NETDEV 0x1A · JULY 2026

Resilient AI Supercomputer Networking using MRC and SRv6

Multipath RDMA with static SRv6

Christoph Paasch · OpenAI

Resilient AI Supercomputer Networking using MRC and SRv6

OpenAI	Microsoft	AMD	Broadcom	NVidia
Joao Araujo	*S. Anantharamu*	*K. Doddapaneni*	*Eric Davis*	*Aviv Barnea*
Alex Chow	*Abhishek Dosi*	*Murali Garimella*	*Dragos Dumitrescu*	*Idan Burstein*
*Mark Handley**	*Mohit Garg*	*Vipin Jain*	*Mohan Kalkunte*	*Elazar Cohen*
Ryder Lewis	*Mahdieh Ghazi*	*Yanfang Le*	*Bhaswar Mitra*	*Yamin Friedman*
Christoph Paasch	*Torsten Hoefler*	*H. Nagulapalli*	*Guglielmo Morandin*	*Noam Katz*
Jitendra Padhye	*Deepal Jayasinghe*	*S. Narayanan*	*Adrian Popa*	*Masoud Moshref*
Michael Papamichael	*Jithin Jose**	*Rong Pan*	*Costin Raiciu*	*Yuval Shpigelman*
Greg Steinbrecher	*Abdul Kabbani*	*Rathina Sabesan*	*Eric Spada**	*Shahaf Shuler*
Amin Tootoonchian	*Guohan Lu*	*Raghava Sivaramu*	*John Spillane*	*Shy Shyman*
Lihua Yuan	*Yang Wang*	*Rip Sohan**	*Niranjan Vaidya*	*Sayantan Sur**

Thousands of GPUs advance one step at a time

One shared step

The same loop, over and over.

- **Compute locally:** Every GPU works on its part of the same training step.
- **Exchange over the network:** GPUs share the data needed to finish the step.
- **Advance together:** Step N+1 waits for every required GPU.

Tail-latency drives step-time

Thousands of GPUs advance one step at a time

One shared step

The same loop, over and over.

- **Compute locally:** Every GPU works on its part of the same training step.
- **Exchange over the network:** GPUs share the data needed to finish the step.
- **Advance together:** Step N+1 waits for every required GPU.

Failure implies restart from snapshot

The network has one job: keep every step moving

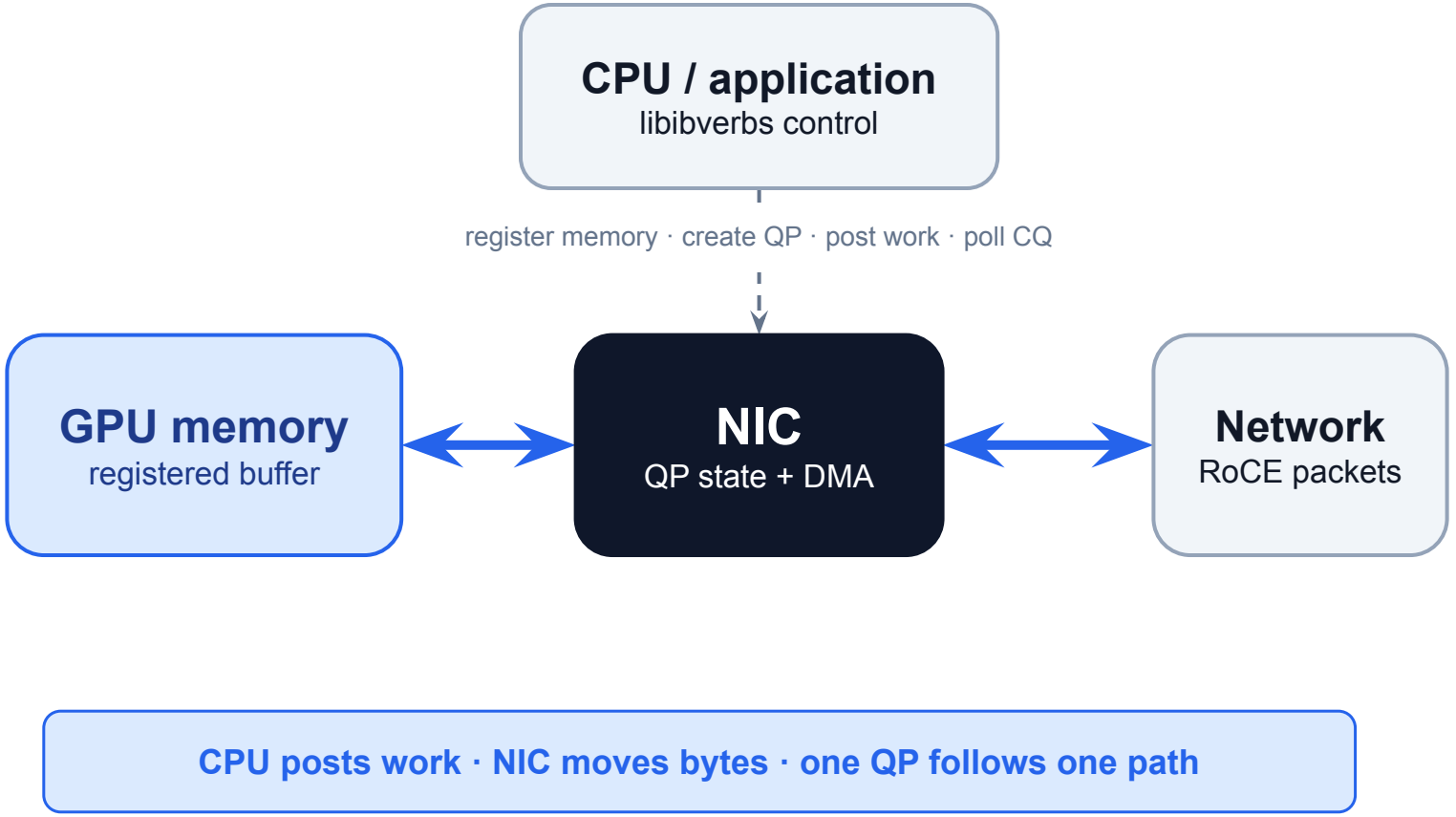
MRC Goals & Requirements

- **Use all healthy capacity:** High throughput and low p100 latency.
- **Lose capacity, not the job:** Restarting the job takes time. So, avoid restarts at the cost of speed.
- **Recover automatically:** Detect, remove, and replace bad paths without a human in the loop.

RoCE 101

RDMA semantics over Ethernet

- **QP = connection-like state.** For path selection, one QP behaves like one flow on one path.
- **CPU setup.** CPU used for control operation and posting work/polling completions.
- **NIC data path.** NICs place the bytes directly into the registered GPU buffer. Memory address information is part of each message.

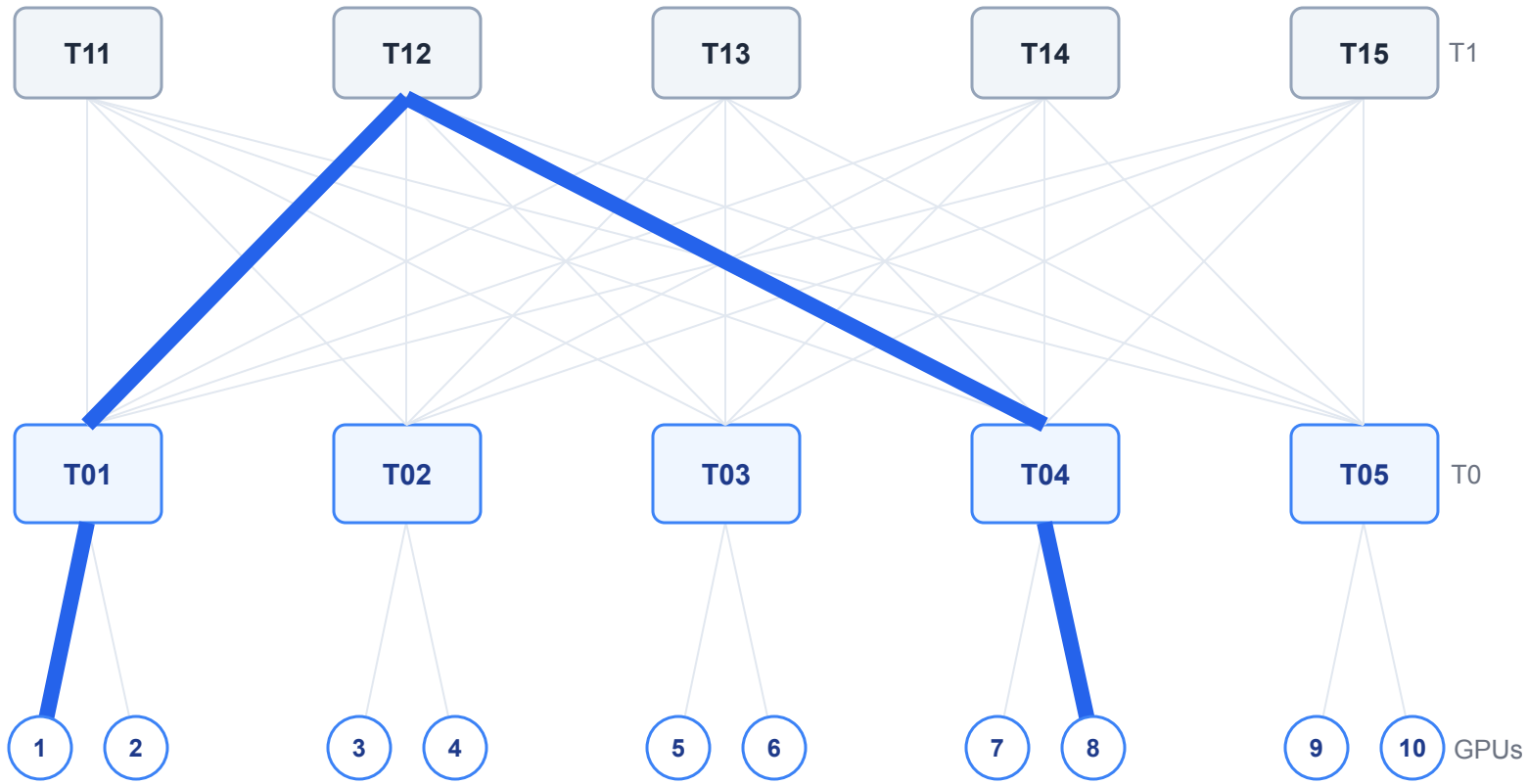


Regular RoCE picks one path

Static Pathing

How Regular RoCE operates

- Each network flow is statically pinned to a single path using packet header hashing.
- Traffic cannot dynamically shift, leaving alternative paths completely idle.
- This static allocation leads directly to path collision and tail latency issues.

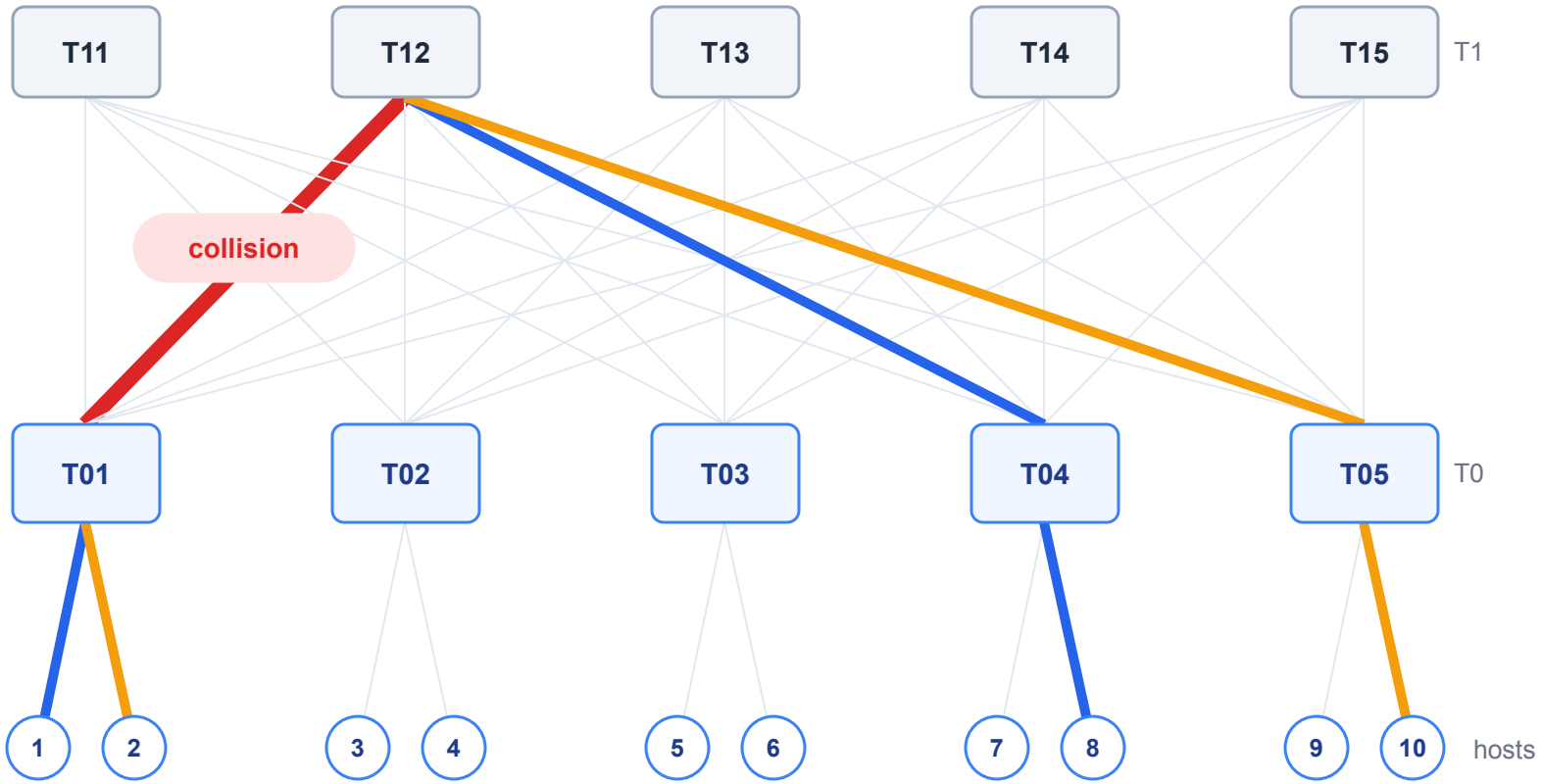


Sometimes several flows pick the same path

Path Collision

The limits of static hashing

- **Flow overlap:** Multiple independent network flows can randomly hash to the exact same physical path.
- **Congestion & queuing:** This overlap creates a critical bottleneck, forcing packets to queue up at the shared link.
- **Tail latency:** While alternative paths remain completely idle, the collided flows suffer from throughput drops.



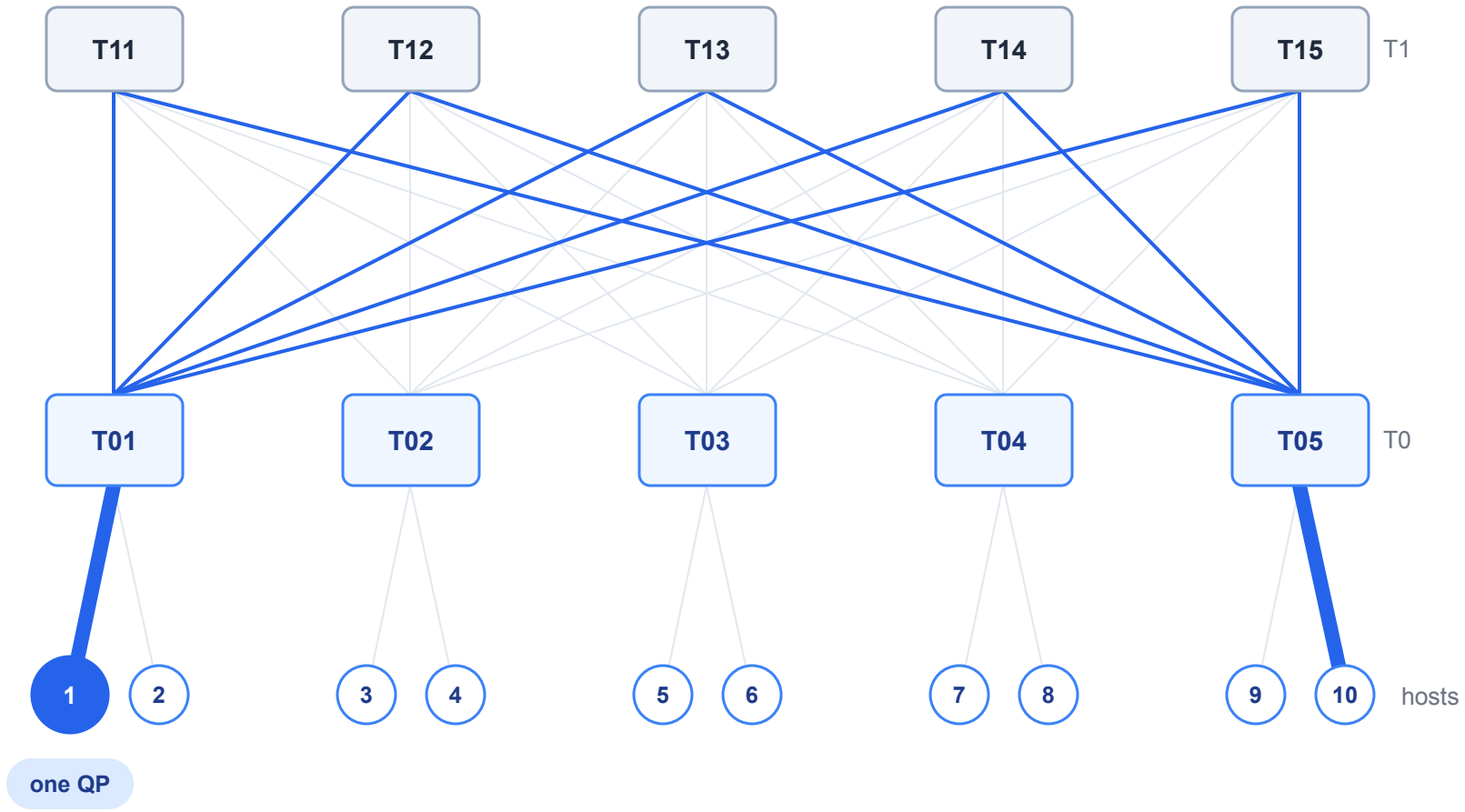
So MRC sprays packets

Packet Spraying

One QP across all available paths

Use all available paths
MRC sprays packets from one QP across the fabric.

One connection visible to the app
The application still creates and uses one logical QP.



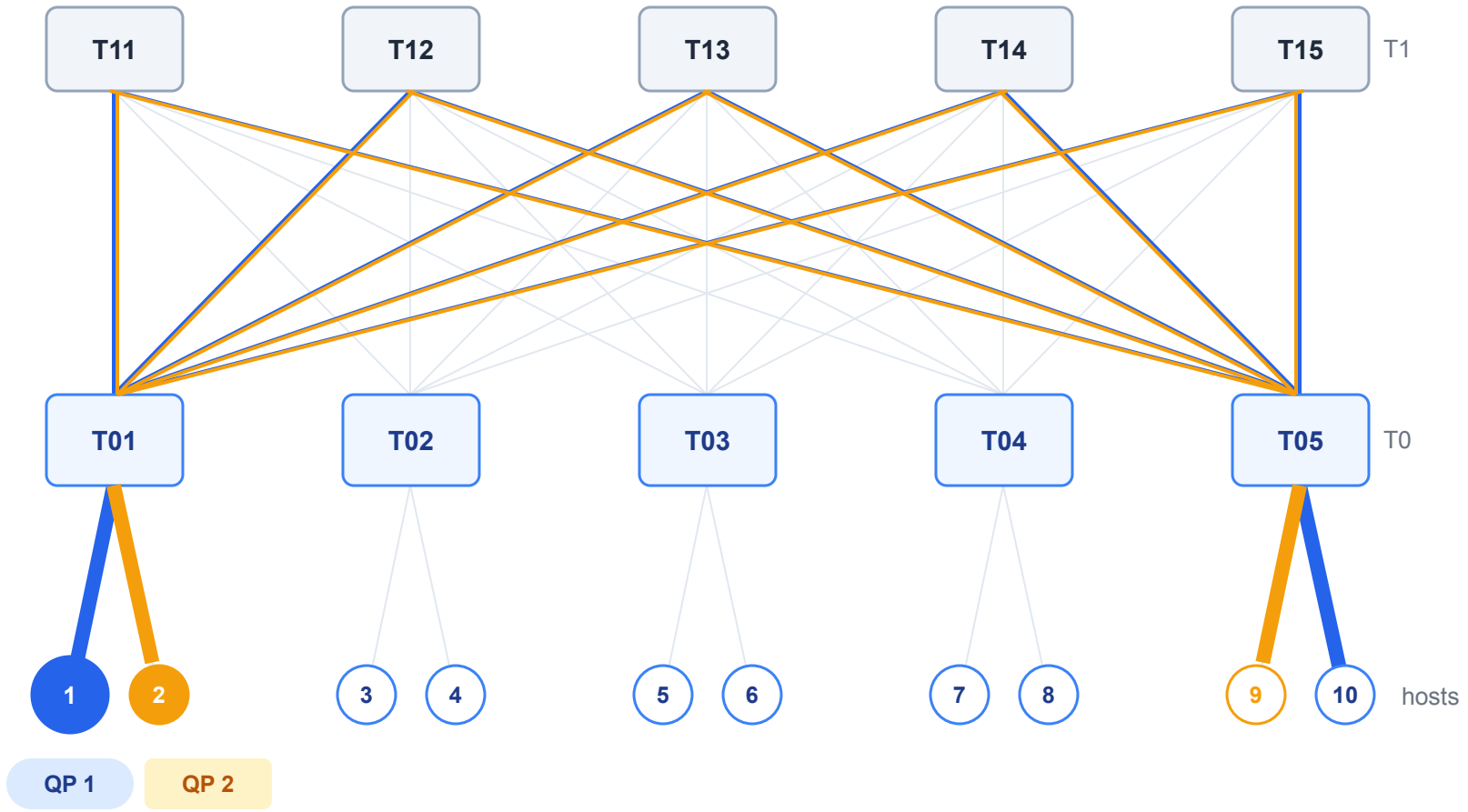
Multiple QPs share every path

Packet Spraying

Two QPs across all available paths

Use all available paths
MRC sprays packets from both QPs across the fabric.

Every path carries both QPs Each T0–T1 link carries a small share of both QPs.

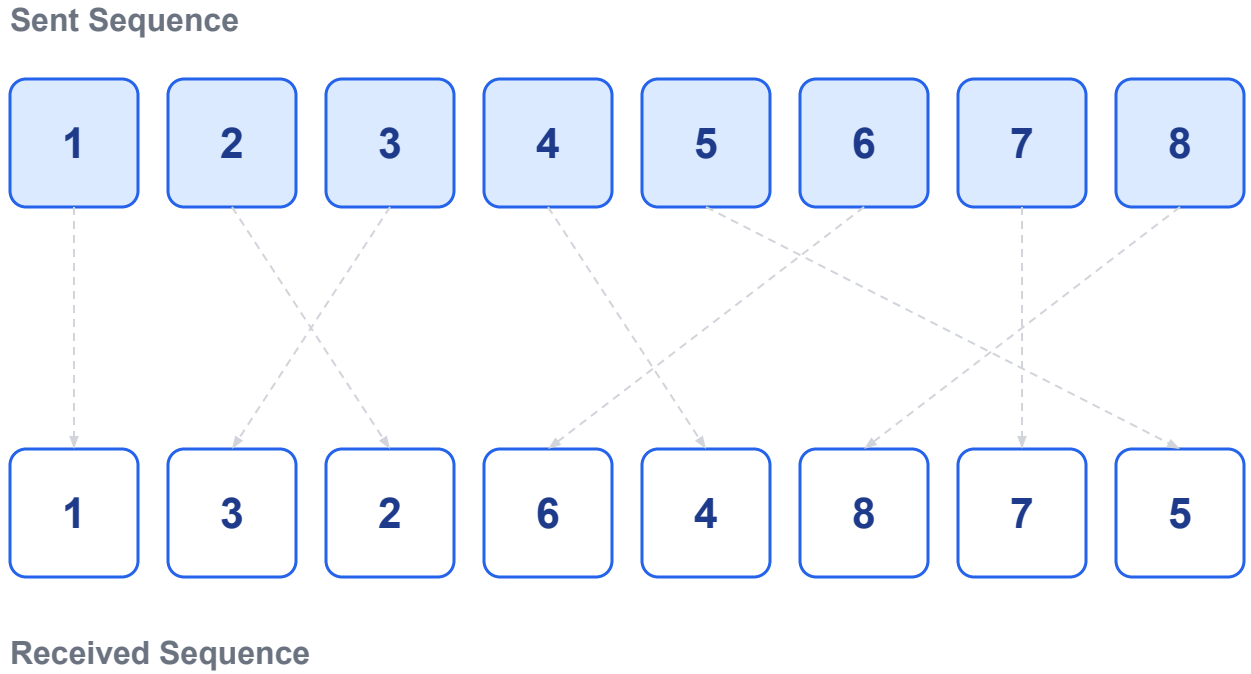


Now packets arrive out of order

Out-of-Order Delivery

The side-effect of packet spraying

- **Collision resolution:** Packet spraying successfully solves physical path collisions by using multiple routes.
- **Out-of-order delivery:** Because packets take paths with different latencies, they arrive out of their original sequence.
- **Receiver impact:** The receiving host can no longer rely purely on chronological arrival order to process data.

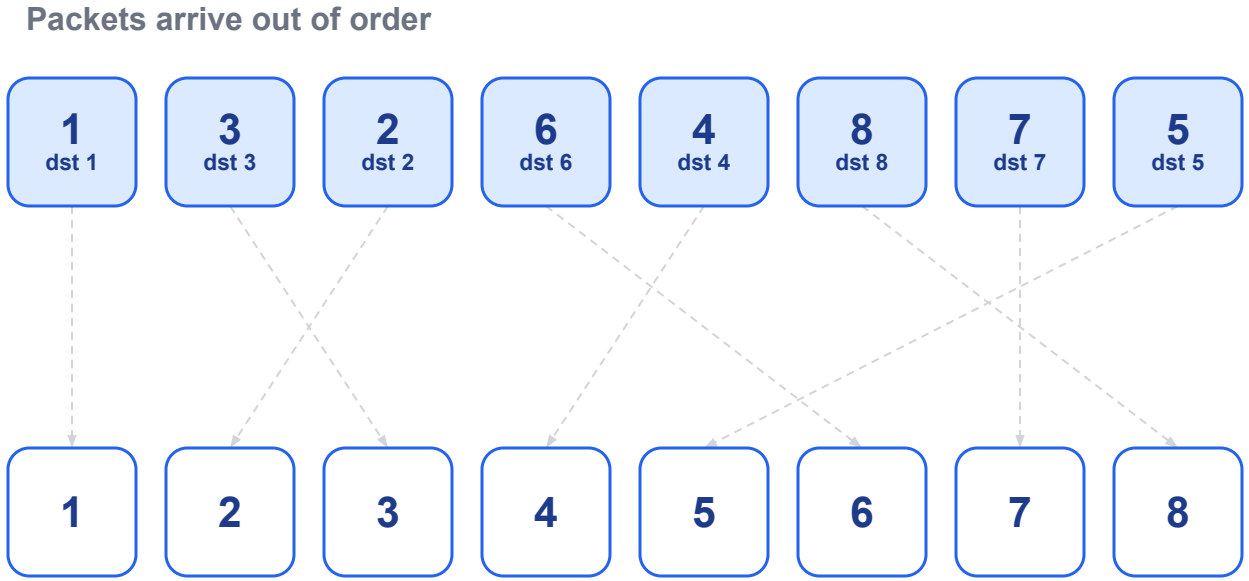


So every packet says where it belongs

Direct Placement

The answer to out-of-order delivery

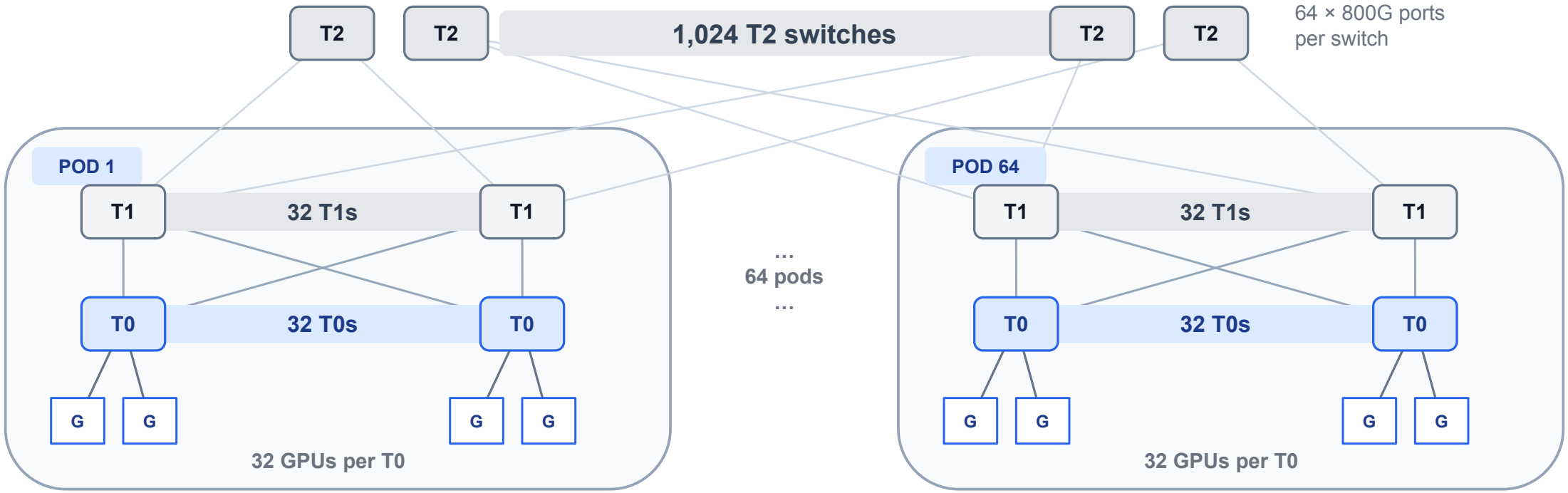
- **Per-packet address:** MRC carries the final GPU-memory address with every packet.
- **Immediate placement:** The NIC writes each packet directly into its destination slot, regardless of arrival order.
- **No reassembly:** No reorder buffer and no intermediate copy inside the NIC.



Direct placement. No reorder buffer. No reassembly.

A traditional three-tier fabric stops at 65K GPUs

51.2 Tbps switches · 64 × 800G ports · one NIC per GPU

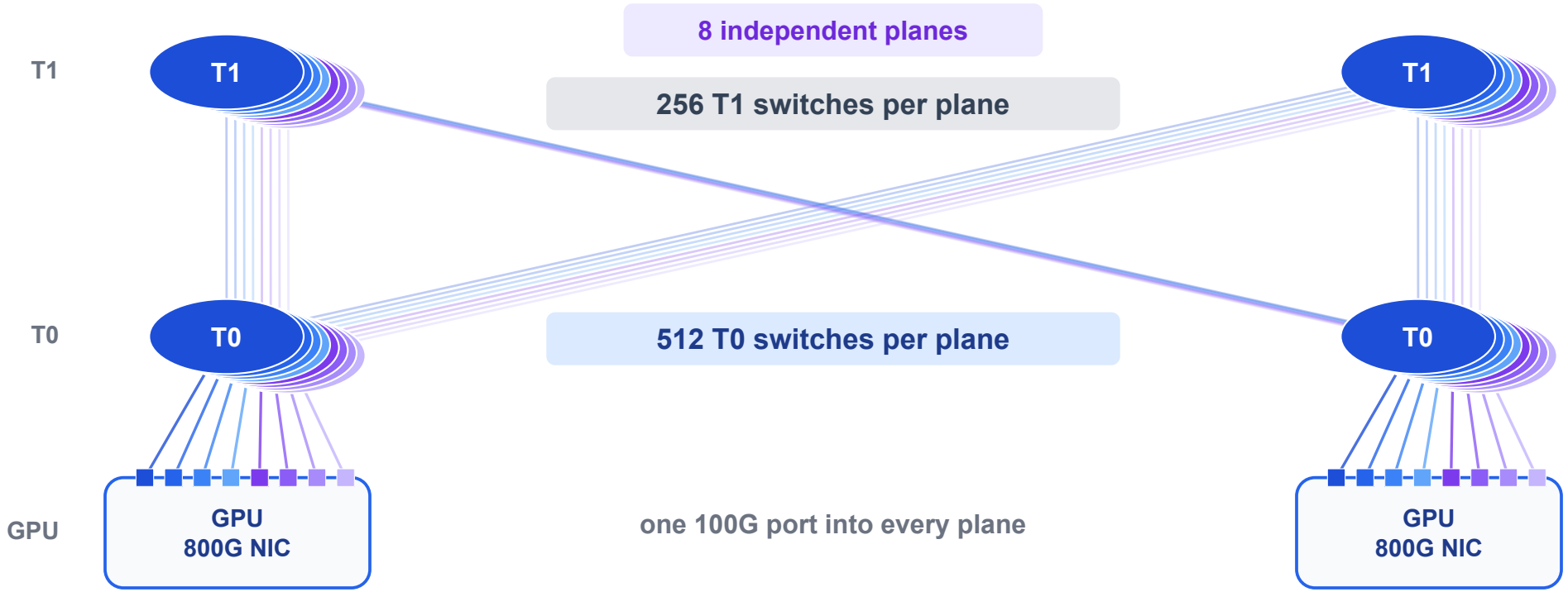


64 pods × 32 T0s × 32 GPUs = 65,536 GPUs

Not enough for the largest training jobs.

Eight two-tier planes reach 131K GPUs

800 Gbps per GPU = eight independent 100 Gbps ports



512 T0s × 256 ports = 131,072 GPUs

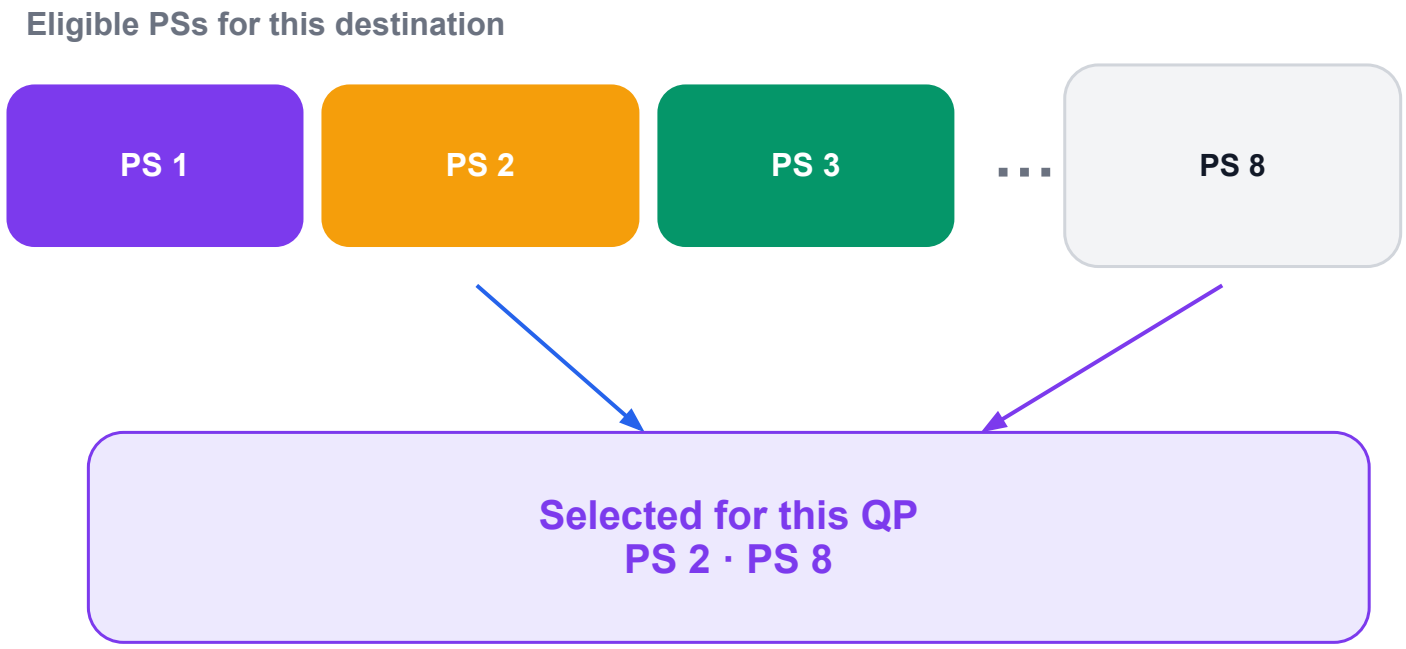
A multi-plane fabric needs a multi-plane transport.

For each destination, a set of Path Selectors is eligible

Path Selector (PS)

One PS represents one path to this destination

- **Eligible set:** Destination and topology determine which PSs can be used.
- **Path choices:** Each PS selects one NIC port → plane and one T0 uplink.
- **QP setup:** MRC chooses a subset of those eligible PSs.



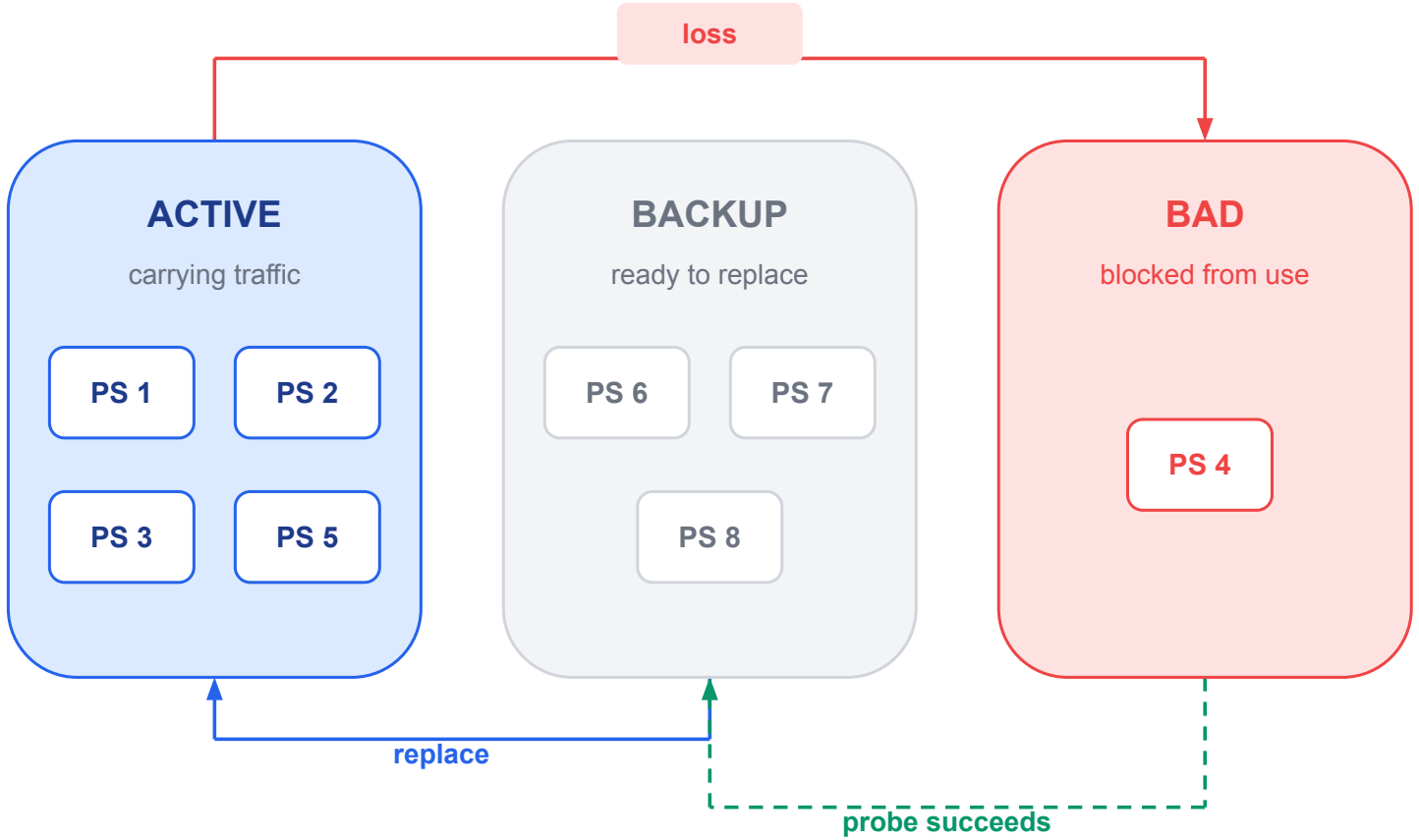
One PS = one path, for this destination.

Bad PS out. Backup PS in. QP stays up.

Path Selector Lifecycle

Each selected PS represents one path to this destination.

- **Active:** PSs currently carry traffic. MRC sends round-robin on them.
- **Backup:** Spare PSs are ready to replace a bad one.
- **Bad:** Loss or a failed probe removes a PS from service.
- **Recovery:** Successful probes can return a PS to the backup set.



Path changes. QP does not.

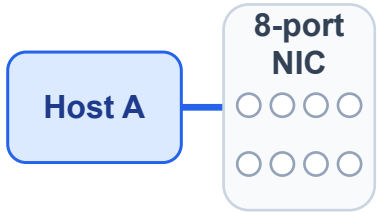
The PS derives the SRv6 destination address

Build the address at send time

Destination template + selected PS

Destination template: Config and destination supply the fixed bits.

- **Selected PS:** Supplies the plane/NIC port and T0 uplink.
- **Result:** One complete SRv6 destination address for this packet.



Address assembled at source before transmission.

locator
32 bits

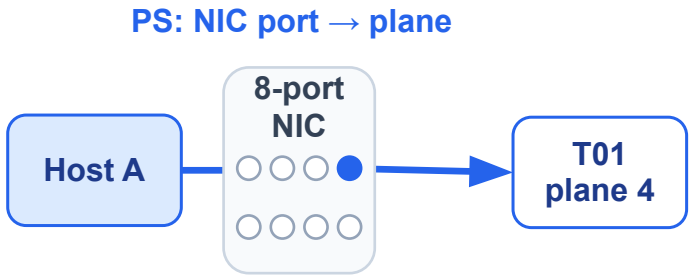
The PS derives the SRv6 destination address

Build the address at send time

Destination template + selected PS

Destination template: Config and destination supply the fixed bits.

- **Selected PS:** Supplies the plane/NIC port and T0 uplink.
- **Result:** One complete SRv6 destination address for this packet.



Address assembled at source before transmission.

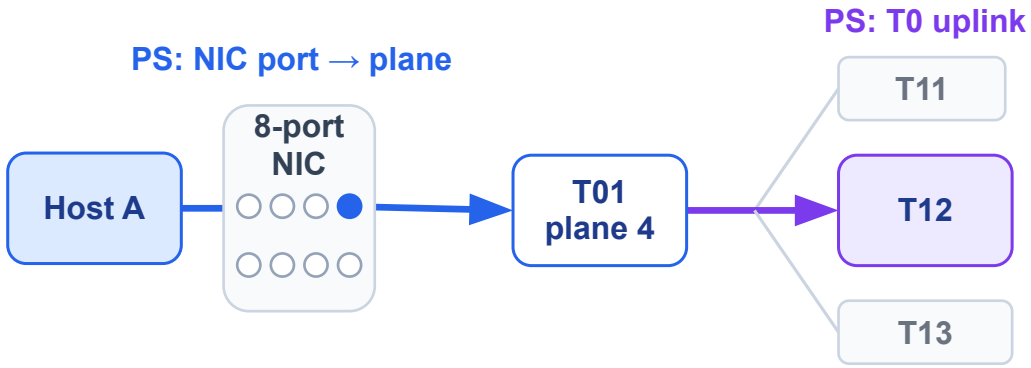
The PS derives the SRv6 destination address

Build the address at send time

Destination template + selected PS

Destination template: Config and destination supply the fixed bits.

- **Selected PS:** Supplies the plane/NIC port and T0 uplink.
- **Result:** One complete SRv6 destination address for this packet.



Address assembled at source before transmission.

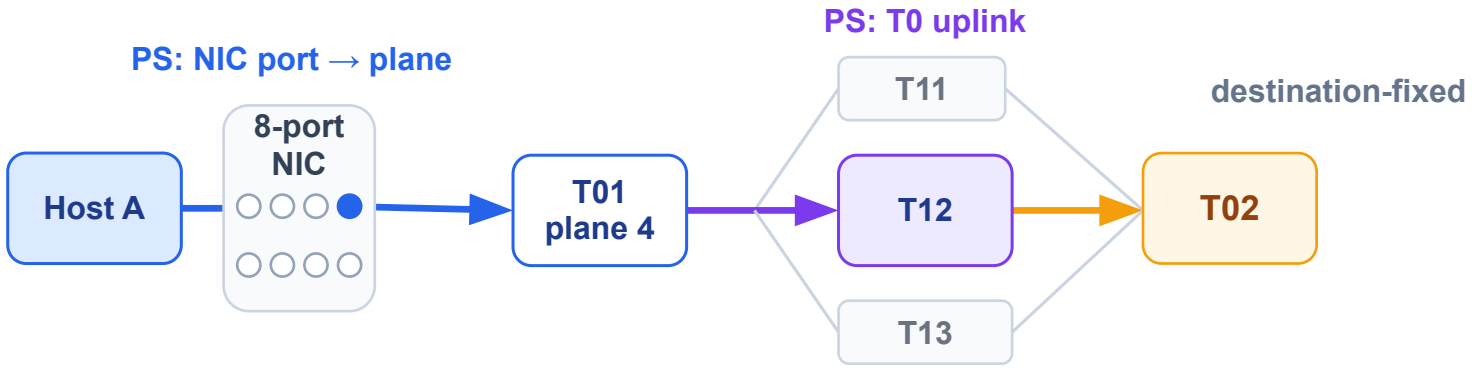
The PS derives the SRv6 destination address

Build the address at send time

Destination template + selected PS

Destination template: Config and destination supply the fixed bits.

- **Selected PS:** Supplies the plane/NIC port and T0 uplink.
- **Result:** One complete SRv6 destination address for this packet.



Address assembled at source before transmission.

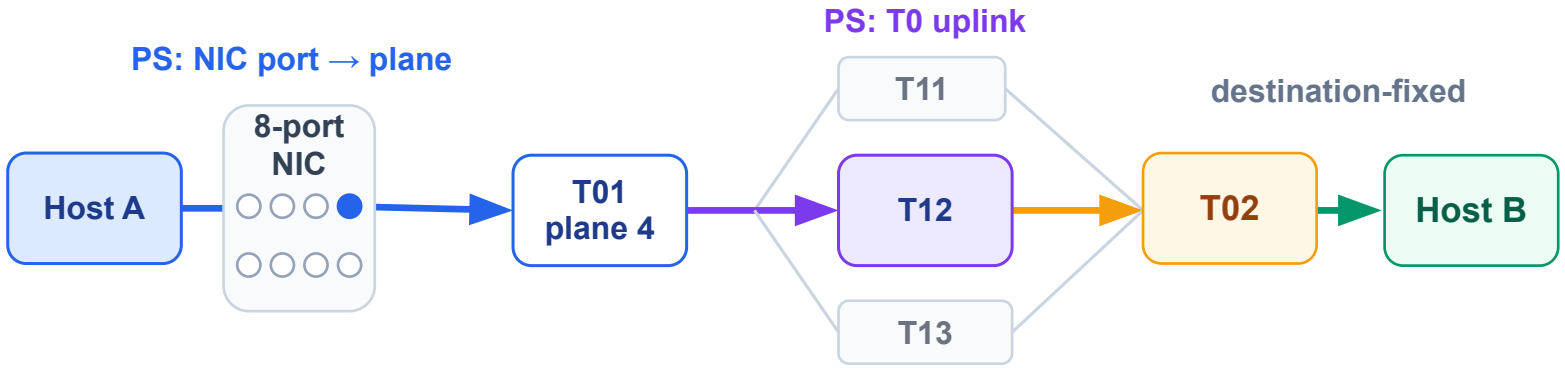
The PS derives the SRv6 destination address

Build the address at send time

Destination template + selected PS

Destination template: Config and destination supply the fixed bits.

- **Selected PS:** Supplies the plane/NIC port and T0 uplink.
- **Result:** One complete SRv6 destination address for this packet.



Address assembled at source before transmission.

Complete SRv6 destination address

SRv6 makes that path deterministic

Benefits of SRv6

- **Allows to keep the network simple:** Switches just forward traffic. Everything is statically configured. No resilient ECMP hashing, no BGP convergence, …
- **End-hosts can deterministically steer traffic:** MRC path selector mechanism routes around failures within one RTT
- **Reliable path-probing without the need for a peer:** `clustermapper` allows to probe paths by creating a loopback SRv6 probe. This can feed as a *denylist* into MRC.

MRC is implemented by three NIC vendors

MRC NIC implementations

400 and 800 Gbps RDMA NICs

NVIDIA — ConnectX-8 & ConnectX-9

AMD — Pollara and Vulcano

Broadcom — Thor Ultra

At OpenAI deployed at large scale

4 x 200 Gbps planes

8 x 100 Gbps planes

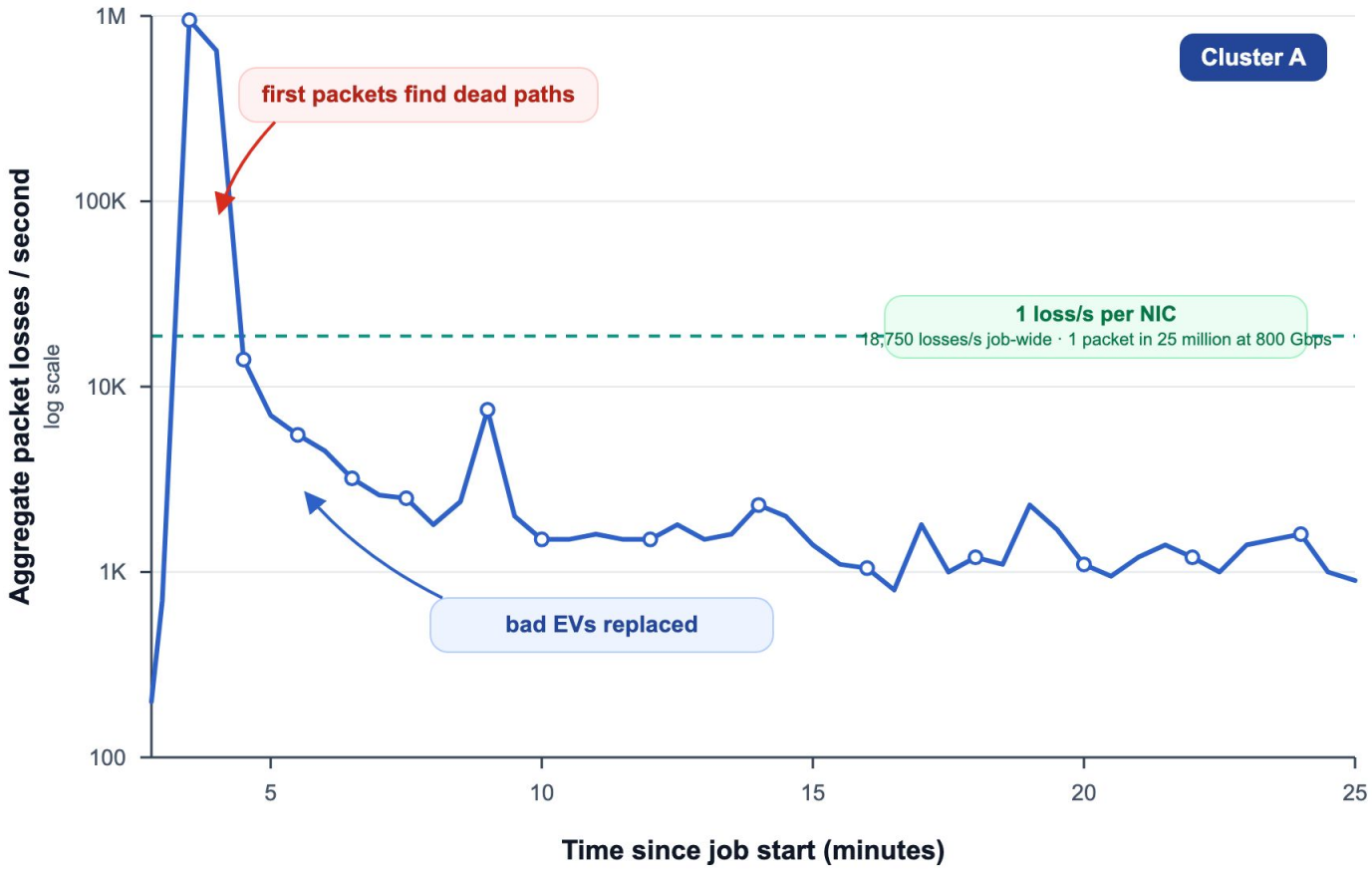
Two-tier, multi-plane production clusters

MRC maps out bad paths on its own

Startup without a denylist

75K-GPU production pretraining job

- **No prior map.** The job starts without pre-populating the denylist.
- **Loss is path feedback.** Packets are retransmitted; bad PSs are replaced from the backup set.
- **Converges quickly.** Below 1 loss/s/NIC within a couple of minutes. <5 packets/QP lost in the first minute.



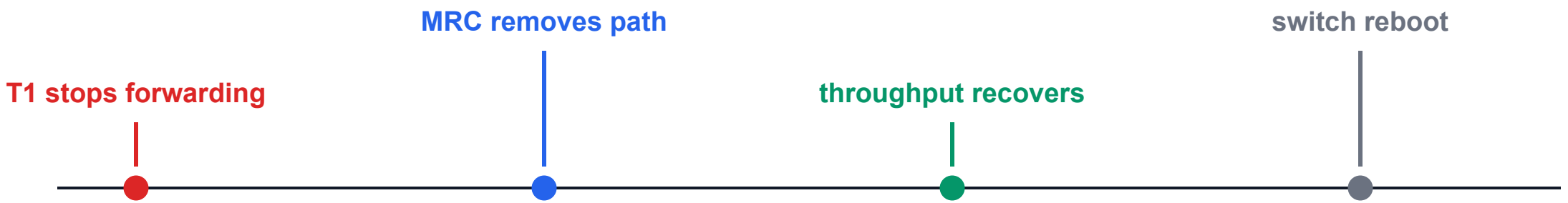
The transport learns the working fabric without operator intervention.

What happened when a T1 stopped forwarding

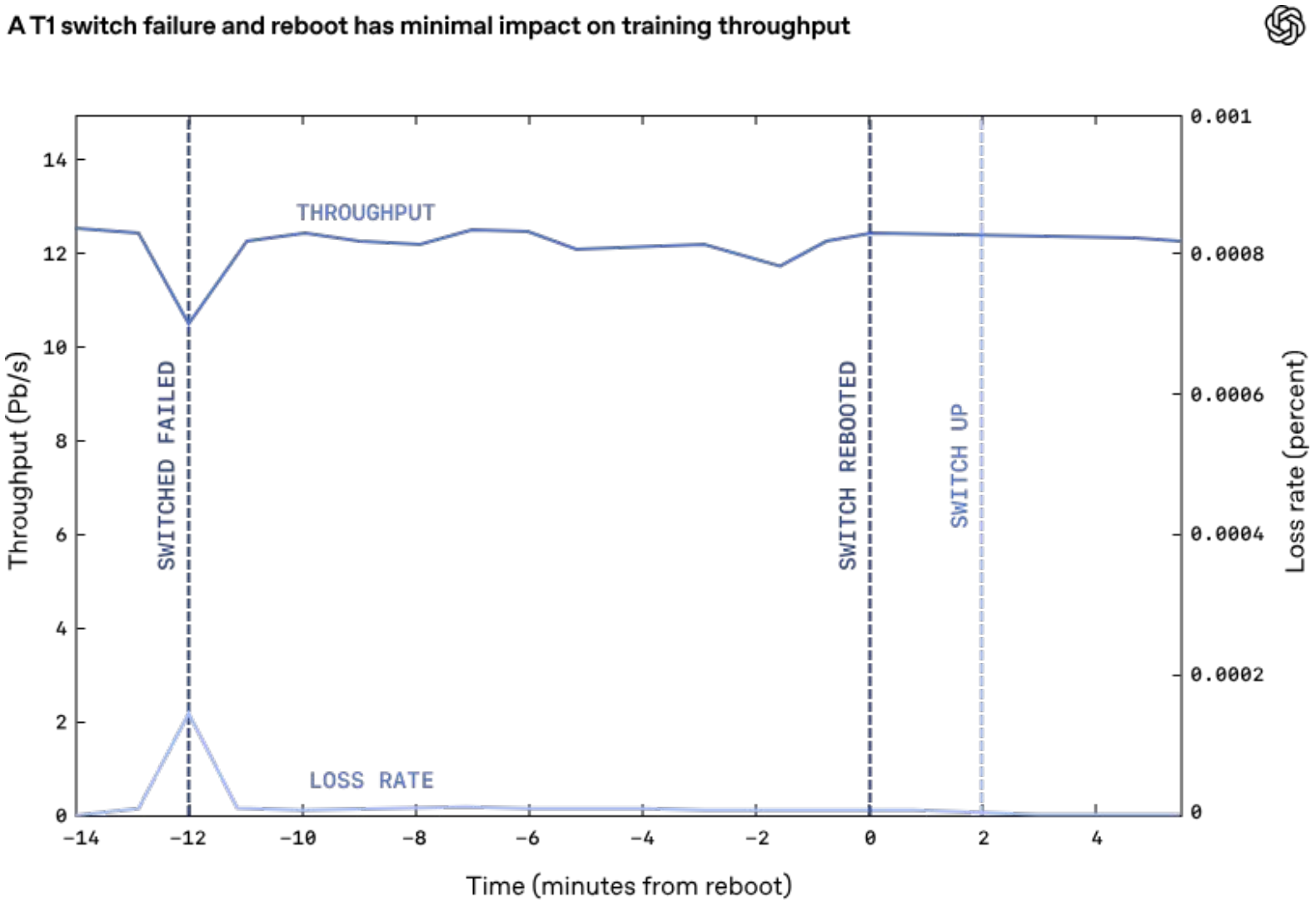
75K
GPUs in the production job

~25%
of QPs affected

~580K
packets dropped



What happened when a T1 stopped forwarding



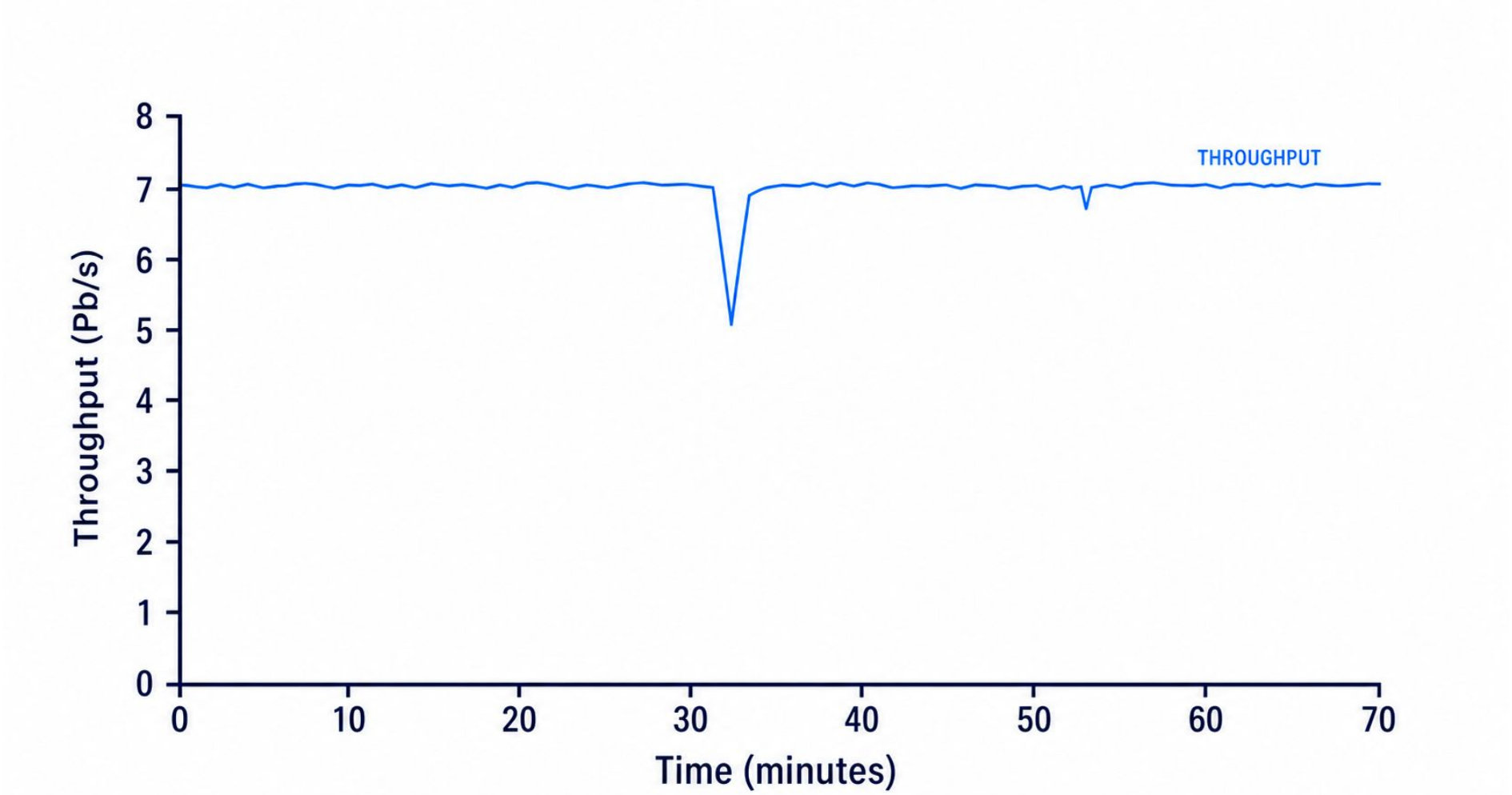
What happened when a T0 switch transceiver flapped

50K
GPUs in the production job

4 × 200
Gbps links flapped

~25%
throughput loss · ~1 min

What happened when a T0 switch transceiver flapped



Conclusion

- **Flaps or maintenance are routine operation:** MRC removes bad fabric paths without causing job failures. No manual intervention is needed.
- **Failures are diagnosable:** Static SRv6 allows end-hosts to have a deterministic view of the network's health.
- **Packet-spraying allows to efficiently use the network:** Efficient use of the network allows for a low tail-latency, which results in a low step-time.

What I didn't cover

- **Packet-trimming & congestion control:** Upon incast to server, t0 switch trims packets & sender adjusts sending rate.
- **ECN & load-balancing:** t0-t1 link load can be unbalanced, ECN allows to restore better balance