Enabling Packet Spraying over Commodity RNICs with In-Network Support

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

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

Kai Chen
Hong Kong University of Science and Technology
Hong Kong, China
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. Although current commodity RNICs support OOO reception, their reliable transport mechanisms misinterpret these arrivals as packet loss, causing spurious retransmissions and unnecessary slow starts.

We propose Themis, a lightweight middleware deployed on programmable switches, enabling packet spraying for commodity RNICs without requiring any modifications to them. Themis applies a PSN-based packet spraying at the source ToR switch, providing an opportunity to infer whether the OOO packet and expected packet traverse the same path based on their packet sequence numbers (PSNs). Building on this opportunity, 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. Experiments demonstrate that Themis reduces the communication completion time of Allreduce and Alltoall by 15.6%~75.3% and 11.5%~40.7%, respectively, compared to the direct combination of commodity RNICs and adaptive routing.

CCS Concepts

• **Networks** → **Transport protocols**; *In-network processing*.

Keywords

Datacenter Networks, Per-hop Flow Control

ACM Reference Format:
Xiangzhou Liu, Wenxue Li, and Kai Chen. 2025. Enabling Packet Spraying over Commodity RNICs with In-Network Support. In *9th Asia-Pacific Workshop on Networking (APNET 2025), August 07–08, 2025, Shang Hai, China.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3735358.3735374

APNET 2025, Shang Hai, China

ACM ISBN 979-8-4007-1401-6/25/08
https://doi.org/10.1145/3735358.3735374

1 Introduction

With the rapid advancement of Artificial Intelligence (AI) [12, 30], AI training jobs have become a significant workload in modern datacenters [11, 15, 19, 21, 31]. To meet the rigorous communication demands of these AI jobs, inter-machine scale-out networks leverage the high-bandwidth and low-latency RDMA technologies. AI workloads exhibit traffic patterns that differ significantly from traditional datacenter workloads [15, 26, 31]. They are mainly composed of a small number of elephant flows with synchronization characteristics, resulting in a low entropy in the traffic pattern [15]. The de-facto RDMA network employs Equal-Cost Multiple-Path (ECMP) [18] for flow-level load balancing. However, this approach severely mismatches the low-entropy traffic patterns of AI workloads: ECMP fails to evenly distribute the small number of elephant flows across all equivalent paths [31]. Consequently, ECMP hash collisions severely degrade the throughput of AI jobs [15, 21, 31].

Packet-level load balancing (e.g., random packet spraying, adaptive routing, etc.) offers a promising alternative to ECMP by avoiding collisions and achieving fine-grained traffic distribution across the network core. However, it introduces out-of-order packet arrivals [13, 20, 35], which are incompatible with the reliable transport of commodity RDMA NICs (RNICs). Previous-generation RNICs (e.g., Mellanox CX-4 and CX-5 [4, 5]) rely on a Go-Back-N (GBN) retransmission mechanism [16], where out-of-order packets are dropped by the receiver, leading to excessive packet loss and performance degradation when packet spraying is employed [35].

Current-generation comodity RNICs (e.g., Mellanox CX-6 [6] and CX-7 [1]) and SmartNICs (e.g., Mellanox BF3 [3]) mitigate this limitation by supporting out-of-order packet reception and using Selective Repeat (i.e., NIC-SR) as the retransmission mechanism [7]. However, NIC-SR is primarily designed for single-path transmission and assumes that all out-of-order arrivals are caused by packet loss. As a result, when packet spraying is used, the receiver blindly generates NACKs for out-of-order packets even when no actual losses occur and out-of-order arrivals are soley due to packet spraying. This behavior introduces two major issues, resulting in bandwidth waste and significant performance degradation:

- **Excessive spurious retransmissions**: Upon receiving a NACK, the sender retransmits the packet indicated by the expected sequence number (ePSN) carried by the NACK, leading to excessive spurious retransmissions.
- **Unnecessary slow starts**: Receiving a NACK also triggers the sender's slow start mechanism [35], reducing its transmission rate unnecessarily.

Previous studies have failed to enable commodity RNICs for packet-level load balancing. Conweave [35] applies flow re-routing

and ensures in-order packet arrival at RNICs by in-network reordering at Top-of-Rack (ToR) switches. However, it does not support packet-level load balancing. Flowlet-based approaches [10, 23, 36], which trade off load balancing granularity to preserve packet order, are poorly suited to RNICs' hardware-based rate pacing. MPRDMA [28] proposes a new transport design for packet-level load balancing, but no off-the-shelf RNICs currently support it.

Motivated by the issues caused by out-of-order packet arrivals and the limitations of previous works, we pose the following question: *Can we design a fine-grained load balancing solution for AI jobs using off-the-shelf, current-generation commodity RNICs and programmable switches that satisfies the following requirements?*

- **Packet-level load balancing**: Achieve the finest (i.e., packet level) granularity of load balancing by fully utilizing all available network paths.
- **Compatibility with commodity RNICs**: Ensure that the solution is directly applicable to current-generation commodity RNICs, requiring no modifications.
- **Minimal deployment overhead**: Minimize the scope of deployment for programmable switches and reduce the state overhead of each switch as much as possible.

We propose Themis, a lightweight middleware deployed only on ToR switches to enable packet-level load balancing for current-generation commodity RNICs[1]. We classify NACKs triggered by OOO arrivals into two categories: valid NACKs, caused by actual packet loss, and invalid NACKs, caused by multi-path delay variation. By enabling destination-side ToR to identify and block invalid NACKs, Themis bridges the gap between packet-level load balancing and NIC-SR.

The core of Themis is a PSN-based packet spraying policy, which distributes packets across all available paths by deterministically assigning each packet to a path based on its packet sequence number (PSN) modulo the number of available paths. The unique opportunity offered by this policy is that it enables inference of whether two packets traverse the same path using only their PSNs. Building on this opportunity, Themis consists of two core 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. Themis-D, at the destination-side ToRs, identifies whether a NACK is valid by analyzing the PSN of OOO packets (tPSN, for short) that triggered the NACK and the ePSN. To achieve this, Themis-D caches each flow's in-flight PSNs at the last hop in a PSN queue, and identifies the tPSN for each NACK by scanning the queue.

Our preliminary NS-3 simulation results show that by blocking invalid NACKs, Themis effectively reduces spurious retransmissions and avoids unnecessary slow starts. Under various DCQCN configurations, Themis reduces the communication completion time by 15.6%~75.3% for Allreduce and 11.5%~40.7% for Alltoall, compared to the direct combination of adaptive routing with commodity RNICs.

[1]Hereafter, "commodity RNIC" and "RNIC" in this paper refer to current-generation RNICs that optionally support out-of-order packet reception and utilize NIC-SR as their reliable transport protocol [1, 3, 6].

2 Background and Motivation

2.1 ECMP's Dilemma in AI Workloads

Datacenters often utilize Clos networks [9, 34] 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) [18] is the most widely used load balancing (LB) mechanism [16], which determines the path of a flow by hashing the 5-tuple in the packet header. ECMP works well for traditional workloads, where there are millions of flows that ensure a relatively even distribution of traffic. However, unlike traditional datacenter workloads, AI training workloads exhibit traffic patterns that are fundamentally mismatched with ECMP's design: (1) Small number of flows: In 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.

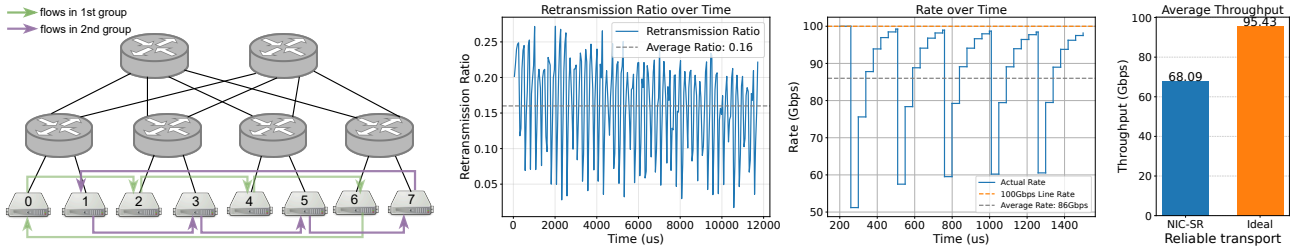
These flow characteristics (i.e., few in number, large in size) result in a high ECMP collision rate, as the small number of flows cannot be evenly distributed across available paths, leading to severe performance degradation in AI training workloads [15, 21, 31].

2.2 Incompatibility between Packet-Level LB and Commodity RNICs

Out-of-Order Arrival. Packet-level load balancing (LB) (e.g., random packet spraying [13], adaptive routing, etc.) is a promising solution to address the limitations of ECMP. Packet-level routing evenly distributes traffic across multiple paths, even with few flows. However, it causes out-of-order (OOO) packet arrivals [13, 20], making it incompatible with reliable transport on commodity RNICs and challenging to deploy in datacenters.

RNICs' Reliable Transmission Mechanism. The latest generation of commodity RNICs [1, 3, 6] supports Selective Repeat (i.e., NIC-SR) as their built-in reliable transmission mechanism [7]. While some SmartNICs [3] enable programmability of congestion control (CC) logic [2], their reliable transports remain fixed and cannot be programmed. These NICs also support optionally leverage NIC-SR for handling out-of-order packet reception. NIC-SR handles out-of-order (OOO) data packets and retransmissions 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



(a) Topo of the motivation simulation. **(b) Retransmission ratio.** **(c) Sending rate.** **(d) Throughput.**

Figure 1: Performance impact of directly combining packet spraying and commodity RNICs.

most one NACK, even if multiple OOO packets arrive.

- The triggered NACK only carry 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 reduceing hardware implementation complexity.

NIC-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. However, the commodity RNIC blindly assumes the packet is lost and accordingly generates a NACK. Therefore, many of these NACKs are unnecessary and should not reach the sender, as they can cause spurious retransmissions and unnecessary slow starts.

To validate this, we conduct an experiment using NS3 with a typical leaf-spine topology, where all links have a bandwidth of 100Gbps. The topology consists of eight nodes, as shown in Figure 1a. Nodes $\{0, 2, 4, 6\}$ form one group, and nodes $\{1, 3, 5, 7\}$ form another group. Each node sends 100MB of data to the next node within the same group. This creates a ring traffic pattern for each group, which is common in collective communications. Random packet spraying is used as the load-balancing method.

Excessive Spurious Retransmissions. When the sender RNIC receives a NACK, it retransmits only the packet indicated by the ePSN. However, in scenarios where OOO arrivals are caused by multi-path delay variations rather than packet loss, these retransmissions are unnecessary and spurious, wasting network resources. In our experiment, we observe that no packet loss occurs. However, as shown in Figure 1b, the retransmission ratio of a chosen flow (from node 0 to 2) remains high throughout the transmission. By the end, the average spurious retransmissions ratio is 16% for all flows, meaning only 84% of the traffic are useful.

Unnecessary Slow Starts. In addition to triggering retransmissions, NACKs also cause the sender to reduce its transmission rate [35], as they are treated as signals of congestion. However, this assumption is incorrect in the context of packet-level LB, where OOO arrivals can occur without any congestion. This unnecessary rate reduction further degrades performance. In our experiment, no congestion occurs. However, the RNIC's congestion control module (DCQCN [41]) reacts to NACKs by reducing the sending rate, causing unnecessary slow starts. Figure 1c illustrates this behavior by showing the sending rate of a chosen flow, where all rate drops are triggered by NACKs. As a result, the average sending rate is reduced to 86% of the line rate.

Figure 1d shows the end-to-end impact: the average throughput of all flows is only 71% of the ideal case (with no spurious retransmissions or slow starts). This degradation results from the combined effects of an 86% average sending rate and an 84% useful transmission ratio ($71\% = 86\% \times 84\%$).

2.3 Limitations of Existing Solutions

In-network reordering. Conweave [35] allows packets from a single flow to traverse up to two distinct paths simultaneously during rerouting phase and relies on the ToR switch to perform in-network reordering of packets from these paths. However, the in-network reordering approach cannot support packet-level LB, as reordering packets from multiple paths exceeds the resource capacity of ToR switches.

Flowlet-based LB. Flowlet-based LB [10, 23, 36] preserves 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.

Multi-path RDMA Transport. Other works [25, 28, 29, 33] propose new transport protocols to enable packet-level LB. 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 [14], integrating and validating new transport protocols on these devices requires hardware developers with domain expertise. Most AI training clusters continue to utilize commodity RNICs [15, 19, 21, 31, 39] 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 [2]), making these transport protocols difficult to integrate into existing clusters.

3 Design

3.1 Key Idea and Design Overview

When using packet-level LB with commodity RNICs, as mentioned in Section 2.2, some NACKs are unnecessary and should not reach the sender. This raises the question: *how to determine whether a NACK is "necessary"*?

We classify NACKs into two categories based on the OOO packet

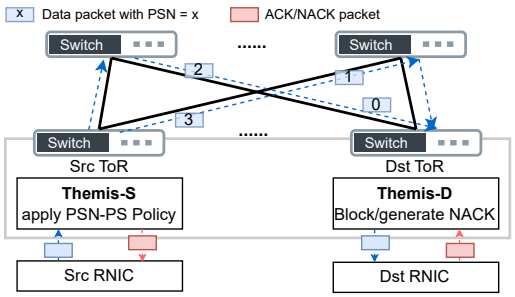


Figure 2: Overview of Themis.

that triggers NACKs and the expected packet:

- **Valid NACKs:** If the OOO packet travels along the same path as the expected packet, it confirms that the expected packet is indeed lost. Therefore, the NACK triggered by it should be forwarded to the sender to trigger the retransmission of the ePSN packet. for example, if the receiver's ePSN in Figure 2 is 0, and the packet with PSN = 2 arrives, the NACK triggered by this packet is valid because the packet with PSN = 2 travels along the same path as the expected packet with PSN = 0.
- **Invalid NACKs:** If the OOO packet travels along a different path than the expected packet, it cannot confirm the loss of the expected packet. Thus, the corresponding NACK is invalid and should not reach the sender. For example, if the receiver's ePSN in Figure 2 is 0, and the packet with PSN = 1 arrives, the NACK it triggers is invalid because the packet with PSN = 1 travels along a different path than the expected packet with PSN = 0.

Key Idea. Our key idea is to use a deterministic packet spraying policy for each flow. This policy allows the receiver-side ToR switch to determine whether the OOO packet and the expected packet travel along the same path. Based on this information, the ToR switch can classify NACKs as valid or invalid and block invalid NACKs accordingly.

Design Overview. Based on this key idea, we propose an in-network middleware, **Themis**, which consists of Themis-Src (Themis-S) and Themis-Dst (Themis-D). Figure 2 illustrates its overview, with both components deployed on the ToR switch and work together to identify and block invalid NACKs.

We adopt a PSN-based packet spraying policy (Section 3.2). The Themis-S on the sender-side ToR modifies the packet header to apply this policy. This spraying policy enables the Themis-D on the receiver-side ToR to determine whether a NACK, based on the tPSN (PSN of the OOO packet that triggered the NACK) and ePSN (expected PSN when the NACK is generated), should be blocked.

One challenge is that NACKs generated by commodity RNICs only contain the ePSN and do not include the tPSN to maintain protocol compatibility (Section 2.2). To address this limitation, we design a ring-based PSN queue structure deployed on the receiver-side ToR switch. The Themis-D leverages this queue to cache the PSNs of all in-flight packets on the ToR-to-NIC hop and identifies the tPSN for each NACK by scanning the queue (Section 3.3).

Another challenge arises when an invalid NACK is blocked for an ePSN while subsequent packets confirm that the ePSN packet is indeed lost. In this case, a valid NACK needs to be triggered and forwarded to the sender. However, as the RNIC can generate

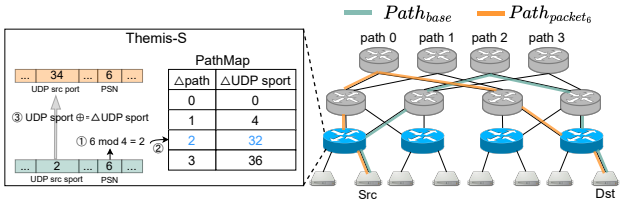


Figure 3: Illustration of PSN-based packet spraying.

only one NACK for an ePSN, it cannot generate this valid NACK (Section 2.2). To address this, we design a NACK compensation mechanism (Section 3.4) that allows the Themis-D to generate and send the necessary NACK on behalf of the RNIC when the RNIC is unable to do so.

3.2 PSN-based Packet Spraying

To achieve packet-level LB and enable NACK validation, we propose a PSN-based packet spraying 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, ECMP hashing algorithm determines its base path index $P_{\text{base}} \in \{0, 1, \ldots, N-1\}$. Under our policy, any $packet_i$ 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 it, we can determine the traveled path of the OOO packet that triggers the NACK (with $tPSN$) and the expected packet (with $ePSN$) below:

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

The NACK is valid if $Path_{ooo} = Path_{expected}$. This condition simplifies to:

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

Thus, for any NACK, if Eq. 3 holds, the NACK is valid; otherwise, it is invalid.

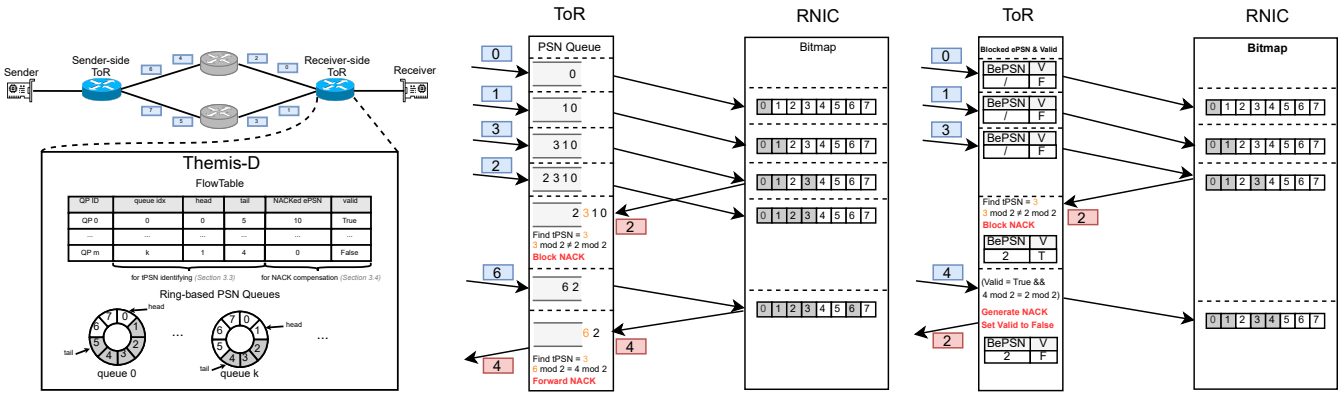
Implementation limited to the ToR switch. To minimize the deployment scope of programmable switches, we limit the necessary modifications to the Top-of-Rack (ToR) switch.

In a 2-tier Clos network, where path selection is entirely determined by the ToR (leaf) switch, this mechanism can be fully implemented 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, the method proposed in prior work [37], which has already been deployed in production AI training clusters [31], can be applied. This method leverages the linearity of ECMP hashing to construct a deterministic *PathMap* offline. As illustrated in Figure 3, when a packet arrives at the ToR switch, the ToR calculates its relative path change by $PSN \bmod N$ (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.

3.3 Identifying tPSN for NACK Validation

To enable the ToR switch to identify valid NACKs using Eq. 3, it must determine the PSN of OOO packet that triggered the NACK, referred



(a) Themis-D design. **(b) Identify tPSN and block invalid NACK.** **(c) Example of NACK compensation.**

Figure 4: Themis-D design and corresponding examples.

to as tPSN. However, NACKs generated by commodity RNICs do not include the tPSN (Section 2.2). To address this limitation without modifying the RNIC, we leverage two key observations:

- The time between when an OOO packet leaves the ToR switch and when the corresponding NACK arrives back at the ToR switch can be roughly bounded by the $\text{RTT}_{\text{last-hop}}$.
- The tPSN of a NACK is the first PSN larger than the ePSN that arrived at the RNIC. This results from the fact that the RNIC generates at most one NACK per ePSN (Section 2.2).

Based on first observation, we propose a per-QP ring-based PSN queue at the ToR to store the PSNs of recently sent packets for tPSN identification. Each queue entry corresponds to a PSN. During connection setup, the ToR intercepts RNIC handshakes and prepares a ring-based PSN queue for each QP, which operated in FIFO manner, ToR stores relevant context (queue index, head/tail pointers) in the Flow Table, as illustrated in Figure 4a. Queue capacity, determined by the bandwidth-delay product (BDP) of the last hop and the network's MTU, is configured to be slightly larger than $\text{BDP}_{\text{last-hop}}/\text{MTU}$ to accommodate fluctuations in $\text{RTT}_{\text{last-hop}}$.

Based on the second observation, when a NACK arrives at the ToR switch, the switch dequeues entries from the PSN queue until it finds the first PSN larger than the ePSN. This PSN is identified as the tPSN, as the dequeuing order of the PSN queue matches the arrival order of the packets. The ToR switch then validates the NACK using Eq. 3 and either forwards or blocks the NACK accordingly.

Example. Assume there are two paths, as shown in Figure 4a, and a packet arrival order, as illustrated in Figure 4b. PSNs of packets (e.g., 0, 1, 3, 2) are enqueued into the PSN queue before they leave the ToR switch. When a NACK with ePSN = 2 arrives at the ToR switch, the switch dequeues PSNs from the queue until it finds the first PSN larger than 2. This PSN, which is 3, is identified as the tPSN. The NACK is determined to be invalid based on Eq. 3 and is therefore blocked. Similarly, when a NACK with ePSN = 4 arrives, the switch dequeues until it finds tPSN = 6. The NACK is determined to be valid and is forwarded to the sender.

3.4 NACK Compensation

Blocking invalid NACKs prevents unnecessary retransmissions but introduces a new challenge: if subsequent packets confirm the ePSN packet is indeed lost, the RNIC cannot regenerate a NACK for the same ePSN (Section 2.2). Consequently, the lost packet is only retransmitted after a timeout, causing performance degradation. To address this, we design a NACK compensation mechanism that enables the ToR to compensate for blocked NACKs. The ToR maintains Blocked ePSN (BePSN) and Valid fields in the flow table for each flow, as shown in Figure 4a. These fields guide the switch in deciding whether to compensate for a blocked NACK when new packets arrive. The mechanism works as follows:

- When a NACK is blocked, the BePSN is set to the ePSN of the blocked NACK, and the Valid field is set to True, indicating that a NACK associated with BePSN may require compensation in the future.
- If Valid is True and a packet with a PSN larger than BePSN arrives such that $PSN \bmod N = BePSN \bmod N$, the ToR determines the BePSN packet is lost and generates a NACK for the BePSN. The Valid field is set to False to prevent generating multiple NACKs for the same BePSN.
- If Valid is True and a packet with a PSN equal to BePSN arrives, the switch determines that the packet with BePSN was not lost, and no compensation for the blocked NACK is needed. The Valid field is then set to False, ensuring no further NACKs are generated for BePSN.

Example. As shown in Figure 4c, when a NACK with ePSN = 2 is blocked, the BePSN is set to 2 and Valid is set to True. When a subsequent packet with PSN = 4 arrives, the switch checks the BePSN field because Valid is True. Since 4 mod 2 = 2 mod 2, the switch determines that the packet with BePSN is lost and generates a NACK for ePSN = 2.

4 Memory Overhead Estimation

Themis-S. The memory consumption of Themis-S primarily comes from storing the PathMap. The PathMap consists of N_{paths} entries, and each entry requires 16 bits to store the Δ(UDP source port). Thus, the memory consumption of the PathMap is:

$$M_{\text{PathMap}} = N_{\text{paths}} \times 2\,\text{bytes}.$$

Themis-D. Themis-D only operates on cross-rack QPs between RNICs connected to different ToR switches. The per-QP memory

Table 1: Symbols and reference values used in the analysis.

Symbol	Description	Ref Value
N_{paths}	Number of equal-cost paths	256
BW	Last-hop bandwidth	400 Gbps
RTT_{last}	Last-hop RTT	2 μs
N_{NIC}	NICs per ToR switch	16
N_{QP}	Cross-rack QPs per RNIC	100
MTU	MTU size	1500 B
F	Queue capacity expansion factor	1.5

overhead includes the flow table entry and the PSN queue.

Each flow table entry requires 20 bytes, consisting of 13 bytes for the QP ID, 3 bytes for the `blocked ePSN`, 1 byte for the `Valid` flag, and 3 bytes for queue metadata (queue index, head/tail pointers).

For each PSN queue, the number of entries is determined by:

$$N_{\text{entries}} = \left\lceil \frac{BW \times RTT_{\text{last}} \times F}{\text{MTU}} \right\rceil,$$

where $F > 1$ is the queue capacity expansion factor for potential RTT fluctuations. Each queue entry requires 1 byte to store the truncated PSN. Therefore, the memory overhead per QP is:

$$M_{\text{QP}} = 20\,\text{bytes} + N_{\text{entries}} \times 1\,\text{byte}$$

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

$$M_{\text{total}} = M_{\text{PathMap}} + M_{\text{QP}} \times N_{\text{QP}} \times N_{\text{NIC}} \tag{4}$$

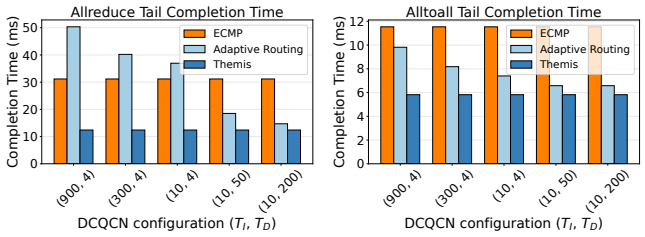
Example. We assume a fat-tree topology [9] with switches with port number $k = 32$. A 3-layer fat-tree with 1:1 subscription consists of $\frac{k^2}{2} = 512$ ToR (leaf) switches, $\frac{k^2}{2} = 512$ spine switches, and $\frac{k^2}{4} = 256$ core switches, supporting up to $k^3/4 = 8192$ GPUs (NICs). In this topology, at most 256 equal-cost paths exist between any source-destination pair, setting $N_{\text{paths}} = 256$. And in this topology, each ToR connects $N_{\text{NIC}} = k/2 = 16$ RNICs.

In AI training workloads, the number of QPs per GPU is usually limited. A recent study [15] 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, we estimate the cross-rack QP count per RNIC to be $N_{\text{QP}} = 100$.

Using these values in Eq. 4 yields $M_{\text{total}} \approx 193$ KB, consuming just 0.6% of the 64 MB SRAM in current Tofino switches[8].

5 Evaluation

Simulation Setup. We use NS-3 simulations to evaluate Themis in a 16×16 leaf-spine topology with 1:1 subscription, where all links are 400 Gbps with 1 μs delay and each switch is equipped with a 64 MB buffer [8]. Each of the 16 ToR switches connects to 16 NICs, and this topology connects a total of 256 NICs (also representing 256 GPUs). NICs use NIC-SR for reliable transmission and DCQCN for congestion control. 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 the experiments, all 16 groups start the same collective communication simultaneously. We conduct separate experiments for Allreduce (300MB) and Alltoall (300MB), using the slowest group's completion time as the metric which reflects the training job's communication bottleneck. Themis is compared with ECMP and Adaptive Routing under various DCQCN configurations.



(a) Allreduce **(b) Alltoall**

Figure 5: Collective communication performance under different DCQCN parameters.

Results and Analysis. Figures 5a and 5b show that Themis consistently outperforms ECMP and Adaptive Routing (AR). Compared to AR, Themis achieves 15.6%∼75.3% and 11.5%∼40.7% lower communication completion time for Allreduce and Alltoall, respectively, across different DCQCN configurations. In DCQCN, the rate decrease interval (T_D) controls the frequency of rate reductions, while the rate increase timer (T_I) sets the interval for recovering the sending rate. With the recommended parameters [27] ($T_I = 900$ μs, $T_D = 4$ μs), AR performs poorly due to frequent rate reductions and slow recovery. Adjusting T_I and T_D improves AR, as smaller T_I and larger T_D mitigate the slow start effect. However, extreme values, such as overly small T_I or overly large T_D, may delay responses to congestion, causing prolonged congestion and degraded performance. This also indicates that using NIC-SR with packet-level LB introduces new challenges for CC parameter tuning.

6 Discussion and Future Work

Link Failure Tolerance. When link failures occur, Themis's PSN-based packet spraying may fail to maintain load balancing across network cores. Upon detecting failures via monitoring tools [17, 37], ToR switches disable Themis and revert to ECMP mode. Future work will explore restricting flows to path subsets and dynamically adjusting pathsets when failures occur.

Implementation on Tofino Switch. Themis needs merely 0.2MB SRAM (Section 4), well within Tofino switch capacity [8]. Studies show that Tofino switches can efficiently support a diverse range of offloading tasks [22, 24, 32, 38, 40]. With its lightweight design, Themis suits Tofino hardware implementation. Future work will implement Themis on real Tofino switches.

7 Conclusion

Themis is a lightweight 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 preventing spurious retransmissions and unnecessary slow starts, Themis enables effective packet spraying with commodity RNICs. Evaluations demonstrate its superior performance.

Acknowledgments

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

References

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

[2] 2025. DOCA PCC. (2025). https://docs.nvidia.com/doca/sdk/doca+pcc/index.html

[3] 2025. Mellanox BlueField-3 Product Brief. (2025). https://www.nvidia.com/en-us/networking/products/data-processing-unit/

[4] 2025. Mellanox ConnectX-4 Product Brief. (2025). https://www.nvidia.com/en-in/networking/ethernet/connectx-4-lx/

[5] 2025. Mellanox ConnectX-5 Product Brief. (2025). https://www.nvidia.com/en-sg/networking/ethernet/connectx-5/

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

[7] 2025. Zero touch RoCE enables RoCE to operate on fabrics where no PFC nor ECN are configured. (2025). https://docs.nvidia.com/networking/display/winof2v25150020/ethernet+network#src-3576232145_EthernetNetwork-RoLN

[8] 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

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

[10] 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.

[11] 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 (ACM SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3651890.3672239

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

[13] 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

[14] 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

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

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

[17] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, Dave Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, Zhi-Wei Lin, and Varugis Kurien. 2015. Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 139–152. https://doi.org/10.1145/2785956.2787496

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

[19] Qinghao Hu, Zhisheng Ye, Zerui Wang, Guoteng Wang, Meng Zhang, Qiaoling Chen, Peng Sun, Dahua Lin, Xiaolin Wang, Yingwei Luo, Yonggang Wen, and Tianwei Zhang. 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

[20] 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

[21] Ziheng Jiang, Haibin Lin, Yinmin Zhong, Qi Huang, Yangrui Chen, Zhi Zhang, Yanghua Peng, Xiang Li, Cong Xie, Shibiao Nong, Yulu Jia, Sun He, Hongmin Chen, Zhihao Bai, Qi Hou, Shipeng Yan, Ding Zhou, Yiyao Sheng, Zhuo Jiang, Haohan Xu, Haoran Wei, Zhang Zhang, Pengfei Nie, Leqi Zou, Sida Zhao, Liang Xiang, Zherui Liu, Zhe Li, Xiaoying Jia, Jianxi Ye, Xin Jin, and Xin Liu. 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

[22] Xin Jin, Xiaozhou Li, Haoyu Zhang, Robert Soulé, Jeongkeun Lee, Nate Foster, Changhoon Kim, and Ion Stoica. 2017. NetCache: Balancing Key-Value Stores with Fast In-Network Caching. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17)*. Association for Computing Machinery, New York, NY, USA, 121–136. https://doi.org/10.1145/3132747.3132764

[23] 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.

[24] ChonLam Lao, Yanfang Le, Kshiteej Mahajan, Yixi Chen, Wenfei Wu, Aditya Akella, and Michael Swift. 2021. ATP: In-network Aggregation for Multi-tenant Learning. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association, 741–761. https://www.usenix.org/conference/nsdi21/presentation/lao

[25] 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. (2024). arXiv:cs.NI/2407.15266 https://arxiv.org/abs/2407.15266

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

[27] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, and Minlan Yu. 2019. HPCC: high precision congestion control. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM '19)*. Association for Computing Machinery, New York, NY, USA, 44–58. https://doi.org/10.1145/3341302.3342085

[28] 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

[29] Rui Miao, Lingjun Zhu, Shu Ma, Kun Qian, Shujun Zhuang, Bo Li, Shuguang Cheng, Jiaqi Gao, Yan Zhuang, Pengcheng Zhang, Rong Liu, Chao Shi, Binzhang Fu, Jiaji Zhu, Jiesheng Wu, Dennis Cai, and Hongqiang Harry Liu. 2022. From luna to solar: the evolutions of the compute-to-storage networks in Alibaba cloud. In *Proceedings of the ACM SIGCOMM 2022 Conference (SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 753–766. https://doi.org/10.1145/3544216.3544238

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

[31] Kun Qian, Yongqing Xi, Jiamin Cao, Jiaqi Gao, Yichi Xu, Yu Guan, Binzhang Fu, Xuemei Shi, Fangbo Zhu, Rui Miao, Chao Wang, Peng Wang, Pengcheng Zhang, Xianlong Zeng, Eddie Ruan, Zhiping Yao, Ennan Zhai, and Dennis Cai. 2024. Alibaba HPN: A Data Center Network for Large Language Model Training. In *Proceedings of the ACM SIGCOMM 2024 Conference (ACM SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA, 691–706. https://doi.org/10.1145/3651890.3672265

[32] Amedeo Sapio, Marco Canini, Chen-Yu Ho, Jacob Nelson, Panos Kalnis, Changhoon Kim, Arvind Krishnamurthy, Masoud Moshref, Dan Ports, and Peter Richtarik. 2021. Scaling Distributed Machine Learning with In-Network Aggregation. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association, 785–808. https://www.usenix.org/conference/nsdi21/presentation/sapio

[33] 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.

[34] 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 (SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 183–197. https://doi.org/10.1145/2785956.2787508

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

[36] 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.

[37] 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

[38] Bohan Zhao, Wenfei Wu, and Wei Xu. 2023. NetRPC: Enabling In-Network Computation in Remote Procedure Calls. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 199–217. https://www.usenix.org/conference/nsdi23/presentation/zhao-bohan

[39] Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Huazuo Gao, Jiashi Li, Liyue Zhang, Panpan Huang, Shangyan Zhou, Shirong Ma, Wenfeng Liang, Ying He, Yuqing Wang, Yuxuan Liu, and Y. X. Wei. 2025. Insights into DeepSeek-V3: Scaling Challenges and Reflections on Hardware for AI Architectures. (2025). arXiv:cs.DC/2505.09343 https://arxiv.org/abs/2505.09343

[40] Hang Zhu, Kostis Kaffes, Zixu Chen, Zhenming Liu, Christos Kozyrakis, Ion Stoica, and Xin Jin. 2020. RackSched: A Microsecond-Scale Scheduler for Rack-Scale Computers. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, 1225–1240. https://www.usenix.org/conference/osdi20/presentation/zhu

[41] 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 (SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 523–536. https://doi.org/10.1145/2785956.2787484