AI CLUSTERS

UEC vs MRC: A Comparative Analysis of Next-Generation Networking Standards for Large-Scale DC Deployments

Load Balancing, Congestion Control, and Architectural Completeness for AI-Scale Networks

OCP PLATINUM Member

Mohammad Hanif *Senior Director, CSG, Broadcom*

Kamran Naqvi *Chief Network Architect-EMEA, CSG, Broadcom*

Agenda

- The Problem
- Two standards, two philosophies
- MRC Origins & Operations
- UEC – a clean-slate, full stack design
- Comparing the two standards
- Summary
- Call to action

The Problem: RDMA wasn’t Built for This

- Traditional RoCEv2 assumes single-path, in-order, lossless delivery per connection
- AI training/inference clusters generate massive, synchronized "elephant flows" across hundreds of thousands of GPUs
- Single-path routing creates hot spots; PFC-based lossless delivery is fragile at scale (pause storms, head-of-line blocking)
- A single stalled flow can stall an entire collective operation — cost of congestion is now a stalled job, not just added latency
- The industry needs multipath, fast-reacting, resilient transport — and two different standards have emerged to deliver it

Two Standards, Two Philosophies

- **MRC (Multipath Reliable Connection):** a modification to RoCEv2 RC, with minimum features to support AI applications, developed by OpenAI with AMD, Broadcom, Intel, Microsoft, and NVIDIA; published through OCP; already in production at OpenAI and Microsoft

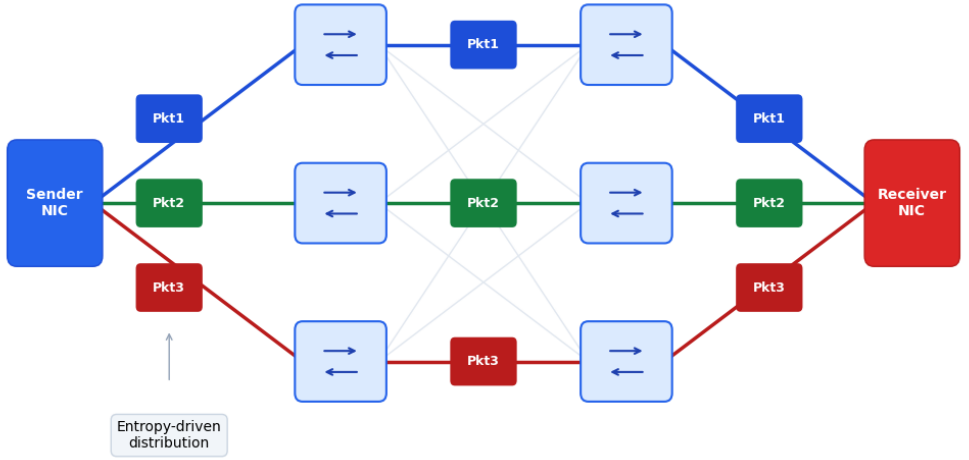
- **UEC (Ultra Ethernet Consortium):** a clean-slate transport (UET) plus link-layer and security work, developed by a broad multi-vendor consortium; Ultra Ethernet Specification v1.0 published in 2025

- Both are open, both target the same class of problem — but they start from different premises: "evolve what's deployed" vs. "redesign the stack"

Common Ground

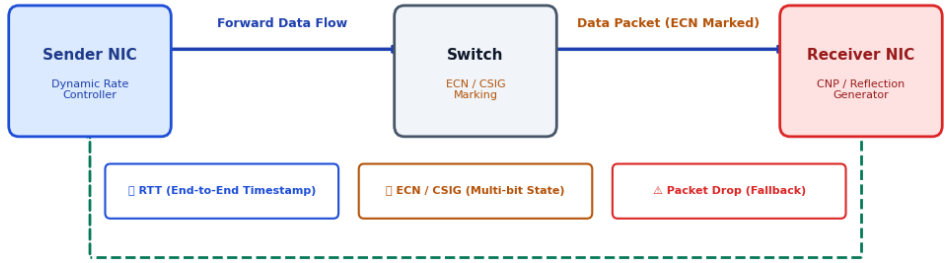
Both Standards converge on three foundational techniques:

- **Packet spraying / multipath by default**: distributing packets from a single flow across many paths using a per-packet entropy value, rather than pinning a flow to one path via traditional ECMP hashing
- **Source-based congestion control**: the sender uses network signals (RTT, ECN, loss) to adjust transmission rate. (A standard advanced cc algorithm to ensure vendor interop.)



Multiswitch based network fabric

End-to-End Congestion Control Feedback Loop

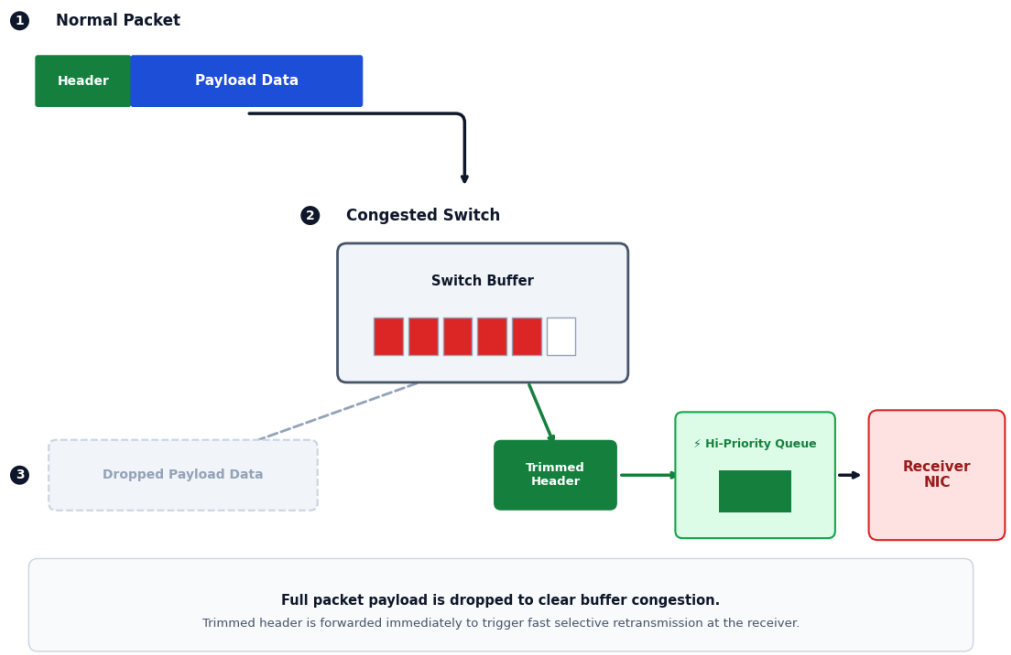


Feedback Path: Congestion Notification Packet (CNP / ACK)

Common Ground

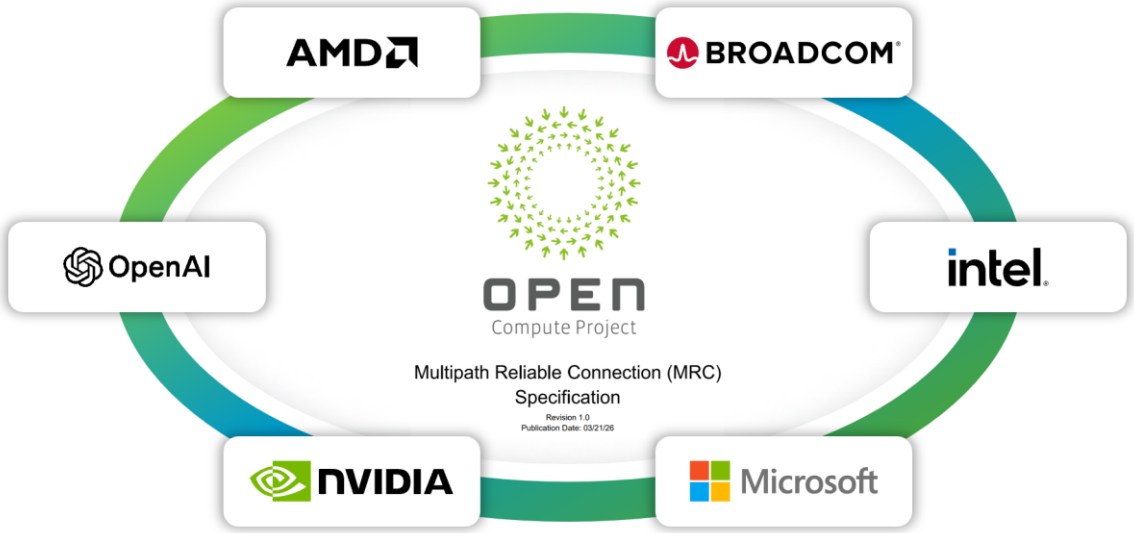
Both Standards converge on three foundational techniques:

- **Packet trimming:** when a switch would otherwise drop a congested packet, it instead truncates it to headers only and forwards it at high priority to the destination — giving the receiver an explicit, low-latency loss signal to trigger immediate retransmission, without waiting for a timeout



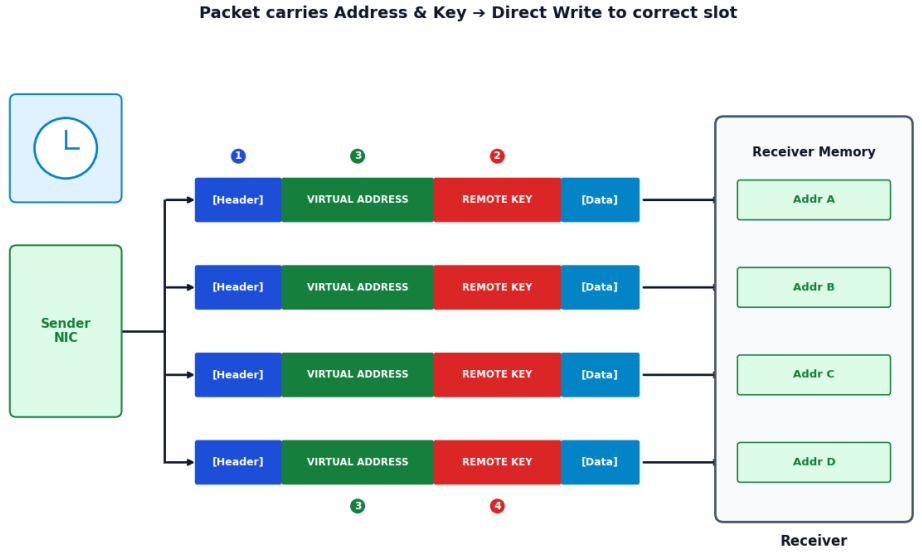
MRC: Origins and Design Constraint

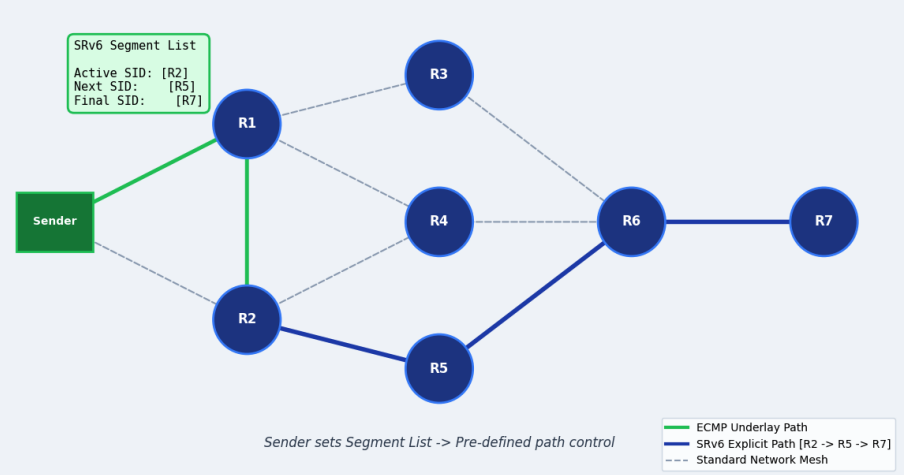
- Extends RoCEv2 Reliable Connection (RC) with a minimal wire-protocol delta: new opcode space (distinct 0101 transport prefix), modified BTH fields (rtx bit, tsh bit, PSN repurposed as request_id), new control packet types
- Non-interoperable with legacy RC endpoints — but reuses the existing RoCEv2 Verbs API and Queue Pair abstraction, so applications largely don't need to change. (Fall back to RC is spec'd in MRC)
- Design goal stated by its authors: prove that RoCEv2's multipath, reliability, and resilience gaps are solvable via a practical, incremental, composable extension — not a rebuild
- Currently supports write and write-with-immediate operations only



MRC: How It Works

- Each data packet carries the RDMA virtual address and remote key directly, so the receiving NIC can write packets to memory immediately regardless of arrival order
- Per-QP entropy value (EV) set — typically 128-256 entries — striped across UDP source port and IPv6 flow label to steer packets across different ECMP paths
- Uses SRv6-based source routing to extend multipath control beyond what ECMP alone provides
- On failure or congestion, MRC reroutes affected traffic within microseconds — far faster than software-based recovery





MRC: Application and Controller APIs

APPLICATION API

`mrc.h`

Familiar verbs model

Transport control and data-plane operations

- **Reuse libibverbs**
  Discovery and memory registration
- **Familiar resources**
  mrc_qp and mrc_cq lifecycle
- **Discover capabilities**
  Device and per-port feature queries
- **Configure and transmit**
  Write and Write-with-Immediate

`query_device · query_port · modify_qp`

OPERATE

DATA PLANE

Reliable connection

BINDS TO

EV PROFILE | CC PROFILE

Connection setup remains out of band

Best-effort Ethernet
RoCEv2 / UDP / IP encapsulation

POLICY

CONTROLLER API

`mrc_ctl.h`

Shared transport policy

Privileged process with CAP_NET_ADMIN

- **Query capabilities**
  Mandatory device and port discovery
- **Create EV profiles**
  Shared path universe and selection policy
- **Create CC profiles**
  Shared congestion-control policy
- **Optional path controls**
  EV-state events and EV probes

Policy and telemetry shared across many QPs

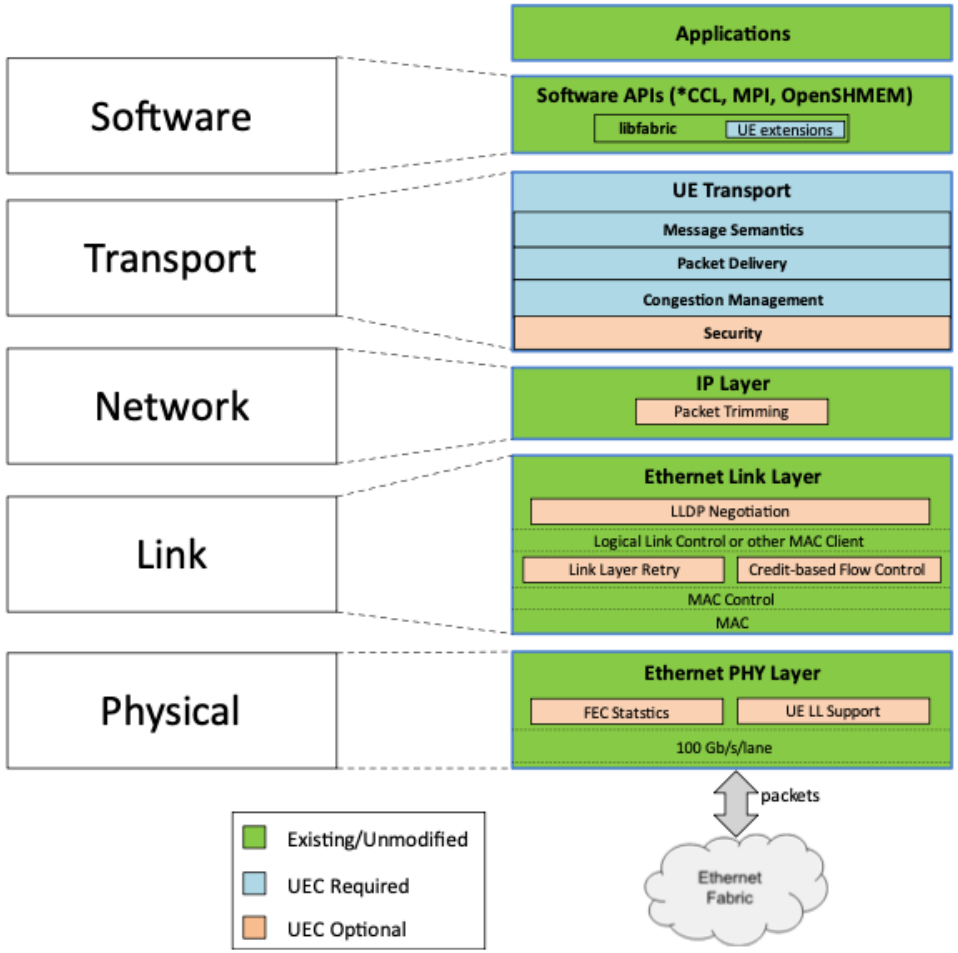
Familiar application semantics • centralized path and congestion policy

MRC: What's Deliberately Left Out

- No receiver-based congestion control — MRC 1.0 spec is sender-based only.
- No link-layer guidance — MRC operates purely at the transport layer; it assumes whatever link-layer behavior the underlying Ethernet already provides
- No network-layer prescription beyond SRv6 source routing for path steering
- This isn't an oversight — it reflects the design philosophy: solve the transport-layer gap fast, on hardware that already exists, without requiring a new link-layer

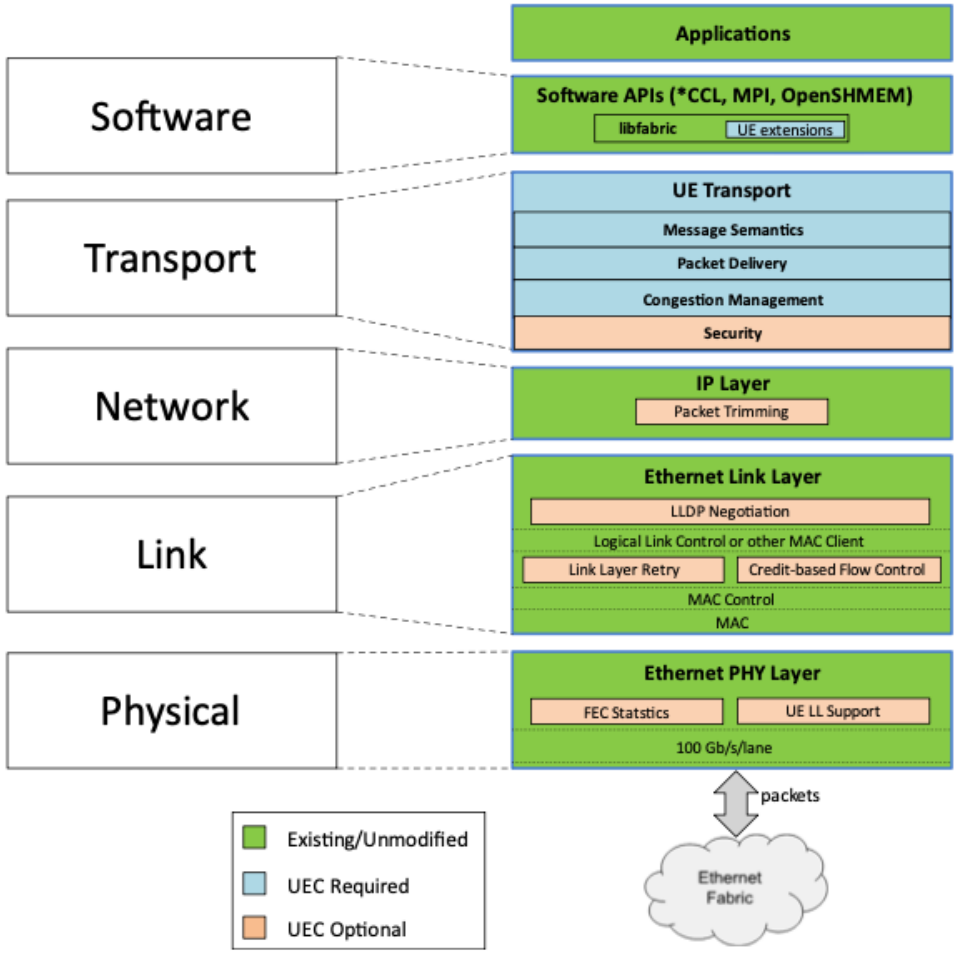
UEC: A Clean-Slate, Full-Stack Design

- Ultra Ethernet Transport (UET) is built from the ground up for AI/HPC traffic patterns — not an extension of RoCEv2 semantics
- Spans multiple layers explicitly:
  - Transport: native multipath, in-order and out-of-order delivery modes, Reliable/Unreliable + Ordered/Unordered delivery combinations
  - Congestion management: both sender-based and receiver-based mechanisms , plus in-band telemetry via congestion signaling (CSIG)



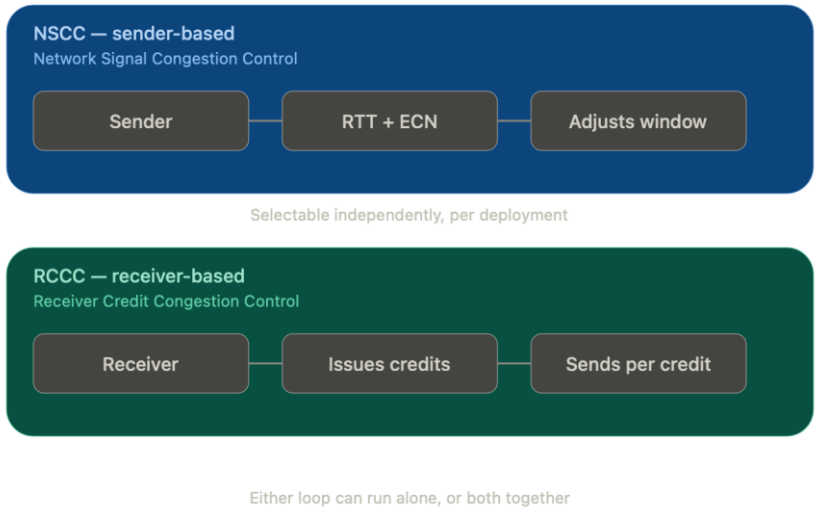
UEC: A Clean-Slate, Full-Stack Design

- Link layer: Link Level Retry (LLR) for hop-by-hop recovery, optional packet trimming for fast loss signaling
- Security: integrated end-to-end encryption/authentication (IPSec/PSP) designed to coexist with packet spraying
- Designed to plug into libfabric 2.0 as a native transport, easing application-level adoption. (Verbs is being added to 1.1 and is in the process of being unstreamed to RDMA core)

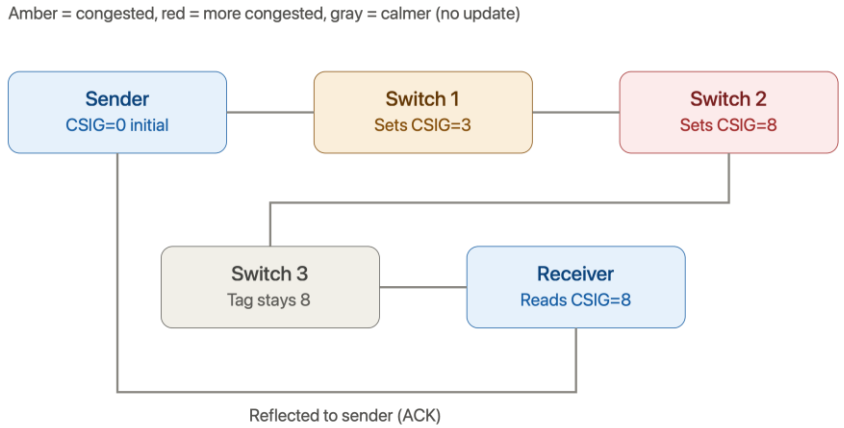


UEC: Congestion Management

- Dual congestion control: sender-based (RTT/loss/ECN-driven) and receiver-based (credit issuance) — selectable based on deployment needs.



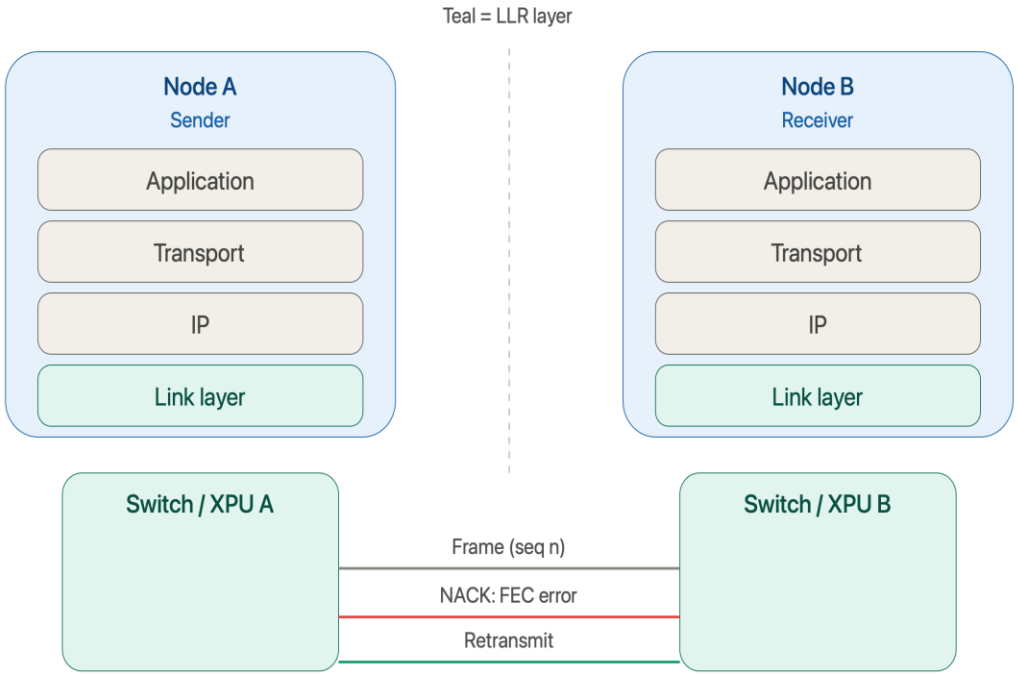
- CSIG (congestion signaling): compare-and-replace in-band telemetry updated at each hop, without growing header size, giving precise congestion location and severity



UEC: Congestion Management

- Packet trimming: congested switches truncate rather than drop packets, giving both sender and receiver an immediate, explicit signal — avoids waiting on timeouts despite reordering from packet spraying

- LLR: recovers lost/corrupted frames at the link level, before the loss ever propagates up to the transport layer or stalls a collective



Side-by-Side Comparison

Dimension	MRC	UEC
Design philosophy	Evolutionary extension of RoCEv2 RC	Clean-slate, purpose-built stack
Load balancing	Packet spraying via entropy value + SRv6 source routing	Native packet spraying, configurable in-order/out-of-order modes
Congestion control	Sender-based only	Sender-based **and** receiver-based
Link-layer scope	None specified — relies on underlying Ethernet	Link Level Retry (LLR), Credit-Based Flow Control (CBFC)
Network-layer guidance	SRv6 source routing only, Trimming	Congestion signalling (CSIG), trimming-aware switch behaviour
Interoperability with legacy	Non-interoperable wire format, but reuses RoCEv2 Verbs API	New transport under UET; broader ecosystem coordination required
Functional Scope	Narrow/Specialised: Tailored strictly for synchronous AI training; supports only RDMA Write and Write-with-Immediate.	Broad/Universal: Covers AI training, inference (MoE/KV cache), storage, and HPC; supports Reads, Writes, Sends, and Atomics.
Deployment status	In production today at OpenAI, Microsoft and OCI	Spec (v1.0.3) published; hardware/software ecosystem maturing

Immediate Value vs. Long-Term Foundation

- MRC’s narrower scope is a feature, not a bug:
  - Focused on AI Training workloads.
  - Preserves familiar ROCEv2/libibverbs programming model – minimal code changes for AI frameworks
  - Available today and already running in production

- UEC's broader scope is the more durable foundation
  - Designed for both AI and HPC environments.
  - Comprehensive, full-stack approach
  - Large Eco-system continuing to evolve

Summary

- Both UEC and MRC converge on packet spraying and sender-based congestion control as the emerging baseline for AI-scale network fabrics
- MRC: pragmatic and deployable today on existing hardware — proven in production at hyperscale — but deliberately narrower in transport scope
- UEC: comprehensive and clean-slate, spanning transport through link layer — but earlier in its deployment and ecosystem maturity curve
- The right choice depends on your deployment timeline, existing infrastructure, and willingness to manage transition complexity

Call to Action

- Get involved in the OCP MRC Project and UEC.
- Contribute silicon validation data, SAI-layer telemetry, and real production deployment feedback to both standards bodies
- Where to find additional information (URL links)
  - Multipath Reliable Connection (MRC) Specification - V1.0
  - Resilient AI Supercomputer Networking using MRC and SRv6
  - The Multipath Reliable Connection (MRC) Transport
  - Ultra Ethernet Consortium
  - Ultra Ethernet Specification v1.0.3
  - Enabling AI Networking @ Scale with Multi-path Reliable Connections (MRC) - Broadcom Blog
  - MRC - Spercomputer Networking - OpenAI Blog
  - OCP SAI Documentation

Thank You!

