Adaptive Packet Routing vs. Flowlet Switching for Distributed Training Under Link Failures

Alon Zeltser
alondovz@post.bgu.ac.il

February 10, 2026

Abstract

Modern AI factories rely on low-latency, high-bandwidth fabrics for distributed training. In Ethernet-based deployments (e.g., RoCE), the routing policy can strongly affect congestion, tail latency, and ultimately training iteration time. This project compares three multipath strategies—ECMP, flowlet switching, and an adaptive (queue-aware) routing policy—using a custom discrete-event simulator that models an AI-factory "scale unit" (SU) like leaf–spine Clos pod, and training-like collective traffic augmented with background "mice" flows.

I report application-level metrics (job completion time, step time mean and tail percentiles) together with network-level indicators (mice flow completion time, queue occupancy, and utilization) under an offered-load sweep and under random link failures. The experiment results indicate that queue-aware adaptive routing can substantially reduce job time and step-time tail latency at high load, and remain robust under failures, while ECMP and flowlet switching are more prone to persistent hotspots and larger queues.

1 Introduction

Large-scale distributed training generates bursty, synchronized communication patterns (e.g., all-reduce) that can stress multi-rooted Clos fabrics and amplify the impact of transient congestion. At the same time, many AI factories deploy RDMA over Ethernet infrastructure (RoCE) where loss and congestion dynamics are tightly coupled with transport behavior. These properties motivate revisiting the routing layer: can fine-grained adaptive routing outperform flowlet switching and classic ECMP under training-like traffic and failures?

1.1 Goal

Evaluate how routing choices (ECMP vs. flowlet switching vs. adaptive routing) affect *training-task* performance—iteration/step time and job completion time—in an SU-like AI-factory pod, including robustness to link failures.

1.2 Hypothesis.

Compared to ECMP and flowlet switching, queue-aware adaptive routing will (i) reduce job time, which will be achieved by (ii) reducing mean step time and (iii) reducing significantly tail step time (p95/p99). This should trend is expected to be more notable in high offered load and under random link failures.

1.3 Contributions

- A Python discrete-event simulator that implements application layer (training workload generator) and a packet-level network layer (hosts, switches, ports, queues, links).
- Two training-inspired workloads (a single heavy-job and simultaneous mixture of jobs), and an offered-load sweep with and without failures.
- A measurement suite covering job time, step time (mean/p95), mice mean/p95 FCT (flow completion time), and queue metrics.

2 Background

2.1 Multipath routing in Clos fabrics

ECMP hashes each flow onto one of multiple equal-cost paths. It is simple and widely deployed, but can suffer from *hash collisions* where large flows share a bottleneck while other paths remain underutilized. **Flowlet switching** mitigates persistent collisions by allowing a long flow to be split into shorter routing sequences (flowlets) potentially sent over different paths when a sufficient inter-packet gap is observed. LetFlow studies flowlet switching as a simple and resilient approach, especially under asymmetry [1]. **Adaptive / packet-spraying approaches** aim for more fine-grained load balancing. Presto demonstrates host-based load balancing that can outperform ECMP and flowlet switching by spreading traffic across available paths [2]. Modern AI Ethernet fabrics also incorporate telemetry and congestion-awareness (e.g., Spectrum-X) [3].

2.2 AI-factory "scale units"

NVIDIA DGX SuperPOD reference architectures describe modular scaling built from *scalable units (SU)* [4]. The networking design includes leaf–spine fabrics and emphasizes redundancy and fault tolerance at the fabric level [5]. This project uses a SU-like Clos pod as the baseline topology (scaled down for simulation runtime).

2.3 RDMA, reordering, and routing

RoCE deployments are sensitive to congestion and loss; recent work (e.g., SeqBalance) highlights that load-balancing schemes must consider reordering implications for RDMA [6]. In this project, we focus on *application-level* step time and tail behavior while measuring queue build-up and mice flow completion time to gauge network congestion.

3 Methodology

3.1 Simulator architecture

The simulator is a discrete-event system with:

- **Application layer**: constructs jobs consisting of steps (train iterations) and phases (gradients compute phase + collectives phase) and injects flows into the network. Each collective listen to the completion of the collective through all the participating servers, via a barrier. The high level structure of the application layer is described in Figure 1. In this implementation there are 2 types of job scenarios within the pod: a single, heavy job, and a mixture of smaller two jobs, yet this application layer support the implementation of additional various scenarios. While the application layer is abstract, it is incorporated into a previously developed discrete simulator that is used for network level simulation. Besides submitting application–level tasks, the application layer finally direct hosts (servers) to send messages, and can register to listen to the arrival of those, for synchronizing collective–tasks.
- **Network layer**: consist of hosts, switches, ports with egress queues, and full-duplex links with bandwidth and propagation delay. The elements are connected similarly to the real world connections, and inform each other on each action using a post message interface. Each element that receives post–message write an event to the global event scheduler that will, when time comes, initiate that event, as shown in Figure 2
- **Routing policies**: pluggable forwarding decisions for ECMP, flowlet switching, and adaptive queue-aware routing.

3.2 Topology

I modeled an SU-like two-layer Clos pod with 32 servers arranged as 8 racks × 4 servers per rack, connected via 8 ToR (leaf) switches to 4 spine switches. A representative run summary reports 32 hosts, 12 switches, and 512 links for this topology as seen in Figure 3 and Table 1

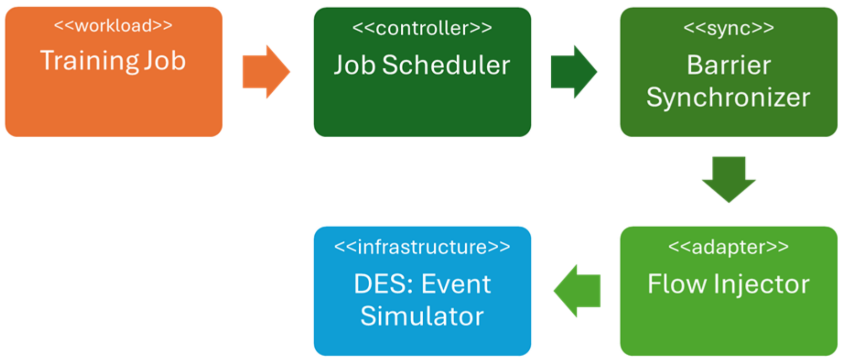


Figure 1: Training Job Schematic Architecture, shows the management of jobs, broken into sequences of dependent steps, phases and collectives by a Job Scheduler, then scheduled in the network simulator by the Flow Injector as scheduled host-to-host messages, and tracks them for completion by a Barrier Synchronizer

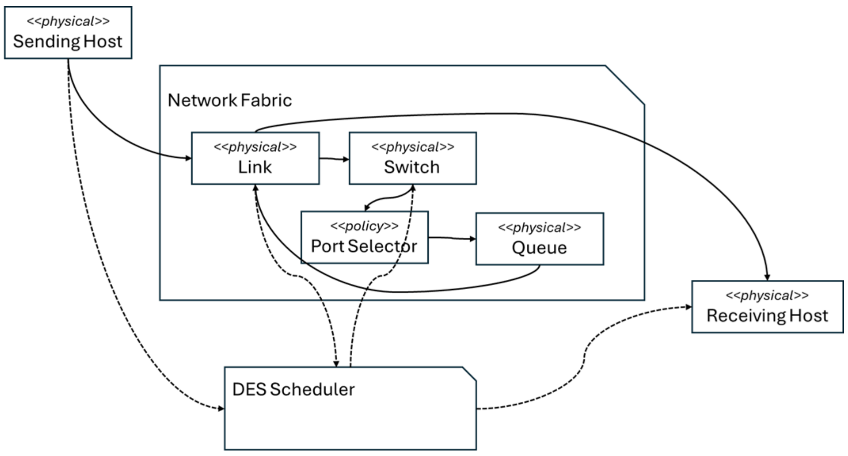


Figure 2: Network Simulator Schematic Diagram. Each network entity (host, switch) has ports with queues, connected to links, and so forth. At time, the network entity decides by the active policy how to select port. Each entity that needs to delays its next action turns to the DES (discrete event simulator) and submit a future event to be triggered at the desired time.

3.3 Routing policies evaluated

- **ECMP**: per-flow hash over equal-cost spine paths.
- **Flowlet switching**: host-managed ECMP where path choice may change at flowlet boundaries (parameterized by a packet-gap for simplicity and persistence).
- **Adaptive (queue-aware)**: chooses among equal-cost candidate next-hops using the minimum observed egress-queue length.

3.4 Workloads

I evaluate two scenarios:

Single heavy workload. A single communication-heavy job is run with 50 steps (chosen to keep test runtime reasonable, while still exercising congestion at higher load).

Mixed workload (two concurrent jobs). A “TP-heavy” job (tensor parallelism) and a “PP+DP” job (pipeline parallelism and data parallelism) run concurrently to emulate mixed training activity (typically by different clients or teams). In the simulator configuration, the TP-heavy job uses 100 steps while the PP+DP job uses 200 steps [7, see log parameters].

Table 1: Baseline SU-like pod configuration used in the simulator (scaled model).

Parameter	Value
Servers (hosts)	32
Racks	8 (4 servers per rack)
Leaf (ToR) switches	8
Spine switches	4
Links	512 (full-duplex modeled)

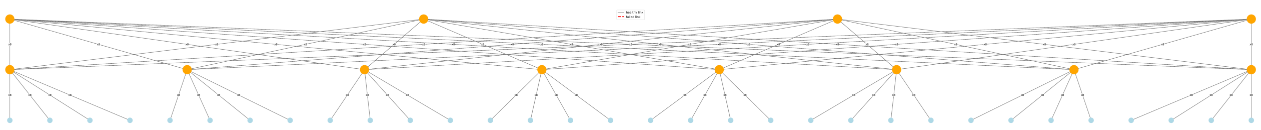


Figure 3: SU Pod Topology as visualized by the simulation: 32 GPU-Servers arranged as 4 in a rack, overall 8 racks with leaf switches and 4 spine switches

3.5 Offered-load sweep and failure model

I sweep offered-load by scaling collective bucket sizes (payload). The sweep points are:

- **Mixed**: X1, X2, X4, and X4 with 5% random link failures.
- **Heavy**: X1, X2, X8, and X8 with 5% random link failures.

Amount of simulated payloads can be found in the project configuration .yaml files. Failures are injected by randomly disabling a fixed fraction of links (5%) prior to the run, producing asymmetric path diversity.

3.6 Metrics

I report metrics aligned with training impact and fabric stress.

Training metrics.

- **Job completion time**: simulation time, from job start to job finish.
- **Step time**: per-step duration for each job; I report mean and p95 (and p99 in the project logs).

Mice traffic metrics. Small background flows are injected with a fixed inter-arrival process; I report their **flow completion time (FCT)** mean and p95 (and optionally p99, in logs). A sample run summary reports mice flows and FCT percentiles [8, see log summary].

Queue and utilization metrics.

- **Queue occupancy**: peak egress queue length per port (global max) and average peak per node.
- **Utilization**: average link utilization and bytes transmitted statistics.

4 Results

4.1 Job Completion time vs. load

4.1.1 Heavy Workload

Figure 4 summarizes the end-to-end completion time of the heavy workload job as traffic is scaled, and 5% link failure is added. At low load (×1), all three schemes complete in roughly the same time (∼10 s), indicating the fabric is uncongested. As load increases, **ECMP** and **flowlet switching** degrade sharply, revealing a congestion “knee” where persistent queueing and hotspot formation dominate completion time. In contrast, **adaptive routing** stays essentially flat (∼10 s) across all load points.

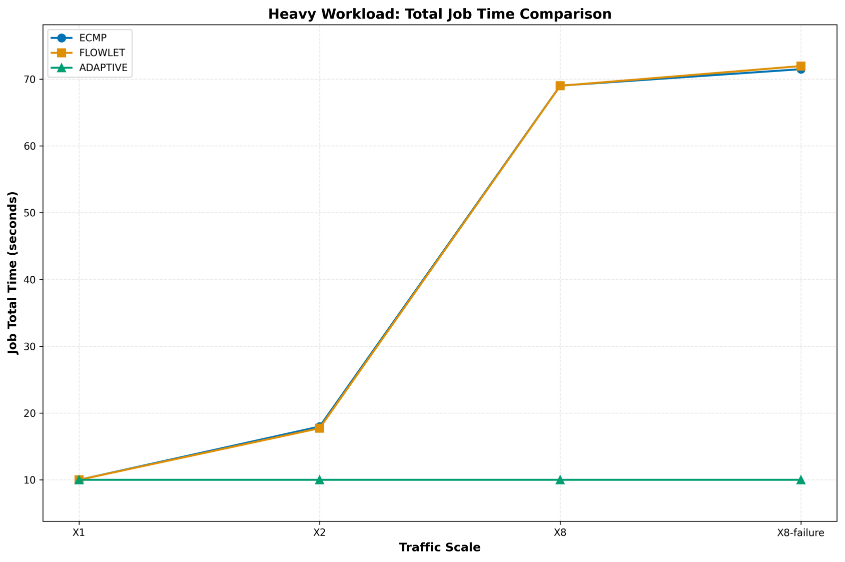


Figure 4: Heavy workload job completion time vs. offered load (and with 5% failures).

4.1.2 Mixed Workload

Figure 5 shows the mixed-workload total job time as traffic is scaled from ×1 to ×2, ×4, and ×4 with a failure. All schemes are identical at ×1 and (about 10 s), implying the network is not loaded, with no congestion . At ×4, **ECMP** increases sharply to ∼18.7 s and rises slightly further under ×4-*failure* (∼19.2 s), while **flowlet switching** increases mildly from 10 s to ∼10.4 s and ∼10.9 s under failure. **Adaptive routing** remains essentially flat at ∼10 s across all points shown.

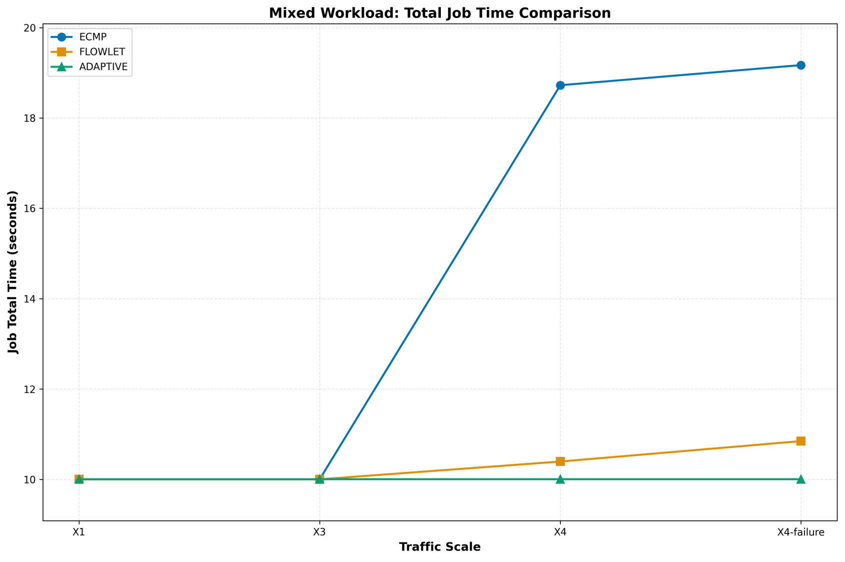
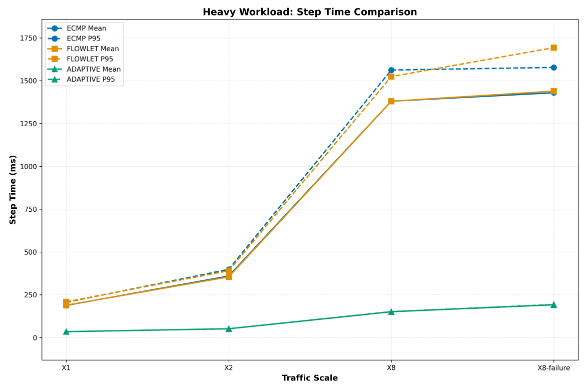


Figure 5: Mixed workload job completion time vs. offered load (and with 5% failures

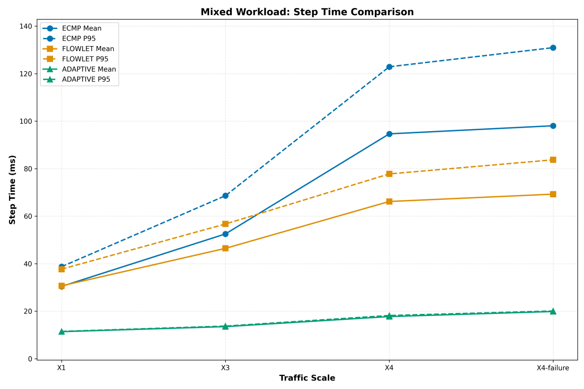
4.2 Step time under load

4.2.1 heavy workload

Figure 6a compares mean and p95 step time across the heavy-workload sweep. The key qualitative behavior is the sharp growth in step time (mean, and moreover – tail) for ECMP and flowlet at higher load, while adaptive routing remains comparatively stable.

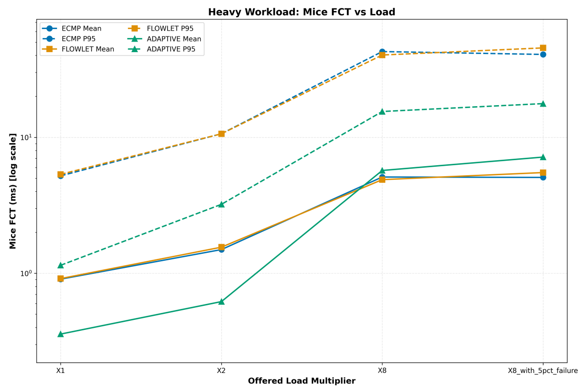


(a) Heavy workload

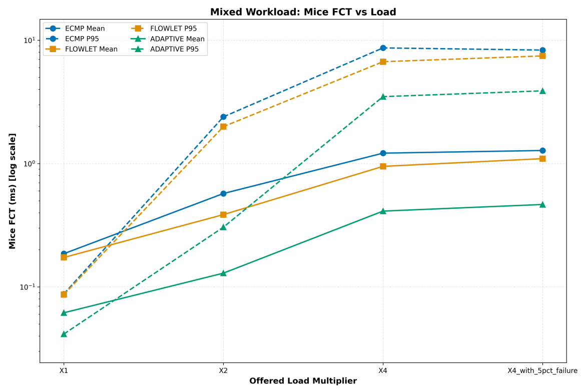


(b) Mixed workload

Figure 6: Step time (mean and p95) versus offered load.



(a) Heavy workload



(b) Mixed workload

Figure 7: Mice flow completion time (mean and p95) versus offered load.

4.2.2 Step time under load: mixed workload

Figure 6b shows the mixed workload. Compared to the single heavy workload, interference between jobs increases split tail from mean for ECMP and flowlets. In this scenario ECMP shows more sensitivity to congestion than flowlets, suggesting existence of the longer tail cases which eased by short flowlets. Adaptive routing exhibits lower mean with almost no variance (tail step time close to mean).

4.3 Mice Flow Completion Time (FCT)

(Figures: Heavy workload — Fig. 7a, Mixed workload — Fig. 7b)

Mice flow completion time (FCT) serves as a sensitive indicator of short-term congestion and transient queuing effects in the fabric. Unlike step-level or job-level completion time, mice FCT captures the experience of latency-sensitive traffic that competes with long-lived collective flows.

Heavy workload In the heavy workload scenario, mice traffic competes with a single dominant, communication-intensive job, creating a more sustained congestion environment.

At X1 and X2, adaptive routing shows better results in both mean and tail. As offered load increases to X8, ECMP and flowlet switching both enter a regime where mean is similar to adaptive, implying high load at average over the links, while the increasing tail mice FCT shows still better results for adaptive routing.

Overall, these results indicate that adaptive routing improves not only bulk training performance but also enables better performance at the tail cases of low-latency service under high load and failure conditions.

Mixed workload Under low offered load (X1), all routing schemes show similar mean and tail mice FCT, indicating that the fabric operates in a largely uncongested regime (the lower tail values are plot artifacts, as they show trend at edge). As load increases to X2 and X4, clear differences between the routing policies begin to appear.

In this case ECMP and flowlet behave quite similarly, with lower performance than adaptive routing. Although in all routing methods the tail cases are higher than their means, adaptive routing's tail still remains lower than ECMP and flowlets'. This indicates that queue-aware path selection helps prevent short flows from being trapped behind congested queues, even when the fabric is heavily loaded or partially impaired.

4.4 Queue Occupancy and Congestion Spread

(Figure: Average switch peak queue versus offered load — Fig. 8)
(Figure: Global queue peak versus offered load — Fig. 9)

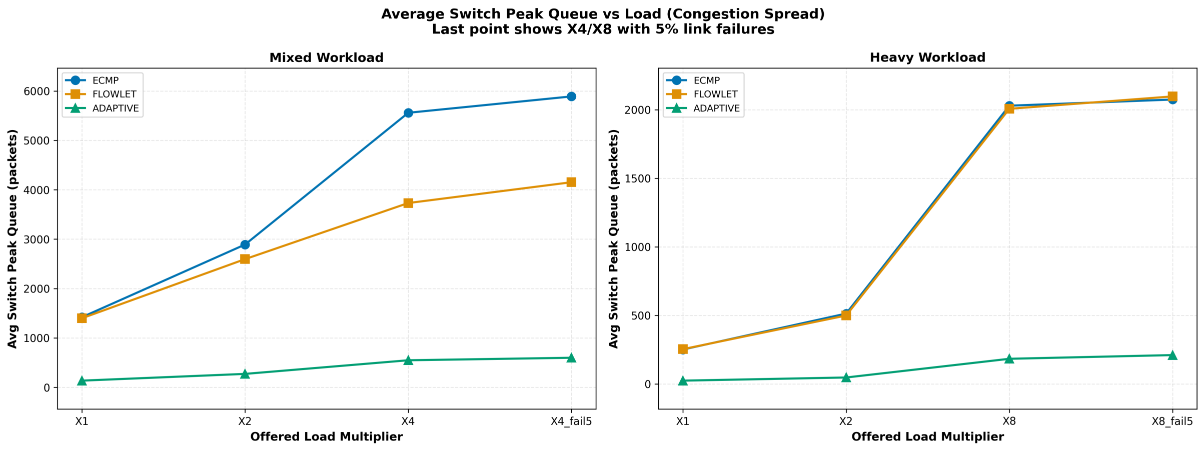


Figure 8: Average switch peak (mean of largest egress instance of each switch) queue vs. load

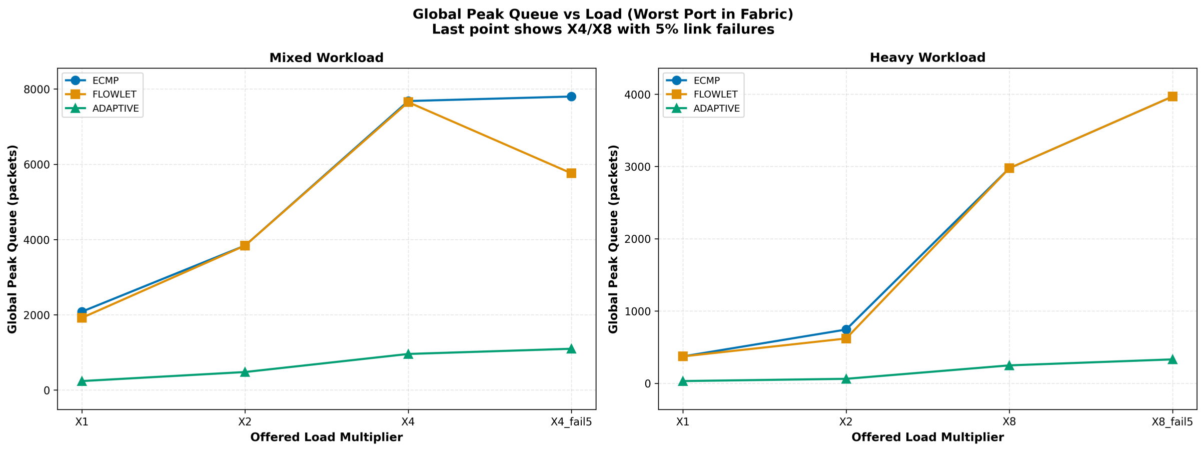


Figure 9: Global queue peak (largest egress over all switches) vs. load

Queue occupancy provides a structural view of congestion inside the fabric, revealing whether load imbalance is confined to a small number of hotspots or spread more evenly across the network.

Mixed workload In the mixed workload scenario, ECMP exhibits a rapid increase in average switch peak queue as offered load increases from X1 to X4. This growth reflects the formation of persistent queues on a subset of uplinks, consistent with hash-based load imbalance. The large peak queue values indicate that congestion is both severe and spatially concentrated.

Flowlet switching reduces average peak queue occupancy relative to ECMP, suggesting improved load distribution over time. However, peak queue sizes still grow substantially at higher load levels, indicating that flowlet-level granularity is insufficient to fully exploit available path diversity under sustained traffic pressure.

Adaptive routing shows a qualitatively different behavior. Across all load points, including X4 under the 5% link failure condition, average switch peak queue occupancy remains significantly lower than both ECMP and flowlet. This indicates that congestion is less severe and more evenly distributed, with queues draining before reaching large depths.

Heavy workload Under the heavy workload, both ECMP and flowlet switching experience sharp increases in average switch peak queue at X8, indicating that a large fraction of switches encounter sustained queuing. The similar behavior of ECMP and flowlet in this regime suggests that when traffic is dominated by a single large communication-intensive job, time-based flowlet switching alone is insufficient to prevent congestion concentration.

Adaptive routing again maintains substantially lower queue occupancy. Even at X8 with link failures, average switch peak queue values remain an order of magnitude smaller than those observed under ECMP and flowlet. This behavior aligns with the improvements observed in tail step time and mice FCT, as queues rarely reach depths that dominate latency.

Interpretation Taken together, the queue occupancy results suggest that the primary advantage of adaptive routing lies not only in reducing worst-case latency, but in fundamentally changing the congestion structure of the fabric. Rather than allowing long-lived queues to form on specific uplinks, adaptive routing continuously redistributes traffic across the available path parallelism, keeping queues shallow and short-lived.

This effect is particularly relevant in NVIDIA SU-style fabrics, where each logical entity-to-entity connection is backed by multiple parallel physical links. ECMP and flowlet switching tend to serialize traffic onto a single link for extended periods, leaving other links underutilized, whereas adaptive routing actively exploits the available parallelism.

5 Discussion

The step-time curves suggest that at higher offered load, ECMP and flowlet switching can enter regimes where traffic collisions create persistent queues on a subset of uplinks, inflating tail step time. Queue-aware adaptive routing, by directly reacting to local queue buildup, is able to steer traffic away from congested ports and reduce tail latency. Moreover, In NVIDIA SU-style fabrics each entity-to-entity connection is realized as an 8-way parallel link bundle. ECMP and flowlet switching typically do not fully exploit this available path parallelism: they map each flow (or each flowlet) onto a single member of the bundle for long intervals, so imbalanced hashing or unlucky flowlet assignments can leave some links underutilized while others become hotspots. A key question for future refinement is *transport realism*: in real RoCE deployments, packet reordering and congestion control (ECN/PFC, retransmission behavior) interact with routing. Integrating a more realistic RDMA transport model that includes congestion avoiding tools would strengthen the conclusions.

6 Conclusion

I presented a simulation-based comparison of ECMP, flowlet switching, and adaptive queue-aware routing under training-like traffic and link failures in an SU-like Clos pod. Across both workloads, adaptive routing demonstrates improved robustness at high load and under 5% random link failures, as reflected in lower step-time tail latency and reduced queue buildup. Future work may include parameter sweeps for flowlet thresholds, congestion avoidance methods, N-Back extreme reordering cases, richer failure models, and closer coupling to RoCE transport behavior.

References

[1] Edoardo Vanini, Radhika Pan, Mohammad Alizadeh, et al. “Let It Flow: Resilient Asymmetric Load Balancing with Flowlet Switching”. In: *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 2017. URL: `https://www.usenix.org/system/files/conference/nsdi17/nsdi17-vanini.pdf`.

[2] Keqiang He et al. “Presto: Edge-based Load Balancing for Fast Datacenter Networks”. In: *Proceedings of the ACM SIGCOMM 2015 Conference*. Available as PDF from University of Wisconsin. 2015. URL: https://pages.cs.wisc.edu/~akella/papers/presto-sigcomm15.pdf.

[3] NVIDIA. *NVIDIA Spectrum-X Network Platform Architecture*. Whitepaper / Architecture overview (PDF). 2024. URL: https://gzhls.at/blob/ldb/e/0/0/b/3ebe47fbbd87fe76a235d40ecedfd77c04a3.pdf.

[4] NVIDIA. *NVIDIA DGX SuperPOD Reference Architecture (DGX GB200)*. Reference Architecture (PDF). 2025. URL: https://docs.nvidia.com/dgx-superpod/reference-architecture-scalable-infrastructure-gb200/latest/_downloads/238ab33ee345a7241fc2f15eedc2ff6c/RA11338001-DSPGB200-ReferenceArch.pdf.

[5] NVIDIA. *Network Fabrics — NVIDIA DGX SuperPOD Reference Architecture Documentation*. Online documentation. 2025. URL: https://docs.nvidia.com/dgx-superpod/reference-architecture/scalable-infrastructure-b300/latest/network-fabrics.html.

[6] Huimin Luo et al. *SeqBalance: Congestion-Aware Load Balancing with no Reordering for RoCE*. arXiv:2407.09808. 2024. URL: https://arxiv.org/abs/2407.09808.

[7] Alon Zeltser. *Simulator log parameters: step counts for mixed workload*. Local experiment logs. 2026. URL: https://github.com/AlonZeltser/networks_for_AI_factories_and_datacenters_final_project/.

[8] Alon Zeltser. *Simulator log summary: mice FCT statistics*. Local experiment logs. 2026. URL: https://github.com/AlonZeltser/networks_for_AI_factories_and_datacenters_final_project/.