LR2: Accelerating Long-Distance RDMA Recovery via In-Network Retransmission Decoupling

Minfei Long*, Jiangping Han*†, Kaiping Xue*†, Jinhao Liu*, Jian Li*
*School of Cyber Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China
†Corresponding author: J. Han, K. Xue (jphan, kpxue)@ustc.edu.cn

IEEE INFOCOM 2026 - IEEE Conference on Computer Communications | 979-8-3315-4961-9/26/$31.00 ©2026 IEEE | DOI: 10.1109/INFOCOM59046.2026.11571220

***Abstract*—Extending RDMA across datacenters is hindered by inefficient loss recovery. High round-trip time (RTT) and large bandwidth-delay product (BDP) cripple the default Go-back-N (GBN) retransmission, while hardware constraints on RDMA NICs (RNICs) limit the feasibility of practical deployment of selective retransmission (SR). In this paper, we present LR2, an in-network collaborative retransmission mechanism that overcomes RNIC limitations to enable reliable long-distance RDMA. At its core, LR2 decouples retransmission responsibilities, assigning fast-loop GBN at the RNIC and reconstructing long-haul SR at datacenter interconnection (DCI) switches. In addition, DCI switches independently mask intra-DC losses through near-sender fast feedback and near-receiver local retransmission. To realize this design, LR2 deploys lightweight programmable modules at both ends of the long-haul link: (1) a near-receiver `Depot` module that maintains buffer-efficient reordering and backup pools, and (2) a near-sender `Sentry` module that generates fast loss feedback and filters out redundant retransmission. We implement LR2 on Tofino switches and evaluate it through testbed experiments and large-scale simulations. LR2 significantly reduces flow completion time (FCT) and improves bandwidth efficiency over baselines, without large system overhead.**

***Index Terms*—RDMA, Datacenter Interconnection, Loss Recovery, Programmable Switches**

I. Introduction

Remote Direct Memory Access (RDMA) is widely adopted in modern datacenters (DCs) for its high throughput and low latency, enabled by zero-copy and kernel-bypass via network interface card (NIC) offloading. As cloud applications increasingly span multiple geographic regions, a growing trend is to extend RDMA across DCs to support highly efficient communication at a global scale [1]–[4].

However, extending RDMA across DCs poses fundamental challenges for loss recovery. Since commercial RDMA NICs (RNICs) are typically equipped with limited on-chip SRAM and optimized for line-rate processing using fixed-function pipelines, they leave little flexibility to support complex recovery logic. RDMA instead relies on lossless fabrics and simple Go-back-N (GBN) retransmission at the NIC level, supported by Priority Flow Control (PFC) to prevent in-network loss [5], [6]. This architecture is highly vulnerable to packet loss [7]–[9], especially over long-haul links across DCs, which span tens to hundreds of kilometers and experience inherent fiber impairments, making packet loss both frequent and unavoidable. Consequently, robust loss recovery becomes essential to sustaining RDMA performance in inter-DC environments.

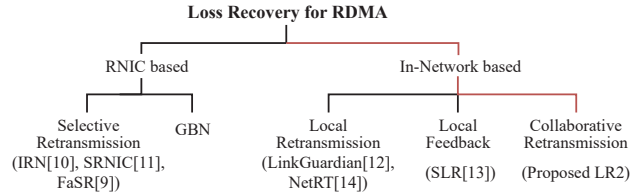


Fig. 1: Design classification of loss recovery for RDMA.

RDMA loss recovery can be enhanced via RNIC-based selective retransmission (SR) [9]–[11] or via in-network retransmission offloaded [12]–[14], as illustrated in Fig. 1. However, existing designs target intra-DC settings and struggle in inter-DC environments with long latency and large bandwidth-delay product (BDP). RNIC-based approaches (IRN [10], SRNIC [11], FaSR [9]) replace GBN with more efficient SR logic in hardware but require extra state stored in on-chip memory. Over inter-DC paths with large BDP and frequent out-of-order (OoO) arrivals, the SR state expands rapidly [9], rendering SR infeasible on commodity RNICs with limited resources. In contrast, in-network based solutions offload retransmission into programmable switches, avoiding resource consumption on RNICs. Local-retransmission designs, such as LinkGuardian [12] and NetRT [14], hide losses from RNICs via switch-based retransmission, but require large buffers and become impractical over high BDP. Local-feedback designs, such as SLR [13], accelerate retransmission via early in-network loss detection and feedback. Yet, over inter-DC paths with high latency, the delayed feedback arrival undermines their benefit. These limitations call for a new approach that retains the efficiency of SR while overcoming hardware constraints in long-distance inter-DC environments.

Our key insight is that inter-DC RDMA paths naturally form a hierarchical structure: short-latency intra-DC segments bridged by a long-latency inter-DC backbone. This asymmetry enables a principled decomposition of end-to-end loss recovery. Building on this, we introduce the idea of collaborative retransmission decomposition, wherein SR logic is partially offloaded to programmable switches while coordination remains at the RNIC to ensure line-rate operation and minimal buffer. The core idea is to reconstruct end-to-end SR semantics in the network by retrieving GBN retransmission from the sender and selectively recovering losses at ingress of the long-haul link with minimal overhead. Complementing this, loss recovery

within short-latency intra-DC segments can be accelerated via upstream fast feedback and near-receiver backup.

Building on the above insight, we propose Loss Recovery for Long-distance RDMA (LR2), tailored for inter-DC environments. LR2 is deployed at both ends of the long-haul link (i.e., datacenter interconnection (DCI) switches) using two programmable switch modules. The near-sender module, `Sentry`, detects packet loss and generates early feedback, while also filtering unnecessary retransmission before entering the long-haul link. The near-receiver module, `Depot`, buffers OoO packets to support SR over the long-haul link and backs up in-order packets to accelerate last-hop recovery. Together, these modules emulate long-distance SR with minimal in-network state. The RNIC remains GBN-compliant, augmented by lightweight coordination logic to cooperate with in-network modules. LR2 thus preserves RNIC simplicity while outsourcing SR complexity to the network, improving responsiveness and reducing memory and bandwidth consumption.

While designed to operate together, `Sentry` and `Depot` can also function independently. We further analyze the performance boundaries of both modules, demonstrating LR2's adaptability across diverse network configurations. To validate its practicality, we build a prototype of LR2 on P4-programmable switches equipped with Barefoot Tofino ASIC, and evaluate it using both testbed experiments and large-scale simulations. Our results show that LR2 reduces flow completion time (FCT) by up to 40–70% over GBN under real production workload, with low system overhead.

We summarize the contributions of this paper as follows:

- LR2 proposes a collaborative retransmission decomposition for long-distance RDMA. It leverages the hierarchical structure of inter-DC paths to decouple loss recovery, enabling in-network SR across long-haul links and fast GBN retransmission near the RNIC.
- The pair of in-network modules, `Sentry` and `Depot` coordinate via a lightweight dual-register abstraction and explicit state transitions to reconstruct SR in the network with minimal state, while preserving RNIC simplicity and integrating cleanly with GBN timeout logic.
- We implement LR2 on P4-programmable switches. Testbed evaluations and large-scale NS-3 simulations demonstrate a significant reduction in FCT and bandwidth overhead compared to baselines, with low hardware cost.

The rest of this paper is organized as follows. Section II includes the background and motivation. Sections III and IV present the design and implementation, respectively. Section V presents the evaluation results and Section VI presents the related work. Finally, we conclude this paper in Section VII.

II. Background and Motivation

A. Long-distance RDMA over Inter-DC Deployment

Extending RDMA across DCs. Cloud service providers (CSPs) increasingly adopt regional cluster architectures that interconnect multiple DCs over dedicated long-haul links. Notable examples include Azure [3] and China's national computing backbone [4]. These deployments typically rely on datacenter interconnection (DCI) switches to connect distributed DCs [1], [15], as illustrated in Fig. 2.

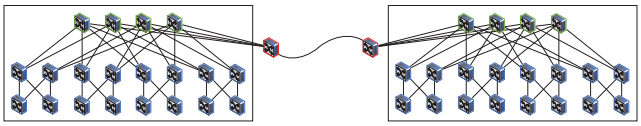


Fig. 2: Datacenter interconnection via DCI switches.

RDMA has become the standard transport layer within DCs and is now being extended to inter-DC environments. Two main RDMA protocols are Infiniband (IB) [16] and RDMA over Converged Ethernet version 2 (RoCEv2) [17]. While IB offers high performance in high performance computing (HPC), extending it across DCs often relies on costly optical gear [18]. In contrast, RoCEv2 operates over standard Ethernet/IP, reducing infrastructure cost and complexity, and is thus preferred for wide-area RDMA deployments [19], [20]. This paper also focuses on extending RoCEv2 across DCs. In the following, we use RDMA and RoCEv2 interchangeably.

Characteristics of long-distance RDMA. DCI long-haul links span hundreds to thousands of kilometers, incurring significant propagation latency. For instance, every 100km of fiber introduces approximately 0.5ms of one-way delay, yielding a base round-trip time (RTT) of about 1ms. Moreover, long-distance optical attenuation and the physical-layer impairments [8], [12], [21] cause frequent non-congestion losses, which RoCEv2's Priority Flow Control (PFC) [6] cannot address. The combination of elevated packet losses and high latency differentiates inter-DC long-distance RDMA from intra-DC settings and fundamentally affects the effectiveness of existing loss recovery mechanisms.

B. Existing Loss Recovery is not Enough

Performance penalty of GBN under long control loop at high latency. RoCEv2 depends on in-order, lossless delivery. Due to limited RNIC capacity, it typically employs a lightweight Go-back-N (GBN) mechanism for loss recovery. Upon receiving an out-of-order (OoO) packet, the receiver discards it and generates a negative acknowledgement (NACK), prompting the sender to retransmit all packets after the last acknowledged one. Over long-haul links with high RTT and large BDP, GBN suffers from long feedback loops and redundant retransmission, which severely degrade performance. Specifically, GBN penalizes performance due to: (i) high recovery latency, caused by slow loss detection and long feedback loops; and (ii) bandwidth waste, due to retransmitting packets that are already successfully delivered.

Large BDP limits RNIC support for selective retransmission. Selective retransmission (SR) improves upon GBN by retransmitting only the missing packets. However, realizing SR at the RNIC requires SR-specific data structures, such as bitmaps, reordering buffers, and outstanding request tables. While some components (e.g., reordering buffers) can be partially offloaded to the application layer, core SR logic

still demands on-chip memory. As described in IRN [10] and SRNIC [11], each queue pair (QP) requires five BDP-sized bitmaps (e.g., 500 slots per bitmap for 100Gbps/40µs). This requirement becomes infeasible over long-distance inter-DC scenarios, where large BDP drastically increase memory pressure [9]. Yet, commodity RNICs typically provide only a few megabytes of on-chip SRAM (e.g., 2MB in Mellanox CX5 [22]) to store all QP states and control structures, fundamentally limiting the scalability of SR over long-distance scenarios.

C. Collaborative Loss Recovery via DCI Switches

As shown in Table I, modern programmable switches provide ample buffer space, making them well-suited to support in-network retransmission offloading and complement the limited on-chip memory of RNICs.

TABLE I: Buffer size of mainstream programmable switches.

Switch Commodity	Buffer Size
Spectrum 3 [23]	64 MB
Tofino 2 [24]	64 MB
Cisco Nexus 9300+ series [25]	$>$ 120MB
Arista 7050+ series [26]	$>$ 132MB

However, the large BDP makes it impractical to offload all retransmission logic to switches, as even switch buffers can become a bottleneck. We instead propose a collaborative approach, where DCI switches and RNICs jointly share retransmission responsibilities. The hierarchical latency of inter-DC paths further enables geographically decomposed recovery: *performing long-loop SR and accelerating short-loop GBN retransmission*, motivating our design as follows:

Mimic end-to-end SR without large buffers. Persistently buffering all in-flight packets at the switch is inefficient. However, sender-side DCI switches are physically close to the RNIC and can fetch packets on demand with low latency, avoiding such overhead. We instead leverage the RNIC's short-loop GBN behavior and near-sender filtering to mimic end-to-end SR. By confining GBN to run over the short intra-DC path and filtering redundant retransmissions before they enter the long-haul link, this design not only avoids bandwidth waste on the long-haul bottleneck, but also obviates the need for large in-network buffers.

Shorten the GBN loop via near-end assistance. Switch-generated direct feedback is used to introduce fast loss signalling within the network. Specifically, assisted switches detect packet loss and generate negative acknowledgements (NACKs), enabling the sender to respond before the receiver observes the loss. Moreover, packet loss near the destination, often caused by congestion or link-layer errors within downstream DCs, is non-negligible but should not trigger costly long-distance retransmission. A near-receiver backup can enable local retransmission within the DC, reducing recovery latency. The buffer requirement remains modest, as it only needs to cover the intra-DC BDP.

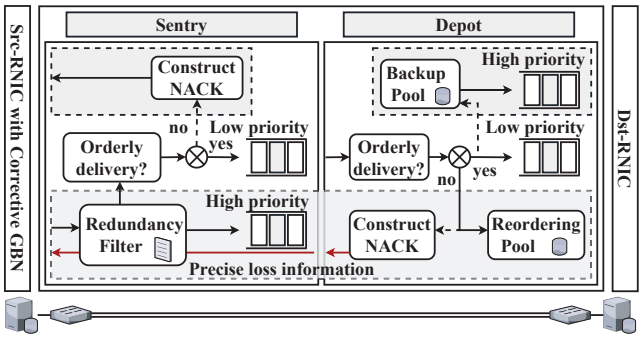


Fig. 3: Framework of LR2.

III. Design of LR2

In this section, we present the design of LR2, a switch-assisted **L**oss **R**ecovery system for **L**ong-distance **R**DMA. We aim to develop a simple yet efficient solution by leveraging existing switch capabilities for collaborative loss recovery. The design of LR2 is guided by the following goals:

- ***Mask near-receiver loss***: Prevent unnecessary long-haul retransmission by masking packet loss within the downstream DC through near-receiver buffering.
- ***Accelerate near-sender feedback***: Enable fast retransmission by generating early NACKs, shortening the feedback loop within the upstream DC.
- ***Reconstruct SR across long-haul links***: Enable SR-like loss recovery by combining lightweight near-receiver buffering and near-sender filtering, with minimized cost.

A. Overview

Fig. 3 presents the framework of LR2, comprising two switch-resident components: `Depot` at the near-receiver DCI switch and `Sentry` at the near-sender DCI switch. They collaborate to enable SR across long-haul links while shortening the retransmission control loop within the local-DC independently. LR2 maintains the RNIC's native GBN behavior, introducing only negligible modifications at src-RNIC for compatibility. `Depot` processes packets before they are delivered to the receiver. It performs two key tasks: 1) buffering out-of-order (OoO) packets arriving from long-haul links into a reordering pool while generating `Depot`-NACK with lost packets information carried; and 2) backing up in-order packets sent to the downstream DC to enable link-level retransmission if needed by the receiver side. `Sentry` inspects packets before they traverse long-haul links. It performs two tasks: 1) parsing `Depot`-NACKs to record SR states and filtering duplicate packets; and 2) masking upstream losses before the long-haul link by cutting near-sender OoO packets and generating direct NACKs for fast sender retransmission.

B. The `Depot` Module

`Depot` resides at the near-receiver DCI switch. It maintains two isolated buffer pools: a reordering pool for temporarily storing out-of-order (OoO) packets from the long-haul link, and a backup pool for retaining in-order packets forwarded

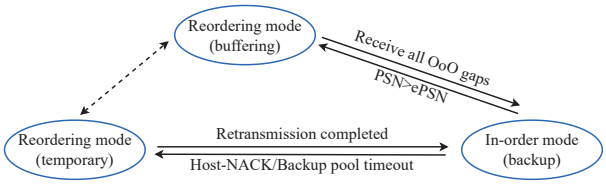


Fig. 4: Reording/In-order state transition.

to the downstream DC. To coordinate retransmission, `Depot` maintains a ternary state machine: reordering (temporary), reordering (buffering), and in-order (backup), to manage the transition from OoO reception to reliable in-order delivery, as illustrated in Fig. 4. We next present the core functionality of `Depot`, with symbols summarized in Table II.

1) ***OoO buffering and SR state management****:* `Depot` tracks the expected packet sequence number ($ePSN$) for each flow using a P4 register. Upon detecting an OoO packet ($PSN > ePSN$), `Depot` transitions into the reordering mode (buffering), redirects subsequent packets into the isolated reordering pool, preventing interference with the normal forwarding path. It then generates a `Depot`-NACK and sends it back to `Sentry`. To distinguish `Depot`-NACKs from host-generated NACKs, a reserved bit in the Base Transport Header (BTH) is reused as a type flag.

Instead of high-overhead bitmap-based tracking, `Depot` adopts a lightweight dual-register mechanism to manage SR state with minimal state and feedback overhead. For each flow, an auxiliary register $ePSN\_D$ complements $ePSN$. During in-order delivery, the two remain synchronized. When OoO packets arrive, $ePSN$ is frozen to mark the in-order boundary, while $ePSN\_D$ advances with subsequent packets, tracking the upper edge of reordering window. This allows efficient gap computation and feedback generation using only $ePSN\_D$, as illustrated in Fig. 5. The `Depot`-NACK uses $ePSN\_D$ as the reported sequence number and encodes only the gap length using the remaining 14 bits in reserved BTH field. This design avoids unnecessary rollbacks and provides sufficient precision under the typically sparse and random loss patterns of long-haul links. In particular, when `Depot` is deployed independently, there is no need to carry the OoO gap.

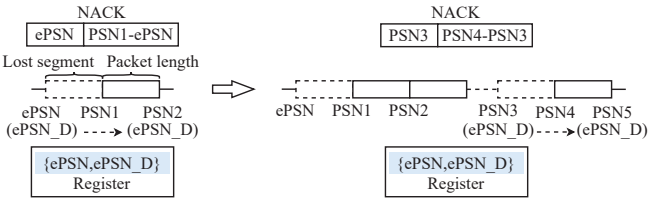


Fig. 5: Dual-register OoO tracking.

By default, the reordering pool is provisioned to match the long-haul link's BDP, covering the worst-case need to buffer all in-flight packets. When the buffer is exhausted, `Depot` stops accepting additional OoO packets. In such a case, `Depot` generates a host-type NACK to request retransmission of subsequent packets.

TABLE II: Symbol Description.

Symbol	Description
PSN	packet sequence number
$ePSN$	expected packet sequence number
ack_seq	acknowledged packet sequence number
$recover\_seq$	recovery point for backup pool
ret_state	retransmission flag for backup pool

2) ***In-order backup and link-level retransmission****:* When packets from the long-haul link arrive in order, `Depot` typically enters the in-order (backup) mode. Each in-order packet is copied into the backup pool and released upon receiving the corresponding ACK. The backup pool, together with the destination RNIC, forms a complete near-receiver retransmission loop to handle downstream DC losses. Upon receiving a host-NACK, `Depot` triggers link-level retransmission and discards the host-NACK to avoid unnecessary end-to-end retransmission. To deal with ACK/NACK losses, the backup pool supports timeout retransmission. Each flow maintains a $timer$ register, which is refreshed upon ACK reception. If the elapsed time exceeds the configured threshold T_{out}, it actively triggers retransmission of all packets in the backup pool for this flow. When link-level retransmission is triggered, the current value of $ePSN$ is recorded in a dedicated register as the recovery point $recover\_seq$. `Depot` then rolls back $ePSN$ and initiates retransmission from the backup pool for packets with PSN in the range [$ePSN$, $recover\_seq$]. The process terminates once the ACK for $recover\_seq$ is received.

To protect the data processing pipeline consistency, `Depot` conservatively redirects all subsequent arriving packets to the reordering pool once retransmission from the backup pool is triggered. It then switches to reordering mode (temporary), and resumes normal forwarding only after recovery completes and buffered packets are drained.

The backup pool must hold at least one BDP within the downstream DC to store all in-flight packets. To support timeout retransmission, a larger capacity of $B \times T_{out}$ is required, where B denotes bandwidth. We configure T_{out} as twice the downstream DC's base RTT to balance retransmission timeliness and buffer efficiency. If the backup pool becomes full and packets requested for retransmission are not backed up, `Depot` disables link-level retransmission and normally forwards host-NACKs to request end-to-end recovery.

3) ***Compatibility with RNIC Timeout Retransmission****:* Although LR2 handles the majority of packet losses in-network, RNIC-side timeout recovery may still occur under unexpected conditions. Without special handling, these retransmissions may be misidentified as duplicates and dropped by `Depot` ($PSN < ePSN$), causing deadlock. To ensure compatibility, `Depot` tracks the latest acknowledged sequence number ack_seq extracted from ACKs. If an incoming packet has $PSN \leq ack\_seq$ (typically with $ack\_seq < ePSN$), it is recognized as an RNIC timeout retransmission. In response, `Depot` treats it as the new expected packet, immediately resets

Algorithm 1: Packet Process of `Depot` Module

Input: pkt
$fid \leftarrow$ hash(pkt) ; ▶ *hash flow index*
packet_classify(pkt) ; ▶ *enter specific process*
Process `Data`
  if $pkt.PSN \leq ack\_seq[fid]$ **then**
    $ePSN[fid] \leftarrow pkt.PSN + pkt.length$;
    $ePSN\_D[fid] \leftarrow ePSN[fid]$;
    clear_reordering_and_backup_pools(fid);
  else if $pkt.PSN = ePSN[fid]$ **then**
    $ePSN[fid] \leftarrow pkt.PSN + pkt.length$;
    $ePSN\_D[fid] \leftarrow \max(ePSN\_D[fid], ePSN[fid])$;
    forward_and_copy_to_backup_recirculation(pkt);
  else if $pkt.PSN > ePSN[fid]$ **then**
    if $pkt.PSN > ePSN\_D[fid]$ **then**
      send_NACK($ePSN\_D[fid]$,$pkt.PSN - ePSN\_D[fid]$) ; ▶ `Deport-NACK`
    $ePSN\_D[fid] \leftarrow pkt.PSN + pkt.length$;
    copy_to_reordering_recirculation(pkt);
  else
    drop(pkt) ; ▶ *duplicate*
Process `ACK`
  $ack\_seq[fid] \leftarrow pkt.ack$;
  refresh_backup_pool_timer(fid);
  if $ack\_seq[fid] >= recover\_seq[fid]$ **then**
    $ret\_state[fid] \leftarrow false$;
Process `NACK`
  $ret\_state[fid] \leftarrow true$; ▶ *recover begins*
  $recover\_seq[fid] \leftarrow ePSN[fid]$;
  $ePSN[fid] \leftarrow pkt.nack$;

Algorithm 2: Buffer Pool of `Depot` Module

Function `backup_recirculation`
  if *check_timer_timeout(*fid*)* **then**
    $ret\_state[fid] \leftarrow true$; ▶ *timeout recover begins*
    $recover\_seq[fid] \leftarrow ePSN[fid]$;
    $ePSN[fid] \leftarrow ack\_seq[fid]$;
  if $pkt.PSN < ack\_seq[fid]$ **then**
    drop(pkt) ; ▶ *release acknowledged packet*
  else if *ret_state[fid]* ***and*** $pkt.PSN = ePSN[fid]$ ***and*** $pkt.PSN < recover\_seq[fid]$ **then**
    copy_to_forward(pkt) ; ▶ *retransmit backup pkt*
    $ePSN[fid] \leftarrow pkt.PSN + pkt.length$;
Function `reordering_recirculation`
  if $pkt.PSN == ePSN[fid]$ **then**
    forward_and_copy_to_backup_recirculation(pkt);
    $ePSN[fid] \leftarrow pkt.PSN + pkt.length$;

$ePSN$ and $ePSN\_D$, releases all packets in the reordering and backup pools for this flow, and re-aligns with the RNIC state to resume collaborative in-network recovery.

Finally, Algorithm 1 and 2 summarize the process logic for new incoming packets and local buffer pool in the `Depot` module, respectively. In particular, the reordering and backup pools are implemented using switch recirculation, where packets are recirculated to the ingress through queues in the internal recirculation port and reprocessed.

C. The `Sentry` Module

`Sentry` resides at the near-sender DCI switch. Rather than buffering and retransmitting lost packets locally, it accelerates recovery by generating early NACKs and suppressing redundant retransmissions near the source.

1) ***Near-sender direct feedback****:* `Sentry` maintains a per-flow expected sequence number $ePSN_s$. It enforces in-order forwarding across the long-haul segment by allowing only packets with $PSN = ePSN_s$ to proceed. OoO packets ($PSN > ePSN_s$) are immediately dropped, and a `Sentry`-NACK carrying the current $ePSN_s$ is sent to the source RNIC to trigger fast retransmission. To distinguish `Sentry`-NACKs, the same reserved bit in the BTH is reused as a type flag, consistent with `Depot`-NACKs.

2) ***SR filter****:* To support collaborative SR with `Depot`, `Sentry` maintains a per-flow bitmap register to track missing sequence ranges indicated in `Depot`-NACKs. Packets with $PSN < ePSN_s$ are identified as retransmissions. Only those matching the bitmap are forwarded, while others are discarded as duplicates. Unlike regular forwarding, valid retransmissions that pass the SR filter are enqueued into high-priority queues for expedited delivery. In particular, when `Sentry` is deployed independently, the SR filter is not required.

3) ***Compatibility with RNIC Timeout/NACK Retransmission****:* Similar to `Depot`, `Sentry` resets its internal state when RNIC timeout retransmissions are detected. It clears the SR bitmap and resets $ePSN_s$ to realign with the sender. Additionally, it performs the same reset when a host-NACK is received to maintain consistency with end-to-end recovery.

D. Src-RNIC Corrective GBN

LR2 preserves conventional GBN behavior at the source RNIC, while introducing minimal coordination logic to enable cooperation with in-network components. `Sentry`-NACKs and `Depot`-NACKs, collectively referred to as switch-NACKs, are generated within the network and detect losses on specific path segments. Packet losses may still occur on other segments, requiring the sender to retransmit packets with PSN earlier than those indicated in switch-NACKs. To support this logic, source RNIC identifies the NACK type by examining the designated flag in the BTH header. Upon receiving a switch-NACK, it retransmits from the indicated PSN but does not update the last acknowledged PSN (lACK), allowing subsequent NACKs or timeout events to trigger recovery from earlier PSN. In contrast, a host-NACK updates both the retransmission point and the lACK, following standard GBN semantics. This correction does not modify the RNIC architecture or transport control logic, and thus incurs no additional overhead.

E. Performance Boundary

We further characterize LR2's performance boundary over a long, lossy path. Specifically, we consider a single-flow model

where N packets are transmitted over a link with one-way delay D, bandwidth B, and packet loss rate e.

For GBN, it retransmits only after detecting loss via an RTT delay. During this period, all in-flight packets are discarded by the receiver. Assume that the sender successfully transmits $n-1$ consecutive packets before encountering a loss on the n-th packet. Given an independent per-packet loss probability e, the expected number of packets per transmission round is $1/e$; we denote this expectation as L, representing the average number of packets successfully delivered per round. The transmission time for each round is:

$$T_{round} = \frac{L}{B} + 2D, \tag{1}$$

where the first term $\frac{L}{B}$ accounts for the effective transmission time and the second term $2D$ is the additional retransmission time. Including an additional round trip for final acknowledgement, the total flow completion time (FCT) becomes:

$$T = T_{round} * \frac{N}{L} + 2D = \frac{N}{B} + 2D(1 + Ne). \tag{2}$$

This result is intuitively consistent: each packet loss introduces an additional delay of $2D$ due to the end-to-end retransmission. Compared to the ideal FCT without loss, given by $T_{ideal} = \frac{N}{B} + 2D$, the FCT slowdown is:

$$S_{GBN} = \frac{T}{T_{ideal}} = 1 + \frac{Ne}{1+\delta}, \tag{3}$$

where $\delta = N/(2D*B)$, denotes the flow size relative to BDP.

With `Sentry` enabled, the delay of direct feedback can be negligible compared to that of the long-haul link, allowing the near-sender link to be considered lossless. Based on Eq. (3), the FCT slowdown of `Sentry` can be expressed as:

$$S_{sentry} = 1 + \frac{Ne_s}{1+\delta}, \tag{4}$$

where e_s denotes the equivalent loss rate.

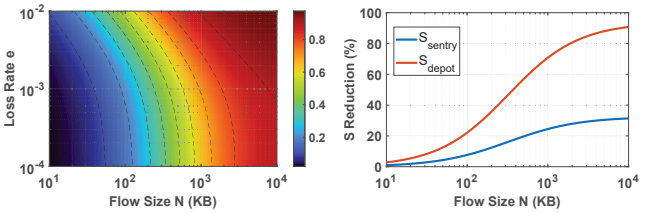
With `Depot` enabled, the near-receiver link can be considered lossless due to negligible link-level retransmission delay. We denote the equivalent loss rate as e_d. Even without the collaboration of `Sentry` to perform SR, `Depot`'s OoO buffering still effectively reduces recovery time. During the blocked RTT period caused by a packet loss, subsequent OoO packets continue to be received; any loss among these packets will trigger a `Depot`-NACK, which is consecutively transmitted to the sender. As a result, packet losses during this period only block one RTT. The FCT slowdown is:

$$S_{depot} \approx 1 + \frac{Ne_d}{(1+\delta)(1+2D*B*e_d)}, \tag{5}$$

which can be interpreted as reducing the number of lost packets by a factor of $(1 + BDP * e_d)$.

Based on Eq. (4) and Eq. (5), we obtain the FCT slowdown for LR2:

$$S_{LR2} \approx 1 + \frac{Ne_l}{(1+\delta)(1+2D*B*e_l)}, \tag{6}$$



(a) $(S_{GBN} - S_{LR2})/S_{GBN}$ (b) Decoupled LR2 ($e = 10^{-3}$)

Fig. 6: Improvement of LR2 over GBN.

where e_l is the equivalent loss rate, i.e., the actual loss rate of the long-haul link.

Taking the topology in Fig. 2 as an example, D and B are 400us and 100Gbps. The average total packet loss rate within DCs is 10^{-3}, derived from Microsoft datacenter links [8]. The loss rate of the long-haul link is e. Fig. 6 shows LR2's reduction in FCT relative to GBN under different N and e. For a single fixed-size flow, when e is less than 10^{-3}, the results are dominated by packet losses within DCs. As e increases, LR2 exhibits more pronounced improvement. Taking e to be 10^{-3} as an example, decoupled LR2 modules effectively reduce FCT especially the long flow, with `Depot` outperforming `Sentry`. The result of LR2 is not plotted, as it closely resembles that of `Depot`. The reason is that S_{LR2} becomes insensitive to e when BDP is large.

We assert that the above analysis establishes a lower bound for LR2's performance, since reducing FCT effectively mitigates network congestion, which is particularly pronounced in high-latency environments, as confirmed in Section V.

IV. Implementation

Hardware support. LR2 is fully implemented in the dataplane using approximately 1,000 lines of P4 code, and most of the required features can be easily satisfied by programmable switches, including packet field parsing, per-flow state tracking, packet replication and modification, etc. A key challenge lies in supporting in-network buffering for reordering and backup at line rate. We deploy LR2 on Tofino switches, utilizing the `recirculation` feature to achieve this. Besides two internal recirculation ports, more ports can be added by running idle ports in loopback mode. With more features supported by advanced programmable switches, such as Tofino2's Advanced Flow Control (AFC), in-networking buffering could theoretically be implemented more efficiently without `recirculation` [12], [27].

NACK construction: LR2 leverages the `mirror` feature to generate NACKs. Upon detecting an OoO packet in the ingress pipeline, the packet is mirrored to the traffic manager (TM) with its payload truncated. It is then converted into a NACK packet in the egress pipeline.

Local isolated buffer pool: By configuring recirculation ports, different buffer spaces can be allocated for reordering and backup pools. In-order packets are backed up via `mirror`. To ensure in-order forwarding, each recirculated packet is

TABLE III: Hardware Resource Consumption.

Resource	LR2	switch	LR2+switch
SRAM	5.1%	58.0%	$\leqslant$ 63.1%
Stateful ALUs	29.2%	31.3%	$\leqslant$ 60.5%
Gateway	25.5%	28.6%	$\leqslant$ 54.1%
VLIW Inst	10.7%	20.3%	$\leqslant$ 31.0%
Hash Bit	10.3%	38.8%	$\leqslant$ 49.1%

checked against $ePSN$ upon returning to the ingress pipeline and sent back repeatedly until it is eligible for forwarding.

Timeout timer: Timers are not typically supported in switches. LR2 emulates timeout detection of the backup pool by using per-flow registers. The register records the timestamp of the first backup packet and is updated with the receipt of ACK/NACK. Each time a backup packet is recirculated to the ingress pipeline, the difference between its timestamp and the stored register value is calculated, and the timeout is triggered if it exceeds the predefined threshold.

Protocol overhead. LR2 introduces no additional headers and only reuses the reserved fields in BTH for control information, minimizing modifications to the native transport architecture while eliminating extra bandwidth consumption.

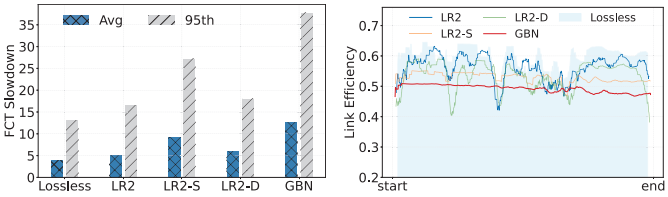
Hardware resource overhead. The related dataplane resource consumption is summarized in Table III. SRAM usage stems from P4-table lookups (1.8%) and stateful registers (3.3%). Stateful ALUs are used for processing register states on a per-flow basis. Gateway is used for conditional judgment, specifically handling sequence number comparisons in LR2. VLIW Instruction primarily evaluates P4-action complexity, and the result indicates that LR2 has a lightweight logical implementation. Other resources, such as Hash Bit, maintain relatively low consumption. For comparison, we provide the overhead of `switch.p4-16`, a networking stack that provides commonly used networking features. As the total utilization remains well below hardware limits, LR2 can be fully integrated alongside standard switch functions.

V. Performance Evaluation of LR2

In this section, we evaluate the performance of LR2 through both testbed experiments and NS-3 [28] simulations.

A. Evaluation in Testbed

Setup. Our testbed comprises four hosts and two 32 $\times$ 100Gbps Barefoot Tofino switches (each connects to two hosts). Switches are connected via an additional delay-host that emulates a long-haul link using DPDK [29]. Specifically, three dedicated threads are deployed: a receiving thread forwards packets from a receive queue to a ring buffer (Rx-ring) while recording precise arrival timestamps; a delay thread retains the packets in the Rx-ring buffer for a predetermined duration D before forwarding them to a Tx-ring buffer; a sending thread transmits packets from the Tx-ring buffer to a send queue. Each host is equipped with an Intel E810 100GbE dual-port RNIC. Key parameters are set as follows: D is set to 400us, corresponding to a distance of 80km; The link loss rate defaults to 10^{-3}; the RNIC line rate is configured at 10Gbps



(a) FCT slowdown (b) Link efficiency

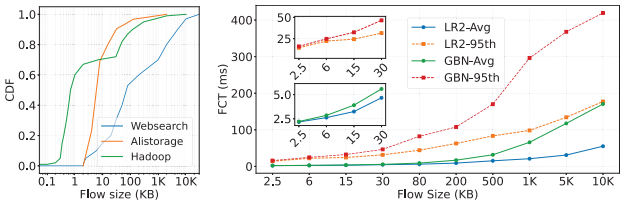
Fig. 7: Individual module contribution.

due to limitation of the delay-host. Consequently, the BDP of our testbed topology is $\sim$ 1MB, equivalent to 1,000 MTU-size packets. The reordering pool size is accordingly restricted to 1MB. Based on our empirical measurements for link delay ($\sim$ 10us), the backup pool size is configured to 50KB. The traffic pattern defaults to Websearch [30] with a load of 60%, senders and receivers are selected randomly, and the flow arrival time follows a Poisson distribution. The default congestion control algorithm is DCQCN with the same parameters in [7].

Ablation study. To assess the contribution of individual components within LR2, we conduct experiments with decoupled modules enabled independently, and compare their results against the complete LR2, GBN, and an idealized lossless baseline. As shown in Fig. 7. Compared to GBN, `Sentry` (LR2-S) and `Depot` (LR2-D) reduce the average FCT slowdown by 28% and 53%, while reducing the 95th percentile tail latency by 28% and 52%. LR2-D performs better than LR2-S, yielding results comparable to LR2. We further measure the utilization efficiency of the long-haul link. Given that the link is not fully saturated, the ideal utilization is approximately 60%. GBN maintains a relatively low utilization of around 50%, primarily due to excessive redundant retransmissions, particularly in the early stages. In contrast, LR2 effectively suppresses redundant retransmissions, achieving link utilization that resonates with the ideal condition.

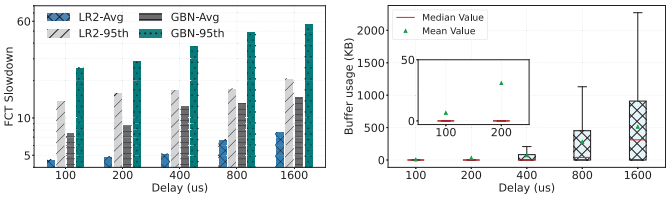
Different flow sizes. We categorize FCT based on different flow sizes to evaluate LR2's performance under different workload types. In addition to Websearch, Fig. 8a presents Hadoop [31] and distributed Alistorage [32] workloads. Hadoop workload is predominantly characterized by small flows, typically less than 1KB, while Alistorage workload mostly comprises flows around 1-10KB. Due to space constraints, results are presented for Websearch only, which exhibits a broader distribution of flow sizes, as shown in Fig. 8b. LR2 demonstrates substantial performance gains for large flows. Specifically, average FCT and tail latency are approximately halved for flows exceeding 500KB. For small flows around 10KB, the reduction is about 10-20%. Note that since the measurement error is on the order of microseconds, its effect on the experimental results is negligible.

Different loss rates. Fig. 9 shows the FCT and buffer occupancy for different loss rates. Even at a low loss rate of (10^{-4}), LR2 achieves an approximately 25% reduction in average FCT. As loss rate increases to 10^{-2}, the average FCT and tail latency are reduced by $\sim$ 40-50%. At low loss rates,



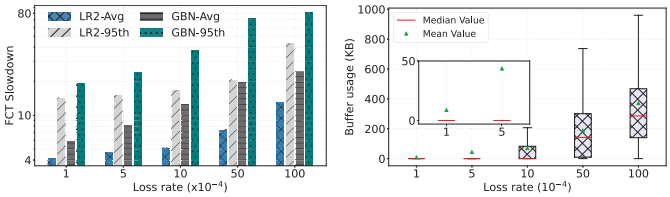
(a) Traffic distribution (b) FCT and tail latency

Fig. 8: FCT categories by different flow sizes.



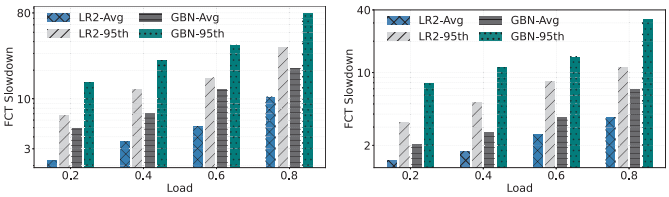
(a) FCT slowdown (b) buffer occupancy

Fig. 10: FCT slowdown and buffer occupancy vs. distance.



(a) FCT slowdown (b) buffer occupancy

Fig. 9: FCT slowdown and buffer occupancy vs. loss rate.



(a) Websearch (b) Alistorage

Fig. 11: FCT slowdown vs. traffic load.

buffer usage is sparse, with the median value close to 0, and the average occupied size not exceeding 50 packets. As the loss rate surpasses 10^{-3}, continuous buffer occupancy emerges. When the loss rate reaches 10^{-2}, the peak occupancy reaches 1MB, which is close to the configured reordering pool size.

Different distances. We configure a varying delay to represent different long-haul link distances. As shown in Fig. 10, as the distance increases, LR2 maintains a stable improvement over GBN. At a relatively short distance (20km/100us), LR2 reduces the average FCT and tail latency by $\sim$ 40-46%, while for longer distances ($>$ 160km/800us), the reduction exceeds 50% and 60%. Moreover, buffer usage increases with longer distances, and the occupancy rate rises steadily. This trend confirms LR2's effectiveness in long-distance scenarios where propagation delay dominates performance characteristics.

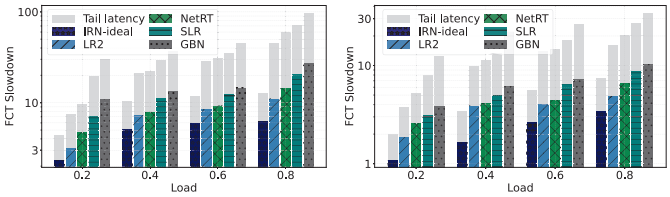
Different traffic loads. Finally, we evaluate the overall performance of LR2 across different workload types, specifically, Websearch and Alistorage, under varying average load. As shown in Fig. 11. LR2 exhibits better improvements under higher load conditions, particularly in reducing tail latency. A systematic analysis of the experimental data reveals that sluggish retransmissions lead to flow accumulation in the network, primarily manifesting as prolonged durations of large background flows. Under high-load scenarios, this accumulation readily triggers congestion. A comparative analysis of the two workloads shows that the workload with generally smaller flow sizes (Alistorage in this case) experiences less performance degradation under identical conditions. This is because a relatively smaller portion of flows is affected by packet loss (a smaller flow size distribution implies more flow numbers), while individual affected small flows are less likely to impact other flows.

Through the testbed experiments, we demonstrate that the decoupled LR2 modules deliver compelling performance with less overhead. Under varying packet loss conditions, LR2 imposes a maximum buffer requirement of one BDP, a level of overhead that is well within the capacities of modern switch hardware. LR2 maintains robustness across diverse workload types. Even under challenging high-loss, high-load conditions, LR2 maintains high efficiency, whereas GBN experiences significant performance degradation.

B. Large-Scale Simulation using NS-3

We further evaluate LR2 in a large-scale network environment using NS-3 simulator with the topology in Fig. 2, where the datacenter is a quaternary Fat-Tree [33] structure. Each ToR switch is connected to 2 servers (omitted in the figure). The long-haul link has a bandwidth of 400Gbps and a latency of 400us, while the others have 100Gbps capacity and 1us latency, resulting in an oversubscription of 1 within datacenters and $\sim$ 820 us baseRTT for cross-DC flows. The average total packet loss rate for a single datacenter is 10^{-3} [8], with the long-haul link maintaining an identical loss rate of 10^{-3} by default. The buffer sizes for intra-DC and DCI switches are 16MB and 132MB, respectively, based on real device configurations [23]–[26], and the buffer sizes for reordering and backup pools are 40MB and 200KB, respectively. The workload pattern is all-to-all traffic, and other setups are consistent with testbed evaluation. We compare LR2 against GBN, IRN, SLR and NetRT. For IRN, we remove the RNIC memory limitation to represent an ideal end-to-end retransmission; the corresponding parameters are referenced from [10]. For SLR, each switch operates in backward mode.

LR2 achieves smaller FCT. Fig. 12 shows the results under Websearch and Alistorage workloads. Except for the ideal IRN, LR2 demonstrates the best performance, with an average FCT slowdown and tail latency improvement of 40-70% and 36-74%, respectively compared to GBN. NetRT also shows competitive results with its OoO packets buffering mechanism, performing comparably to LR2 under light load. SLR effectively reduces tail latency via its fast NACK feedback mechanism. Comparing testbed results shows that, under the



(a) Websearch (b) Alistorgae

Fig. 12: Average FCT slowdown and tail latency under different workloads.

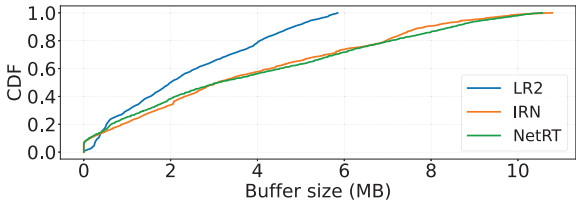


Fig. 13: Buffer occupancy comparison.

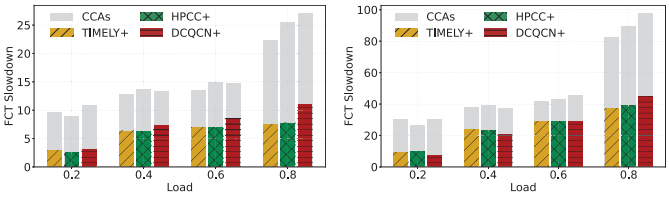
same conditions, the FCT of large-scale simulation is generally larger, particularly under light loads. This discrepancy is attributed to increased flow interactions in large-scale networks (which can carry more traffic). This aligns with our previous analysis in the testbed evaluation.

LR2 requires less buffer occupancy. To evaluate the cost of improvement, we measure buffer overhead, as shown in Fig. 13. For IRN, we calculate the receiver RNIC's buffer usage. For LR2 and NetRT, we calculate the total buffer usage of switches and average it across all servers. The SLR result is omitted since it does not require additional buffer resources. It can be seen that the buffer usage reaches the MB level, with LR2 using up to 6MB (corresponding to a total usage of 96MB for 16 servers in a single datacenter), significantly lower than IRN and NetRT. Since NetRT deploys out-of-order packet caching at the receiver-side ToR switches, the result is similar to that of IRN. A characteristic of LR2 is that it exhibits higher occupancy in the low buffer size region, which is attributed to packets in the backup pool.

LR2 enhances congestion control efficiency. We evaluate different congestion control algorithms (CCAs), with Fig. 14 comparing the results of TIMELY [34], HPCC [32], and DCQCN [7] before and after deploying LR2, where the gray area represents the original CCAs' results. Overall, there is no apparent difference between the results of these CCAs. Our experimental data indicate that CCAs suffer a significant performance penalty due to high latency, regardless of workload type and size (results under Alistorage are omitted due to space constraints). However, with LR2 enabled, both average FCT and tail latency are effectively improved. We believe that LR2 has better compatibility with congestion control algorithms designed explicitly for long-distance scenarios.

VI. Related Work

Congestion control for inter-DC scenarios. For TCP-based inter-DC transmission, Annulus [35] uses dual control loops to handle inter/intra-DC coexistence, while GEMINI [36] integrates ECN and delay signals. For RDMA-based inter-DC transmission, LSCC [37] and BiCC [19] deploy segmented control at DCI switches for upstream/downstream regulation.



(a) Average FCT slowdown (b) 95th percentile FCT slowdown

Fig. 14: Congestion control w/ vs. w/o LR2.

Flow control for long-distance RDMA. Long-haul links impose buffer pressure on DCI switches and degrade the RDMA flow control efficiency. SWING [38] uses PFC relays to expose downstream rates to upstream switches, improving link utilization efficiency and alleviating buffer pressure. Bifrost [20] infers traffic patterns from historical PFC events to enhance bandwidth efficiency and reduce latency.

Switch-assisted loss recovery for RDMA. Linkguardian [12] duplicates and caches each forwarded packet, and retransmits them upon receiving NACKs, achieving link-level retransmission. SLR [13] discards all subsequent packets upon detecting OoO arrivals and constructs NACK feedback for fast retransmission (backward mode). NetRT [14] offloads retransmission to ToR switches connected to end-hosts.

Improved RNIC loss recovery. IRN [10] pioneers SR offloading to RNICs by utilizing bitmaps to track both lost and received packets. SRNIC [11] implements a personalized reordering buffer, performs SR relying on CPUs. Flor [39], a unified framework for heterogeneous RNICs, leverages Unreliable Connections (UCs) to handle OoO delivery. FaSR [9] implements a reorder buffer and designs a shared SR pool to manage the overhead of SR state recording.

VII. Conclusion

In this paper, we presented LR2, a switch-assisted solution that reconstructs SR semantics without requiring large in-network buffers or complex RNIC modifications. By leveraging near-sender filtering and near-receiver buffering, LR2 shortens the control loop and mitigates retransmission overhead. We implemented the prototype on Intel Tofino switches and evaluated it via both testbed experiments and large-scale simulations. LR2 effectively addresses both recovery efficiency and memory constraints, providing a practical solution for reliable inter-DC communication.

Acknowledgment

We would like to thank the anonymous reviewers for their constructive comments. This work is supported in part by the National Natural Science Foundation of China under Grant No. 62302472, and the Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093.

REFERENCES

[1] V. Dukic, G. Khanna, C. Gkantsidis, T. Karagiannis, F. Parmigiani, A. Singla, M. Filer, J. L. Cox, A. Ptasznik, N. Harland *et al.*, "Beyond the mega-data center: Networking multi-data center regions," in *Proceedings of the 2020 ACM Special Interest Group on Data Communication (SIGCOMM)*. ACM, 2020, pp. 765–781.

[2] Y. Chen, Y. Lu, and J. Shu, "Scalable RDMA RPC on reliable connection with efficient resource sharing," in *Proceedings of the 14th ACM European Conference on Computer Systems (EuroSys)*. ACM, 2019, pp. 1–14.

[3] W. Bai, S. S. Abdeen, A. Agrawal, K. K. Attre, P. Bahl, A. Bhagat, G. Bhaskara, T. Brokhman, L. Cao, A. Cheema *et al.*, "Empowering azure storage with RDMA," in *Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX, 2023, pp. 49–67.

[4] Z. Fu, "Understanding china's eastern data western. calculation project. PINGWEST," 2022, accessed on Jul. 2025. [Online]. Available: https://en.pingwest.com/a/9940

[5] C. Guo, H. Wu, Z. Deng *et al.*, "RDMA over commodity ethernet at scale," in *Proceedings of the 2016 ACM Special Interest Group on Data Communication (SIGCOMM)*. ACM, 2016, pp. 202–215.

[6] "802.1qbb - priority-based flow control," Accessed: Jan., 2026. [Online]. Available: http://www.ieee802.org/1/pages/802.1bb.html

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

[8] D. Zhuo, M. Ghobadi, R. Mahajan, K.-T. Förster, A. Krishnamurthy, and T. Anderson, "Understanding and mitigating packet corruption in data center networks," in *Proceedings of the 2017 ACM Special Interest Group on Data Communication (SIGCOMM)*. ACM, 2017, pp. 362–375.

[9] P. Huang, G. Chen, X. Zhang, C. Liu, H. Wang, H. Shen, Y. Bian, Y. Lu, Z. Ruan, B. Li *et al.*, "Fast and scalable selective retransmission for RDMA," in *Proceedings of the 2025 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2025, pp. 1–10.

[10] R. Mittal, A. Shpiner, A. Panda, E. Zahavi, A. Krishnamurthy, S. Ratnasamy, and S. Shenker, "Revisiting network support for RDMA," in *Proceedings of the 2018 ACM Special Interest Group on Data Communication (SIGCOMM)*. ACM, 2018, pp. 313–326.

[11] Z. Wang, L. Luo, Q. Ning, C. Zeng, W. Li, X. Wan, P. Xie, T. Feng, K. Cheng, X. Geng *et al.*, "SRNIC: A scalable architecture for RDMA NICs," in *Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX, 2023, pp. 1–14.

[12] R. Joshi, C. H. Song, X. Z. Khooi, N. Budhdev, A. Mishra, M. C. Chan, and B. Leong, "Masking corruption packet losses in datacenter networks with link-local retransmission," in *Proceedings of the 2023 ACM Special Interest Group on Data Communication (SIGCOMM)*. ACM, 2023, pp. 288–304.

[13] Q. Meng, Y. Zhang, S. Zhang, Z. Wang, T. Zhang, H. Luo, and F. Ren, "Switch-assistant loss recovery for rdma transport control," *IEEE/ACM Transactions on Networking*, vol. 32, no. 3, pp. 2069–2084, 2023.

[14] W. Wang, J. Han, K. Xue, J. Li, K. Ding, and R. Li, "NetRT: Enhancing RDMA with retransmission offloading in data center networks," in *Proceedings of the 2025 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2025, pp. 1–10.

[15] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano *et al.*, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," *ACM SIGCOMM computer communication review*, vol. 45, no. 4, pp. 183–197, 2015.

[16] I. T. Association, "Infiniband architecture volume 1, general specifications," 2008.

[17] I. T. Association *et al.*, "Infiniband architecture specification release 1.4 annex a17: RoCEv2. 2020."

[18] NVIDIA, "Infiniband long-haul systems," accessed on Jul. 2025. [Online]. Available: https://www.nvidia.com/en-us/networking/infiniband-long-haul-systems/

[19] Z. Wan, J. Zhang, M. Yu, J. Liu, J. Yao, X. Zhao, and T. Huang, "BiCC: Bilateral congestion control in cross-datacenter RDMA networks," in *Proceedings of the 2024 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2024, pp. 1381–1390.

[20] C. Huang, F. Xue, P. Yu, X. Wang, Y. Chen, T. Wu, L. Han, Z. Han, B. Wang, X. Gong *et al.*, "Minimizing buffer utilization for lossless Inter-DC links," *IEEE/ACM Transactions on Networking*, vol. 32, no. 6, pp. 4960–4975, 2024.

[21] Y. Zhou, C. Sun, H. H. Liu, R. Miao, S. Bai, B. Li, Z. Zheng, L. Zhu, Z. Shen, Y. Xi *et al.*, "Flow event telemetry on programmable data plane," in *Proceedings of the Conference of the 2020 ACM Special Interest Group on Data Communication (SIGCOMM)*. ACM, 2020, pp. 76–89.

[22] A. Kalia, M. Kaminsky, and D. Andersen, "Datacenter RPCs can be general and fast," in *Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX, 2019, pp. 1–16.

[23] "NVIDIA Mellanox Spectrum-3," https://network.nvidia.com/files/doc-2020/pb-spectrum-3.pdf, accessed on Jul. 2025.

[24] "Intel® tofino™ 2," Accessed: Jan., 2026. [Online]. Available: https://www.intel.com/content/www/us/en/products/details/network-io/intelligent-fabric-processors/tofino-2/products.html

[25] "Cisco nexus 9000 series switches," Accessed: Jan., 2026. [Online]. Available: https://www.cisco.com/site/us/en/products/networking/cloud-networking-switches/nexus-9000-switches/index.html

[26] "Arista networks," Accessed: Jan., 2026. [Online]. Available: https://www.arista.com/en/products/platforms

[27] C. H. Song, X. Z. Khooi, R. Joshi, I. Choi, J. Li, and M. C. Chan, "Network load balancing with in-network reordering support for rdma," in *Proceedings of the 2023 ACM Special Interest Group on Data Communication (SIGCOMM)*. ACM, 2023, pp. 816–831.

[28] "NS-3 simulator," Accessed: Jan., 2026. [Online]. Available: https://www.nsnam.org/

[29] "Intel dpdk," Accessed: Jan., 2026. [Online]. Available: https://www.dpdk.org/

[30] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," in *Proceedings of the 2010 ACM Special Interest Group on Data Communication (SIGCOMM)*. ACM, 2010, pp. 63–74.

[31] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proceedings of the 2015 ACM Special Interest Group on Data Communication (SIGCOMM)*. ACM, 2015, pp. 123–137.

[32] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh *et al.*, "HPCC: High precision congestion control," in *Proceedings of the 2019 ACM Special Interest Group on Data Communication (SIGCOMM)*. ACM, 2019, pp. 44–58.

[33] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM computer communication review*, vol. 38, no. 4, pp. 63–74, 2008.

[34] R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "TIMELY: RTT-based congestion control for the datacenter," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 537–550, 2015.

[35] A. Saeed, V. Gupta, P. Goyal, M. Sharif, R. Pan, M. Ammar, E. Zegura, K. Jang, M. Alizadeh, A. Kabbani *et al.*, "Annulus: A dual congestion control loop for datacenter and WAN traffic aggregates," in *Proceedings of the 2020 Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication (SIGCOMM)*, 2020, pp. 735–749.

[36] G. Zeng, W. Bai, G. Chen, K. Chen, D. Han, Y. Zhu, and L. Cui, "Congestion control for cross-datacenter networks," *IEEE/ACM Transactions on Networking*, vol. 30, no. 5, pp. 2074–2089, 2022.

[37] M. Long, J. Han, W. Wang, J. Yang, and K. Xue, "LSCC: Link-segmented congestion control for rdma in cross-datacenter networks," in *Proceedings of the 2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*. IEEE, 2024, pp. 1–10.

[38] Y. Chen, C. Tian, J. Dong, S. Feng, X. Zhang, C. Liu, P. Yu, N. Xia, W. Dou, and G. Chen, "Swing: Providing long-range lossless RDMA via pfc-relay," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 1, pp. 63–75, 2022.

[39] Q. Li, Y. Gao, X. Wang, H. Qiu, Y. Le, D. Liu, Q. Xiang, F. Feng, P. Zhang, B. Li *et al.*, "Flor: An open high performance RDMA framework over heterogeneous RNICs," in *Proceedings of the 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2023, pp. 931–948.