NETWORKING



MRC: A Purpose-Built RDMA Transport for AI Training Networks

Abdul Kabbani *Partner, Microsoft*

Wael Noureddine *VP, Microsoft*



Overview

- Foundations
  - Multipathing
  - Congestion control
  - Loss recovery
- MRC in action at scale
- MRC moving forward

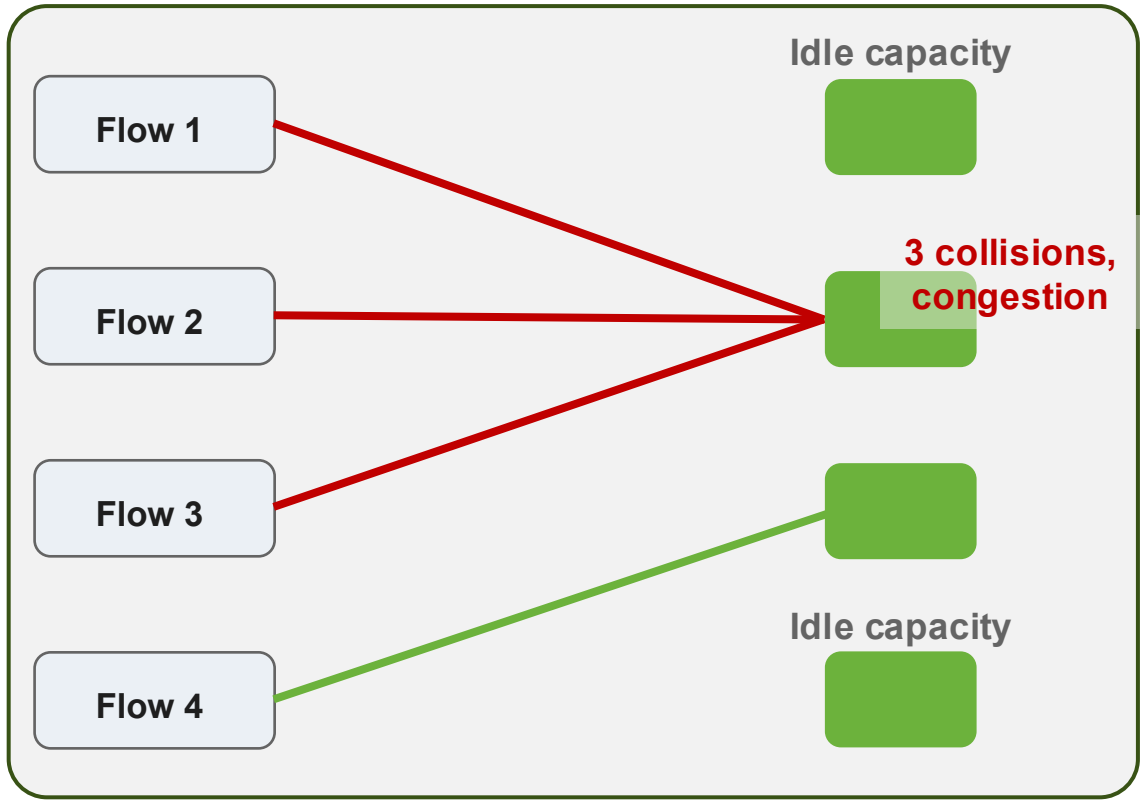
Background: Fundamentals and Key pre-MRC Transport Efforts



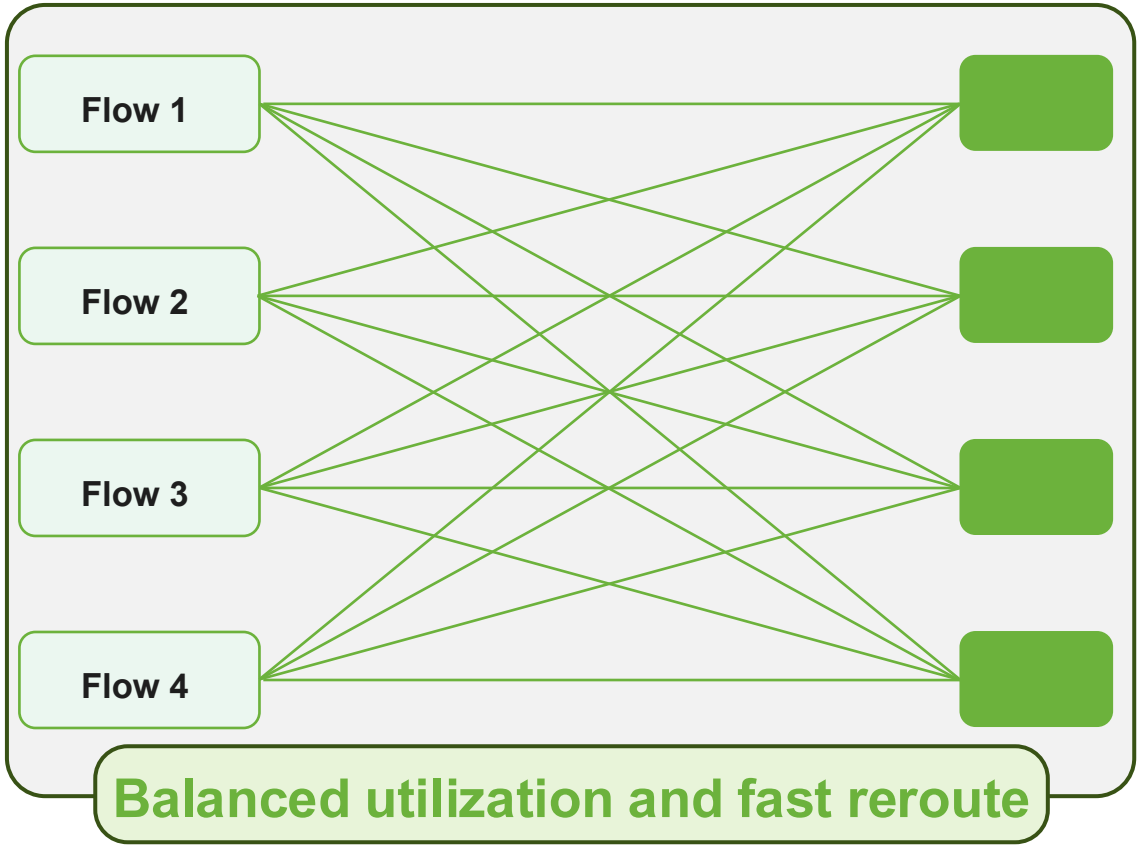
Static Hashing Causes Network Imbalance and Hotspots

The transport must react when equal-cost paths are not equally loaded.

ECMP: one flow, one fixed path



Adaptive multipath: Continuously rebalance good paths



Why Multipathing Is Foundational to AI Transport

AI fabrics expose abundant paths, but transports must use them intelligently.

1. Path diversity

Fat-tree networks provide many equal-cost paths.

A single fixed path per connection leaves capacity stranded.

2. Dynamic imbalance

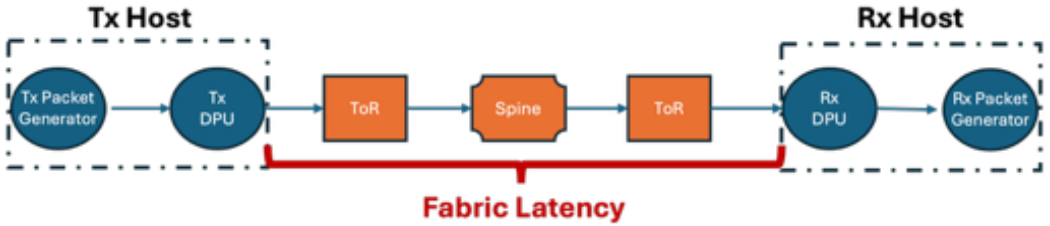
Collisions, incast, asymmetry, and failures shift the best path over time.

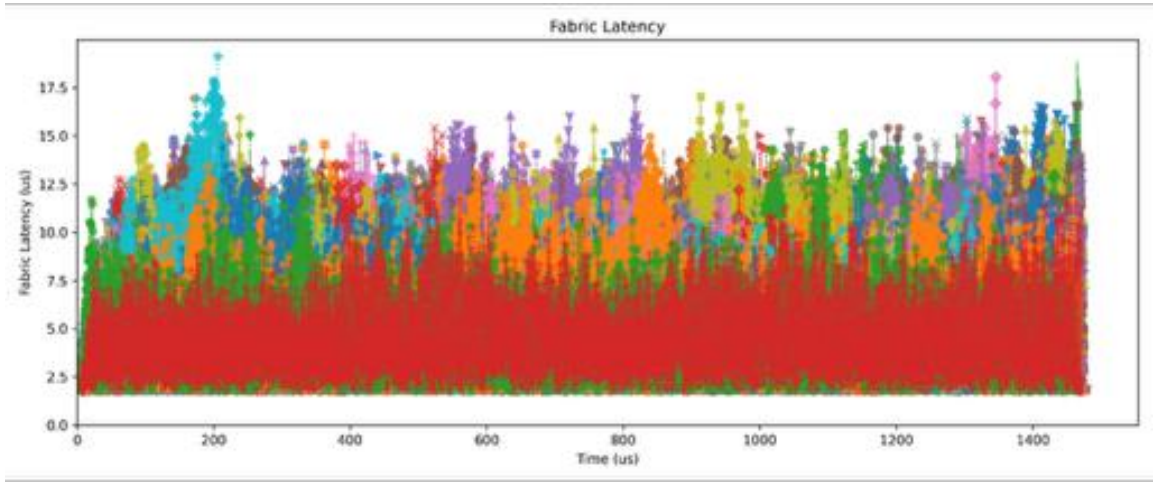
3. Transport response

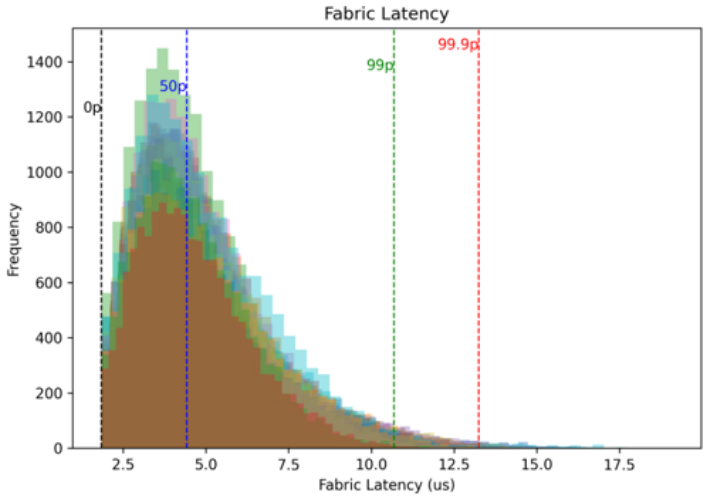
Multipath makes path choice a responsive transport function, not a static routing outcome.

Fungible's Fabric Control Protocol

- First production packet spray DC transport
  - Expands high-end router fabric design to DC scale
  - End-to-end flow and congestion control
  - Reliable underlay transport – designed to carry RDMA, TCP, UDP…
- Two-tier full cross-section bandwidth network topology







All-to-all traffic pattern, fabric latencies are short-tailed, with a 99p value of 10.5us and 99.9p value of only 13us.

Must Co-design Load Balancing and Congestion Control

Path placement determines the signal seen by congestion control, and vice versa.

Load balancing

Question: Where should the next packet go?

Goal: spread traffic over available capacity

Signal: ECN, delay, and trimming

Path quality

Sending rate

Congestion control

Question: How much data should be in flight?

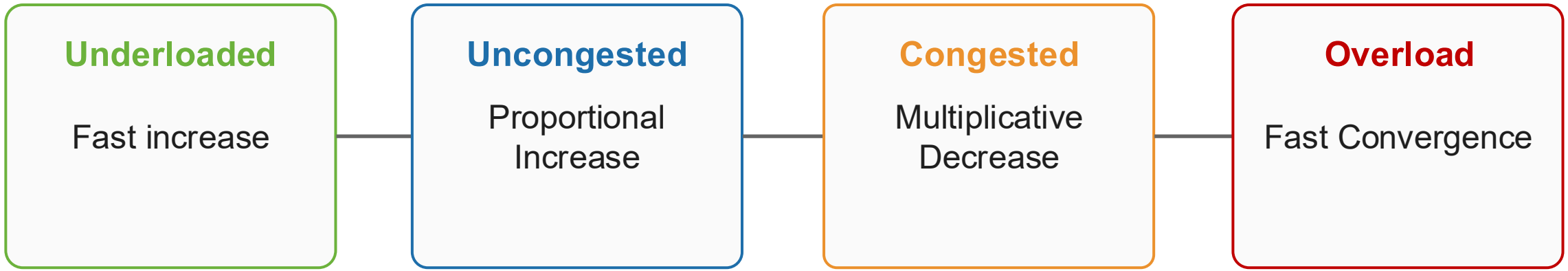
Goal: control queues while sustaining throughput

Signal: ECN, delay, and trimming

MRC uses focused mechanisms that cooperate rather than treating these as independent problems.

SMaRTT: Fast, Multi-Signal Congestion Control

Window-based control combines ECN, delay, and optional trimming.



Inputs

- **ECN: leading indicator**
- **RTT: multi-bit congestion level**
- **Trimming: rapid overload response**

Design intent: responsive during incast and ramp-up, gradual in steady state, simple enough for hardware.

Reactive Subflow Spraying (ReSS)

A small set of subflows provides path agility while preserving useful ordering.

Sender

Maintains a small active subflow set

Fabric

Multiple paths

Receiver

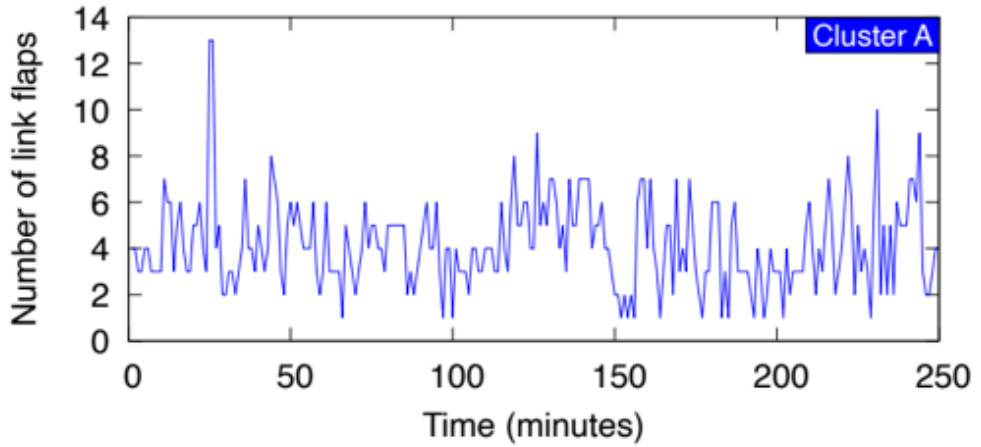
Returns ECN / RTT feedback

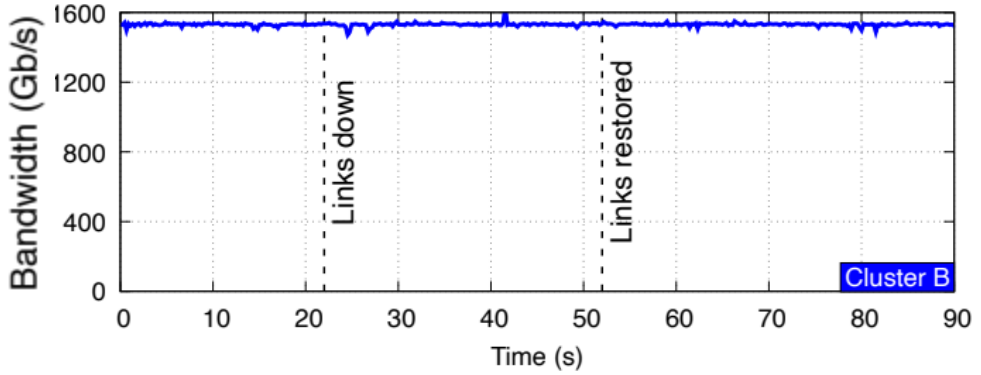


- **Randomly reroute the most congested subflow**
- **Congestion control at the flow level**
- **Few active subflows can still converge across many available paths.**

MRC is Deployed on Production AI-Training Supercomputers

- Cluster A is a large production AI-training supercomputer
  - Compute: NVIDIA GB200 GPUs, used to run 50K- and 75K- GPU training jobs
  - NICs: NVIDIA ConnectX-8, each providing 800 Gb/s.
  - Fabric: Two-tier Clos network with T0 and T1 switches.
  - Planes: Four independent 200-Gb/s planes. Each NIC's 800-Gb/s connection is split into 4 × 200 Gb/s links.
  - Switches: A mixture of NVIDIA Spectrum-4 and Broadcom Tomahawk-5.
  - During production training, frequent T0–T1 link flaps occur as shown in the figure on the right, yet MRC transparently reroutes traffic and keeps their impact on the job negligible.
- Cluster B is an experiment demonstrating bi-directional throughput given an accelerated 20-link-failure scenario



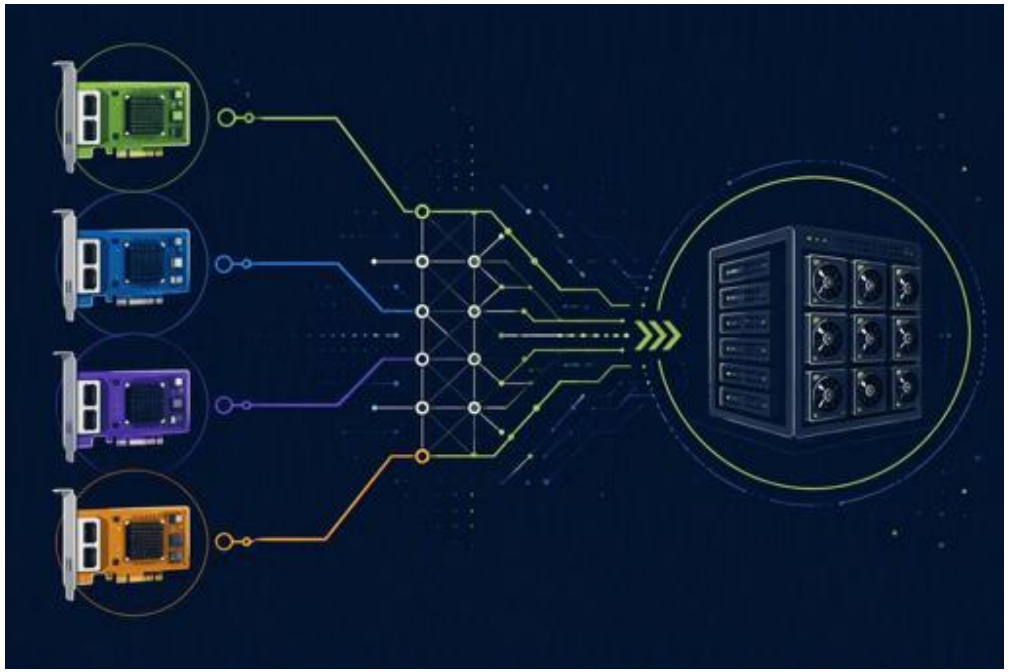


MRC: Moving Forward



Towards Heterogenous MRC Deployments

- Early MRC deployments assumed a single NIC vendor across the fleet
- Broad adoption requires interoperability across multiple NIC implementations
- Mixed-vendor fabrics are becoming the norm for AI inference networks



Example MRC Specification Gaps

Reserved bits, unclear ownership

Some implementations repurpose “reserved” header bits for vendor-specific signaling (e.g., plane health). The spec doesn’t say if “reserved” means reserved-for-vendor or reserved-for-future-use, so wire-format compliance becomes ambiguous

RTT measurement left open

MRC spec doesn’t state whether RTT should be derived from data packets, SACKs, or dedicated probe packets — so implementations diverge on how (and how often) they sense congestion

Reliability algorithm unspecified

The spec doesn’t define reliability/retransmission behavior, which directly affects RTO handling — reducing interoperability effectiveness even when wire format matches

Defining Interoperability Requirements

*What does "**interoperable**" actually mean?* Full interoperability requires **Level 3**.

LEVEL 3

Behavior compatibility

Matching signal frequency, reaction latency, and timing dynamics, so mixed-vendor endpoints perform well in the shared network

LEVEL 2

Algorithm compatibility

Comparable load balancing, congestion control, and reliability logic

RoCEv2 RC (with DCQCN)

LEVEL 1

Wire-format compatibility

A common packet format — headers, opcodes, and field semantics — so vendors can at least parse each other's packets

MRC Interoperability Requirements

Each level builds on the one below it; full interoperability requires Level 3.

Requirement	LEVEL 1	LEVEL 2	LEVEL 3
Identical header field sizes and offsets	✓	✓	✓
SMaRTT congestion control (CC)		✓	✓
ReSS load balancing (LB)		✓	✓
No Go-Back-N retransmission		✓	✓
Per-packet / per-connection retransmission timeout (RTO)		✓	✓
CC grows the congestion window at the same rate when uncongested			✓
CC shrinks the congestion window equally at equal congestion			✓
Same LB reaction time to link failures			✓

Building Consensus & Planning the MRC Plugfest



Build Consensus

- Invite vendors and operators to share implementation experience and known interop pain points
- Present the identified gaps and the Level 1–3 requirements as a shared starting point
- Align on which gaps are must-fix vs. nice-to-have
- Agree on how decisions get ratified within the OCP networking community



Plan the Plugfest

- Stand up a shared interoperability lab / testbed for cross-vendor validation
- Define a concrete test suite — e.g., congestion-control test cases and failure/degradation scenarios
- Agree on a process and timeline for an initial cross-vendor plugfest

Call for Action

Interoperability

- Hone the specs so basic interoperability is guaranteed

Evolution

- Transport management should be malleable as workloads evolve; best to furnish unified programmable congestion management solutions.

Generality

- Expand specs to encompass semantics beyond only writes

Open Discussion