



# ECMP Collisions, Adaptive Routing and Packet Spraying for AI

Plan and build private AI compute with GPU Smith. We connect workload requirements with GPU hardware, networking, deployment and operating decisions so your infrastructure fits the work it must perform.

Published by GPU Smith · 2026-10-29 · ecmp hash collision ai training network · adaptive routing infiniband vs spectrum-x · packet spraying ultra ethernet · nccl\_ib\_qps\_per\_connection · elephant flows ai training fabric · programmable switches dynamic load balancing ai · nvidia spectrum-x vs infiniband

Original article: <https://gpusmith.com/articles/en/adaptive-routing-packet-spraying-ai-fabric-load-balancing>



## In this article

1. The symptom: idle uplinks while collectives crawl
  2. Why AI traffic breaks ECMP
  3. Hash polarization and what the hash can see
  4. A worked example: how bad is random hashing?
  5. Meta's progression: from pinning to E-ECMP
  6. NCCL knobs that change path diversity
  7. Per-packet adaptive routing: InfiniBand and Spectrum-X
  8. Packet spraying in Ultra Ethernet (UEC 1.1)
  9. Flowlets: the in-between option
  10. Diagnosing imbalance
  11. Choosing a strategy by fabric
  12. Operational caveats
- 

## The symptom: idle uplinks while collectives crawl

The typical complaint is a fabric that looks under-utilized on average while collectives run well below their expected bus bandwidth. Per-uplink counters on a leaf show some links near line rate and others near idle, and the busy set changes from job to job. The cause is usually not capacity but placement: equal-cost multipath (ECMP) hashes each flow onto one path, and AI training produces a few large, synchronized flows that collide on the same uplinks. The fixes, in rising order of disruption, are more entropy (more queue pairs plus a hash that sees them), adaptive routing that picks paths from live switch state, and packet spraying with endpoints that tolerate reordering. All of this assumes the topology is right; if the leaf-to-spine ratio is wrong, start with [fabric sizing and oversubscription](#).

## Why AI traffic breaks ECMP

Meta's description of its RoCE training fabrics names the three properties that matter ([Meta Engineering, 2024](#)):

- **Low entropy:** "the number and the diversity of flows for AI workloads are much smaller and the flow patterns are usually repetitive and predictable."
- **Burstiness:** flows are "on and off" at millisecond granularity.
- **Elephant flows:** "For each burst, the intensity of each flow could reach up to the line rate of NICs."

ECMP works when thousands of small, independent flows average out a random assignment. Meta's conclusion was blunt: "ECMP rendered poor performance for the training workload due to the low flow entropy" ([Meta Engineering](#)).

Synchronization makes it worse. A collective step waits for its slowest transfer, so one uplink carrying two line-rate flows halves the whole step even while other uplinks sit idle. Average utilization stays low while job time climbs.

## Hash polarization and what the hash can see

ECMP picks a next hop by hashing header fields. Meta's fabrics hashed "the five-tuple: source and destination IPs, source and destination UDP ports, and protocol" ([Meta Engineering](#)). Two separate problems hide in that sentence.

**Too few distinct inputs.** If a GPU pair talks over one RDMA connection, the five-tuple is constant for the whole job, so the path is fixed for the whole job. Adding connections helps only if they differ in a field the switch actually hashes. Meta had to configure switches to "additionally hash on the destination QP field of a RoCE packet using the UDF capability of the switch ASIC" before more queue pairs (QPs) produced more paths. Before relying on QP scaling, confirm on your own NIC and switch which header field changes between QPs and whether the hash includes it.

**Correlated hashing across tiers.** Polarization is the older, classic failure: when successive switch tiers run the same hash algorithm on the same inputs, they make correlated choices and some redundant paths "remain completely unused" ([Cisco TAC](#), [CEF Polarization](#)). Cisco's mitigations are to alternate hash inputs between layers or to seed each router's hash with a router-specific 32-bit value. On an AI fabric, check that leaf and spine do not share hash function and seed.

## A worked example: how bad is random hashing?

The numbers below are derived arithmetic, computed exactly with a short Python script and checked by Monte Carlo simulation (50,000 trials). They are a model, not a measurement.

### Assumptions (hypothetical):

- One leaf with  $M = 8$  uplinks, each at the same rate as a GPU NIC (say 400 Gb/s).
- $N = 8$  flows leave the leaf at the same moment, each wanting line rate, all bound for other leaves.
- The hash places each flow on an uplink uniformly and independently at random.
- Flows sharing an uplink split it evenly; the collective step finishes when the slowest flow finishes.

**Step 1: probability of zero collisions.** Every flow must land on a different uplink:  $8!/8^8 = 40,320/16,777,216 = 0.0024$ . A perfect layout happens about once in 416 attempts.

**Step 2: expected idle uplinks.** Each uplink is missed by all 8 flows with probability  $(7/8)^8 = 0.344$ , so the expected number of idle uplinks is  $8 \times 0.344 = 2.75$ . On a typical step, almost three of eight uplinks carry nothing.

**Step 3: distribution of the busiest uplink.** Exact probabilities for the maximum number of flows on one uplink:

| Flows on busiest uplink | Probability |
|-------------------------|-------------|
| 1                       | 0.24%       |
| 2                       | 49.8%       |
| 3                       | 40.9%       |

| Flows on busiest uplink | Probability |
|-------------------------|-------------|
| 4                       | 8.0%        |
| 5 or more               | 1.0%        |

Expected maximum load is 2.60 flows.

**Step 4: effective rate.** The slowest flow runs at  $400/k$  Gb/s when  $k$  flows share its uplink. Averaging  $1/k$  over the distribution gives 0.41, so the expected step rate is about  $0.41 \times 400 = 164$  Gb/s per flow instead of 400. Under these assumptions random ECMP delivers roughly 41% of the fabric's capacity to a synchronized collective.

**Step 5: what more flows buy.** Now give each GPU  $q$  queue pairs, with the hash able to tell them apart and bytes spread evenly across them, so  $N = 8q$  flows each carry  $1/q$  of the data. The ideal is  $N/M$  flows per uplink; the slowdown is the busiest uplink's load divided by that ideal.

| QPs per GPU ( $q$ ) | Flows $N$ | Expected max load | Slowdown vs ideal | Expected efficiency | P(busiest uplink at 2x ideal or more) |
|---------------------|-----------|-------------------|-------------------|---------------------|---------------------------------------|
| 1                   | 8         | 2.60              | 2.60x             | 41%                 | 99.8%                                 |
| 2                   | 16        | 4.20              | 2.10x             | 50%                 | 79.2%                                 |
| 4                   | 32        | 7.05              | 1.76x             | 58%                 | 30.1%                                 |
| 8                   | 64        | 12.24             | 1.53x             | 67%                 | 3.7%                                  |
| 16                  | 128       | 21.91             | 1.37x             | 74%                 | 0.07%                                 |

Entropy helps, but slowly: imbalance shrinks roughly with the square root of flows per link, so even 16 QPs per GPU leave an expected 1.37x penalty in this model. That is consistent with Meta's observation that QP scaling plus E-ECMP improved AllReduce "up to 40%" while "the underlying probabilistic nature of hashing was a persistent downside" ([Meta Engineering](#)). The model overstates penalties for loosely synchronized flows and understates them for multi-tier fabrics, where each tier adds collisions.

**Step 6: what doubling uplinks buys.** With 16 uplinks at NIC rate and the same 8 flows, the zero-collision probability is  $16!/(8! \times 16^8) = 0.121$ . There is still an 88% chance that at least one uplink carries two flows, and expected efficiency rises only to 53%. Over-provisioning lowers the odds without removing them.

## Meta's progression: from pinning to E-ECMP

Meta's published history is the best public account of these trade-offs on a production RoCE fabric ([Meta Engineering](#)):

1. **Plain ECMP** performed poorly because of low flow entropy.
2. **Path pinning** by destination "slice" (the rack switch downlink index) "worked well if each rack was fully assigned to the same job and there was no failure in the network. However, this was seldom true." Partial rack allocation "degraded the training performance up to more than 30%", and failures

reassigned flows unevenly.

3. **2x uplink capacity** ("1:2 under-subscribed") mitigated collisions but "required 2x network capacity"; Meta treated it as short-term.
4. **E-ECMP plus QP scaling** gave up to 40% better AllReduce versus baseline ECMP. Of two strategies, splitting messages across QPs or posting whole messages round-robin, round-robin performed well for Meta's production NCCL message sizes.

The cost: tuning the QP scaling factor per workload type "presented long-term operational complexity." Entropy tuning is a workable stopgap on a fabric without adaptive routing, not a set-and-forget fix.

## NCCL knobs that change path diversity

The relevant NCCL environment variables, quoted from the NCCL 2.32.3 documentation (v2.32.3-1, released 2026-09-17 and still the latest release on 2026-10-28; [NVIDIA NCCL docs](#)):

| Variable                                | What it does                                                                                                                     | Default                                                |
|-----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------|
| NCCL_IB_QPS_PER_CONNECTION (since 2.10) | "Number of IB queue pairs to use for each connection between two ranks"; useful on multi-level fabrics that need routing entropy | 1 (valid 1 to 128)                                     |
| NCCL_IB_SPLIT_DATA_ON_QPS (since 2.18)  | 1 splits each message evenly across QPs; 0 uses QPs round-robin per message                                                      | 0 since NCCL 2.20 (split was default in 2.18 and 2.19) |
| NCCL_IB_ADAPTIVE_ROUTING (since 2.16)   | Enables adaptive-routing-capable data transfers on the IB Verbs transport                                                        | 1 on InfiniBand, 0 on RoCE                             |
| NCCL_IB_SL                              | InfiniBand service level; an AR-enabled SL must be selected for adaptive routing                                                 | AUTO (since 2.32u1)                                    |

Raising NCCL\_IB\_QPS\_PER\_CONNECTION only adds paths if the switch hash distinguishes the QPs, which step 5 assumed; on RoCE, verify the hash fields first. NCCL's round-robin default matches what Meta found worked; split mode produces smaller on-wire messages and more ACKs. Test the QP count per job type and record it with each benchmark. The [NCCL algorithms and protocols](#) article covers how to confirm what NCCL actually selected.

## Per-packet adaptive routing: InfiniBand and Spectrum-X

Adaptive routing moves the path decision from a static hash to the switch's live state. NVIDIA's NetQ documentation defines it on Spectrum switches as "a load balancing feature that improves network utilization for eligible IP packets by selecting forwarding paths dynamically based on the state of the switch, such as queue occupancy and port utilization" ([NVIDIA NetQ 5.3](#)).

The catch is reordering. Once packets of one connection take different paths, they arrive out of order, and classic RoCE transports treat a gap as loss. Both NVIDIA fabrics push the fix into the NIC:

- **InfiniBand.** NVIDIA's DOCA documentation says "InfiniBand packets belonging to the same QP may traverse different network paths," and its out-of-order data placement feature accepts such packets rather than dropping them, placing data "in host memory in the order it arrives," on ConnectX-5 and later for RC, XRC and DC transports ([NVIDIA DOCA 3.5](#)). Quantum-X800 lists adaptive routing alongside telemetry-based congestion control and SHARP v4 on a 144-port, 800 Gb/s switch ([NVIDIA Quantum-X800](#)). NCCL enables AR by default on IB but requires an AR-enabled service level ([NCCL docs](#)).
- **Spectrum-X Ethernet.** NVIDIA describes adaptive routing as "a feature where the Spectrum-X Ethernet switch and SuperNIC work in tight coordination," targeted at lossless networking ([NVIDIA Spectrum-X](#)). An NVIDIA technical blog states that the Spectrum-4 switch "evenly sprays packets across multiple paths," that "the challenge of out-of-order packet delivery arises," and that SuperNICs resolve it by "placing packets directly in order into the buffer as they arrive" ([NVIDIA Technical Blog, 2024-10-15](#)). NVIDIA does not publish the path-selection algorithm itself; this blog is the most specific NVIDIA description we found. NetQ displays adaptive routing data only with RoCE lossless mode enabled, on Spectrum-4 and Cumulus Linux 5.5.0 or later ([NetQ 5.3](#)).

**Vendor claims, labelled.** NVIDIA's product page claims Spectrum-X accelerates "AI network performance by 1.6x over off-the-shelf (OTS) Ethernet" and "1.9x higher NCCL performance in cross-data center environments" ([NVIDIA Spectrum-X](#)). The page gives no test conditions (the 1.6x figure links to an NVIDIA whitepaper), and we found no independent measurement; measure on your own topology. NCCL leaves `NCCL_IB_ADAPTIVE_ROUTING` off by default on RoCE, so confirm what the vendor's reference configuration expects. For the broader platform choice, see [InfiniBand vs Spectrum-X vs RoCE](#).

Per-packet adaptive routing replaces the worked example's random placement with load-aware placement, but only with a matched switch and NIC. Spraying to a NIC that cannot place out-of-order data converts imbalance into retransmissions.

## Packet spraying in Ultra Ethernet (UEC 1.1)

The Ultra Ethernet Consortium released Specification 1.0 on June 11, 2025, followed by 1.0.1, 1.0.2 and 1.0.3; the current published version is 1.1, released October 12, 2026, and still current on October 28, 2026 ([UEC specification history](#)). The wording below is from the 1.1 PDF ([UE Specification 1.1](#)).

UEC takes a different route from Spectrum-X: the endpoint, not the switch, does the spraying, over ordinary ECMP. The specification says "UE switches use equal cost multipath (ECMP) routing for load balancing where the entropy values are managed by the UET congestion management sublayer," and expects switches not to modify entropy values, so packets with the same value take the same path (section 1.6). The entropy value is a header field such as the UDP source port; varying it lets packet delivery contexts ("PDCs") "with unordered flows" use "all paths to the destination."

Section 3.6.16 (Multipath Path Selection) defines three modes:

- **Single path**, used only by reliable ordered delivery (ROD); the entropy value changes only occasionally, and it is recommended that this happen no more often than every `t_reroute` RTTs (default at least 10 RTTs).

- **Oblivious spraying:** the entropy value is varied "through a large space (typically at least 256 values)," chosen pseudo-randomly.
- **Path-aware spraying:** the entropy value is changed adaptively "to avoid congested paths, based on feedback provided by the destination as to which EVs saw congestion."

Multipath load balancing is RECOMMENDED for unordered (RUD and RUDI) traffic, with implementation left to vendors. The spec is candid that "Hashing is not guaranteed to spread the packets evenly," which is why path-aware spraying exists. One line matters for mixed fabrics: switches that "perform packet spraying themselves" should not do so for UET traffic when path-aware spraying is used, "as it prevents UET getting a clean signal as to which paths are congested." Switch-side spraying and endpoint-side path-aware spraying are alternatives, not layers to stack.

Reordering makes loss detection harder, so the spec adds supporting signals: optional packet trimming, in which switches truncate packets and forward the header as a congestion signal (optional for switches, mandatory for endpoints to receive); ECN marked at dequeue; Link Layer Retry (section 5.1); and Congestion Signaling (CSIG), a lightweight in-band telemetry mechanism.

## Flowlets: the in-between option

Flowlet switching re-hashes a flow only at idle gaps. The LetFlow paper defines a flowlet as "a burst of packets that is separated in time from other bursts by a sufficient gap," where the gap "must exceed a threshold (the flowlet timeout) that is larger than the difference in latency among the paths," so packets still arrive in order (Vanini et al., NSDI 2017). Juniper's AI-ML documentation implements this as a dynamic load balancing (DLB) flowlet mode with an "inactivity interval" set "higher than the maximum latency," alongside a per-packet mode that "might experience packet reordering problems" and should be enabled only for endpoints "capable of handling out-of-order packets" (Juniper selective DLB).

Design reasoning, not measurement: flowlets fit AI traffic less well than TCP traffic. LetFlow's resilience comes from TCP congestion control shrinking flowlets on slow paths. RDMA elephants run at line rate with few gaps inside a burst, so a flowlet often equals the whole burst, and rebalancing happens between collective steps rather than within one. Flowlet DLB is a reasonable upgrade for a RoCE fabric whose NICs cannot reorder; test it rather than assume per-packet benefits.

## Diagnosing imbalance

Use counters, not averages:

1. **Per-uplink transmit utilization during a collective, at short intervals.** A max/mean near 2.6 with 8 flows on 8 uplinks is random hashing at work, not a fault.
2. **Queue-depth distributions.** NetQ reports P95, standard deviation, mean and maximum queue lengths and flags "ECMP traffic imbalances" on Spectrum-4 (NetQ 5.3).
3. **ECN, CNP and PFC pause counters per port.** Concentration on a few uplinks points to collisions; uniform marking points to undersized capacity.
4. **NIC out-of-sequence and retransmission counters** after enabling any spraying mode.

5. **nccl-tests bus bandwidth across several job placements.** Placement-dependent variance is the signature of hash collisions.

The [RoCE acceptance testing](#) runbook gives the counter-baseline procedure; capture a baseline before changing any load-balancing setting.

## Choosing a strategy by fabric

| Fabric                       | Primary mechanism                                      | Endpoint requirement                           | Main risk                                                                                        |
|------------------------------|--------------------------------------------------------|------------------------------------------------|--------------------------------------------------------------------------------------------------|
| InfiniBand (Quantum class)   | Switch adaptive routing, on in NCCL by default         | ConnectX out-of-order placement; AR-enabled SL | AR-enabled SL not selected, so traffic does not use AR                                           |
| Spectrum-X Ethernet          | Switch spraying plus SuperNIC placement                | Spectrum-X SuperNIC; lossless RoCE mode        | Gains are vendor claims until measured; NIC and switch must be matched                           |
| Generic RoCE, static hashing | E-ECMP on QP field plus NCCL QP scaling                | None beyond hash configuration                 | Probabilistic imbalance remains (1.37x expected at 16 QPs/GPU in the model); per-workload tuning |
| Generic RoCE with switch DLB | Flowlet mode; per-packet only for reorder-capable NICs | Reordering support for per-packet              | Few flowlet gaps inside RDMA bursts                                                              |
| Ultra Ethernet (UEC 1.1)     | Endpoint spraying over standard ECMP                   | UET-capable NICs                               | Product maturity; do not stack switch spraying on path-aware spraying                            |

## Operational caveats

- **Rails change the problem.** Same-rail traffic that stays on one leaf never hashes onto uplinks; apply the collision math to spine-bound traffic only.
- **Failures re-hash.** Meta's experience shows that a failed uplink or spine reassigns flows unevenly. Re-measure imbalance in degraded states.
- **Mixed vendors.** Adaptive routing needs agreement between switch and NIC. A Spectrum-X switch with a non-SuperNIC endpoint, or a UEC endpoint behind a spraying switch, may lose the benefit or add reordering.
- **Pin versions.** NCCL defaults changed between 2.18 and 2.20; record NCCL, NIC firmware and switch OS versions with every benchmark.

GPU Smith helps teams size and validate fabrics like these, including load-balancing tests in failure states.

## Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. <https://engineering.fb.com/2024/08/05/data-center-engineering/roce-network-distributed-ai-training-at-scale>
2. <https://www.cisco.com/c/en/us/support/docs/ip/express-forwarding-cef/116376-technote-cef-00.html>

3. <https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/env.html>
4. <https://docs.nvidia.com/networking-ethernet-software/cumulus-netq-53/Monitor-Operations/Monitor-ECMP>
5. <https://networking-docs.nvidia.com/doca/sdk/out-of-order-data-placement>
6. <https://www.nvidia.com/en-us/networking/products/infiniband/quantum-x800>
7. <https://www.nvidia.com/en-us/networking/spectrumx>
8. <https://developer.nvidia.com/blog/powering-next-generation-ai-networking-with-nvidia-supernics>
9. <https://ultraethernet.org/specification-history>
10. <https://ultraethernet.org/wp-content/uploads/sites/20/2026/10/UE-Specification-1.1.pdf>
11. <https://people.csail.mit.edu/alizadeh/papers/letflow-nsdi17.pdf>
12. <https://juniper.net/documentation/us/en/software/junos/ai-ml-evo/topics/topic-map/dlb-selective.html>

## THE PUBLISHER

# About GPU Smith

GPU Smith provides independent engineering and research for private AI infrastructure. We help technical buyers and operators reason about GPU workload sizing, hardware selection, procurement, deployment and inference operations, from the compute system to the networking, power and cooling requirements around it.

## Start with the workload

Useful infrastructure decisions begin with the models, data, throughput, latency and operating constraints a team actually has. GPU Smith helps connect those requirements with system design and deployment choices. The goal is a privately controlled AI environment whose capacity and operational demands are understood before a purchasing decision.

## Hardware and supplier research

Our public reference library covers GPUs, complete systems and networking components. The server-vendor directory and manufacturing research help teams investigate suppliers, compare options and follow sources behind technical claims. We distinguish manufacturer specifications from measured benchmarks and advertised capabilities from tested configurations. Reference pages are research resources, not a live inventory listing or a binding equipment quote.

## Deployment and operations

GPU Smith's engineering scope includes infrastructure integration, networking, power, cooling and the practical operation of inference workloads. Our published articles explain the tradeoffs behind deployment, procurement and ongoing operations so teams can ask better questions and document decisions.

## Work with GPU Smith

Explore the [hardware library](#), [GPU references](#), [networking references](#) and [vendor research](#). [Contact GPU Smith](#) with your workloads and deployment constraints to discuss a project and confirm current scope, pricing and availability.

Inclusion of a manufacturer or product in our research does not imply a partnership, certification or endorsement.

**Website:** <https://gpusmith.com>

---

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.