comparision between DCP, PFC, IRN, and MP-RDMA under the WebSearch workload.



(a) AllReduce, JCT. (b) AllReduce, CDF of FCT. (c) AllToAll, JCT. (d) AllToAll, CDF of FCT.

Figure 14: Comparision between DCP, PFC, IRN, and MP-RDMA under the AllReduce and AllToAll workloads.



(a) 500us delay, P50. (b) 500us delay, P95. (c) 5ms delay, P50. (d) 5ms delay, P95.

Figure 15: Comparision between DCP, PFC, IRN, and MP-RDMA under the cross-DC (100 and 1000 km) scenarios.

network is a single RDMA domain.

We compare DCP with PFC, IRN [42], and MP-RDMA [38]. MP-RDMA includes its own CC component, i.e., an adaptive congestion window, while IRN only employs a BDP-based flow control. By default, DCP, PFC, and IRN are combined with AR, ECMP, and AR as the load balancing schemes in the network. We select IRN+AR as the default configuration because we observe that IRN+AR outperforms IRN+ECMP in most of our experiments. We also compare DCP, PFC, and IRN with CC integrated (we choose DCQCN [56] as it is representative). The traffic loads used in the simulations include general workloads and AI workloads.

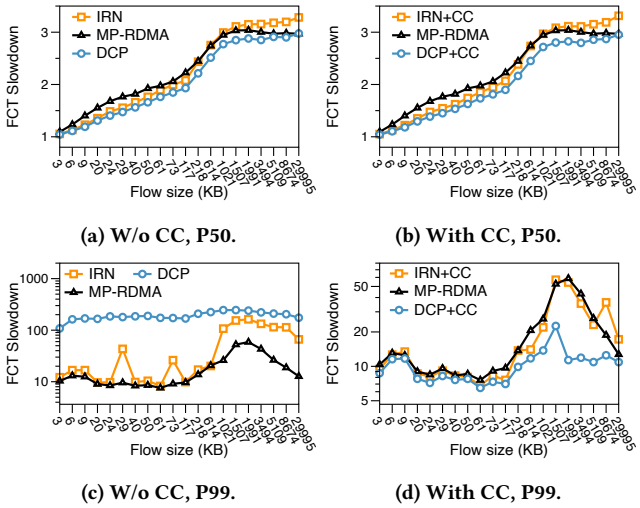
General workload. We evaluate the WebSearch [15] workload, which consists of 60% of flows below 200 KB, 37% of flows between 200 KB and 10 MB, and 3% of flows exceeding 10 MB. Fig. 13 illustrates the results under the WebSearch workload with average loads of 0.3 and 0.5. At each load, we measure the P50 and P95 (tail) flow completion time (FCT).

As shown, fine-grained LB solutions, such as MP-RDMA and AR, consistently outperform ECMP, as they can better balance traffic. Among the fine-grained LB solutions, DCP achieves the best performance, with an average of 5% and 16% lower tail FCT at 0.3 load, and 10% and 12% lower tail FCT at 0.5 load, compared to IRN and MP-RDMA, respectively. The performance advantage of DCP over IRN is due to IRN's susceptibility to spurious retransmissions when combined with AR. Additionally, we observe that MP-RDMA fails to effectively control the out-of-order degree below its expected threshold, which leads to its inferior performance.

AI workload. We arrange 256 servers into 16 groups, with 16 servers per group. Each group executes an AllReduce or AllToAll operation, starting execution at the same time. The total traffic for one AllReduce/AllToAll operation is 300 MB. For AllReduce, the total traffic is partitioned into 16 slices (i.e., flows), and then transmitted following a RingAllReduce procedure. For AllToAll, the total traffic is partitioned into 16 slices, with one slice transmitted to each group member. We measure the time of the last completed flow within each group as the Job Completion Time (JCT) for that group, and we also measure the CDF of individual flows' FCT.

The results for AllReduce and AllToAll are shown in Fig. 14. As shown, DCP achieves average 38%, 44% and 61% lower JCT under the AllReduce workload (Fig. 14a), and average 5%, 45% and 46% lower JCT under the AllToAll workload (Fig. 14c), compared to MP-RDMA, IRN and PFC, respectively. As shown in Fig. 14b and Fig. 14d, DCP achieves the best tail FCT for individual flows, which explains why DCP achieves the best JCT. This is because AI workloads are synchronized, meaning that if just one flow is delayed, it impacts the entire collective communication performance.

Cross-DC scenarios. Fig. 15a and Fig. 15b illustrate the P50 and P95 (tail) FCT slowdown under the 100 km cross-DC scenario (i.e.,



(a) W/o CC, P50. (b) With CC, P50.

(c) W/o CC, P99. (d) With CC, P99.

Figure 16: FCT slowdown of DCP, MP-RDMA and IRN under the incast workload w/ and w/o CC.

Settings	N=22; 128 to 1	N=22; 255 to 1	N=16; 128 to 1	N=16; 255 to 1
Loss rate w/o CC	0	0	0	0.16%
Loss rate w/ CC	0	0	0	0

Table 5: Loss rate of HO packets under severe incast degree.

the propagation delay between leaf and spine switches is 500 us), while Fig. 15c and Fig. 15d show the evaluation results under the 1000 km cross-DC scenario (i.e., 5 ms propagation delay). The workload consists of WebSearch traffic with an average load of 0.5. For PFC and MP-RDMA, we increase the switch buffer from 32 MB to 600 MB and 6 GB for the 100 km and 1000 km distances, respectively, ensuring the buffer is larger than the PFC headroom. In contrast, for IRN and DCP, the switch buffer remains at 32 MB.

As shown, DCP achieves approximately 89%, 81%, and 46% lower tail FCT under the 100 km distance, and 84%, 95%, and 51% lower tail FCT under the 100 km distance, compared to PFC, MP-RDMA, and IRN, respectively. Compared to intra-DC scenarios (Fig. 13), DCP achieves a larger improvement in cross-DC scenarios. This is because servers generate more traffic in cross-DC scenarios due to the larger BDP capacity. As a result, congestion is more severe in cross-DC scenarios than in intra-DC scenarios, making the performance improvement of DCP more pronounced.

6.3 Deep Dive

DCP+CC achieves the best performance under high loads. We observe that in highly congested situations, such as extreme incast, DCP alone (i.e., without CC) struggles with tail FCT performance. For example, we evaluate a workload comprising WebSearch traffic at 50% average load and 128-to-1 incast traffic at 5% average load. Fig. 16a and Fig. 16b illustrates the P50 FCT slowdown of DCP, IRN, and MP-RDMA, with and without CC (specifically DCQCN) integration. As shown, DCP exhibits the lowest P50 FCT in both cases. However, Fig. 16c highlights the P99 FCT slowdown when DCP and IRN are evaluated without any CC integration, indicating that DCP alone exhibits the worst P99 FCT.

DCP's poor tail FCT performance occurs because large-scale incast induces severe congestion, leading to significant packet loss. This triggers numerous HO packets arriving at the sender, which in turn triggers a large number of retransmitted packets that further exacerbate congestion and ultimately degrade overall performance. In contrast, MP-RDMA includes a native adaptive congestion window, which reduces the traffic load when severe congestion occurs. Similarly, in IRN, each packet loss prompts only one retransmission from the sender. If the retransmitted packet is dropped again, the IRN sender will not retransmit it but wait for a timeout. During this timeout period, no additional traffic is generated, which helps reduce the average traffic load.

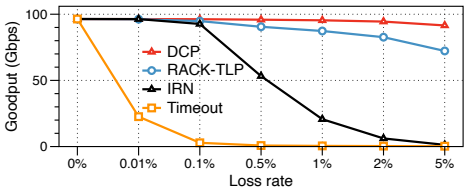


Figure 17: Loss recovery efficiency of DCP, RACK-TLP, IRN, and Timeout-based scheme under various loss rates.

As noted, DCP focuses solely on the reliability aspect, leaving rate control to the CC modules. Therefore, we integrate DCQCN into DCP and IRN and evaluate their performance. Fig. 16d shows the P99 FCT slowdown after CC integration. As demonstrated, DCP achieves the best P99 FCT after CC integration, with a reduction of about 31% and 29% compared to MP-RDMA and IRN, respectively.

Robustness of the lossless control plane. DCP controls the maximum affordable incast scale by adjusting the scheduling weight ($\frac{N-1}{r-N+1}$: 1), as described in §4.2. $N-1$ represents the maximum incast scale that can be handled at local switches. Ideally, N should equal the switch radix, allowing DCP to handle any incast scale. However, if the ratio r is small, N cannot be set to the switch radix. We evaluate extreme incast scales of 128-to-1 and 255-to-1 with N of 22 and 16 and measure the ratio of lost HO packets over all HO packets. The background traffic is WebSearch with an average load of 0.3. We evaluate DCP both with and without CC enabled. Table 5 shows the results. As shown, when CC is disabled, no HO packets are lost with $N = 22$ at any incast scale, and only 0.16% of HO packets are lost under the extreme 255-to-1 incast scale with $N = 16$. When CC is enabled, there are no HO packet losses in any case. This demonstrates that DCP's lossless control plane maintains robustness even under severe incast scales.

Comparison with Timeout and RACK-TLP. The NVIDIA Spectrum platform [13] supports adaptive routing (AR), where the Spectrum Switch dynamically changes packet paths, and the SuperNIC relies on timeouts for loss recovery to avoid spurious retransmissions. However, Spectrum AR only supports RDMA Write operations, and using timeouts results in significant performance degradation upon packet loss. Falcon [5] introduces RACK-TLP [19] for TCP networks to address packet reordering and retransmission/tail loss issues. RACK-TLP maintains transmission timestamps for every data packet (including retransmissions). If a packet remains unacknowledged for an estimated RTT after being sent, it is considered lost. This mechanism tolerates a reordering window of one RTT and helps avoid timeouts for retransmitted packet losses. However, it delays retransmissions by one RTT and incurs significant

memory overhead due to the need to maintain per-packet timestamps, making it impractical for hardware offloading.

We conducted a simulation to evaluate the loss recovery efficiency of DCP, IRN, RACK-TLP, and timeout-based mechanisms. We measured the goodput of a long-running flow under ECMP with various packet loss rates artificially enforced at switches. As shown in Fig. 17, DCP achieves up to 22%, 98%, and 99% higher goodput than RACK-TLP, IRN, and timeout, respectively. As packet loss increases, the performance of the timeout-based scheme degrades sharply. IRN suffers due to increased timeouts caused by the loss of retransmitted packets. RACK-TLP performs better than IRN by avoiding such timeouts, but this comes at the cost of high memory overhead from maintaining timestamps. If timestamps are removed, RACK-TLP would likely perform worse than IRN due to delayed retransmissions. Note that our simulations do not emulate the end-host overhead of retransmitting packets, so all schemes appear more efficient than they would in testbed implementations. However, the relative performance comparison remains consistent.

7 Discussion

Differences with NDP and CP. While the DCP-Switch shares similarities with the switch-side logic of both NDP [26] and CP [18], the end-host design in DCP (i.e., DCP-RNIC) differs from both of them. Specifically, NDP primarily focuses on leveraging packet trimming to realize a receiver-driven CC mechanism. Its end-host implementation is based on DPDK in software, while its Tofino and FPGA implementations are limited to switch-side logic only. On the other hand, CP places more emphasis on reliability for TCP. It introduces PACK packets and relies on sender- and receiver-side bitmaps to track losses and determine retransmissions. In contrast, DCP-RNIC is specifically tailored for RDMA networks. Our DCP-RNIC design is grounded in an in-depth analysis of existing RNIC architectures, with design decisions made incrementally based on the needs of RDMA. The key difference in DCP-RNIC, such as HO-based retransmission, an extended packet header, and bitmap-free packet tracking, are not present in either NDP or CP.

Distinction of SR- and HO-based retransmission. HO-based retransmission cannot be handled in the same way as SR-based retransmission. SR operates in a stateful manner: it uses a sender-side bitmap to track lost packets, where gaps in the bitmap indicate packet loss. Based on this information, SR can retrieve and retransmit specific payloads at any later time. In contrast, DCP adopts a stateless design without bitmaps, and loss events are indicated through individual HO packets. As a result, DCP must maintain a queue of these loss events to trigger further retransmissions. Since the number of HO packets can be large, this queue must be implemented in software rather than hardware.

Congestion Control for DCP. Theoretically, DCP is compatible with any reactive [12, 37, 41, 56] and proactive [17, 20, 43, 45] CC design. Currently, we adopt DCQCN at RNICs, which reduces the sending rate upon receiving CNPs. The received CNPs of a flow may result from multiple paths, but the sender does not currently distinguish between them and simply reacts in the standard DCQCN manner. The question of what CC strategy should be used when combined with in-network packet-level LBs is beyond the scope of this paper. We plan to explore this in future work.

Onloading bitmaps to host memory? SRNIC [53] focuses on single path transmission, so all OOO packets (i.e., those that trigger bitmap access) are assumed to be caused by packet loss. Consequently, the access frequency to the bitmap is low, making it acceptable for SRNIC to place the bitmap in host memory. In contrast, DCP employs packet-level LB, which can cause OOO packets even in the absence of packet loss. As a result, the frequency of bitmap access is much higher than what SRNIC assumes. Therefore, placing the bitmap in host memory is not feasible in our case.

Back-to-sender for HO packets. Currently, header-only (HO) packets must be sent to the destination before being sent back to the sender. The reason is as follows. The RDMA RC mode relies on QPs, which consist of a sender QP and a receiver QP. The packet's header includes only the destination QP number (QPN) and does not carry the source QPN. Moreover, a packet is accepted by the RNIC only if its header contains the correct destination QPN. Therefore, when a switch generates an HO packet and attempts to send it back to the sender directly, it must set the destination QPN to that of the sender QP. However, the switch does not know the sender's QPN; only the receiver's QP context contains this information. As a result, the HO packet must first be forwarded to the receiver to obtain the correct destination QPN before it can be sent back to the sender. In theory, if the switch maintained a mapping table between sender and receiver QPNs, it could extract the data packet's destination QPN, find the corresponding sender QPN, and modify the HO packet accordingly, allowing it to be returned directly to the sender. However, maintaining such a table would introduce significant state overhead and computational complexity.

8 Related Work

Load balancing. Various congestion-aware rerouting solutions [31, 46] and flowlet-based LBs [14, 32, 51] require a sufficiently large packet interval within flows to trigger path changes. However, such intervals are rare in RDMA traffic [38]. Moreover, finer-grained load balancing typically yields better performance in terms of path utilization and latency. DCP is designed to support per-packet load balancing to fully exploit this potential. ConWeave [50] reorders packets in the network to deliver an in-order packet sequence to RNICs, enabling packet-level load balancing as well. However, ConWeave restricts a flow to at most two paths, limiting flexibility, and imposes non-trivial queuing overhead on switches. In contrast, DCP's packet trimming module is lightweight and stateless. In summary, DCP is orthogonal to any specific LB approach; it focuses on reliability and is natively compatible with all LB schemes.

Multipath in lossless fabrics. To be compatible with Spectrum AR [13], the NVIDIA SuperNIC supports OOO reception for RDMA Write operations. It achieves this by converting all Write packets (e.g., Write First, Write Middle, Write Last) into Write-Only packets, each of which contains a destination memory address and can thus be directly written to application memory. However, this mechanism applies only to RDMA Write and does not generalize to other RDMA operations. In contrast, our proposed OOO-tolerant reception aims to provide a unified solution that supports both one-sided and two-sided RDMA operations. Similarly, the recently proposed LEFT [30] also targets compatibility with packet-level load balancing by enabling the correct delivery of OOO payloads to

application memory. However, both NVIDIA Spectrum and LEFT work only in lossless fabrics, as they focus solely on data placement and rely solely on timeouts for loss recovery, which leads to significant performance degradation upon packet loss (§6.3).

Industrial lossy solutions. RACK-TLP [19] trades loss recovery efficiency for packet reordering tolerance. Specifically, for each ACK received, the sender calculates the latest RTT measurement and checks whether there are any packets that are unacknowledged for an estimated RTT (i.e., "reordering window"). If this condition is met, RACK marks the packet as lost and retransmits it. This approach can avoid most spurious retransmissions, but the retransmission is delayed by a full RTT, resulting in inefficient loss recovery. SRD [49] provides reliable datagram semantics but requires applications to handle reordering themselves. Solar [40] implements an ordering-resilient network stack, using a one-packet-one-block approach, in FPGAs for specific storage applications. Solar maintains four paths in its control plane and relies on OOO packet arrival to detect packet loss within each path. In contrast, DCP achieves efficient loss recovery and reordering tolerance simultaneously, delivers reliable connection semantics without placing any additional burden on the application, and natively adapts to packet-level LBs.

UEC [11] mentions the packet trimming technique as well but does not discuss any RNIC solutions based on packet trimming. To the best of our knowledge, DCP is the first work to propose a comprehensive RNIC design leveraging the packet trimming technique. Note that if UEC is eventually supported by switch vendors, DCP can directly leverage the UEC-defined trimming functionality and eliminate the switch-side implementation overhead.

9 Conclusion

We present DCP, a transport architecture that rethinks RDMA reliability for lossy networks. By integrating a lightweight lossless control plane in switches with a hardware-efficient RNIC design, DCP eliminates dependence on PFC, supports packet-level load balancing, and avoids RTOs. Our prototype and evaluation show that DCP significantly outperforms existing RDMA solutions, advancing the practicality of high-performance RDMA over lossy fabrics. *This work does not raise any ethical issues.*

Acknowledgments

We thank the anonymous SIGCOMM reviewers and our shepherd, Prof. Gianni Antichi, for their constructive suggestions. We also thank Zeke Wang and Xuzheng Chen for their valuable discussions during this project. This work is supported in part by the Hong Kong RGC TRS T41-603/20R, the GRF 16213621, the ITC ACCESS, the TACC [54], and the NSFC 62402407. Bingyang Liu and Kai Chen are the corresponding authors.

References

[1] 2020. 802.1Qbb – Priority-based Flow Control. https://1.ieee802.org/dcb/802-1qbb/.
[2] 2023. EdgeCore AS9516. https://www.edge-core.com/_upload/images/2023-061-DCS810_AS9516-32D_DS_R07_20230503.pdf.
[3] 2023. NVIDIA InfiniBand Adaptive Routing Technology - Accelerating HPC and AI Applications. https://resources.nvidia.com/en-us-cloud-native-supercomputing-dpus-campaign/infiniband-white-paper-adaptive-routing.
[4] 2024. AMD Alveo™ U250 Data Center Accelerator Card. https://www.amd.com/en/products/accelerators/alveo/u250/a-u250-a64g-pq-g.html.
[5] 2024. Google Falcon. https://cloud.google.com/blog/topics/systems/introducing-falcon-a-reliable-low-latency-hardware-transport.
[6] 2024. Libibverbs. https://github.com/linux-rdma/rdma-core/blob/master/Documentation/libibverbs.md.
[7] 2024. NVIDIA ConnectX-5. https://www.nvidia.com/en-sg/networking/ethernet/connectx-5/.
[8] 2024. NVIDIA ConnectX-6 Dx. https://resources.nvidia.com/en-us-accelerated-networking-resource-library/networking-overal-dp.
[9] 2024. NVIDIA ConnectX-7. https://resources.nvidia.com/en-us-accelerated-networking-resource-library/connectx-7-datasheet.
[10] 2024. OpenMPI. https://www.open-mpi.org/.
[11] 2024. Ultra Ethernet Consortium. https://ultraethernet.org/wp-content/uploads/sites/20/2023/10/23.07.12-UEC-1.0-Overview-FINAL-WITH-LOGO.pdf.
[12] 2024. Zero Touch RoCE. https://docs.nvidia.com/networking/display/winof2v237/ethernet+network.
[13] 2025. NVIDIA Spectrum Platform. https://www.nvidia.com/en-us/networking/spectrumx/.
[14] 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.
[15] 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*. 63–74.
[16] 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.
[17] Qizhe Cai, Mina Tahmasbi Arashloo, and Rachit Agarwal. 2022. dcPIM: Near-optimal proactive datacenter transport. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 53–65.
[18] 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.
[19] Yuchung Cheng, Neal Cardwell, Nandita Dukkipati, and Priyaranjan Jha. 2021. The RACK-TLP Loss Detection Algorithm for TCP. RFC 8985. doi: 10.17487/RFC8985
[20] Inho Cho, Keon Jang, and Dongsu Han. 2017. Credit-scheduled delay-bounded congestion control for datacenters. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. 239–252.
[21] 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, 2130–2138.
[22] Adithya Gangidi, Rui Miao, Shengbao Zheng, Sai Jayesh Bondu, Guilherme Goes, Hany Morsy, Rohit Puri, Mohammad Riftadi, Ashmitha Jeevaraj Shetty, Jingyi Yang, et al. 2024. Rdma over ethernet for distributed training at meta scale. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 57–70.
[23] Yixiao Gao, Qiang Li, Lingbo Tang, Yongqing Xi, Pengcheng Zhang, Wenwen Peng, Bo Li, Yaohui Wu, Shaozong Liu, Lei Yan, et al. 2021. When cloud storage meets {RDMA}. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. 519–533.
[24] Prateesh Goyal, Preey Shah, Naveen Kr Sharma, Mohammad Alizadeh, and Thomas E Anderson. 2019. Backpressure flow control. In *Proceedings of the 2019 Workshop on Buffer Sizing*. 1–3.
[25] 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.
[26] Mark Handley, Costin Raiciu, Alexandru Agache, Andrei Voinescu, Andrew W Moore, Gianni Antichi, and Marcin Wójcik. 2017. Re-architecting datacenter networks and stacks for low latency and high performance. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. 29–42.
[27] Jinbin Hu, Wenxue Li, Xiangzhou Liu, Junfeng Wang, Bowen Liu, Ping Yin, Jianxin Wang, Jiawei Huang, and Kai Chen. 2025. {FLB}: Fine-grained Load Balancing for Lossless Datacenter Networks. In *2025 USENIX Annual Technical Conference (USENIX ATC 25)*. 365–380.
[28] Shuihai Hu, Wei Bai, Gaoxiong Zeng, Zilong Wang, Baochen Qiao, Kai Chen, Kun Tan, and Yi Wang. 2020. Aeolus: A building block for proactive transport in datacenters. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 422–434.
[29] Shuihai Hu, Yibo Zhu, Peng Cheng, Chuanxiong Guo, Kun Tan, Jitendra Padhye, and Kai Chen. 2017. Tagger: Practical PFC deadlock prevention in data center networks. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*. 451–463.

[30] Peihao Huang, Xin Zhang, Zhigang Chen, Can Liu, and Guo Chen. 2024. LEFT: LightwEight and FasT packet Reordering for RDMA. In *Proceedings of the 8th Asia-Pacific Workshop on Networking*. 67–73.
[31] 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*. 149–160.
[32] 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*. 1–12.
[33] Jongyul Kim, Insu Jang, Waleed Reda, Jaeseong Im, Marco Canini, Dejan Kostić, Youngjin Kwon, Simon Peter, and Emmett Witchel. 2021. Linefs: Efficient smartnic offload of a distributed file system with pipeline parallelism. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*. 756–771.
[34] 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).
[35] 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*. 1–8.
[36] Wenxue Li, Chaoliang Zeng, Jinbin Hu, and Kai Chen. 2023. Towards fine-grained and practical flow control for datacenter networks. In *2023 IEEE 31st International Conference on Network Protocols (ICNP)*. IEEE, 1–11.
[37] 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.
[38] 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.
[39] Yuanwei Lu, Guo Chen, Zhenyuan Ruan, Wencong Xiao, Bojie Li, Jiansong Zhang, Yongqiang Xiong, Peng Cheng, and Enhong Chen. 2017. Memory efficient loss recovery for hardware-based transport in datacenter. In *Proceedings of the First Asia-Pacific Workshop on Networking*. 22–28.
[40] Rui Miao, Lingjun Zhu, Shu Ma, Kun Qian, Shujun Zhuang, Bo Li, Shuguang Cheng, Jiaqi Gao, Yan Zhuang, Pengcheng Zhang, et al. 2022. From luna to solar: the evolutions of the compute-to-storage networks in alibaba cloud. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 753–766.
[41] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. 2015. TIMELY: RTT-based congestion control for the datacenter. *ACM SIGCOMM Computer Communication Review* 45, 4 (2015), 537–550.
[42] 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.
[43] 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*. 221–235.
[44] NCCL. 2024. https://github.com/NVIDIA/nccl.
[45] 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*. 307–318.
[46] 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.
[47] Costin Raiciu, Sebastien Barre, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley. 2011. Improving datacenter performance and robustness with multipath TCP. *ACM SIGCOMM Computer Communication Review* 41, 4 (2011), 266–277.
[48] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C Snoeren. 2015. Inside the social network's (datacenter) network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. 123–137.
[49] Leah Shalev, Hani Ayoub, Nafea Bshara, and Erez Sabbag. 2020. A cloud-optimized transport protocol for elastic and scalable hpc. *IEEE micro* 40, 6 (2020), 67–73.
[50] Cha Hwan Song, Xin Zhe Khooi, Raj Joshi, Inho Choi, Jialin Li, and Mun Choon Chan. 2023. Network load balancing with in-network reordering support for rdma. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 816–831.
[51] 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.
[52] Zirui Wan, Jiao Zhang, Mingxuan Yu, Junwei Liu, Jun Yao, Xinghua Zhao, and Tao Huang. 2024. Bicc: Bilateral congestion control in cross-datacenter rdma networks. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 1381–1390.
[53] Zilong Wang, Layong Luo, Qingsong Ning, Chaoliang Zeng, Wenxue Li, Xinchen Wan, Peng Xie, Tao Feng, Ke Cheng, Xiongfei Geng, et al. 2023. {SRNIC}: A Scalable Architecture for {RDMA}{NICs}. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 1–14.
[54] 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.
[55] 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*. 253–266.
[56] 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.

A Additional Information

Appendices are supporting material that has not been peer-reviewed.

A.1 More Explanations for Results in §6

Performance fluctuation of IRN w/ AR. In general, the performance disadvantage of IRN w/ AR is less pronounced under low-load or non-congested scenarios, where the degree of packet reordering is minimal. However, under high-load or congested conditions, IRN w/ AR suffers more significantly. This trend is consistently observed across all our experiments. Specifically, in Fig. 13a and Fig. 13b, the workload load is 0.3, a relatively light load. As a result, the P50 and P95 latencies of IRN w/ AR are comparable to DCP, showing no obvious disadvantage. In contrast, Fig. 13c and Fig. 13d have a higher load of 0.5. The P50 latency typically reflects performance under non-congested conditions, so IRN w/ AR and DCP still show similar P50 latency. By contrast, the P95 latency better captures performance under congestion, where IRN w/ AR shows a noticeable performance gap compared to DCP.

For AI workloads, they inherently consist of coflows, and the reported JCT reflects the tail latency among these coflows. Furthermore, AI workloads are often highly synchronized, which leads to bursts of traffic and transient high load. Under these conditions, the limitation of IRN w/ AR in handling reordering and congestion are more pronounced, resulting in significantly worse JCT compared to DCP under the AI workloads (as shown in Fig. 14).

PFC's oscillating behavior in cross-DC experiments. In cross-DC/long-distance scenarios (as shown in Fig. 15), all lossless schemes exhibit obvious performance variability across flows. For example, both PFC and MP-RDMA show oscillating behavior in Fig. 15b and Fig. 15d, although the effect is less pronounced for MP-RDMA. This variability is due to that we increase the switch buffer sizes for lossless schemes from 32 MB to 600 MB and 6 GB for 100 km and 1000 km distances, respectively. The large buffers can cause some large flows to perform exceptionally well when PAUSE is not triggered while others perform poorly.