Enabling Packet Spraying over Commodity RNICs with In-Network Support

Xiangzhou Liu, Wenxue Li, Zihao Wang, Kai Chen
iSING Lab, Hong Kong University of Science and Technology
{xliugg,wlicv,zwangnw}@connect.ust.hk, kaichen@cse.ust.hk

Abstract

AI training workloads exhibit unique traffic patterns that mismatch the ECMP load balancing of RDMA networks, leading to severe throughput degradation. While packet-level load balancing (e.g., random packet spraying, adaptive routing, etc.) offers a promising alternative to ECMP by providing fine-grained traffic distribution, it introduces out-of-order (OOO) packet arrivals. The reliable transport mechanism of current commodity RDMA NICs (RNICs) misinterprets these OOO arrivals as packet loss, causing spurious retransmissions and unnecessary slow starts.

We propose Themis, a lightweight middleware deployed on programmable switches, that eliminates the incompatibility between packet spraying and reliable transport in commodity RNICs without requiring any modifications to them. At its core, Themis applies PSN-based packet spraying at the source ToR switch, enabling it to infer whether an OOO packet and the expected packet traverse the same path. Building on this capability, Themis at the destination ToR analyzes NACKs triggered by OOO arrivals to determine whether they result from actual packet loss. It then blocks invalid NACKs while allowing valid ones to pass through. We implement a fully functional prototype of Themis using Tofino2. Experiments demonstrate that Themis reduces the communication completion time of Allreduce and Alltoall by 22.7%∼35.4% and 29.3%∼47.3%, respectively, compared to directly combining commodity RNICs and adaptive routing.

1 Introduction

With the rapid advancement of Artificial Intelligence (AI) [20, 43], AI training workloads have become a dominant component of modern datacenters [18, 24, 30, 33, 46]. These workloads rely on high-bandwidth RDMA technologies for inter-machine communication. However, their low-entropy traffic patterns are poorly aligned with Equal-Cost Multiple-Path (ECMP) [26], the de-facto load balancing mechanism in RDMA networks [24, 46]. Consequently, ECMP hash collisions can cause severe throughput degradation in AI training jobs [24, 33, 46].

Packet-level load balancing (e.g., random packet spraying [21], adaptive routing [9]) offers a promising alternative to ECMP by enabling fine-grained traffic distribution across all available paths. However, packet-level load balancing inevitably introduces out-of-order (OOO) packet arrivals [21, 32, 50]. To handle this, commodity RNICs [1, 7, 8] first support OOO packet reception through the Direct Data Placement (DDP) mechanism [11, 42], in which each packet carries an IB RETH header containing the destination memory address, enabling the RNIC to directly place incoming data at the target memory location regardless of arrival order. In addition, for reliability, commodity RNICs provide alternative mechanisms beyond the traditional Go-Back-N protocol, including timeout-only and selective repeat (RNIC-SR). However, both of them have fundamental limitations that undermine the efficiency of packet-level load balancing.

On one hand, the timeout-only mechanism suffers from inefficient packet loss recovery. Since packet loss is inevitable in large-scale datacenters (§2.2), a single lost packet can stall the entire training process, as all workers must wait for retransmission triggered only after a timeout [55]. On the other hand, RNIC-SR is incompatible with packet-level load balancing. In particular, RNIC-SR is primarily designed for single-path transmission and assumes that all OOO arrivals are caused by packet loss. It therefore blindly generates a NACK to trigger fast retransmission of the expected packet. As a result, when incorporated with packet-level load balancing, the receiver generates NACKs for OOO packets even when no actual loss has occurred and the reordering is solely due to path delay variation. This incompatibility introduces three major issues:

- **Unnecessary slow starts**: NACKs trigger the sender's slow start mechanism, causing unnecessary transmission rate reduction.
- **Retransmissions disrupt TX data path**: Excessive spurious retransmissions disrupt the sender's RNIC TX data path, causing low transmission rates.
- **Bandwidth waste**: Spurious retransmissions waste bandwidth by retransmitting packets that were not lost.

Some works propose new transport protocols that require rearchitecting RNICs [35, 40, 41, 48] to enable fine-grained multipath transmission. However, such specialized hardware requires a long time to develop, deploy, and validate, which limits its adoption. Most AI training clusters still rely on commodity RNICs [24, 33, 46], such as ConnectX-6 [8], ConnectX-7 [1] and BlueField3 [7], due to their proven stability and availability. Yet, these commodity RNICs lack the

full transport-layer programmability needed to implement the aforementioned protocols.

Motivated by the above analysis, we pose the following question: *Can packet-level load balancing be enabled while still achieving efficient loss recovery using off-the-shelf commodity RNICs?* We answer this question optimistically with Themis, which is deployed on programmable ToR switches that work in conjunction with the RNIC-SR mechanism of commodity RNICs[1]. We classify NACKs triggered by OOO arrivals into three categories: valid NACKs (packet loss confirmed), invalid NACKs (packet already received), and undetermined NACKs (loss status unclear). Themis reconciles the gap between packet-level load balancing and RNIC-SR by enabling destination-side ToR switches to identify and block invalid NACKs while forwarding valid ones.

The key challenge in determining whether a NACK is valid lies in verifying whether the corresponding missing packet has actually been lost. To address this challenge, the core of Themis is a PSN-based packet spraying policy that deterministically assigns each packet to a path based on its sequence number (PSN). This allows the receiver to infer the designated path of a missing packet directly from its PSN. This also offers a unique opportunity for accurate NACK validation: for a missing packet, Themis can determine whether it is lost by checking if other later-sent packets that traverse the same path have already been received, thus confirming whether the corresponding NACK is valid.

Building on this idea, Themis consists of two components: Themis-Source (Themis-S) and Themis-Destination (Themis-D), both deployed on ToR switches. Themis-S enforces the PSN-based packet spraying by modifying packet headers at source-side ToRs (§4.1). Themis-D maintains per-QP states to classify received NACKs as valid, invalid, or undetermined, forwarding valid NACKs while blocking invalid ones (§4.2). For undetermined NACKs, Themis-D applies a lazy dropping scheme that waits for confirmation of their validity (§4.3). Themis also incorporates several optimizations to prevent duplicate drops of retransmitted packets and to handle topology asymmetries caused by link/path failures (§4.4).

We implement a Themis prototype on Tofino2 [12] programmable switches (§5). Testbed experiments (§6.1) with ConnectX-7 RNICs demonstrate that, under packet spraying, Themis reduces tail FCT by 25.2%∼72.5% and 46.5%∼77.8% compared to RNIC-SR without and with congestion control, respectively, and by up to 89.6% compared to the timeout-only mechanism. Simulations conducted in NS-3 (§6.2) further show that, for AI workloads, Themis reduces tail collective completion time (CCT) by 22.7%∼47.3% compared to the direct combination of RNIC-SR and adaptive routing, by 58.5%∼66.5% compared to RNIC-SR with ECMP, and by 22.5%∼39.2% compared to Conweave [50]. Furthermore, Themis successfully remains robust under network failures. When a ToR-to-Spine link failure occurs during collective communication, Themis experiences only 8% tail CCT degradation.

Although our current implementation of Themis is built on programmable switches, its high-level principles and detailed NACK validation logic are general and can be extended to end-host NICs once full transport programmability is available (still rare today, but emerging in some RNIC products). We adopt programmable switches because they are mature, production-ready, and compatible with diverse generations of RNICs, making them a broadly applicable choice.

2 Background and Motivation

2.1 ECMP's Dilemma in AI Workloads

Datacenters often utilize Clos networks [13, 49] as their underlying fabric, which provides multiple equal-cost paths between any source and destination. To distribute traffic across these paths, Equal-Cost Multi-Path (ECMP) [26] is the most widely used load balancing (LB) mechanism [25], which determines the path of a flow by hashing the 5-tuple in the packet header. ECMP works well for traditional workloads, where traffic exhibits a long-tail distribution with a large number of short flows and only a few large flows [15, 17, 19, 47]. However, unlike traditional datacenter workloads, AI workloads, especially training, exhibit traffic patterns that are fundamentally mismatched with ECMP's design: (1) Small number of flows: In an AI training job, each node establishes very few connections, as communication is only required with a limited set of peers. (2) Large flow sizes: The flow sizes typically range from several MBs to hundreds of MBs. (3) Bursty traffic: AI training is an inherently synchronized process, where most nodes enter the communication phase almost simultaneously. This synchronization results in a bursty traffic pattern, with large volumes of data being exchanged in a short period of time.

These flow characteristics (i.e., few in number, large in size, bursty traffic) result in a high ECMP collision rate, as the small number of flows cannot be evenly distributed across available paths, leading to severe performance degradation in AI training workloads [24, 33, 46].

2.2 Commodity RNICs' Packet-Level Load Balancing

Packet-level LB (e.g., random packet spraying [21], adaptive routing, etc.) is a promising solution to address the limitations of ECMP. Packet-level LB evenly distributes traffic across multiple paths, even with few flows. Commodity RNICs, such as ConnectX-6 [8], ConnectX-7 [1] and BlueField3 [7], support adaptive routing (AR) for `RDMA Write` and `RDMA Read` operations, which cooperates with specific switch series [9]. Specifically, when RNICs send data packets, a bit is set in the reserved field of the IB BTH header to

[1]Hereafter, "commodity RNIC" and "RNIC" in this paper refer to current-generation RNICs that support out-of-order packet reception and utilize RNIC-SR as their reliable transport protocol [1, 7, 8].

indicate that the packet should be adaptively routed. Upon receiving such packets, switches choose the port with the minimum queue length among all routable ports leading to the destination. While this method achieves packet-level path selection ability, it inevitably causes out-of-order (OOO) packet arrivals [21, 32].

RNICs' Direct Data Placement capability. To address the OOO packet arrivals resulting from adaptive routing, commodity RNICs [1, 7, 8] support OOO packet reception through the Direct Data Placement (DDP) mechanism [11, 42]. Under DDP, each packet carries an IB RETH header containing the destination memory address, enabling the RNIC to place incoming data directly at the target memory location regardless of arrival order. This capability eliminates the need for packet reordering buffers in the data path.

Timeout-only mechanism. When operating in adaptive routing mode, commodity RNICs still lack an efficient reliability protocol for loss recovery. The current, and most straightforward, solution is to rely on PFC to minimize packet loss and to use timeout-only mechanisms for recovery in extreme cases. However, this approach is ad hoc and fundamentally limited, as packet loss is unavoidable. First, gray failures and packet corruptions are inevitable in large-scale datacenters [55, 57]. For example, prior studies show that the number of packet losses due to corruption is comparable to congestion losses on switch-to-switch links in Microsoft datacenters [62]. Second, lossless RDMA networks have inherent limitations in supporting emerging scenarios, such as long-distance cross-DC communication [22, 51], and therefore cannot serve as a long-term solution. Ultimately, we must embrace PFC-free lossy fabrics [10, 36, 41, 42, 48, 56], where congestion-induced losses occur frequently.

Therefore, this timeout-only reliability approach can cause significant performance degradation in AI workloads [55], since a single packet loss forces all workers to wait, stalling the entire training process. The problem becomes even more severe considering that timeout parameters are often explicitly set to larger values in training workloads to prevent frequent timeouts that lead to NCCL completion errors [33].

To quantify the inefficiency of the timeout-only mechanism, we conduct an experiment using NS-3 with a 16×16 leaf-spine topology with 1:1 subscription. All links operate at 200 Gbps with 1μs delay, and each ToR switch connects 16 NICs for a total of 256 NICs. We divide the 256 NICs into 16 communication groups of 16 NICs each, with each NIC connecting to a different ToR switch. All groups simultaneously start AllReduce with message sizes of 512MB, 1GB, and 2GB. We set RTO to 4ms and introduce different packet loss rates[2] on one of the leaf-to-spine links. Figure 1 shows that the tail collective completion time (CCT) significantly increases with higher packet loss rates.

[2]According to Microsoft's study [62], nearly 50% of links that experience corruption have loss rates above 10^{-5}.

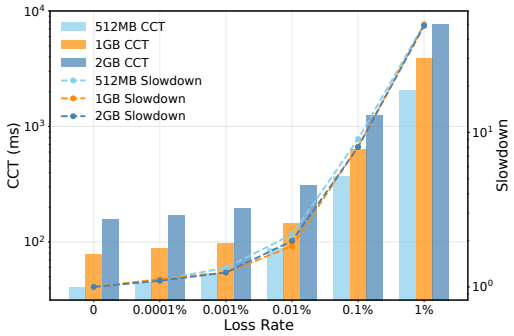


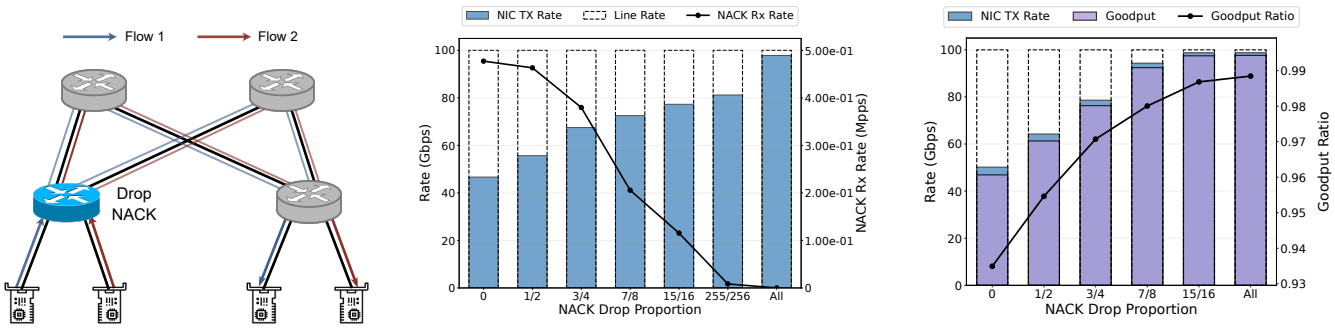
Figure 1. Tail CCT of AllReduce with various loss rates.

2.3 Incompatibility between Packet-Level Load Balancing and RNIC's Selective Repeat

Commodity RNICs also support the selective repeat mechanism (RNIC-SR), which provides much higher loss recovery efficiency than the timeout-only mechanism. However, RNIC-SR was originally designed for single-path transmission and is inherently incompatible with packet-level load balancing. Importantly, the selective repeat logic in commodity RNICs is fixed and cannot be customized by developers. While some SmartNICs [7] allow programmability of CC logic [5], their selective repeat logic remains fixed and non-programmable.

RNIC-SR details. With RNIC-SR, RNICs leverage the same DDP mechanism to handle out-of-order (OOO) packet arrivals caused by packet loss. In addition, RNIC-SR handles retransmissions based on these OOO arrivals as follows:

- The RNIC maintains an expected packet sequence number (ePSN), which indicates the PSN of the next expected packet in sequence. All packets with PSNs smaller than the ePSN have been successfully received. For OOO packets with PSNs larger than the ePSN, the RNIC maintains a bitmap to track their PSNs.
- Upon receiving a packet, the RNIC checks whether its PSN matches the ePSN. If so, the ePSN is updated based on the bitmap: it advances to the smallest PSN for which the corresponding packet has not yet been received.
- If a packet's PSN is larger than the ePSN (i.e., an OOO packet), the RNIC assumes that the packet with the ePSN was lost and generates a Negative Acknowledgment (NACK) to request retransmission of the lost packet. Notably, each ePSN triggers at most one NACK, even if multiple OOO packets arrive. Moreover, once a NACK is triggered for the current expected packet, the next NACK can only be generated after this expected packet is received and the ePSN advances.
- The triggered NACK only carries the ePSN of the receiver, rather than including the PSN of the OOO packet. This design choice does not introduce new packet types, thus maintaining consistency with the Go-Back-N protocol and reducing hardware implementation complexity.



(a) Experiment topology. **(b)** Enabling DCQCN. **(c)** Disabling DCQCN.

Figure 2. Testbed experiment illustrating the impact of unnecessary NACKs on RNIC performance.

- When the sender receives the NACK, it retransmits two packets: the packet indicated by the ePSN carried in the NACK, and the previous packet it just sent.

RNIC-SR performs well in ECMP routing scenarios, where OOO arrivals are caused by packet loss. However, it struggles to handle packet-level LB scenarios effectively. Specifically, when the RNIC receives a packet with PSN > ePSN, it cannot determine whether the packet indicated by the ePSN is actually lost. Instead, it blindly assumes the packet is lost and accordingly generates a NACK. Therefore, many of these NACKs are unnecessary and should not reach the sender.

Quantifying the effect of unnecessary NACKs. To evaluate how unnecessary NACKs affect performance, we conduct a testbed experiment with two senders and two receivers connected using the topology shown in Figure 2a. The sender-side ToR switch is a Tofino2 programmable switch [12], and all senders and receivers are equipped with NVIDIA Mellanox ConnectX-7 (CX7) RNICs with RNIC-SR enabled. We run standard *perftest* [2] to continuously generate traffic between each sender-receiver pair and measure one of the sender RNIC's physical TX rate via *mlnx_perf* [3]. To artificially apply random packet spraying, we configure the sender-side ToR switch to randomly select one of the two paths for each packet. In this scenario, receivers generate NACKs due to OOO packet arrivals, and since no actual packet loss occurs, these NACKs are all unnecessary. We then use the sender-side ToR switch to drop varying proportions of NACKs that come back from receivers to evaluate how unnecessary NACKs influence throughput.

Issue #1: Unnecessary slow starts. When the sender RNIC receives a NACK, it treats this as a signal of packet loss caused by congestion, causing the sender to reduce its transmission rate [50]. This congestion control (CC) response, however, is inappropriate in the context of packet-level load balancing, where OOO arrivals can occur without any congestion. As shown in Figure 2b, with DCQCN [61] enabled (the default CC on CX7 RNIC, with the same parameters as in §6.1), if we do not intervene in the transmission process, the system converges to a state where NACK packets arrive at the sender at approximately 0.48Mpps, causing the sender's transmission rate to drop to less than half the line rate (46.6Gbps). Even when we drop 255/256 of the NACK packets, the rate reduction triggered by the remaining NACKs still limits the sender's transmission rate to only 81% of the line rate.

Issue #2: Retransmissions disrupt TX data path. Even with DCQCN disabled, excessive unnecessary NACKs still cause low transmission rates. As shown in Figure 2c, the sender achieves only 50% of the line rate when all NACKs are passed to the sender. This performance anomaly indicates potential issues with the ConnectX-7 RNICs' retransmission module. A possible reason could be that retransmitted packets are required to be sent as quickly as possible [16], and fetching retransmitted packets may block the normal TX data path[3]. We exclude the possibility that receiver-side out-of-order direct data placement is the performance bottleneck, as the RNIC can achieve line rate when we block all NACKs at the sender-side ToR switch, regardless of whether congestion control is enabled or disabled.

Issue #3: Bandwidth waste due to spurious retransmissions. Unnecessary NACKs trigger the sender to retransmit packets that were not actually lost, resulting in bandwidth waste. To quantify this impact, we define goodput as the transmission rate actually utilized by the application[4] and calculate the goodput ratio as the ratio between goodput and physical transmission rate. Figure 2c shows that when all NACKs are passed to the sender, the goodput ratio drops to 93.5%. This 6.5% bandwidth waste is directly attributed to spurious retransmissions.

[3]Since commodity RNICs are black boxes, we cannot analyze their microarchitecture to identify the root cause of this performance anomaly.
[4]This includes the rate consumed by corresponding packet headers.

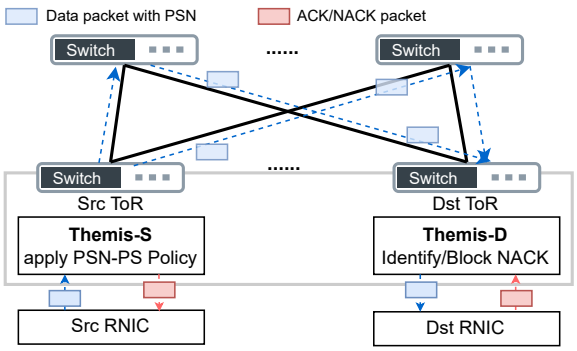


Figure 3. Overview of Themis.

3 Design Overview

We aim to enable packet spraying while preserving efficient loss recovery by leveraging the selective repeat mechanism in commodity RNICs (RNIC-SR) and resolving its incompatibility with packet spraying. To this end, we propose Themis, a middleware deployed on off-the-shelf programmable ToR switches that works in conjunction with the RNIC-SR mechanism. Themis consists of two components: Themis-Src (Themis-S) and Themis-Dst (Themis-D). As illustrated in Figure 3, both components are deployed on ToR switches and work collaboratively to identify and block unnecessary NACKs, thereby reconciling the gap between packet-level load balancing and RNIC-SR.

To distinguish whether a packet is actually lost, Themis exploits the fact that if a received OOO packet traverses the same path as the unreceived expected packet, then the expected packet can be confirmed as lost. However, for any unreceived packet, the receiver knows nothing about which path the packet was supposed to take. The only thing the receiver knows is the packet's sequence number (PSN). Therefore, Themis uses the PSN as the path selection entropy for packet spraying (§4.1). This approach enables the receiver to infer the unreceived packet's designated path directly from its PSN alone, offering Themis-D an opportunity to confirm whether the expected packet indicated by a NACK (NACK.ePSN packet) is actually lost. Themis-D on the receiver-side ToR maintains states for each RDMA queue pair (QP) and uses these states to determine the loss status of the NACK.ePSN packet, thereby classifying received NACKs into one of the following three categories (§4.2):

- **Invalid NACKs:** If the NACK.ePSN packet is subsequently received after the NACK generation, the NACK becomes invalid since retransmission is unnecessary.
- **Valid NACKs:** If the NACK.ePSN packet is confirmed lost by a received OOO packet that was sent later than the NACK.ePSN packet and traverses the same path, the NACK is valid.
- **Undetermined NACKs:** If the NACK.ePSN packet has not been received and no later-sent packet on the same path has been received, packet loss cannot be confirmed and Themis-D classifies the NACK as undetermined.

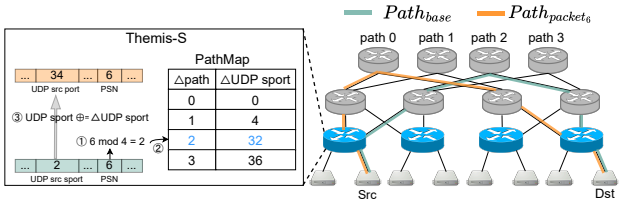


Figure 4. Illustration of PSN-based packet spraying in 3-tier or multi-tier topologies.

Themis-D forwards valid NACKs to the sender to trigger retransmission of the expected packet and drops invalid NACKs to prevent performance degradation. For undetermined NACKs, Themis-D applies a lazy dropping mechanism (§4.3) that waits for confirmation of their validity. Themis also incorporates additional optimizations (§4.4), including retransmission rerouting to avoid timeouts caused by duplicate drops of retransmitted packets and a failure handling scheme to maintain performance robustness when link failures occur.

4 Design Details

4.1 PSN-based Packet Spraying

To enable packet spraying while allowing NACK validation, we propose a PSN-based packet spraying policy (PSN-PS Policy). Specifically, assume there are N equal-cost paths between a source (src) and a destination (dst), indexed as $0, 1, \ldots, N-1$. For a given flow, the ECMP hashing algorithm determines its base path index $P_{\text{base}} \in \{0, 1, \ldots, N-1\}$. Under this policy, every $packet_i$ of this flow with PSN = PSN_i is deterministically assigned to the path:

$$Path_i = (PSN_i \bmod N + P_{\text{base}}) \bmod N \tag{1}$$

This policy ensures deterministic and uniform distribution of packets across all N paths. Based on this policy, we can determine the traveled path of a received OOO packet (with oPSN) and the expected packet (with ePSN) as follows:

$$\begin{aligned} Path_{ooo} &= (oPSN \bmod N + P_{\text{base}}) \bmod N, \\ Path_{expected} &= (ePSN \bmod N + P_{\text{base}}) \bmod N \end{aligned} \tag{2}$$

To determine whether $Path_{ooo} = Path_{expected}$, we only need to check:

$$oPSN \bmod N = ePSN \bmod N \tag{3}$$

Themis-S deployed on the sender-side ToR switch applies this PSN-based packet spraying policy. In a 2-tier Clos network, where path selection is entirely determined by the ToR (leaf) switch, this mechanism can be achieved by allowing the ToR switch to select the egress port for each packet based on its PSN, without involving spine switches.

In 3-tier or multi-tier Clos networks, existing relative path control methods [38, 58] available on commodity switches can be leveraged. For instance, the approach from prior

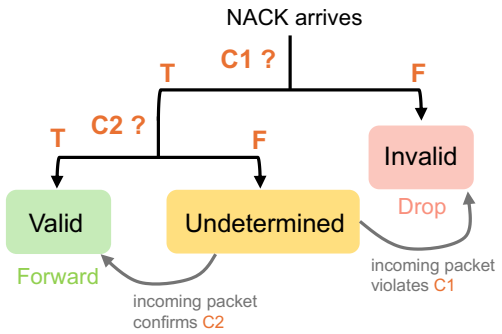


Figure 5. Process of NACK validation at Themis-D.

work [58], which has been successfully deployed in production AI training clusters [46], exploits the linearity of ECMP hashing to construct a deterministic *PathMap* offline. PSN-based packet spraying can be achieved with this PathMap. As illustrated in Figure 4, when a packet arrives at the ToR switch, the ToR calculates its relative path change by PSN mod N (①). The ToR then uses the PathMap to determine the corresponding header modification (②) and applies the modification accordingly (③). This method requires programmability only at the ToR switch.

Notably, Themis is orthogonal to existing relative path control methods. These mechanisms have their own PathMap construction/maintenance schemes and run in the network control plane on the switch's board-management CPU, while Themis primarily operates in the ToR switch data-plane P4 pipeline using the PathMap provided by the control plane. Therefore, they use separate compute and memory resources.

4.2 NACK Validation

Themis-D classifies each incoming NACK as invalid, valid, or undetermined according to the categorization described in §3, and takes corresponding actions for each type. The validation process involves two sequential checks: packet delivery status verification, followed by path-based packet loss confirmation. The complete validation workflow is illustrated in Figure 5, with detailed pseudocode provided in Algorithm 1.

(C1) Packet delivery status verification. When a NACK arrives at the receiver-side ToR, the NACK.ePSN packet may have already reached the RNIC, potentially updating the RNIC's ePSN. The first condition for a valid NACK is that the NACK.ePSN packet has not yet been delivered to the receiver RNIC. Since all packets pass through the ToR switch before reaching the RNIC, the ToR can maintain records of packets successfully transmitted to the receiver RNIC, thereby inferring whether the NACK.ePSN packet has been successfully delivered to the RNIC.

To enable this inference, Themis-D maintains a per-QP *Proxy Bitmap* to record PSNs of packets transmitted to the RNIC. This ring-based bitmap has a size equal to the sender-side RNIC's maximum transmission window[5]. When a data packet completes queuing in the egress queue and is sent to the receiver RNIC, the corresponding bit in the proxy bitmap is set to 1 (Algorithm 1, line 20).

Algorithm 1 NACK validation at Themis-D[6]

```
1:  // bitmap: ProxyBitmap, table: PathMaxPSN Table
2:  Data Structures: bitmap[][], table[][]
3:
4:  procedure onNACK(NACK.ePSN, QP)
5:      // Check Condition C1: Packet delivery status
6:      if bitmap[QP][NACK.ePSN] == True then
7:          return INVALID
8:      end if
9:      // Check Condition C2: Path-based validation
10:     path_index ← NACK.ePSN mod N
11:     if table[QP][path_index] > NACK.ePSN then
12:         return VALID
13:     else
14:         return UNDETERMINED                    ▷ Sec. 4.3
15:     end if
16: end procedure
17:
18: procedure onDataPacket(PSN, QP)
19:     // Update proxy bitmap
20:     bitmap[QP][PSN] ← True
21:     // Update path max PSN
22:     path_index ← PSN mod N
23:     table[QP][path_index] ← max(table[QP][path_index], PSN)
24: end procedure
```

When a NACK arrives at the receiver-side ToR, the validation process first checks the corresponding bit for NACK.ePSN in the QP's bitmap. If the bit is set to 1, indicating that the NACK.ePSN packet has already reached the receiver-side ToR and been delivered to the receiver RNIC, retransmission of this packet is unnecessary, and the NACK is classified as invalid and dropped (Algorithm 1, line 5 to 7). Otherwise, the validation proceeds to the second condition.

(C2) Path-based packet loss confirmation. Passing condition C1 means that the NACK.ePSN packet has not been received yet. The second condition for a valid NACK is that the NACK.ePSN packet has actually been lost. A NACK.ePSN packet is confirmed lost when a later packet (with PSN > NACK.ePSN) on the same path has arrived. To enable this verification, Themis-D maintains a per-QP *PathMaxPSN Table* that records the maximum PSN transmitted to the RNIC from each path.

Before a data packet is transmitted to the receiver RNIC, Themis-D calculates its path index using PSN mod N and updates the corresponding entry in the PathMaxPSN Table if the current PSN is larger (line 22 to 23). For NACK validation, Themis-D computes the path index for NACK.ePSN packet

[5]The maximum transmission window is a static parameter representing the maximum distance between sent packet PSNs and unacknowledged PSNs, configurable in commodity RNICs.

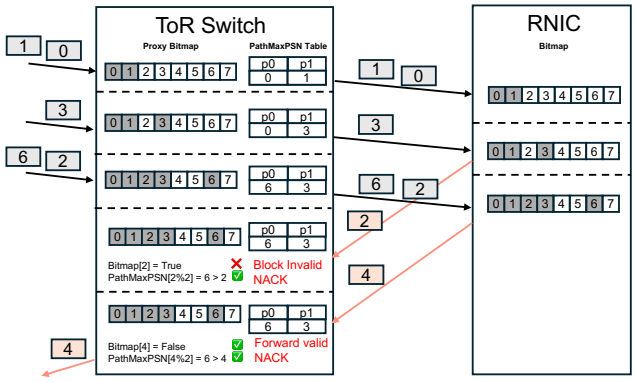


Figure 6. Example of NACK validation.

and checks whether the maximum PSN recorded for that path exceeds NACK.ePSN. If so, NACK.ePSN packet is confirmed lost, and the NACK is classified as valid and forwarded to the sender (line 10 to 12). Otherwise, the NACK is classified as undetermined and handled by the lazy dropping mechanism described in §4.3 (line 14).

Example. As shown in Figure 6, assume there are two paths, with packets having even PSNs sent via Path 0 and packets having odd PSNs sent via Path 1. Data packets with PSNs 0, 1, 3 arrive at the ToR switch and are sent to the RNIC. Before transmission, Themis-D updates the corresponding `bitmap` and `table` entries. When packet 3 arrives at the RNIC, it triggers a NACK with ePSN = 2. When this NACK arrives at the ToR switch, condition C1 is not satisfied since `bitmap[2] = True`, indicating packet 2 has already been sent to the RNIC. Therefore, this NACK is determined to be invalid and dropped.

When the data packet with PSN = 6 arrives at the RNIC, it triggers a NACK with ePSN = 4. When this NACK arrives at the ToR switch, condition C1 is satisfied since `bitmap[4] = False`. For condition C2, the path index is 4 mod 2 = 0, and `table[0]` = 6 > 4 confirms packet 4 is lost. The NACK is determined to be valid and forwarded to the sender.

4.3 Lazy Dropping

Directly dropping undetermined NACKs poses a significant problem: if subsequent packets confirm that the NACK.ePSN packet is indeed lost, the RNIC cannot regenerate a NACK for the same ePSN (§2.3). Consequently, the lost packet will only be retransmitted after a timeout, leading to performance degradation. To address this issue, Themis-D employs a lazy dropping mechanism for undetermined NACKs.

When a NACK is classified as undetermined, instead of dropping it immediately, Themis-D stashes its ePSN for later validation, allowing subsequently arriving data packets to determine whether the stashed NACK is valid or invalid, as illustrated in Figure 5. To achieve this, Themis-D maintains

[6]ProxyBitmap is maintained in a ring buffer manner. For simplicity, the ring buffer maintenance code is omitted here, and bitmap[qp][psn] represents the bit corresponding to PSN in the bitmap.

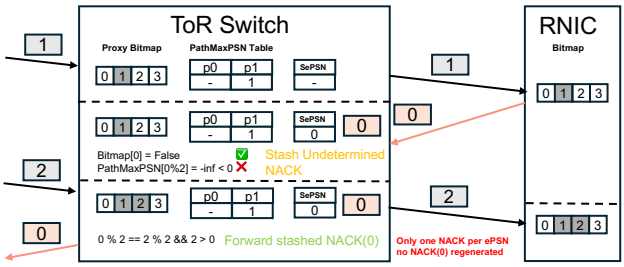


Figure 7. Example of lazy dropping.

one *Stashed ePSN* (SePSN) and a corresponding *Valid* field for each QP. Themis-D sets the SePSN to the NACK.ePSN of the undetermined NACK and marks the *Valid* field as True. The rationale for maintaining only one Stashed ePSN per QP is that RNIC-SR only generates NACKs for the current ePSN. If a NACK with NACK.ePSN > SePSN arrives, it indicates that the earlier NACKed packet (the packet corresponding to SePSN) has been successfully received by the RNIC, thus invalidating the earlier undetermined NACK. Consequently, the RNIC is waiting for the packet indicated by the new NACK.ePSN. The mechanism operates as follows:

- If *Valid* is True and a packet with PSN larger than SePSN arrives such that PSN mod N = SePSN mod N, this indicates that the packet was sent later and traverses the same path as the packet indicated by SePSN (SePSN packet). Themis-D determines that the SePSN packet is lost, and the stashed NACK becomes valid and is forwarded to the sender to trigger retransmission. The *Valid* field is then set to False to prevent generating multiple NACKs for the same ePSN.
- If *Valid* is True and a packet with PSN equal to SePSN arrives, Themis-D determines that the SePSN packet was not lost. The stashed undetermined NACK is then confirmed to be invalid and dropped, with the *Valid* field set to False.

Example. As shown in Figure 7, assume there are two paths, with packets having even PSNs sent via Path 0 and packets having odd PSNs sent via Path 1. Data packet with PSN = 1 arrives at the ToR switch and is sent to the RNIC. Before transmission, Themis-D updates the corresponding `bitmap` and `table` entries. When packet 1 arrives at the RNIC, it triggers a NACK with ePSN = 0. When this NACK arrives at the ToR switch, condition C1 is satisfied since `bitmap[0] = False`, but condition C2 is not satisfied since `table[0%2]` = $-\infty$ < 0. Therefore, this NACK is determined to be undetermined and stashed. Later, when packet 2 arrives at the ToR switch, Themis-D uses Eq. 3 to determine that packet 2 and packet 0 use the same path. Since PSN 2 > SePSN 0, Themis-D confirms that packet 0 is indeed lost and the stashed NACK is valid, then forwards the stashed NACK with ePSN = 0 to trigger retransmission of packet 0.

4.4 Other Optimizations

Retransmission rerouting. Since RNIC-SR generates only one NACK per ePSN, if the retransmitted packet triggered by this NACK is lost again, packet loss recovery can only rely on timeout mechanisms. To minimize timeouts caused by retransmitted packet losses, Themis-S maintains one *lastNackPsn* field for each QP. Upon receiving a NACK, Themis-S updates *lastNackPsn* with the NACK.ePSN value. For each data packet, if its PSN = *lastNackPsn*, Themis-S randomly selects an alternative path for this packet, avoiding the path specified by the PSN-based packet spraying policy. This approach reduces the probability that retransmitted packets will be lost again for the same reasons.

Although rerouting a retransmitted packet to an alternative path breaks its PSN-to-path determinism in Eq. 1, this does not affect Algorithm 1. At the receiver-side ToR, this determinism is only used to update the *PathMaxPSN* table (Algorithm 1, Lines 22–23). Moreover, rerouting is triggered only after the ToR has validated and forwarded the corresponding NACK (ePSN = *lastNackPsn*), at which point $table[\text{QP}][lastNackPsn \bmod N] > lastNackPsn$ already holds. Therefore, $table[\text{QP}][lastNackPsn \bmod N]$ will not be updated by the retransmitted packet regardless of which path it takes (Algorithm 1, line 23), and subsequent NACK validation remains correct.

Failure handling. The PSN-based packet spraying policy (§4.1) ensures balanced path utilization in symmetric topologies. However, network failures (e.g., link flaps) can introduce asymmetry, creating slow or failed paths that disrupt load balancing. To maintain high performance when such asymmetry arises, Themis incorporates a failure handling scheme that detects these problematic paths, temporarily marks them as unavailable, and redistributes packets originally destined for these paths across remaining healthy paths.

Themis-D detects problematic paths by monitoring the OOO degree. Whenever a data packet is received, if the current stashed NACK is valid, Themis-D computes the OOO degree as PSN − SePSN. If this degree exceeds `OOO_THRESHOLD`, it identifies the path corresponding to SePSN mod N as problematic and generates a NACK for the SePSN, bypassing standard validation for undetermined NACKs. A specific reserved bit is then set in the NACK's IB BTH header to designate it as a Path Avoidance Signal. Themis-S maintains a per-QP, per-path avoidance counter. Upon receiving this signal, Themis-S sets the counter for the problematic path to `AVOIDANCE_WINDOW`. During subsequent packet transmission, if the path assigned by the PSN-based packet spray policy for a packet has a non-zero counter, Themis-S randomly selects an alternative path and decrements the counter, temporarily disabling the problematic path until the counter expires.

The packet redistribution used when handling failures breaks the deterministic PSN-to-path mapping used in NACK validation, potentially allowing some invalid NACKs to pass condition C2. However, this impact is partial and affects only packets originally assigned to problematic paths. Overall performance remains robust, as the majority of traffic benefits from accurate NACK validation on healthy paths. Experiments described in §6.3 successfully demonstrate the effectiveness of our failure handling scheme.

5 Implementation

We implement the Themis prototype using P4 on Tofino2 switches [12]. Themis-S runs in the ingress pipeline while Themis-D runs in the egress pipeline. This placement ensures only packets that have completed queuing and are ready for transmission to the RNIC are processed by Themis-D.

Cross-pipeline processing. Tofino2 switches comprise four parallel pipelines for high throughput. However, each pipeline processes packets independently. If a data packet and a NACK packet from the same QP use different pipelines, as shown in Figure 8, they cannot access the same state for Themis's operation. We overcome this challenge by leveraging Tofino2's mirroring feature. For each arriving NACK packet, if its egress port's pipeline is different from its incoming port's pipeline, the ToR switch first routes it back to its incoming port (e.g., Port 0 in Figure 8). The NACK is then processed by Themis-D in the same egress pipeline as data packets from the same QP. After validation, if the NACK is valid, the ToR switch generates a mirrored packet to the target egress port (e.g., Port 31 in Figure 8) while dropping the original NACK. We adopt the same processing approach for ACK packets.

Stashing undetermined NACKs. When a NACK is classified as undetermined, Themis-D does not store the whole NACK packet to conserve switch resources. Instead, it retains only the essential information including the NACK.ePSN and destination QP number (QPN), and then drops the original NACK packet. When a subsequent data packet validates the undetermined NACK, Themis-D uses Tofino2's mirror feature to generate a mirrored packet and sets its packet length to the length of a NACK. When the mirrored packet is re-processed in the egress pipeline, Themis-D modifies its opcode, PSN, QPN in BTH header, packet_len in the IP header, and swaps the source and destination IP and MAC addresses. This transforms the mirrored packet into a NACK. The NACK is then sent back to the sender.

Maintaining bitmap. To accommodate continuously increasing PSNs, bitmaps are usually implemented in a ring manner. However, this requires clearing used bits before reusing them to represent new PSNs, which involves loops with dynamic iteration numbers. Since Tofino2 does not support such dynamic loops to ensure line-rate packet processing, we opt for a double-length variable base bitmap solution (Appendix A) to emulate the behavior of a ring-like structure. This approach eliminates the need to clear used

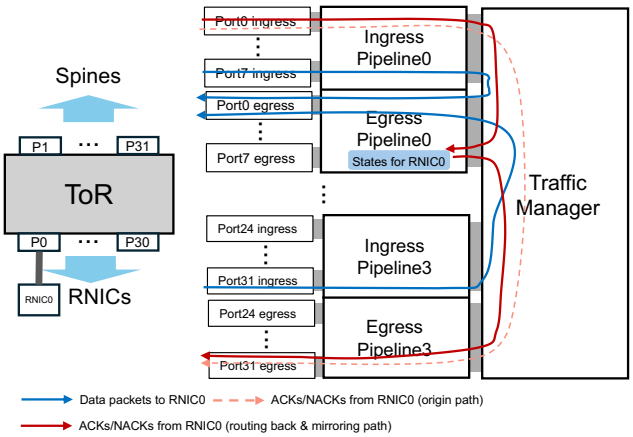


Figure 8. Illustration of cross-pipeline processing via leveraging the mirroring function in Tofino2 switches.

Table 1. Resource consumption of Themis on Tofino2.

Module	Gateway	VLIWs	ALUs	SRAM	Map RAM
Percentage	34.38%	10.24%	37.50%	13.96%	20.95%

bits and instead only requires updating a base bit, at the cost of slightly higher state maintenance.

Resource consumption. Themis's memory consumption at each ToR switch primarily comes from storing per-QP state for cross-rack QPs. For N_{NIC} RNICs with N_{QP} cross-rack QPs each, per-QP state mainly consists of a bitmap (proportional to the transmission window size N_{wnd}) and a PathMaxPSN table (proportional to the number of paths N_{paths}). Total memory consumption is:

$$\text{Mem consumption} \propto N_{\text{NIC}} \times N_{\text{QP}} \times (c_1 \cdot N_{\text{wnd}} + c_2 \cdot N_{\text{paths}}) \quad (4)$$

where c_1 and c_2 are constants determined by implementations. Detailed analysis of memory consumption for each cross-rack QP is provided in the Appendix B.

Table 1 shows the resource consumption when Themis supports 512 cross-rack QPs per RNIC with each QP utilizing 128 paths. Themis requires 34.38% of gateway resources for conditional logic processing, 13.96% of SRAM and 20.95% of Map RAM for storing per-QP states, and 10.24% of VLIWs and 37.50% of ALUs for state maintenance operations. Supporting 512 cross-rack QPs per RNIC is sufficient for AI training scenarios, as training traffic is typically sparse and localized, resulting in low QP counts per GPU [24]. Additionally, 128 paths per QP suffice for ideal load balancing in large-scale AI clusters [39]. Resource consumption around 30% represents a modest overhead. If needed, Themis can support more QPs and paths given the available resource headroom.

6 Evaluation

We evaluate Themis through both testbed experiments with ConnectX-7 RNICs [1] and simulation experiments using NS-3. We compare Themis with existing approaches that combine commodity RNIC reliability mechanisms with load balancing methods. Our comprehensive evaluation reveals the following key findings:

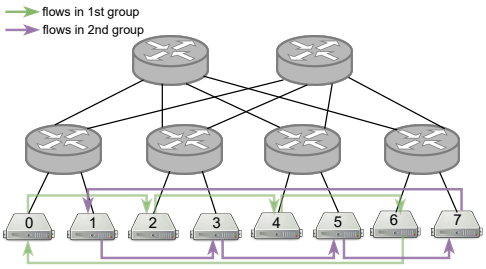


Figure 9. Testbed topologies.

- **Testbed experiments:** Under packet spraying and across different loss rates, Themis reduces tail FCT by 25.2%~72.5% and 46.5%~77.8% compared to RNIC-SR without and with congestion control, respectively. In addition, Themis reduces tail FCT by 25.2%~89.6% compared to timeout-only mechanism when packet loss occurs.
- **Simulation experiments:** Across different loss rates, Themis reduces tail CCT by 22.7%~35.4% and 29.3%~47.3% compared to RNIC-SR with adaptive routing, by 58.5%~65.6% and 58.8%~66.5% compared to RNIC-SR with ECMP, and by 24.3%~33.2% and 22.5%~39.2% compared to Conweave for AllReduce and AllToAll workloads, respectively.
- **Link failure resilience:** When a ToR-to-Spine link failure occurs during collective communication, Themis experiences only 8% CCT degradation compared to failure-free scenarios, demonstrating strong failure resilience.

6.1 Testbed Experiments

Network topology. Figure 9 shows our testbed network topology. We implement the topology by virtualizing one Tofino2 switch as two spine switches while another Tofino2 switch serves as four ToR switches running Themis. All links operate at 100Gbps. One ToR-to-spine link is configured with a specified packet loss rate. Each ToR switch connects to two Mellanox ConnectX-7 RNICs [1].

Workload. The topology consists of eight RNICs, where RNICs {0, 2, 4, 6} form one group and RNICs {1, 3, 5, 7} form another group. Each RNIC performs an `RDMA Write` operation with 256MB of data to the next RNIC within the same group. This creates a ring permutation traffic pattern for each group, which is common in collective communications.

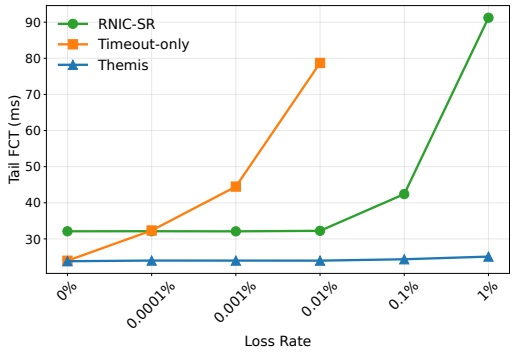


Figure 10. Tail FCT under different loss rates without CC.

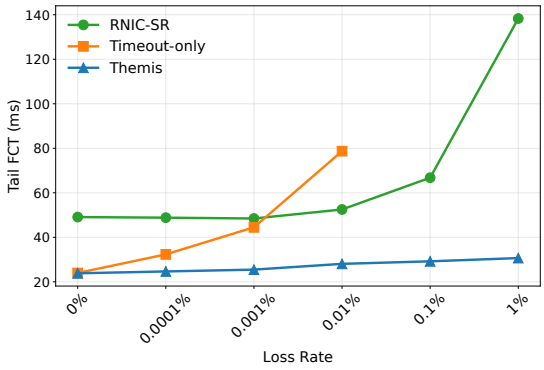


Figure 11. Tail FCT under different loss rates with DCQCN.

Schemes compared. We compare Themis with two baselines—CX7 RNIC's RNIC-SR and the timeout-only mechanism—both evaluated under switch-supported packet spraying[7]. The timeout value is set to 4ms. All schemes use the CX7 RNIC's default TX window size of 512 packets.

Congestion control. We evaluate Themis under both disabled and enabled congestion control scenarios. When congestion control is enabled, we use DCQCN [61], the standard congestion control scheme for commodity RNICs, with $(K_{min}, K_{max}, P_{max}) = (400KB, 1600KB, 0.2)$ as recommended in [37]. For RNIC parameters, we follow the recommendations in [45], which are specifically tuned for AI training workloads.

Performance metrics. We use the slowest flow's completion time (i.e., Tail FCT) as the primary metric.

Figures 10 and 11 show that across different loss rates, Themis reduces tail FCT by 25.2%~72.5% compared to RNIC-SR without congestion control and 46.5%~77.8% compared to RNIC-SR with congestion control enabled. When there is no packet loss, this improvement stems from Themis blocking invalid NACKs, avoiding the performance degradation caused by invalid NACKs (Section 2.3).

[7]We use Tofino2 to implement a packet spraying strategy to validate the RNIC-SR and timeout-only mechanisms. Since the commodity RNIC-AR mode can only be enabled when the RNIC is connected to specific switch series [9] (excluding Tofino 1/2), we emulate the timeout-only mechanism by configuring the Tofino2 switch to drop all NACKs.

The timeout-only mechanism achieves similar performance to Themis when there is no packet loss. However, when packet loss occurs, its performance degradation is severe. At a packet loss rate of 0.001%, the tail FCT of the timeout-only mechanism is 3.28× larger than Themis. Therefore, we do not plot the results for higher loss rates.

RNIC-SR shows consistent performance when the packet loss rate is below 0.1%, as losses can be recovered through NACK-triggered retransmissions. However, when the packet loss rate reaches 0.1% or higher, its performance becomes even worse because retransmitted packets are also lost, forcing the RNIC to fall back to timeout-based loss recovery. Themis avoids this performance degradation by identifying valid NACKs and performing retransmission rerouting to avoid the loss of retransmitted packets (§4.4). Consequently, at 1% packet loss rate, Themis reduces tail FCT by 72.5% and 77.8% compared to RNIC-SR without and with CC, respectively.

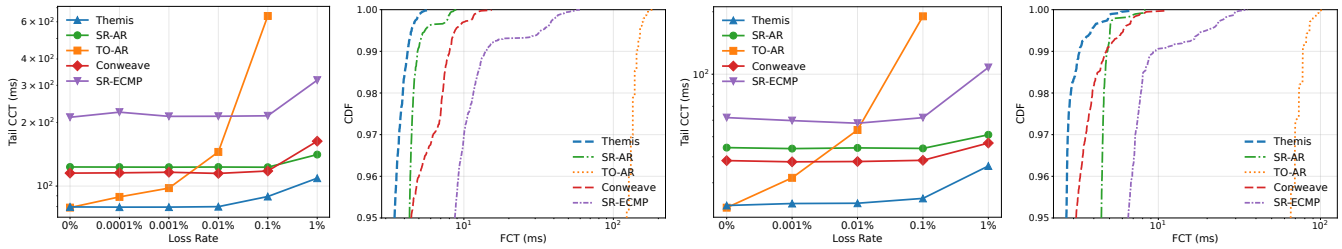
6.2 Simulation Experiments

Network topology. We use NS-3 simulator to evaluate Themis in a 16 × 16 leaf-spine topology with 1:1 oversubscription ratio. All links operate at 200Gbps with 1μs propagation delay, and each switch is equipped with a 64MB buffer [12]. Each of the 16 ToR switches connects to 16 NICs, resulting in a total of 256 NICs (representing 256 GPUs). One ToR-to-spine link is configured with a specified packet loss rate.

Workloads. We evaluate Themis using the most commonly used collective communication workloads: AllReduce and AllToAll. The 256 NICs are divided into 16 communication groups, each containing 16 NICs, with each NIC in a group connected to a different ToR switch. In our experiments, all 16 groups simultaneously initiate the same collective communication. The total traffic for one AllReduce operation is 1GB, and for AllToAll is 512MB.

Schemes compared. We compare Themis against RNIC-SR with switch adaptive routing (SR-AR), RNIC-SR with ECMP (SR-ECMP), timeout-only with switch adaptive routing (TO-AR), and Conweave [50], another in-network load balancing scheme for commodity RNICs, using its default parameter settings. For adaptive routing schemes (SR-AR and TO-AR), switches select the port with minimum queue length among all available ports. All schemes use the CX7 RNIC's default TX window size of 512 packets. The timeout value is set to 4ms. For Themis, we set `OOO_THRESHOLD` to 448 and `AVOIDANCE_WINDOW` to 2×10^6.

Congestion control. We use DCQCN [61] as the congestion control protocol with RNIC parameters following the same settings as our hardware testbed evaluation. For switch-side RED parameters, we configure $(K_{min}, K_{max}, P_{max}) = (800KB, 3200KB, 0.2)$ as recommended in [37] for 200Gbps link capacity.



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

Figure 12. Comparison between Themis, SR-AR, TO-AR, SR-ECMP, and Conweave under AllReduce and AllToAll workloads.

Performance metrics. We use the slowest group's collective completion time (tail CCT) as our primary metric, which reflects the communication bottleneck of training jobs.

Figure 12 shows the performance of different schemes under varying packet loss rates on the specified link. Across different loss rates, Themis consistently outperforms all baseline schemes. For the AllReduce workload, Themis reduces tail CCT by 22.7%~35.4%, 58.5%~65.6%, and 24.3%~33.2% compared to SR-AR, SR-ECMP, and Conweave, respectively. For the AllToAll workload, the performance improvements are 29.3%~47.3%, 58.8%~66.5%, and 22.5%~39.2%, respectively.

When there is no packet loss, TO-AR achieves similar performance as Themis since both distribute packets evenly across all available paths. SR-AR suffers from unnecessary slow starts and bandwidth waste caused by invalid NACKs. SR-ECMP experiences inefficient flow-level load balancing that leads to uneven path utilization. While Conweave performs better than SR-ECMP by rerouting flows upon detecting congestion, it remains less efficient than packet-level load balancing approaches.

When packet loss occurs, TO-AR's performance degrades significantly due to frequent timeouts. Themis continues to outperform other methods as it can achieve packet-level load balancing while maintaining efficient packet loss recovery capability. At a loss rate of 1%, Themis reduces tail CCT by 22.7%, 65.6%, and 33.2% compared to SR-AR, SR-ECMP, and Conweave under the AllReduce workload (Figure 12a), and 29.3%, 66.5%, and 22.5% compared to SR-AR, SR-ECMP, and Conweave under the AllToAll workload (Figure 12c).

As shown in Figures 12b and 12d, Themis achieves the best tail FCT for individual flows at a 1% loss rate, which explains why Themis achieves superior CCT performance. This is because AI workloads are synchronized, meaning that if even one flow is delayed, it impacts the entire collective communication performance.

6.3 Deep Dive

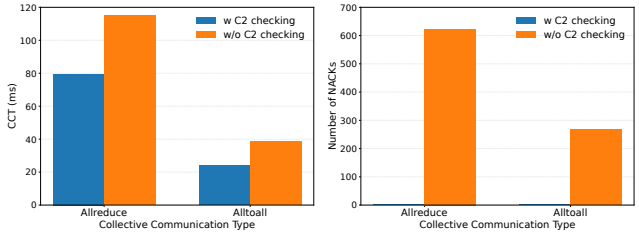
We conduct ablation studies using the same simulation settings as in §6.2 to evaluate the effectiveness of Themis's key components.

Path-based packet loss confirmation. We evaluate a variant of Themis without path-based packet loss confirmation (C2 checking in §4.2). In this configuration, Themis only maintains a bitmap to track packet delivery status, and any NACK passing the C1 condition check is considered valid and forwarded to the sender. Under a 0.001% packet loss rate, disabling C2 checking results in 44.7% and 62.7% CCT degradation for AllReduce and AllToAll workloads, respectively (Figure 13a). Figure 13b shows the number of NACKs received by the QP that receives the most NACKs, demonstrating that the performance degradation is caused by the significant increase in receiving unnecessary NACKs.

Lazy dropping. We evaluate Themis without the lazy dropping, where undetermined NACKs are immediately dropped rather than temporarily stashed. Under a 1% packet loss rate, disabling lazy drop results in 16.9% and 78.3% CCT degradation for AllReduce and AllToAll workloads, respectively (Figure 14a). Figure 14b demonstrates that this performance loss is caused by increased timeouts resulting from prematurely dropping undetermined NACKs. Note that the timeouts observed in the AllToAll workload with lazy drop enabled are due to tail packet losses.

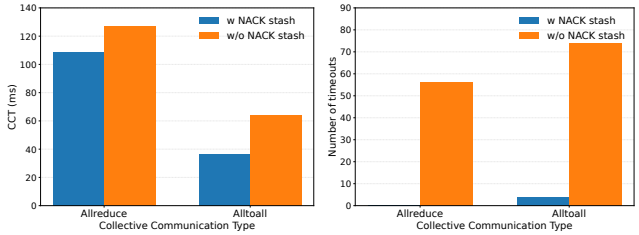
Retransmission rerouting. We evaluate Themis without retransmission rerouting, where retransmitted packets follow the same PSN-based packet spray policy as original packets instead of being randomly sent to alternative paths. Under a 1% packet loss rate, without this mechanism, retransmitted packets face a higher risk of being lost again on the same paths. Disabling retransmission rerouting results in 57.2% and 47.7% CCT degradation for AllReduce and AllToAll workloads, respectively (Figure 15a). Figure 15b shows that this degradation is caused by increased timeouts due to retransmitted packet losses.

Failure handling. We evaluate Themis's failure handling scheme under link failure scenarios by triggering a link down event on one ToR-to-Spine link during collective communication. With the failure handling scheme, Themis experiences only 8% CCT degradation compared to the scenario with no link failures (Figure 16a). Figure 16b shows the average throughput over time of all uplink ports connecting the affected ToR switch to spine switches. When the link failure occurs, all flows are impacted and throughput drops dramatically. The failure handling scheme then quickly takes effect to avoid continued use of the failed path, allowing throughput to rapidly recover and remain stable.



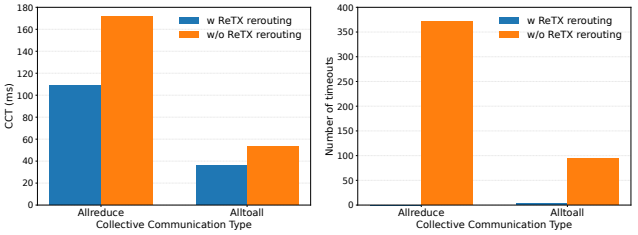
(a) CCT with/without path-based loss confirmation. **(b)** NACK count with/without path-based loss confirmation.

Figure 13. Effectiveness of path-based loss confirmation.



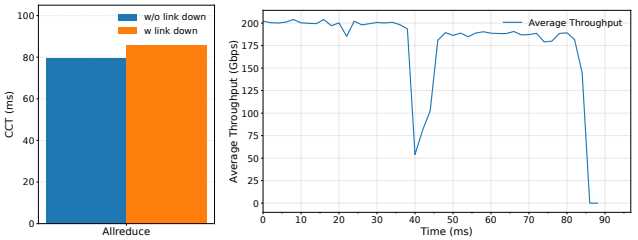
(a) CCT with/without lazy dropping. **(b)** Timeout count with/without lazy dropping.

Figure 14. Effectiveness of lazy dropping.



(a) CCT with/without retransmission rerouting. **(b)** Timeout count with/without retransmission rerouting.

Figure 15. Effectiveness of retransmission rerouting.



(a) CCT with/without link failure. **(b)** Port throughput recovery with failure handling scheme.

Figure 16. Efficiency of failure handling scheme.

7 Discussion

Integration with upper-layer applications. Themis adopts commodity RNICs that ensure in-order message notification (CQE) even under out-of-order reception scenarios, meaning that the CQE is delivered to the application only after all preceding data has been successfully received [42, 56]. Therefore, Themis is fully compatible with existing upper-layer applications without requiring any header changes. Taking NCCL as an example, NCCL supports packet spraying by posting one or more `RDMA Write` operations that directly push data into the destination node's buffer, followed by a zero-payload `RDMA Write with immediate` for completion notification [31]. Themis operates on the `RDMA Write` packets without affecting the notification semantics of `RDMA Write with immediate`, and does not spray packets from other RDMA operations.

Last-hop loss. When data packets have completed queuing at the destination ToR switch and are being sent to the RNIC, they may still be lost due to data corruption or slow receiver issues. In such cases, Themis cannot correctly validate the corresponding NACKs as valid, and must rely on timeouts as a backup mechanism for packet loss recovery. However, since the last-hop distance is typically short, the probability of data corruption is relatively low. Moreover, we recommend enabling PFC on the ToR–RNIC link when deploying Themis. This converts RNIC buffer overflows into ToR congestion, and NACKs from such congestion losses are what Themis can correctly classify and quickly recover via selective repeat. Themis primarily focuses on recovering from packet corruption on switch-to-switch links and congestion-induced packet drops at switches, which occur with higher probability.

Cases for false negatives/positives. For condition C1, misclassification only occurs when there is last-hop loss. In that case, Themis-D incorrectly classifies a valid NACK from last-hop loss as invalid (false negative), forcing Themis to rely on a timeout for recovery.

For condition C2, misclassification only occurs when failure handling is active. In that mode, some invalid NACKs may be classified as valid (false positives). However, as discussed in §4.4, failure handling is path-granular and only affects NACKs whose ePSN maps to the problematic path, so overall performance remains robust as shown in §6.3.

Topology assumptions. Themis performs best in symmetric topologies where all paths between a source-destination pair have the same link capacity, which are common in practice. In clusters with heterogeneous link capacities across paths, failure handling may be falsely triggered, leading to false positive NACKs and reducing Themis's benefits.

Themis on SmartNIC. While Themis is designed to exploit the capabilities of commodity programmable switches to reconcile the gap between packet-level load balancing and RNIC-SR, its key idea of PSN-based packet spraying can also be implemented at the endpoint. However, widely deployed SmartNICs [6, 7] cannot support per-packet path assignment and NACK logic modification due to limited transport-layer programmability. Although some emerging programmable RNICs, such as AMD Pensando DPU [4], can support full transport-layer programmability, they still require time for large-scale deployment and performance stability validation.

Tofino-family switches have been widely deployed in production for years. Although Intel has stopped developing new Tofino chips, it continues to sell and support existing Tofino-family ASICs, and many cloud providers still operate large Tofino deployments in production [44]. Moreover, other vendors continue to develop new P4 architectures [60]. Therefore, Themis on programmable switches is more general and practical for widely deployed RNICs.

8 Related Work

In-network reordering. Conweave [50] reroutes flows away from congested links, and uses destination-side ToR switches to perform in-network reordering of packets when performing rerouting. However, the in-network reordering approach cannot support packet-level LB, as reordering packets from multiple paths exceeds the resource capacity of ToR switches. Some works [28, 29, 32] use switches to select a path for an arriving packet based on the queuing delay of the last-arriving packet in the same flow. These methods attempt to preserve packet order while achieving packet-level load balancing. However, such approaches cannot guarantee that packets always arrive in order.

Load balancing in datacenters. Some works [14, 34, 54] propose flowlet-based LB to preserve packet order by trading off the granularity of load balancing. It relies on hosts creating time gaps within flows to form flowlets. However, RNICs, which use hardware-based rate pacing, cannot produce sufficiently large time gaps, rendering flowlet-based approaches incompatible with RNICs.

Some works [53] propose software-based RDMA load balancing, which creates multiple backend QPs per user QP and splits user-posted WQEs into smaller ones, posting them to different backend QPs. However, splitting large WQEs into packet-sized ones incurs non-negligible software overhead, making fine-grained packet-level LB difficult to achieve. Moreover, commodity RNICs have connection scalability issues [56] due to limited on-chip SRAM. Maintaining multiple backend QPs per user QP increases per-connection state and, at scale, can exceed the SRAM budget, leading to performance degradation. Themis therefore focuses on a hardware-offload design.

Other works [35, 40, 41, 48] propose new transport protocols to enable load balancing for RDMA networks. However, while cloud providers can develop custom ASICs incorporating these protocols, such specialized hardware is predominantly deployed for internal infrastructure rather than offered as general-purpose commodities. Although FPGA-based SmartNICs have seen adoption in some cloud environments [23], integrating and validating new transport protocols on these devices requires hardware developers with domain expertise. Most AI training clusters continue to utilize commodity RNICs [24, 30, 33, 46, 59] due to their proven performance stability and commercial availability. Yet, these widely deployed commodity RNICs and SmartNICs lack the full transport-layer programmability to implement these protocols without hardware modifications (e.g., NVIDIA BF3 only allows programming of the CC portion of RDMA transport [5]), making these transport protocols difficult to integrate into existing clusters.

Packet coding. Some works [27, 52] propose packet coding techniques, where the sender transmits random linear combinations of packets with controlled redundancy to tolerate packet loss without stalling the flow. However, these designs require adding a network-coding layer below the transport layer. This is difficult to achieve with commodity RNICs, where the transport layer is offloaded to hardware, making it infeasible to add the network-coding layer underneath.

9 Conclusion

Themis is a middleware deployed on ToR switches that applies PSN-based packet spraying at the source ToR switch while identifying and blocking invalid NACKs at the destination ToR switch. By avoiding the performance degradation caused by invalid NACKs while preserving fast packet loss recovery capability, Themis enables efficient packet spraying with commodity RNICs. Evaluation results show its superior performance over existing schemes.

Acknowledgments

We thank the anonymous EuroSys reviewers and our shepherd, Prof. Saksham Agarwal, for their constructive suggestions. This work is supported in part by the Hong Kong RGC TRS T41- 603/20R. Wenxue Li and Kai Chen are the corresponding authors.

References

[1] 2021. Mellanox ConnectX-7 Product Brief. https://www.nvidia.com/content/dam/en-zz/Solutions/networking/ethernet-adapters/connectx-7-datasheet-Final.pdf.

[2] 2021. OFED Perftest. https://github.com/linux-rdma/perftest/.

[3] 2024. Mellanox userland tools and scripts. https://github.com/Mellanox/mlnx-tools.

[4] 2025. AMD Pensando Pollara 400 AI NIC. https://www.amd.com/en/products/network-interface-cards/pensando.html.

[5] 2025. DOCA PCC. https://docs.nvidia.com/doca/sdk/doca+pcc/index.html.

[6] 2025. Mellanox BlueField-2 Product Brief. https://resources.nvidia.com/en-us-accelerated-networking-resource-library/bluefield-2-dpu-datasheet.

[7] 2025. Mellanox BlueField-3 Product Brief. https://resources.nvidia.com/en-us-accelerated-networking-resource-library/datasheet-nvidia-bluefield.

[8] 2025. Mellanox ConnectX-6 Product Brief. https://www.nvidia.com/en-sg/networking/ethernet/connectx-6/.

[9] 2025. Mellanox ConnectX-7 Adaptive Routing Mode. https://www.nvidia.com/en-us/networking/spectrumx/.

[10] 2025. Zero touch RoCE enables RoCE to operate on fabrics where no PFC nor ECN are configured. https://docs.nvidia.com/networking/display/winof2v25150020/ethernet+network.

[11] RFC 4296. 2005. The Architecture of Direct Data Placement (DDP) and Remote Direct Memory Access (RDMA) on Internet Protocols.

[12] Anurag Agrawal and Changhoon Kim. 2020. Intel Tofino2 – A 12.9Tbps P4-Programmable Ethernet Switch. In *2020 IEEE Hot Chips 32 Symposium (HCS)*. 1–32. https://doi.org/10.1109/HCS49909.2020.9220636

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

[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* (New Delhi, India) *(SIGCOMM '10)*. Association for Computing Machinery, New York, NY, USA, 63–74. https://doi.org/10.1145/1851182.1851192

[16] Mina Tahmasbi Arashloo, Alexey Lavrov, Manya Ghobadi, Jennifer Rexford, David Walker, and David Wentzlaff. 2020. Enabling Programmable Transport Protocols in High-Speed NICs. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA, 93–109. https://www.usenix.org/conference/nsdi20/presentation/arashloo

[17] Wei Bai, Li Chen, Kai Chen, Dongsu Han, Chen Tian, and Hao Wang. 2015. Information-Agnostic Flow Scheduling for Commodity Data Centers. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. USENIX Association, Oakland, CA, 455–468. https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/bai

[18] Jiamin Cao, Yu Guan, Kun Qian, Jiaqi Gao, Wencong Xiao, Jianbo Dong, Binzhang Fu, Dennis Cai, and Ennan Zhai. 2024. Crux: GPU-Efficient Communication Scheduling for Deep Learning Training. In *Proceedings of the ACM SIGCOMM 2024 Conference* (Sydney, NSW, Australia) *(ACM SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3651890.3672239

[19] Li Chen, Justinas Lingys, Kai Chen, and Feng Liu. 2018. AuTO: scaling deep reinforcement learning for datacenter-scale automatic traffic optimization. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (Budapest, Hungary) *(SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 191–205. https://doi.org/10.1145/3230543.3230551

[20] DeepSeek-AI and Aixin Liu et al. 2024. DeepSeek-V3 Technical Report. arXiv:2412.19437 [cs.CL] https://arxiv.org/abs/2412.19437

[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*. 2130–2138. https://doi.org/10.1109/INFCOM.2013.6567015

[22] Haotian Dong, Jingyan Jiang, Rongwei Lu, Jiajun Luo, Jiajun Song, Bowen Li, Ying Shen, and Zhi Wang. 2025. Beyond A Single AI Cluster: A Survey of Decentralized LLM Training. *arXiv preprint arXiv:2503.11023* (2025).

[23] Daniel Firestone and Andrew Putnam et al. 2018. Azure Accelerated Networking: SmartNICs in the Public Cloud. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. USENIX Association, Renton, WA, 51–66. https://www.usenix.org/conference/nsdi18/presentation/firestone

[24] 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* (Sydney, NSW, Australia) *(ACM SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA, 57–70. https://doi.org/10.1145/3651890.3672233

[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* (Florianopolis, Brazil) *(SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 202–215. https://doi.org/10.1145/2934872.2934908

[26] C. Hopps. 2000. RFC2992: Analysis of an Equal-Cost Multi-Path Algorithm.

[27] Jinbin Hu, Jiawei Huang, Wenjun Lv, Yutao Zhou, Jianxin Wang, and Tian He. 2018. CAPS: Coding-based Adaptive Packet Spraying to Reduce Flow Completion Time in Data Center. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 2294–2302. https://doi.org/10.1109/INFOCOM.2018.8486354

[28] 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 *Proceedings of the 2025 USENIX Conference on Usenix Annual Technical Conference* (Boston, MA, USA) *(USENIX ATC '25)*. USENIX Association, USA, Article 22, 16 pages.

[29] Jinbin Hu, Chaoliang Zeng, Zilong Wang, Junxue Zhang, Kun Guo, Hong Xu, Jiawei Huang, and Kai Chen. 2023. Enabling Load Balancing for Lossless Datacenters. In *2023 IEEE 31st International Conference on Network Protocols (ICNP)*. 1–11. https://doi.org/10.1109/ICNP59255.2023.10355615

[30] Qinghao Hu, Zhisheng Ye, Zerui Wang, Guoteng Wang, Meng Zhang, Qiaoling Chen, Peng Sun, Dahua Lin, Xiaolin Wang, Yingwei Luo, et al. 2024. Characterization of Large Language Model Development in the Datacenter. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Santa Clara, CA, 709–729. https://www.usenix.org/conference/nsdi24/presentation/hu

[31] Zhiyi Hu, Siyuan Shen, Tommaso Bonato, Sylvain Jeaugey, Cedell Alexander, Eric Spada, James Dinan, Jeff Hammond, and Torsten Hoefler. 2025. Demystifying NCCL: An In-depth Analysis of GPU Communication Protocols and Algorithms. arXiv:2507.04786 [cs.DC] https://arxiv.org/abs/2507.04786

[32] Jiawei Huang, Wenjun Lv, Weihe Li, Jianxin Wang, and Tian He. 2018. QDAPS: Queueing Delay Aware Packet Spraying for Load Balancing in Data Center. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. 66–76. https://doi.org/10.1109/ICNP.2018.00017

[33] Ziheng Jiang, Haibin Lin, Yinmin Zhong, Qi Huang, Yangrui Chen, Zhi Zhang, Yanghua Peng, Xiang Li, Cong Xie, Shibiao Nong, et al. 2024. MegaScale: Scaling Large Language Model Training to More Than 10,000 GPUs. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Santa Clara, CA, 745–760. https://www.usenix.org/conference/nsdi24/presentation/jiang-ziheng

[34] 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.

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

[36] Wenxue Li, Xiangzhou Liu, Yunxuan Zhang, Zihao Wang, Wei Gu, Tao Qian, Gaoxiong Zeng, Shoushou Ren, Xinyang Huang, Zhenghang Ren, et al. 2025. Revisiting RDMA Reliability for Lossy Fabrics. In *Proceedings of the ACM SIGCOMM 2025 Conference* (São Francisco Convent, Coimbra, Portugal) *(SIGCOMM '25)*. Association for Computing Machinery, New York, NY, USA, 85–98. https://doi.org/10.1145/3718958.3750480

[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* (Beijing, China) *(SIGCOMM '19)*. Association for Computing Machinery, New York, NY, USA, 44–58. https://doi.org/10.1145/3341302.3342085

[38] Yadong Liu, Yunming Xiao, Xuan Zhang, Weizhen Dang, Huihui Liu, Xiang Li, Zekun He, Jilong Wang, Aleksandar Kuzmanovic, Ang Chen, et al. 2025. Unlocking ECMP Programmability for Precise Traffic Control. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. USENIX Association, Philadelphia, PA, 87–106. https://www.usenix.org/conference/nsdi25/presentation/liu-yadong

[39] Jie Lu, Jiaqi Gao, Fei Feng, Zhiqiang He, Menglei Zheng, Kun Liu, Jun He, Binbin Liao, Suwei Xu, Ke Sun, et al. 2025. Alibaba Stellar: A New Generation RDMA Network for Cloud AI. In *Proceedings of the ACM SIGCOMM 2025 Conference* (São Francisco Convent, Coimbra, Portugal) *(SIGCOMM '25)*. Association for Computing Machinery, New York, NY, USA, 453–466. https://doi.org/10.1145/3718958.3750539

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

[41] 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* (Amsterdam, Netherlands) *(SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 753–766. https://doi.org/10.1145/3544216.3544238

[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] OpenAI and Josh Achiam et al. 2024. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL] https://arxiv.org/abs/2303.08774

[44] Tian Pan, Kun Liu, Xionglie Wei, Yisong Qiao, Jun Hu, Zhiguo Li, Jun Liang, Tiesheng Cheng, Wenqiang Su, Jie Lu, et al. 2024. LuoShen: A Hyper-Converged Programmable Gateway for Multi-Tenant Multi-Service Edge Clouds. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Santa Clara, CA, 877–892. https://www.usenix.org/conference/nsdi24/presentation/pan

[45] Yajuan Peng, Zicheng Wang, Yang Bai, Zhuo Jiang, Jianxi Ye, Xiaoliang Wang, Haoran Wei, Xiaolong Zhong, Junkai Huang, Haohan Xu, et al. 2025. Barre: Empowering Simplified and Versatile Programmable Congestion Control in High-Speed AI Clusters. In *Proceedings of the 2025 USENIX Annual Technical Conference (USENIX ATC '25)*. USENIX Association, Boston, MA, USA. https://www.usenix.org/conference/atc25/presentation/peng-yajuan Open access; co-located with OSDI '25.

[46] Kun Qian, Yongqing Xi, Jiamin Cao, Jiaqi Gao, Yichi Xu, Yu Guan, Binzhang Fu, Xuemei Shi, Fangbo Zhu, Rui Miao, et al. 2024. Alibaba HPN: A Data Center Network for Large Language Model Training. In *Proceedings of the ACM SIGCOMM 2024 Conference* (Sydney, NSW, Australia) *(ACM SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA, 691–706. https://doi.org/10.1145/3651890.3672265

[47] 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* (London, United Kingdom) *(SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 123–137. https://doi.org/10.1145/2785956.2787472

[48] 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.

[49] Arjun Singh and Joon et al. Ong. 2015. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) *(SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 183–197. https://doi.org/10.1145/2785956.2787508

[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* (New York, NY, USA) *(ACM SIGCOMM '23)*. Association for Computing Machinery, New York, NY, USA, 816–831. https://doi.org/10.1145/3603269.3604849

[51] Foteini Strati, Paul Elvinger, Tolga Kerimoglu, and Ana Klimovic. 2024. ML Training with Cloud GPU Shortages: Is Cross-Region the Answer?. In *Proceedings of the 4th Workshop on Machine Learning and Systems* (Athens, Greece) *(EuroMLSys '24)*. Association for Computing Machinery, New York, NY, USA, 107–116. https://doi.org/10.1145/3642970.3655843

[52] J. K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, and J. Barros. 2009. Network Coding Meets TCP. In *IEEE INFOCOM 2009*. 280–288. https://doi.org/10.1109/INFCOM.2009.5061931

[53] Feng Tian, Yang Zhang, Wei Ye, Cheng Jin, Ziyan Wu, and Zhi-Li Zhang. 2021. Accelerating Distributed Deep Learning using Multi-Path RDMA in Data Center Networks. In *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)* (Virtual Event, USA) *(SOSR '21)*. Association for Computing Machinery, New York, NY, USA, 88–100. https://doi.org/10.1145/3482898.3483363

[54] 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.

[55] Hao Wang, Han Tian, Jingrong Chen, Xinchen Wan, Jiacheng Xia, Gaoxiong Zeng, Wei Bai, Junchen Jiang, Yong Wang, and Kai Chen. 2024. Towards Domain-Specific Network Transport for Distributed DNN Training. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Santa Clara, CA, 1421–1443. https://www.usenix.org/conference/nsdi24/presentation/wang-hao

[56] 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)*. USENIX Association, Boston, MA, 1–14. https://www.usenix.org/conference/nsdi23/presentation/wang-zilong

[57] 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* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 253–266. https://doi.org/10.1145/3098822.3098841

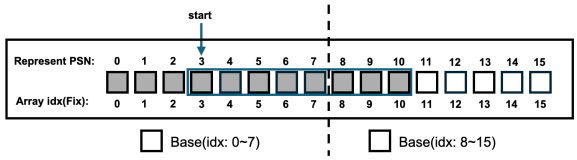
[58] Zhehui Zhang, Haiyang Zheng, Jiayao Hu, Xiangning Yu, Chenchen Qi, Xuemei Shi, and Guohui Wang. 2021. Hashing Linearity Enables Relative Path Control in Data Centers. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, 855–862. https://www.usenix.org/conference/atc21/presentation/zhang-zhehui

[59] Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Huazuo Gao, Jiashi Li, Liyue Zhang, Panpan Huang, Shangyan Zhou, Shirong Ma, et al. 2025. Insights into DeepSeek-V3: Scaling Challenges and Reflections on Hardware for AI Architectures. arXiv:2505.09343 [cs.DC] https://arxiv.org/abs/2505.09343

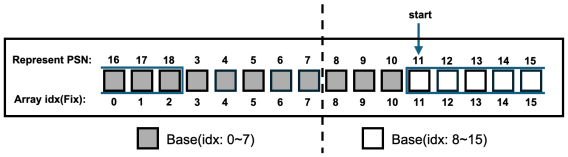
[60] Hao Zheng, Xin Yan, Wenbo Li, Jiaqi Zheng, Xiaoliang Wang, Qingqing Zhao, Luyou He, Xiaofei Lai, Feng Gao, Fuguang Huang, et al. 2025. When P4 Meets Run-to-completion Architecture. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. USENIX Association, Philadelphia, PA, 1487–1505. https://www.usenix.org/conference/nsdi25/presentation/zheng-hao

[61] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. 2015. Congestion Control for Large-Scale RDMA Deployments. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) *(SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 523–536. https://doi.org/10.1145/2785956.2787484

[62] Danyang Zhuo, Monia Ghobadi, Ratul Mahajan, Klaus-Tycho Förster, Arvind Krishnamurthy, and Thomas Anderson. 2017. Understanding and Mitigating Packet Corruption in Data Center Networks. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 362–375. https://doi.org/10.1145/3098822.3098849



(a) Start pointer at first half.



(b) Start pointer at second half.

Figure 17. Illustration of double-length variable base bitmap.

Appendix

A Maintaining bitmap.

To eliminate the need to clear used bits in a ring-based bitmap, we adopt a double-length variable base bitmap solution. As illustrated in Figure 17, the entire bitmap is divided into two halves: the first half and the second half, with each half sized to match the sender's maximum transmission window. Each half maintains its own base bit. When the base bit is 0, a bit value of 0 indicates that the corresponding packet has not been received, while 1 indicates successful packet reception. When the base bit is 1, the semantics are reversed. Upon packet arrival, Themis-D updates the corresponding bit position based on the base bit of the respective half. When a NACK arrives, Themis checks the corresponding bit and compares it with the base bit to determine whether the packet has been received. Upon receiving an ACK, Themis-D updates the start pointer (ePSN). If the start pointer moves from one half to another—for example, from the first half shown in Figure 17a to the second half shown in Figure 17b—Themis flips the base bit of the first half, thereby enabling reuse of the first half to track packet reception status.

B Switch Memory Overhead.

Table 2. Symbols and reference values used in the analysis.

Symbol	Description	Ref Value
N_{paths}	Number of paths per QP	128
N_{NIC}	NICs per ToR switch	16
N_{QP}	Cross-rack QPs per RNIC	512
$N_{\text{TX window}}$	TX window size	512

Themis-S. The memory consumption of Themis-S includes storing the PathMap and per-QP states. The PathMap consists of N_{paths} entries, and each entry requires 16 bits to store the Δ(UDP source port), where $M_{\text{PathMap}} = N_{\text{paths}} \times 2$ bytes.

Table 3 shows the per-QP memory overhead for Themis-S when using the reference values from Table 2.

Table 3. Per-QP memory overhead for Themis-S.

Component	Size (Bytes)
recved_nack_psn	4
path_adaptive_count	$128 \times 4 = 512$
Total	**516**

Themis-D. Themis-D only operates on cross-rack QPs between RNICs connected to different ToR switches. Table 4 shows the per-QP memory overhead for Themis-D when using the reference values from Table 2. Note that the bitmap array size is determined by the TX window size, requiring $N_{\text{TX window}} \times 2/8 = 128$ bytes to track packet delivery status using a double-length bitmap.

Table 4. Per-QP memory overhead for Themis-D.

Component	Size (Bytes)
bitmap_array	128
epsn_array	4
base_first_half	1
base_second_half	1
path_psn_max_array	$128 \times 4 = 512$
stashed_nack_psn	4
stashed_nack_valid	1
stash_nack_path_id	4
stash_nack_qpn	4
stash_nack_egress_port	2
Total	**661**

The total memory overhead for a ToR switch connected to N_{NIC} RNICs, each hosting N_{QP} cross-rack QPs, is given by:

$$M_{\text{total}} = M_{\text{PathMap}} + (M_{\text{QP-S}} + M_{\text{QP-D}}) \times N_{\text{QP}} \times N_{\text{NIC}} \quad (5)$$

where $M_{\text{QP-S}}$ and $M_{\text{QP-D}}$ are the per-QP memory overhead for Themis-S and Themis-D respectively, corresponding to the total values from Tables 3 and 4 (516 bytes and 661 bytes).

Using the reference values from Table 2, the total memory overhead is approximately 9.7 MB. A Tofino2 switch provides four pipelines, each with 20 stages, where each stage contains 80 pages of 1K entries with 128-bit RAMs. These RAMs are dedicated to packet processing logic and are independent of the buffer memory used for packet queuing. Therefore, Themis's memory overhead is reasonable and does not affect the switch's packet buffering capacity.

It is important to note that Themis only operates on cross-rack QPs, which limits the number of QPs that need to be tracked. In AI training workloads, the number of QPs per GPU is typically constrained. A recent study [24] shows that the average number of QPs per GPU for operations such as AllReduce, AllGather, ReduceScatter, and AllToAll is 4, 4, 4, and 10, respectively. Therefore, supporting 512 cross-rack QPs per RNIC is sufficient for most practical scenarios.