Networking for AI and HPC, and Ultra Ethernet

ITNOG9

2025-05-20

Massimo Magnani, Arista Networks

What's the Problem?

AI and HPC networks are different

- Endpoints are fast, Load is high
- Flows are few and high BW
- RTTs are short
- Flows are synchronized
- Completion time determined by slowest flow

Vanilla networking doesn't meet the needs

Source; https://engineering.fb.com/2022/10/18/open-source/ocp-summit-2022-grand-teton/

UEC background

Who why?

Ultra Ethernet Consortium

Mission:

Advance an Ethernet-Based Open, Interoperable, High-Performance Full-Stack architecture to meet the Growing Demands of AI and HPC at Scale

>100 member companies

>1300 active participants

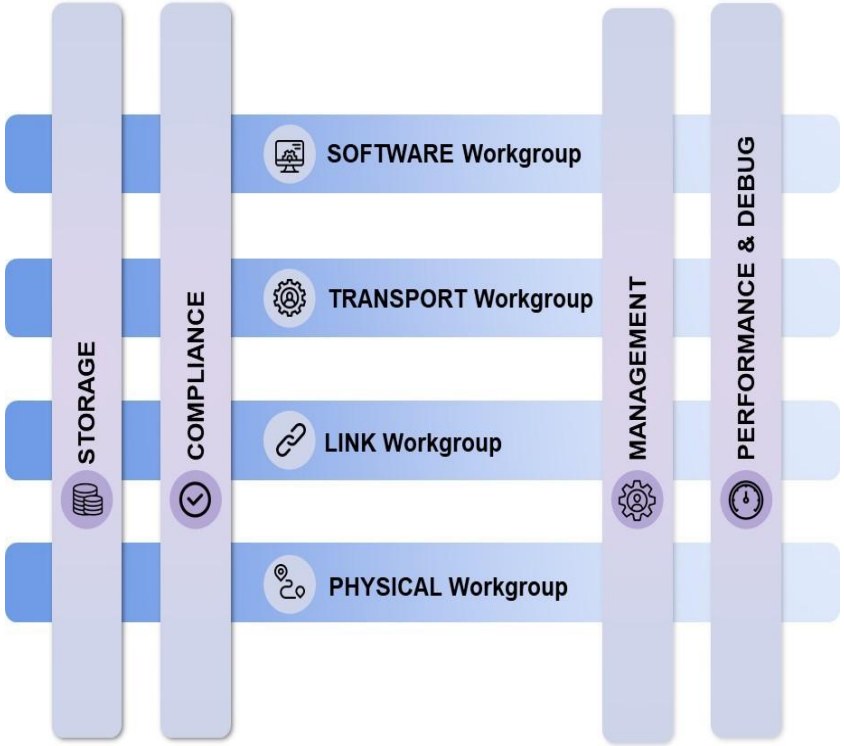
*not all members listed snapshot as of 2025-05

Source:ultraethernet.org

Ultra Ethernet Activities

- Many working groups
- One specification, many layers
- The spec will be big
- Expect it early 2025

UEC is a JDF project and an International Standards Organization

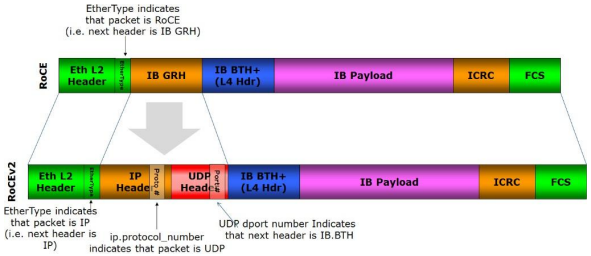


Source:ultraethernet.org

RMA is critical to performance

Remote Memory Access

- Accelerators today communicate with RMA
- RMA is hardware delivery straight to/from memory
  - Kernel bypass, zero-copy
  - Hardware loss detection, retrans, loss recovery
- RDMA over IP (RoCEv2) is a widely deployed RMA implementation

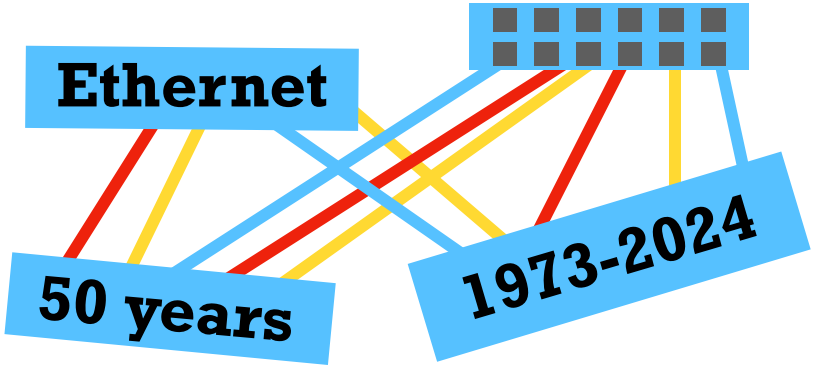


RoCE headers

RMA is a *great* concept

Ethernet is the right foundation for RMA

for all the reasons…



- broad ecosystem
  - NICs, switches, optics, cables
  - multi-vendor at all layers
- rapid innovation
- many tools for operations, management, testing
- scales to millions: addressing, routing, management, provisioning
- universally understood - books, courses, websites, classes, …

UEC builds on Ethernet

Why revisit RMA?

…specifically RoCE?

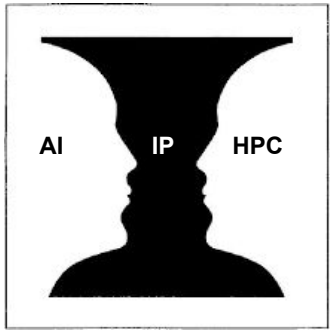
- **Lack of multipathing**
  - in-order packet delivery is limiting
- **Go-back-N** Recovery is inefficient, forcing lossless networks
- Congestion control (DCQCN) is hard to tune, not easy to (inter)operate
- Scale requirements are increasing
- Integrated security is important

RMA is great, but it’s time to revisit the protocol

Ultra Ethernet Transport

An RMA protocol for the future

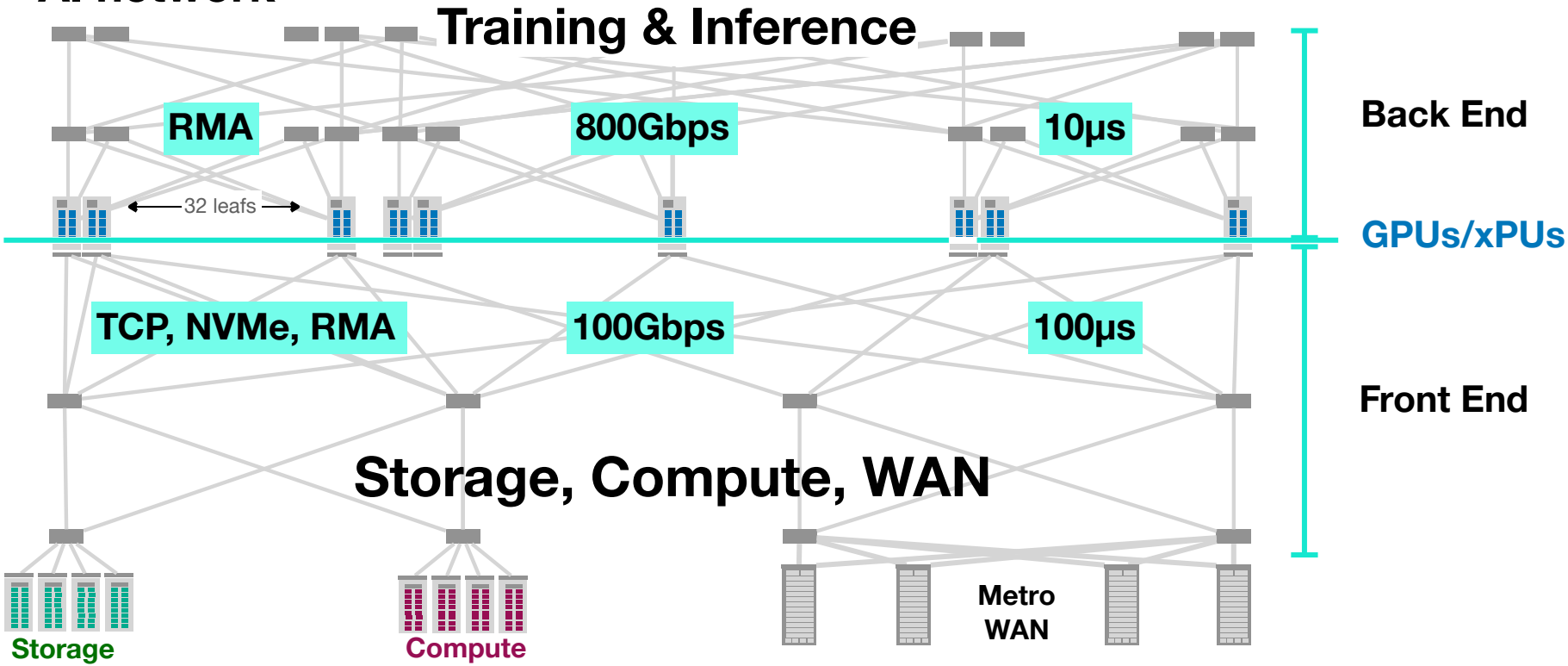
- Multipathing RMA
- Relaxed Delivery Ordering
- Rapid loss recovery
- Modern congestion control for the DC
  - Rapid startup and slowdown
  - multi-path aware



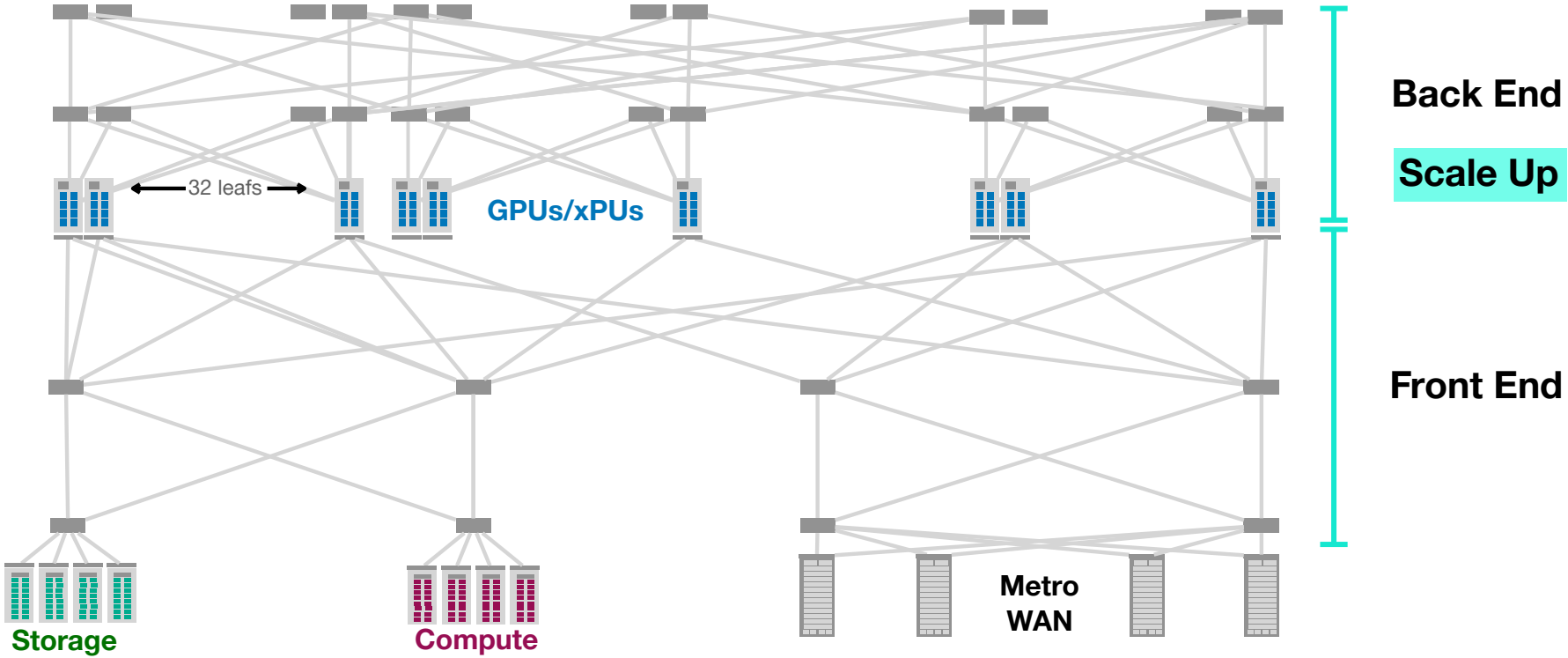
- Run on IPv4/v6 and Ethernet
- Lossy and Lossless operation
- Ordered and Unordered Delivery
- Design for high scale at low cost
- Day-1 Security

Preserve the applications above, use Ethernet and IP below a new transport in the middle

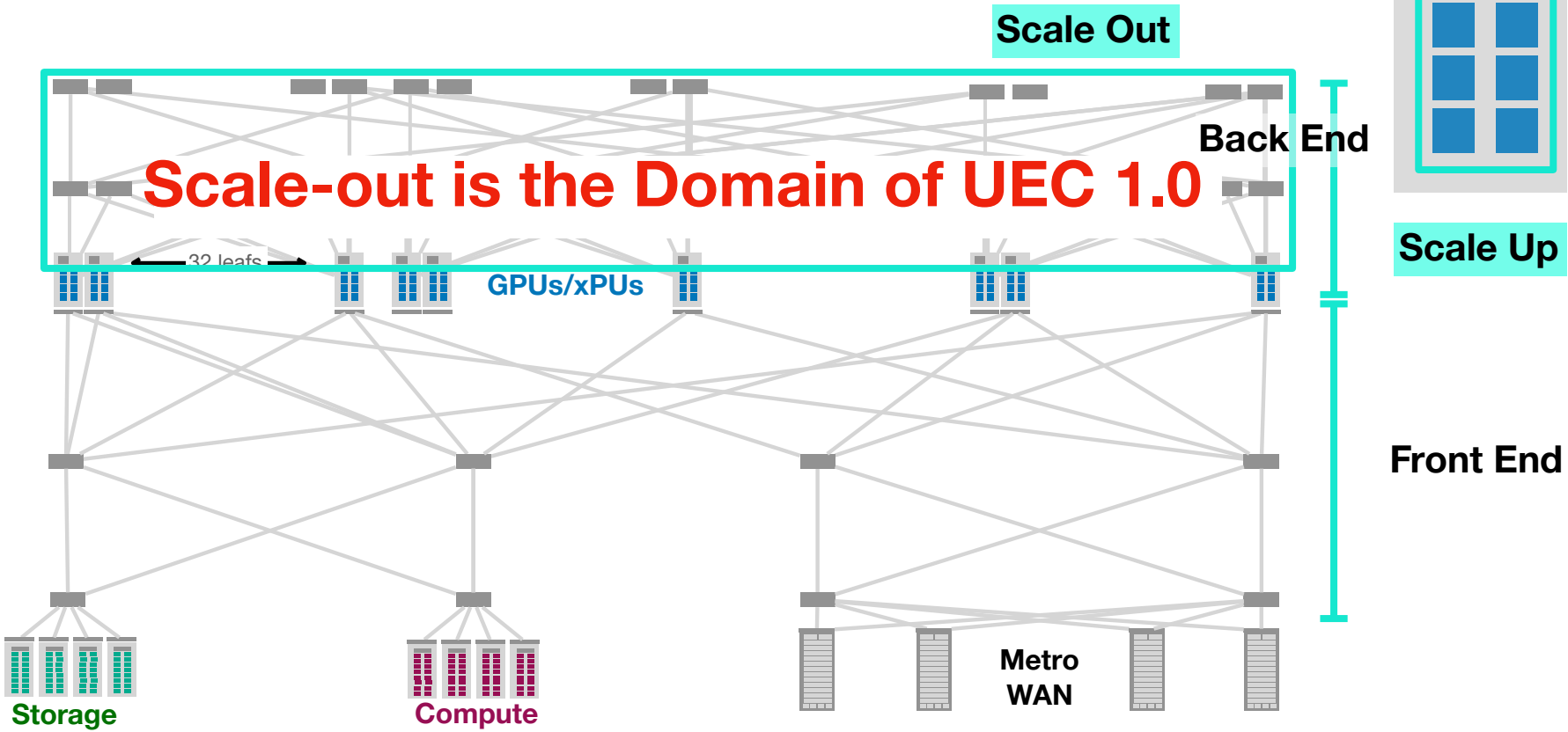
Front-end and Back-end



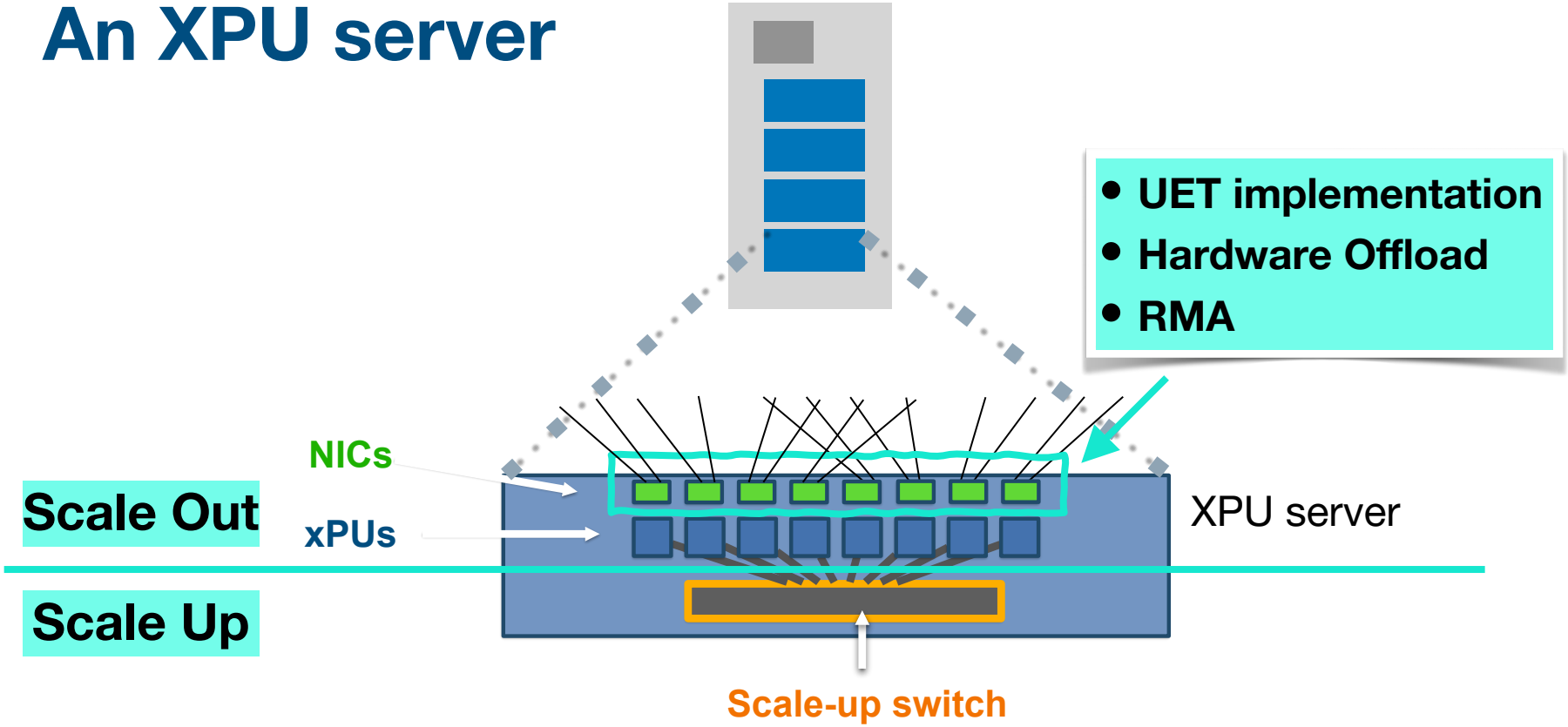
Front-end and Back-end



Front-end and Back-end

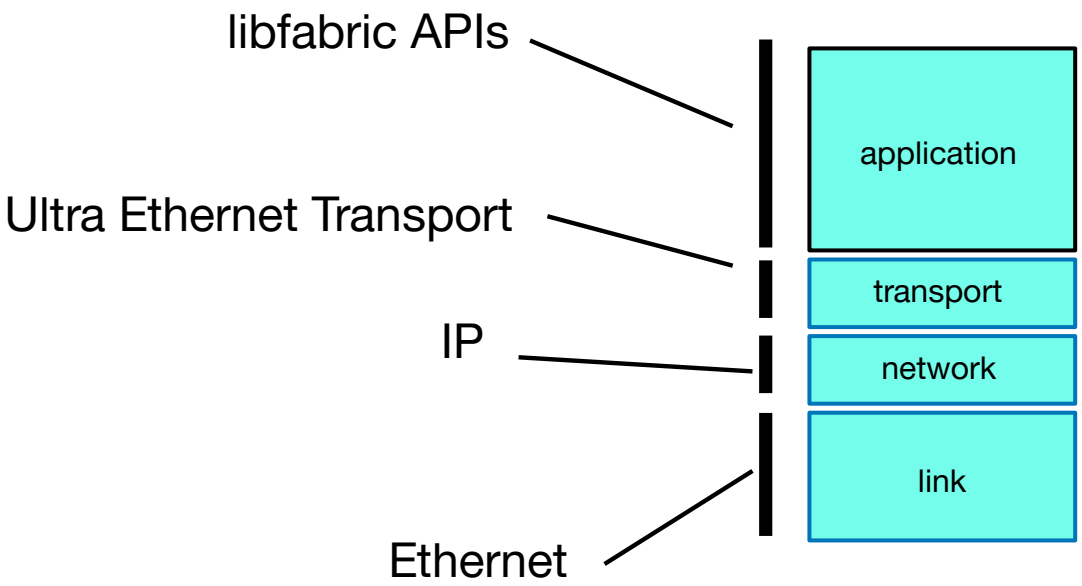


An XPU server

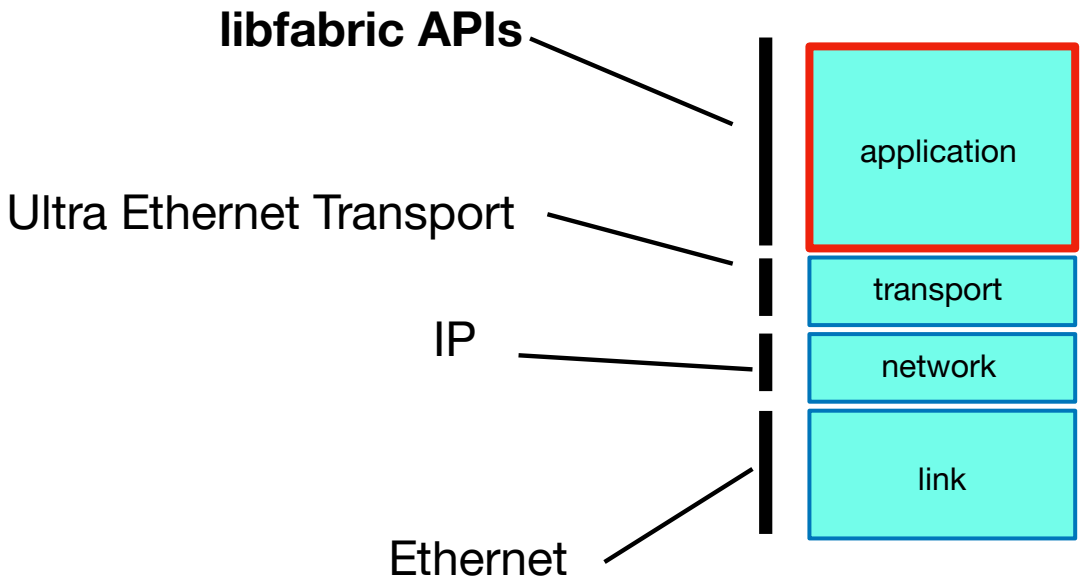


NICs play an important role in Ultra Ethernet

Ultra Ethernet up and down the stack

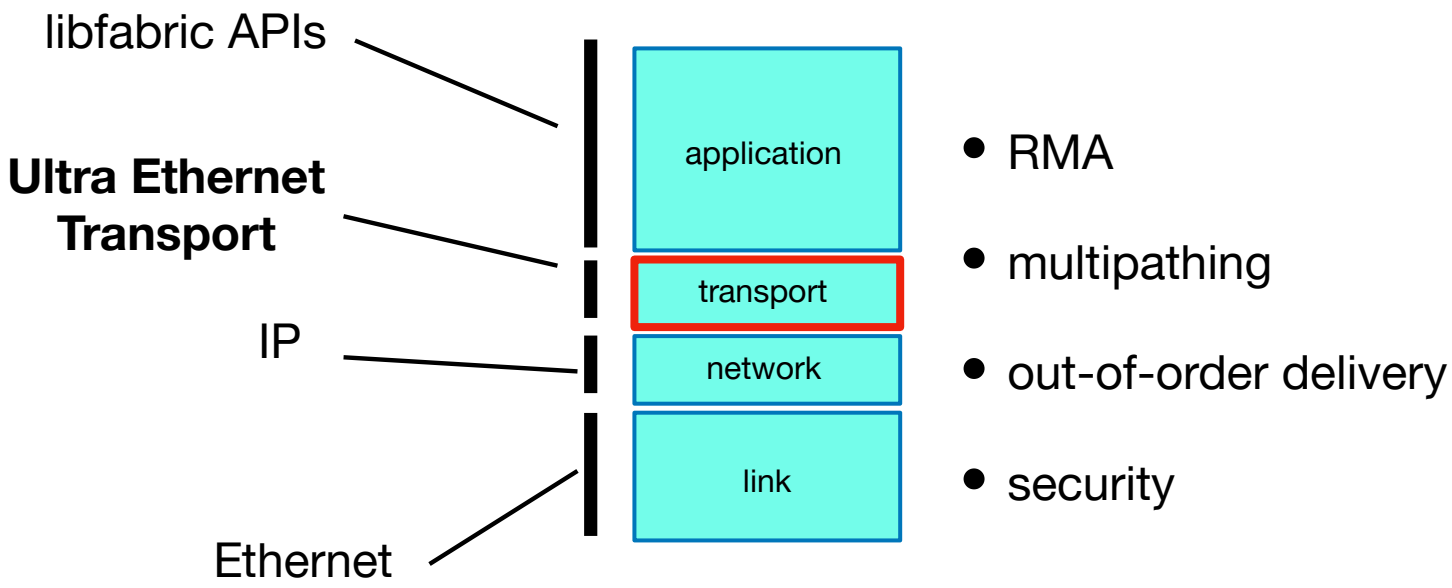


Ultra Ethernet up and down the stack



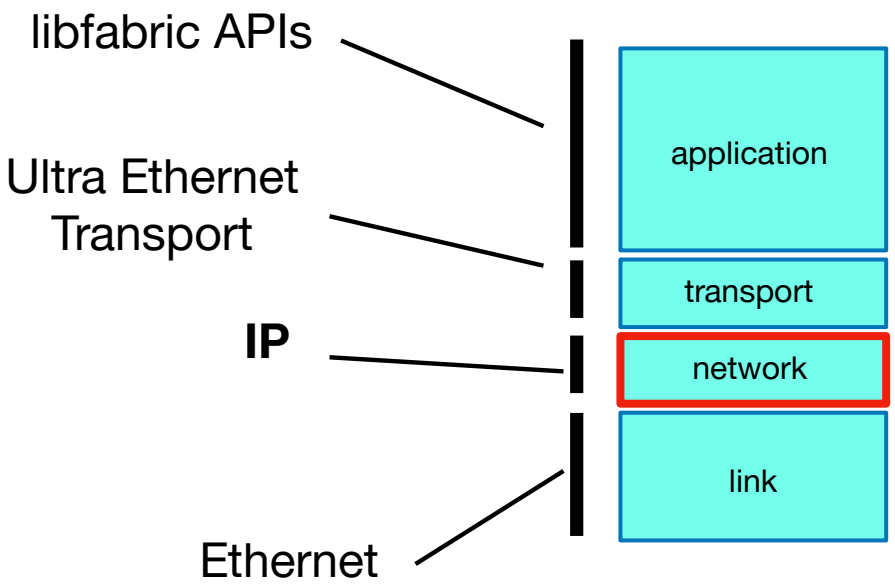
- modern RMA API for HPC, AI
- Richer than sockets

Ultra Ethernet up and down the stack



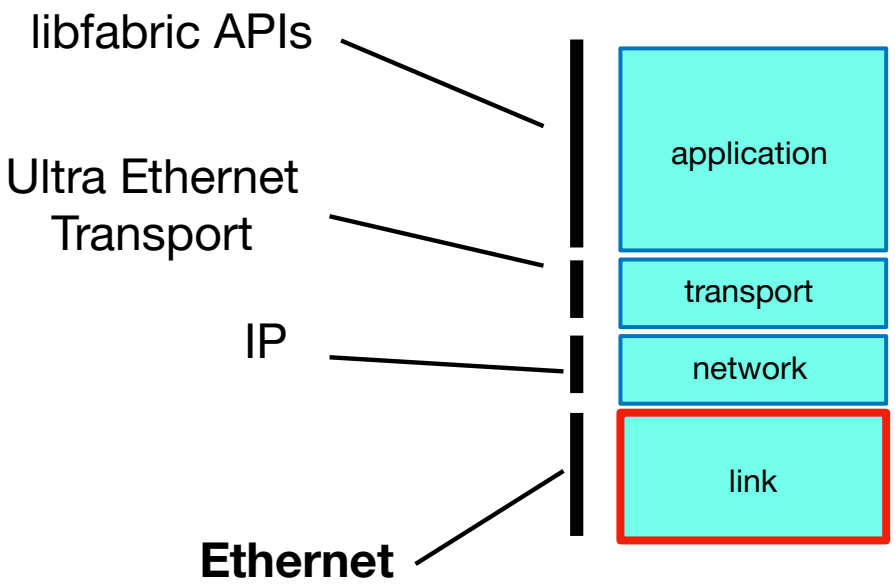
- RMA
- multipathing
- out-of-order delivery
- security

Ultra Ethernet up and down the stack



- **Standard IP** - v4 or v6
  - DSCP, QoS
- Packet Trimming (optional)

Ultra Ethernet up and down the stack

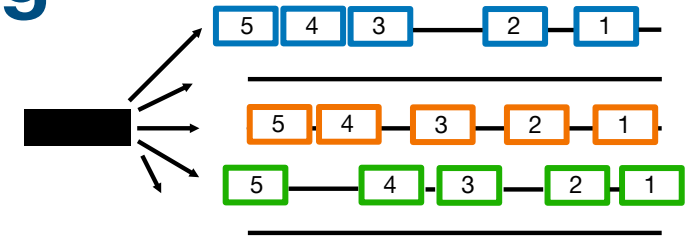


- **Standard Ethernet**
- Optional Link-layer Retransmit

Load balancing

The key problem to solve

Flows and packet ordering

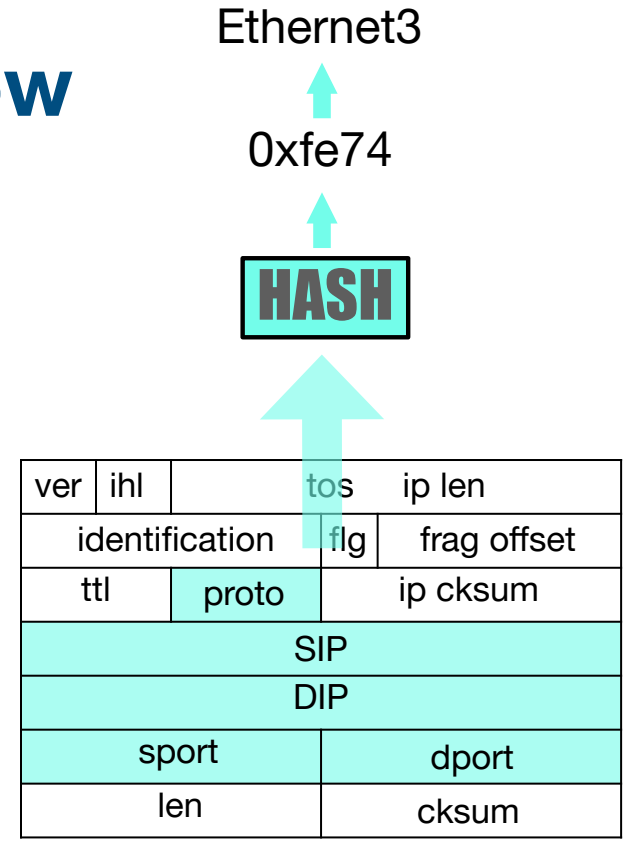


- Networks today keep packets within a single L4 flow in order
- Because transport protocols (TCP, RDMA) don't like out of order packets
  - out-of-order packets are interpreted as loss
  - repeated loss is interpreted as congestion
  - congestion results in slowing down

so don't reorder packets within a flow

Choosing a path for each flow

Spreading flows over all ECMP paths

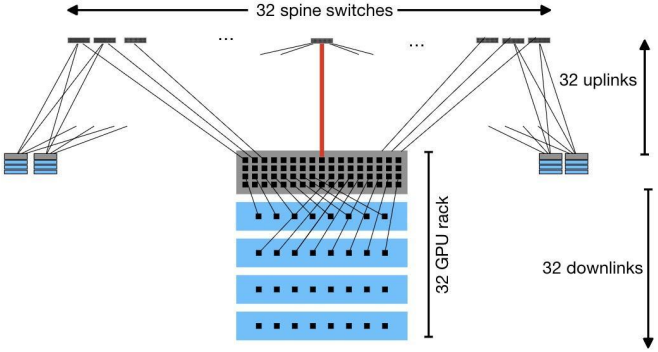


- Generally, with a hash of L4 ports and IP
- Works great if many small flows per link

...but it's hard to spread flows evenly when there are not many

So, how good is flow hashing?

Load Balancing simulations



- 1 Rack - 32 GPU - 32 Uplinks - No Oversubscription
- 80% offered load per link - 32 Uplinks - N/S only traffic
- Case 1 - Vanilla traffic
  - 80 flows each - 1Gb over 100Gb links
  - Average LB efficiency 99,95% - Great
- Case 2 - Simulated AI Traffic
  - 8 flows each - 10 Gb over 100Gb links
  - Each flow is divided into 1MB chunks (256 packets - 4k bytes each)
  - Average LB efficiency 96,8% - Very good … BUT

In the Simulated AI Traffic, on worst case scenario, links received 14 flows, thus exceeding the 100Gb bandwidth availability by 40%

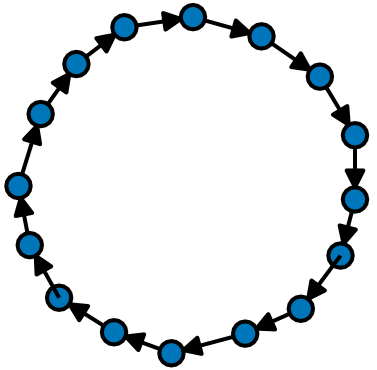
And this reduced the average efficiency, in worst case scenario to 71% … Very BAD!

Why the slowest link matters

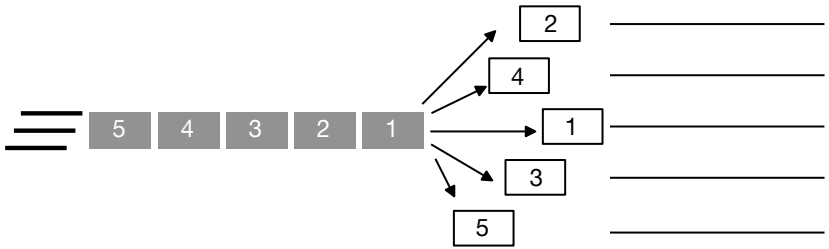
Collective Communications

- **Collectives are core to AI and HPC apps**
- Distributed computations from MPI
  - Reduce, Scatter, AllReduce, Gather, AllToAll, Broadcast, … **so slow links are bad**
  - e.g., average and broadcast gradients / sum and distribute vectors
  - Commonly use a ring or a tree (logical)
    - of 32, 64 or more, nodes

```
doRing() {
  send chunk
  while (more data) {
    receive chunk
    merge with next chunk
    send merged chunk
  }
}
```



Communication in a ring (or tree) is limited by the speed of the **slowest** link

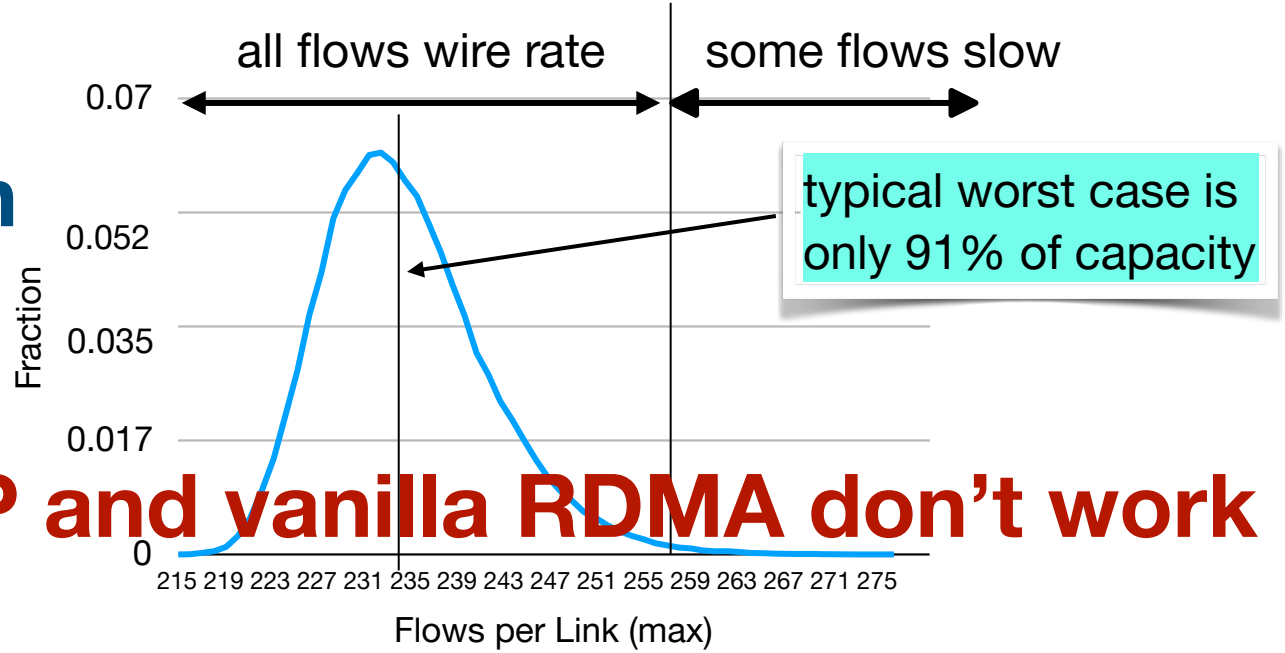


forget about keeping packets of a flow in order…

What if…

One flow could use *ALL the* paths?

Max Utilization



but TCP and vanilla RDMA don't work

32 servers, packet-sprayed 204 ways on 32 uplinks

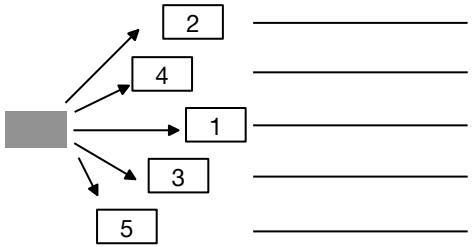
80% offered load

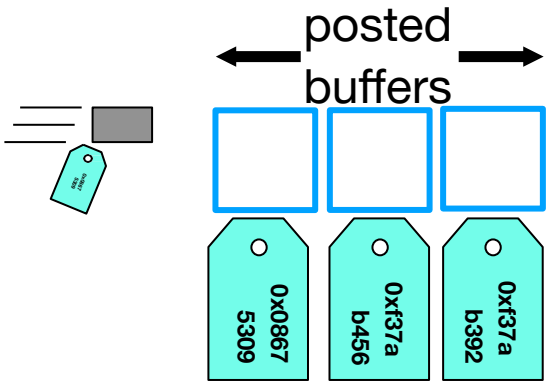
99.98% efficient for an application driven by worst-case

Ultra Ethernet Transport

So enable the transport protocol to spray!

A key tenet of the UET



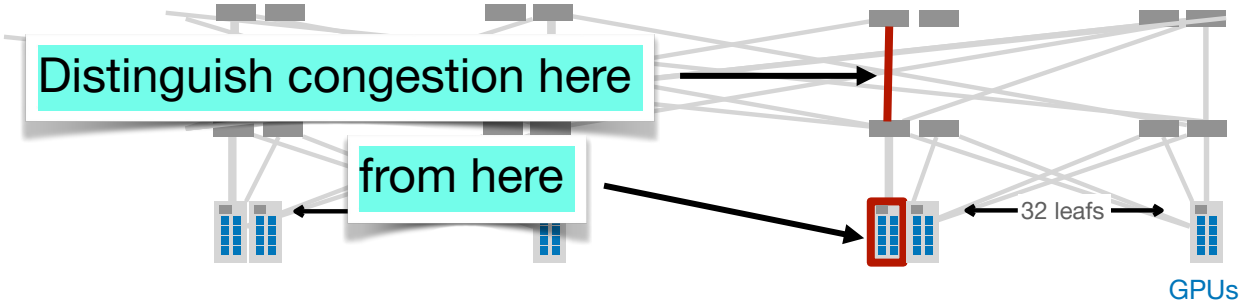


- Don’t insist on packet ordering within a flow
- Tag packets with their ultimate destination
  - eliminates the need to reorder on arrival
  - packets can be immediately placed in memory

UET: RMA with out-of-order arrivals

Packet Spraying Challenge (1)

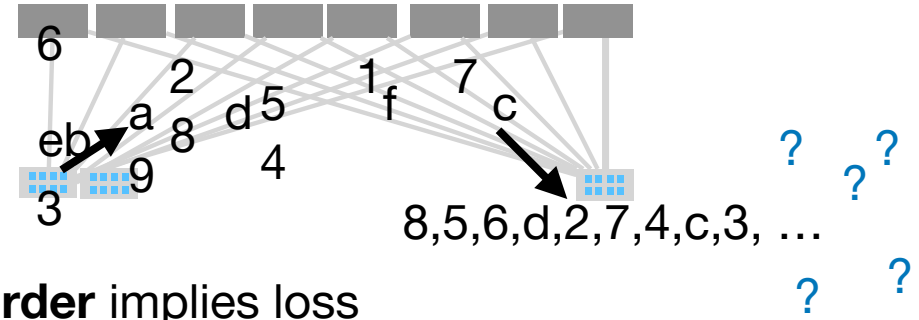
Path selection and congestion avoidance



- Need enough entropy so that all paths get used equally
- Avoid entropy values that drop, reuse ones that don't
  - choose the right amount of entropy values (too many can slow reaction)

Packet Spraying Challenge (2)

Loss Detection in an OOO protocol

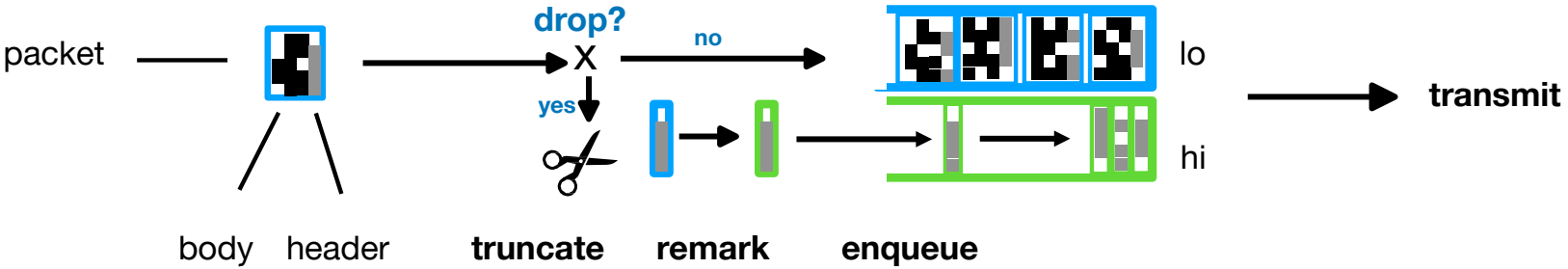


- Generally, **timeout** or **out-of-order** implies loss
- With ***spraying,*** *out of order* is not a simple concept
  - packets taking different paths can arrive in any order
- Fast *timeouts* are made harder because of variable delay across paths

need new methods to detect loss

Packet trimming

chop, don't drop!



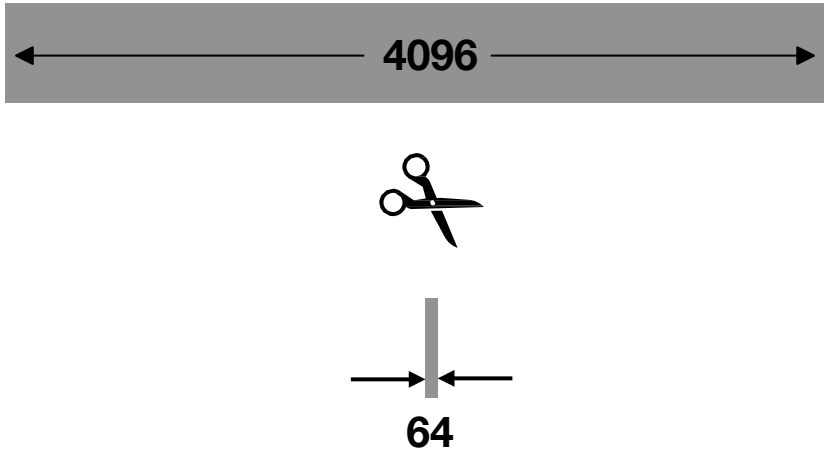
- **Truncate** ("trim") to 64 bytes instead of dropping
- **Mark** the DSCP as "trimmed"
- **Enqueue** truncated pkt in high priority queue for a *faster* congestion signal

switch support for fast loss detection

Packet trimming

switch support

- 4096 to 64 bytes: **64x** reduction
- **Only trim eligible** (DSCP) packets
  - trimming would confuse TCP, UDP, …
- **Trimmed packet signals receiver to:**
  - slow down
  - request retransmission

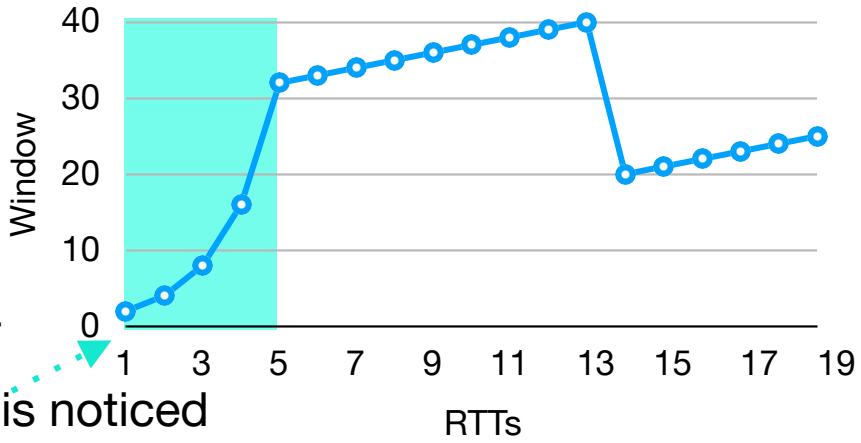


precise and fast loss detection

Packet Spraying Challenge (3)

high bandwidth and short RTT

- How is UET CC different from TCP?
- Get to wire rate very quickly
  - 1MB takes 10 usec at 800gbps = 1 RTT
  - Must back off quickly when congestion is noticed
- No time to wait for TCP slow start



UET flows can be short but large

Fast Speed-Up and Slow-Down

- We need to ramp quickly and slow down quickly
- Losses and/or delays tell the transport to slow-down
- UET needs **new algorithms** for a sprayed network

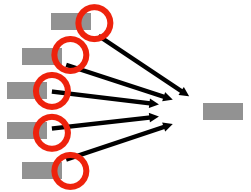
existing transports are too slow and/or depend on ordering

UET congestion control

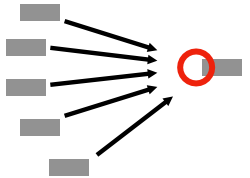
two flavors - that can work together

- **Sender-based** (default)
  - fast ramp, fast slowdown
  - uses delay, mark, trim as indicator of congestion
- **Receiver-based** (optional)
  - receiver-generated credit manages incast
  - optimistic transmission before credits received

sender control



receiver control



both are designed to deal with spraying and OOO

Ephemeral Connections

fast startup

- Eliminate the delay of a round-trip handshake before transmitting
- Connection is established on-demand by the first data packet
- Fast startup means I don’t need to keep state around when it’s done
- Reduces costly connection state on NICs

UET - faster startup latency and less state

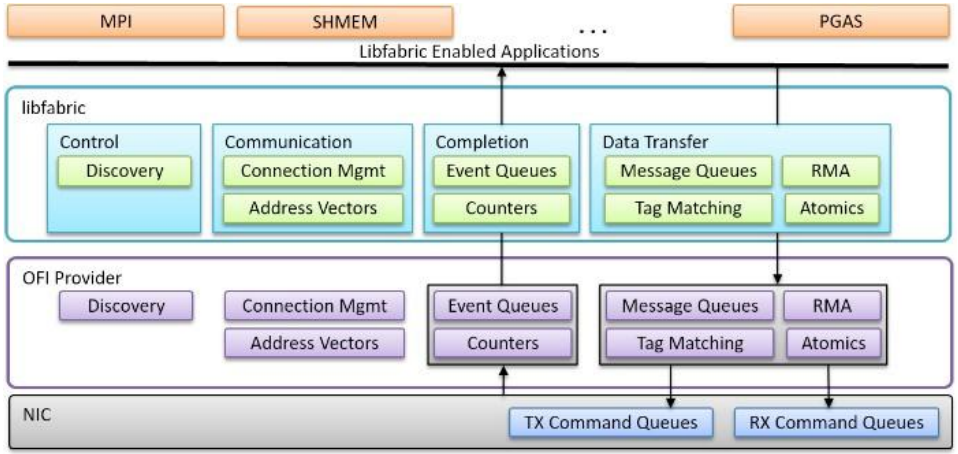
Ultra Ethernet across the layers

Application, Transport, Network, Link Layer

libfabric

by the OpenFabrics Alliance

- UEC selected libfabric 2.0 as a modern API
- Generic APIs for High Performance Communication
  - RMA
  - Tagged messages, Atomics
  - Collective operations
  - event queues, completion queues



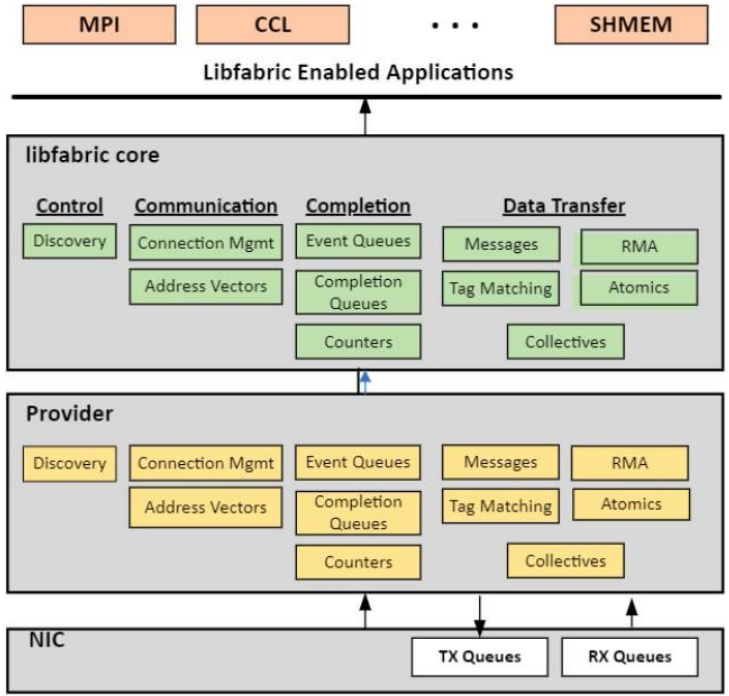
sockets API isn't rich enough for HPC/AI

Source: https://ofiwg.github.io/libfabric/

libfabric

expresses the UEC “Semantic” layer

- UEC
  - extends libfabric 2.0
  - creates a libfabric “provider” over UET
  - makes OFI contributions
    - e.g. reference implementation

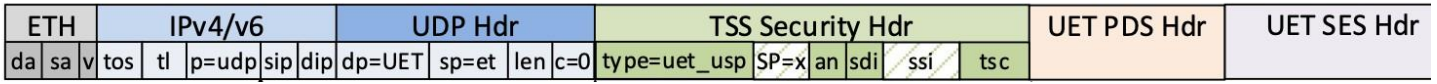


source: Ultra Ethernet Consortium,

UET Security

Integrated Security in Ultra Ethernet Transport

UEC Secure Transport with UDP header



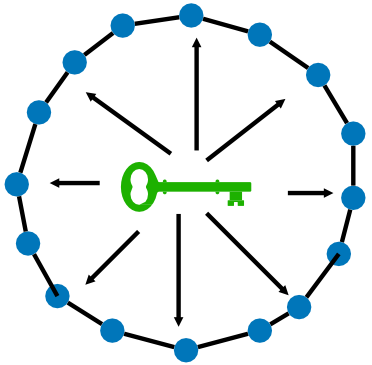
- Builds on core principles from IPSec and PSP
  - AES-GCM, KDFs, IVs, Key Rotation, Anti-Replay
- designed for high scale and group and client-server communication
- includes a model for host-level security and authorization

Integrated security to protect data, connection setup, …

source: Ultra Ethernet Consortium,

UET security group keying

- Security for **group applications**: Security Domains
  - Group Keying
  - Jobs exist in Security Domains
- Members trust others in the same group



efficient security for groups, integrated into UET

Link-layer retransmission

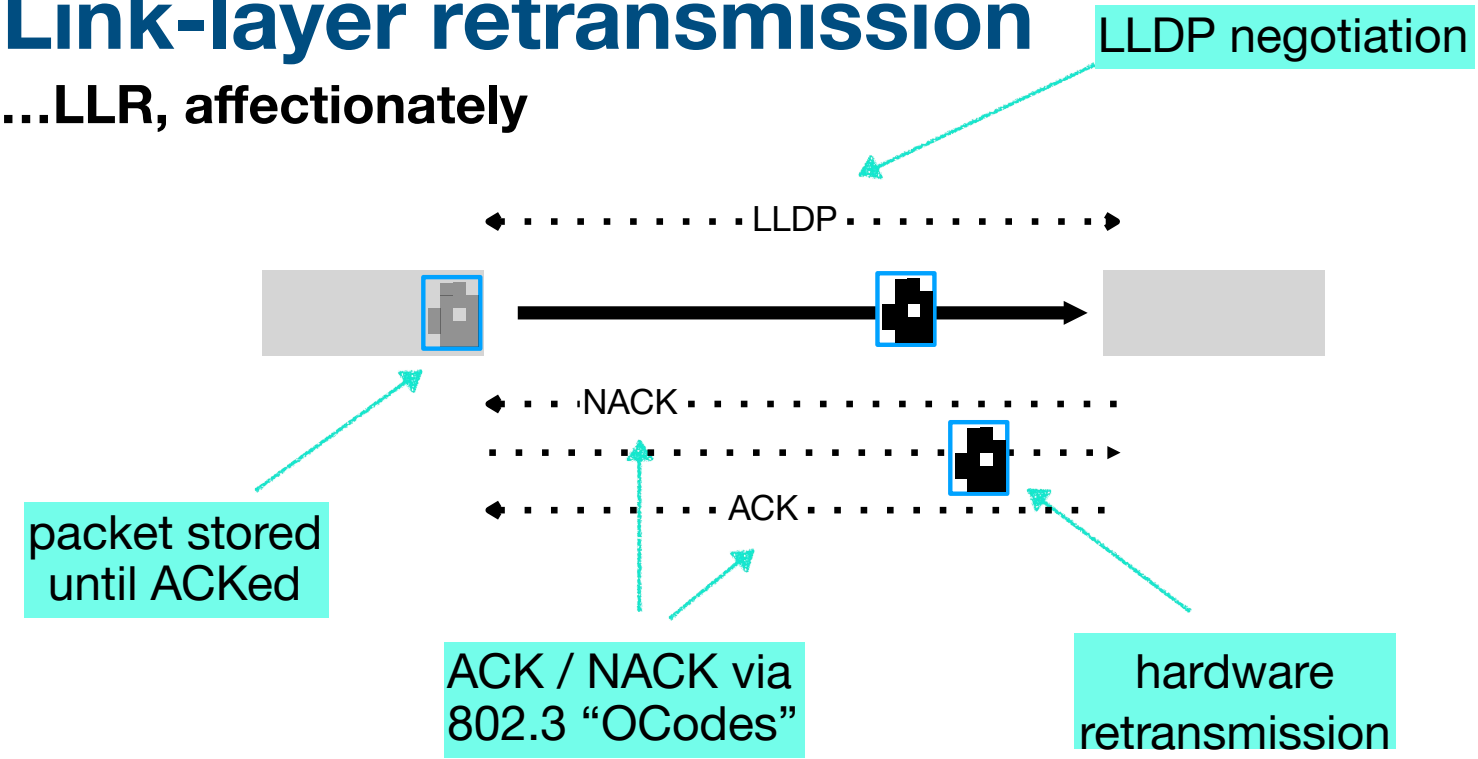
…LLR, affectionately

- Link and transceiver failures are a fact — and impact workloads
- An AI/HPC datacenter could have 256,000-512,000 transceivers
- Local retransmission to avoid end-to-end rxmit

improves tail latency

Link-layer retransmission

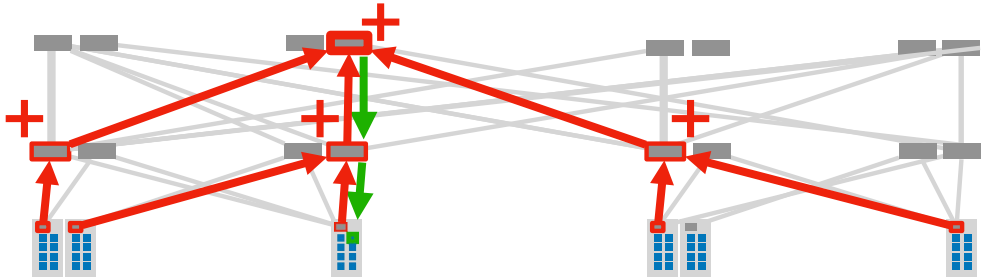
…LLR, affectionately



improves tail latency

In-Network Compute (INC)

Vector arithmetic in the network



INC
Reduce

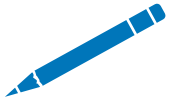
- Switch support for **Collective** operations : AllReduce, Broadcast, AllGather, …
- Switch(es) implement a **simpler**, transport protocol
  - tailored for point-to-point usage
- **APIs** coordinate the nodes

Saces bandwidth and reduces latency

Futures

UEC will continue after the 1.0 release

Sooner



- Storage - Storage APIs on UET
- Management - OpenConfig / RedFish
- Compliance and Testing, for profiles and optional features
- Performance and Debugging
- Telemetry - CSIG and BTS

Later, maybe…

- Programmable congestion control
- More topologies - DragonFly, DragonFly+, Slimfly, xFly
- More INC
- UET for regional / metro?
- Scale-up?

Summary

How is this relevant to ITNOG?

- Datacenters are not isolated - they will be interconnected
  - This is what many datacenters are doing internally
  - AI applications will inevitably spread to metro, regional, and WAN networks
  - Large flows and high BDP apply there too
- AI/HPC is an important new class of endpoints and flows
- **We need your vision** on:
  - the next round of problems
  - creative solutions!

In Conclusion

Networks for AI

- **Ethernet**: the standard solution for AI and HPC networks
  - Ethernet does and will support the features critical to AI and HPC
  - Ethernet will scale to 1,000,000s of GPUs
- **UltraEthernet** is ready for AI and HPC of the future

Join UEC and shape the future of AI and HPC networking

Thank you!