Transport and Load balancing Approaches in AI/ML Networks

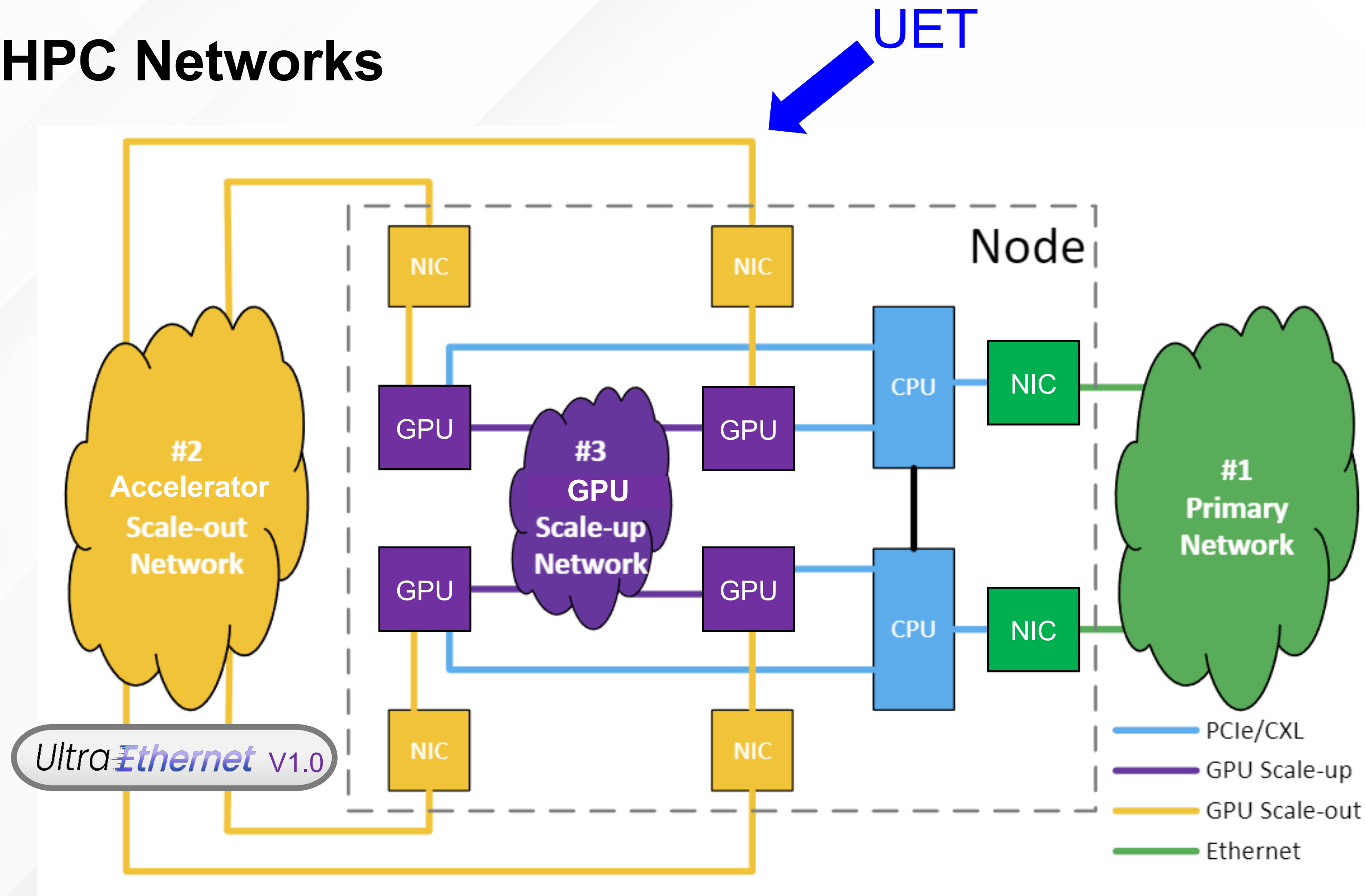
Costin Raiciu

Broadcom and Politehnica of Bucharest

Network Landscape Changing Dramatically

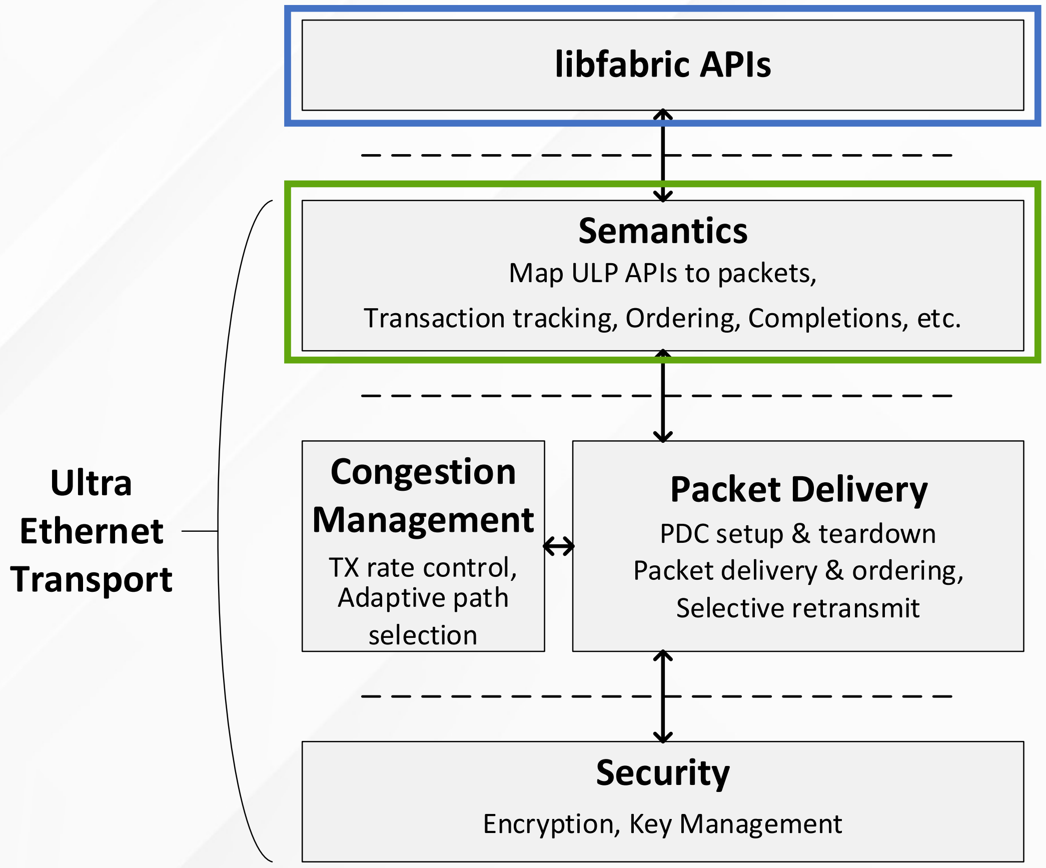
	CPU Cloud Networks		Large-Scale GPU Networks
Network	Single network	→ → →	Scale-up / Scale-out
Topology	3-4 tier folded Clos	→ → →	2-tier Clos, multi-rail, multi-plane
Switch support	ECN, PFC.	→ → →	Trimming, ECN, PFC.
Transport	TCP/IP, QUIC	→ → →	RoCE, Multipath / Ultra Ethernet
Transport impl.	Kernel (software)	→ → →	NIC
Datacenter Scale	100,000+	→ → →	~1M
Job Scale	10K	→ → →	~1M
Flows per host	High	→ → →	Low
Flow throughput	1-10Gbps	→ → →	Line-rate (e.g. 800Gbps)
Load Balancing	ECMP	→ → →	Multipath

AI/HPC Networks



UEC V1.0 targets the Scale Out Network

UET – UltraEthernet Transport

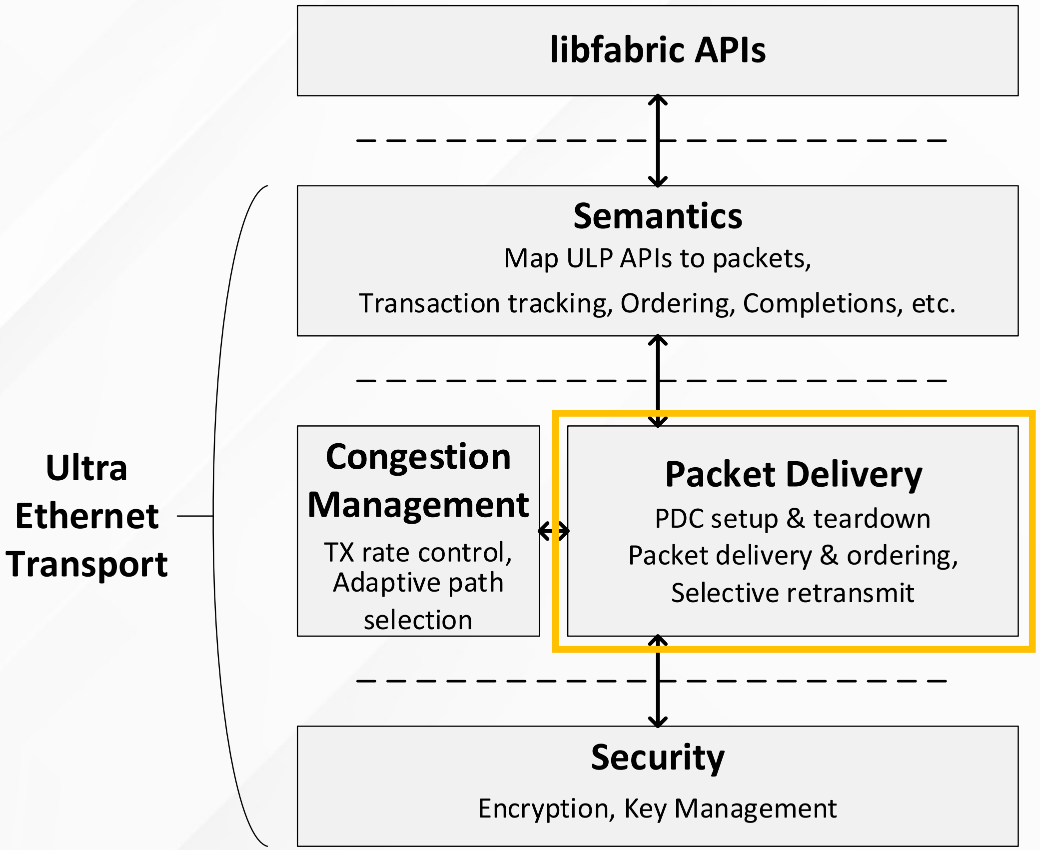


High scale, high utilization, low tail latency

- Simplify adoption by using existing, open standard
- Easy to port applications

- RDMA services: Send/Rcv, Write, Read, Atomics
  - Focus on MPI and *CCL
  - Shared receive queue
- Scalable addressing
- Optimized extensions:
  - Deferred Send improves RNR
  - Rendezvous using Send/Read
  - Exact match tags for HW offload of ordering between endpoints using shared receive queues

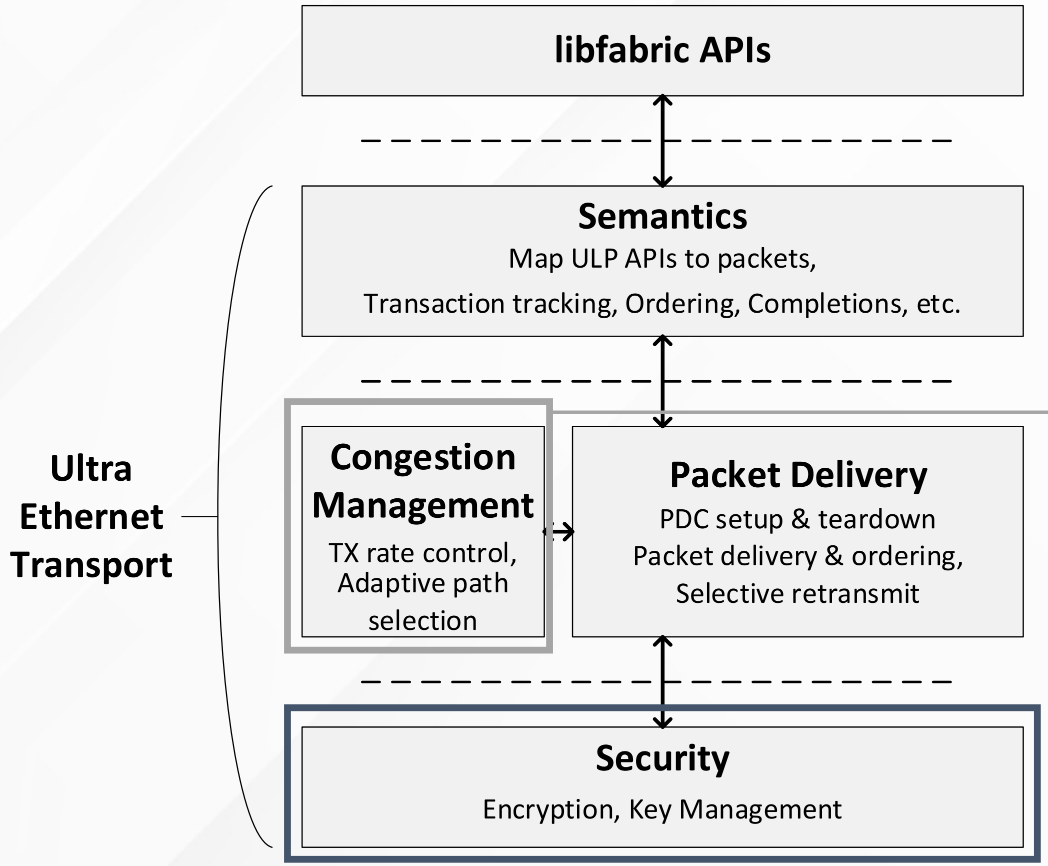
UET – UltraEthernet Transport



High scale, high utilization, low tail latency

- Dynamic, ephemeral connections
  - Zero start up time, 1-RTT close
- 4 delivery services:
  - ROD – Reliable, ordered
  - RUD – Reliable, unordered
  - RUDI – Reliable, unordered, idempotent (Write/Read)
  - UUD – Unreliable, unordered
- Out-of-order packet arrival
- Selective acknowledgement and retransmission for RUD & RUDI
  - ROD uses Go-BackN

UET – UltraEthernet Transport

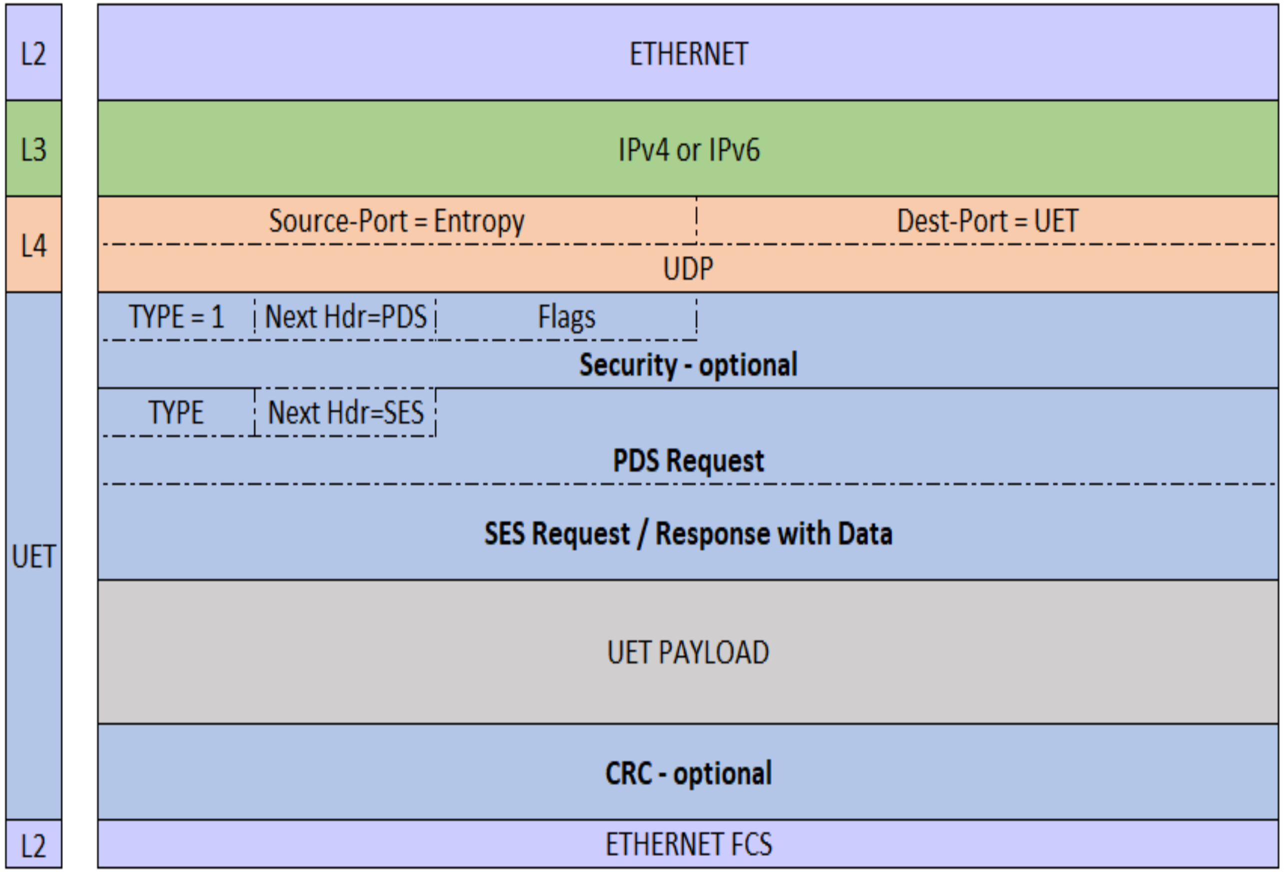


High scale, high utilization, low tail latency

- Multipath with congestion avoidance
  - Leveraging ECMP
- Trimming with NACK signal
- Network Signaled CC (NSCC)
  - Window based at sender using RTT and ECN
- Receiver Controlled CC (RCCC)
  - Credit based at receiver

- End-to-end AES encryption
- Key derivation for additional security
- Replay protection
- Scalable security domains
- Optional within UET

Network Header Stack



- UDP is optional
  - If UDP removed, a 4B entropy header is added
- Security is 16B
  - Optional
  - Includes 16B ICV after UET payload (no CRC in this case)
- PDS header – 12B
  - 16B for RCCC
- SES header – 44B
  - Compact options available
- CRC – 4B
  - Or 16B ICV if using crypto

Now for some details

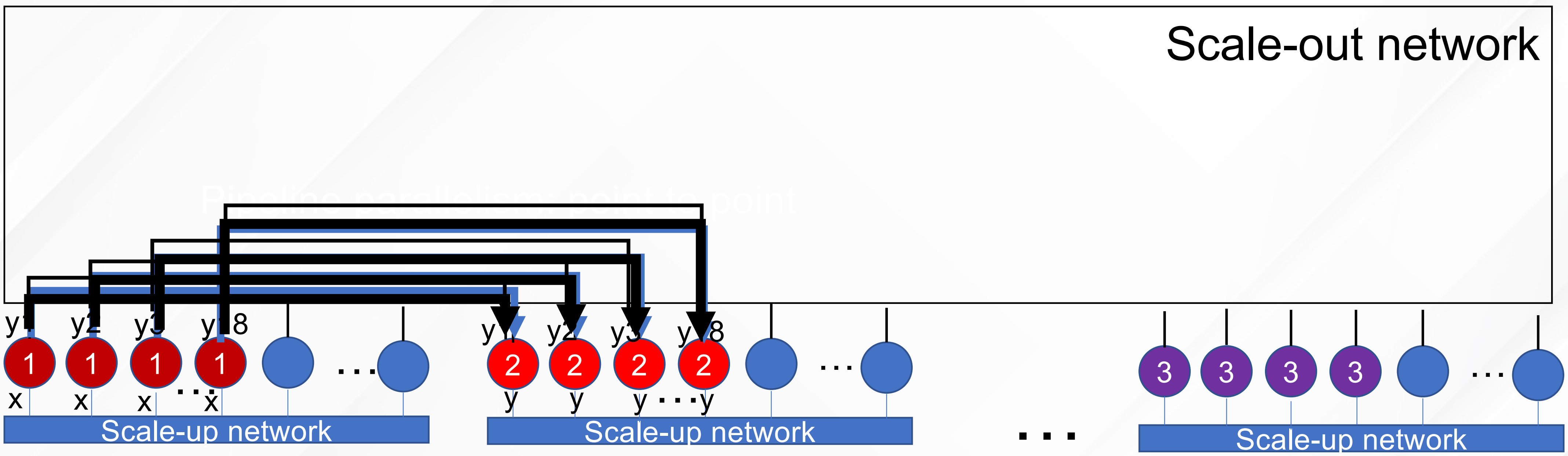
What are the requirements of AIML workloads?

How do you build a packet spraying protocol?

How does UET congestion control work?

What is packet trimming?

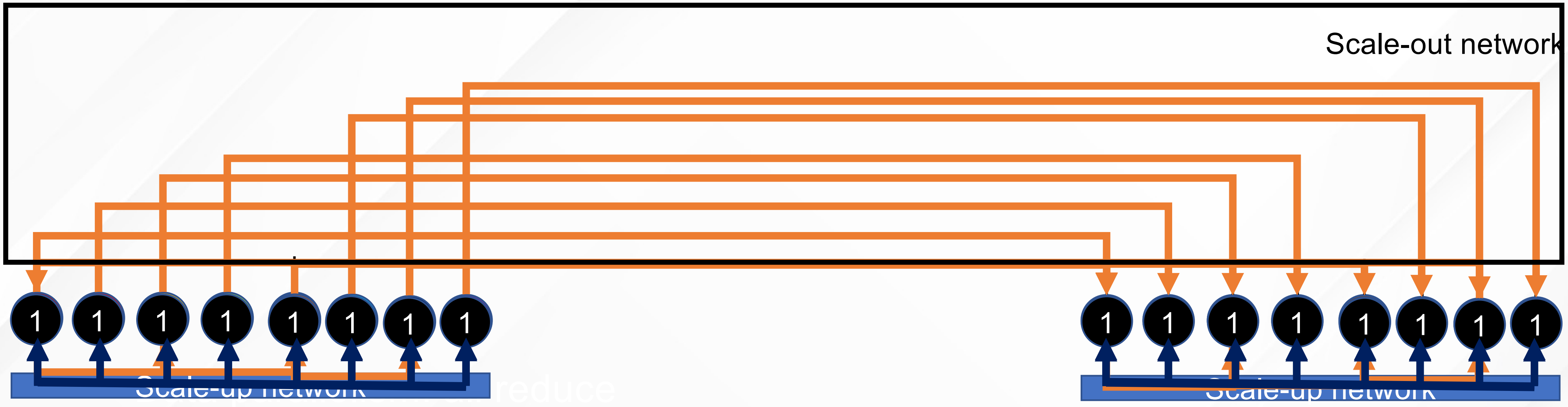
How do AIML workloads use the network?



100T model prediction:

- 18 B200 GPUs needed to hold 1 layer.
- size (x) = size (y) = 3.86GB
- Exposed networking: 38ms with 130ms fw / 260ms backward computation.
- Shard data to use scale-out efficiently: 215MB per GPU + all-gather at destination: 2.15ms only!

How do AI/ML workloads use the network?

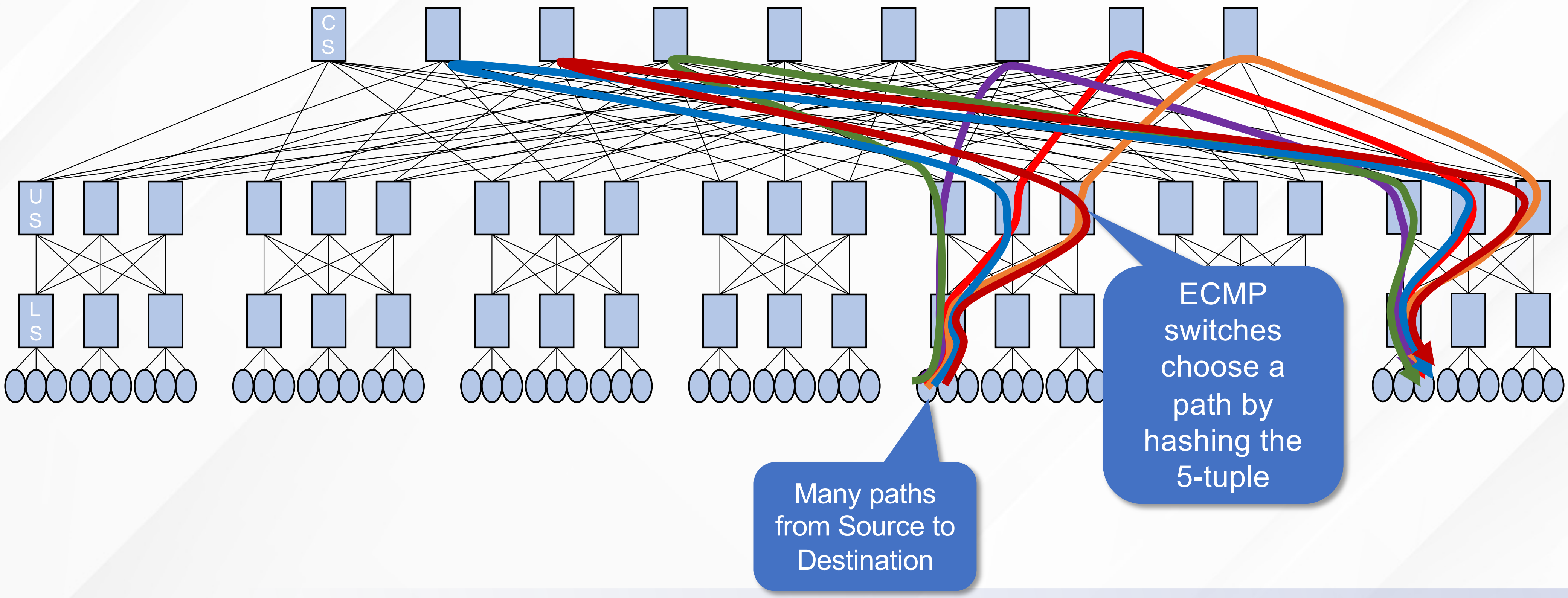


<u>Takeaways</u>: Scale-out collectives also use scale-up, have poor locality!

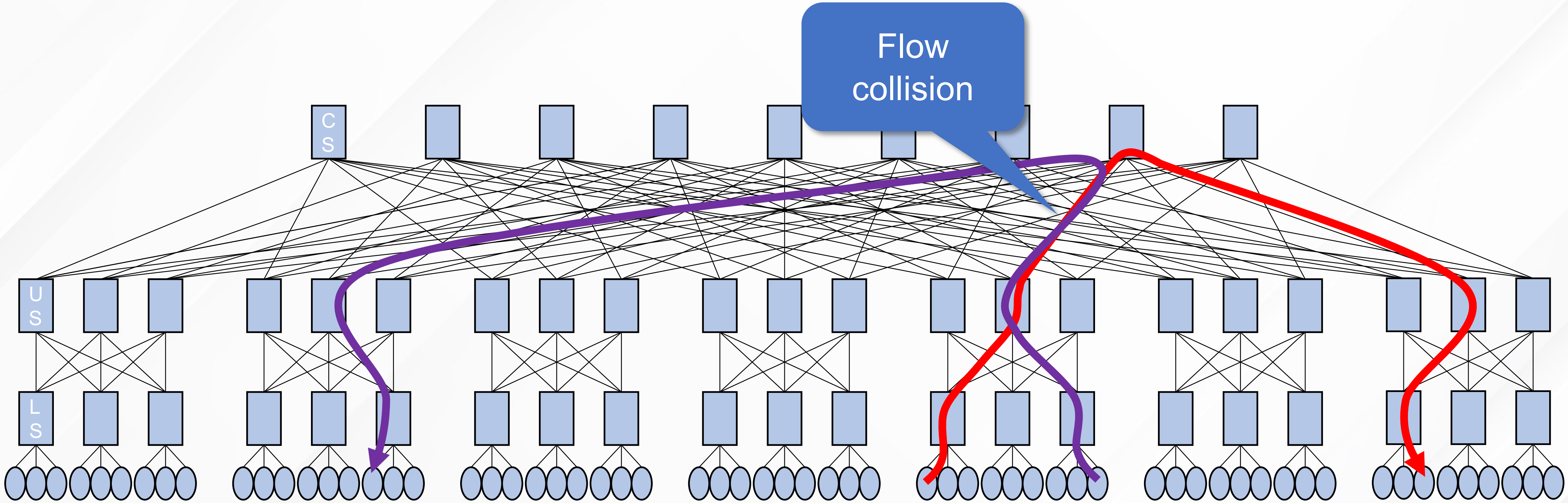
Pipelining used to reduce latency =>

- scale-out flows will be small and have little locality.
- scale-out flows start simultaneously, fully load network.
- coupling between scale-up and scale-out flows.

Flow routing in a Clos topology

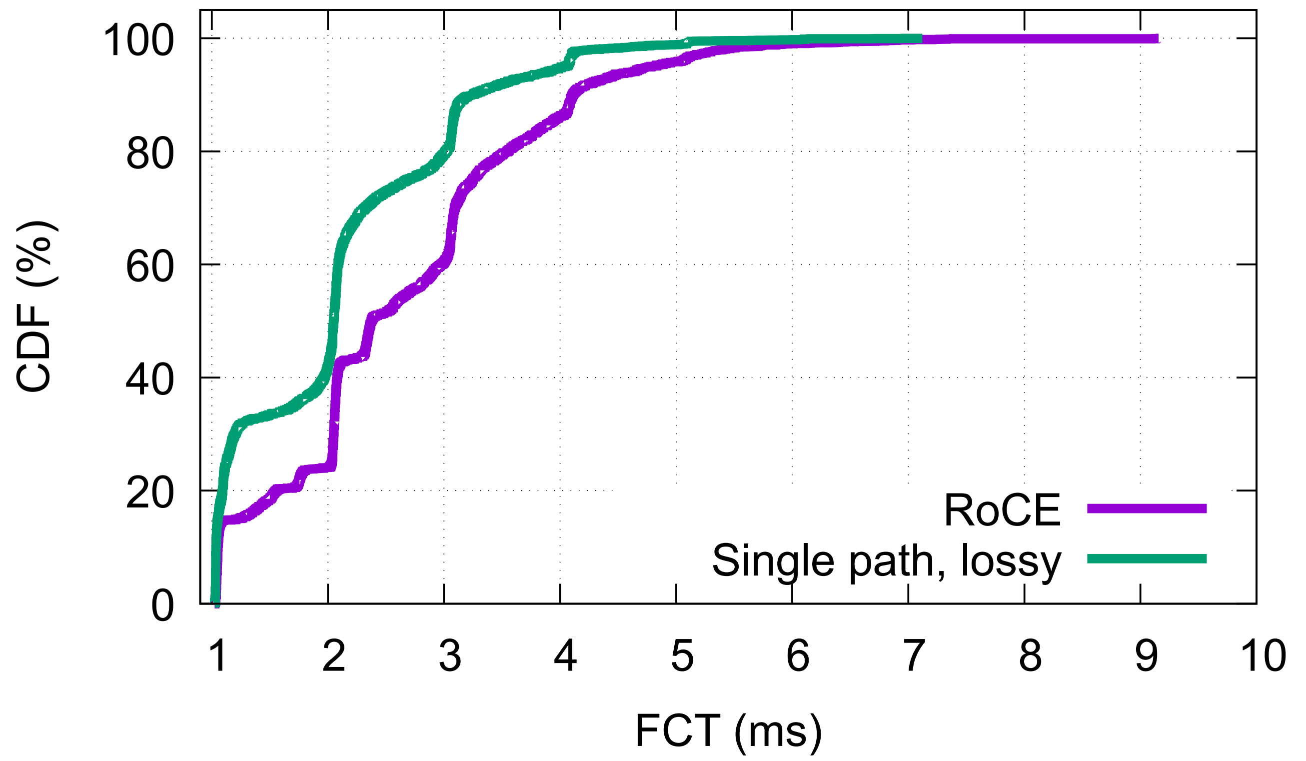


Flow collisions



Are RoCE / Infiniband good-enough for AIML?

Simulated pipeline parallelism, 800Gbps links, 8K nodes leaf-spine, **100% load**

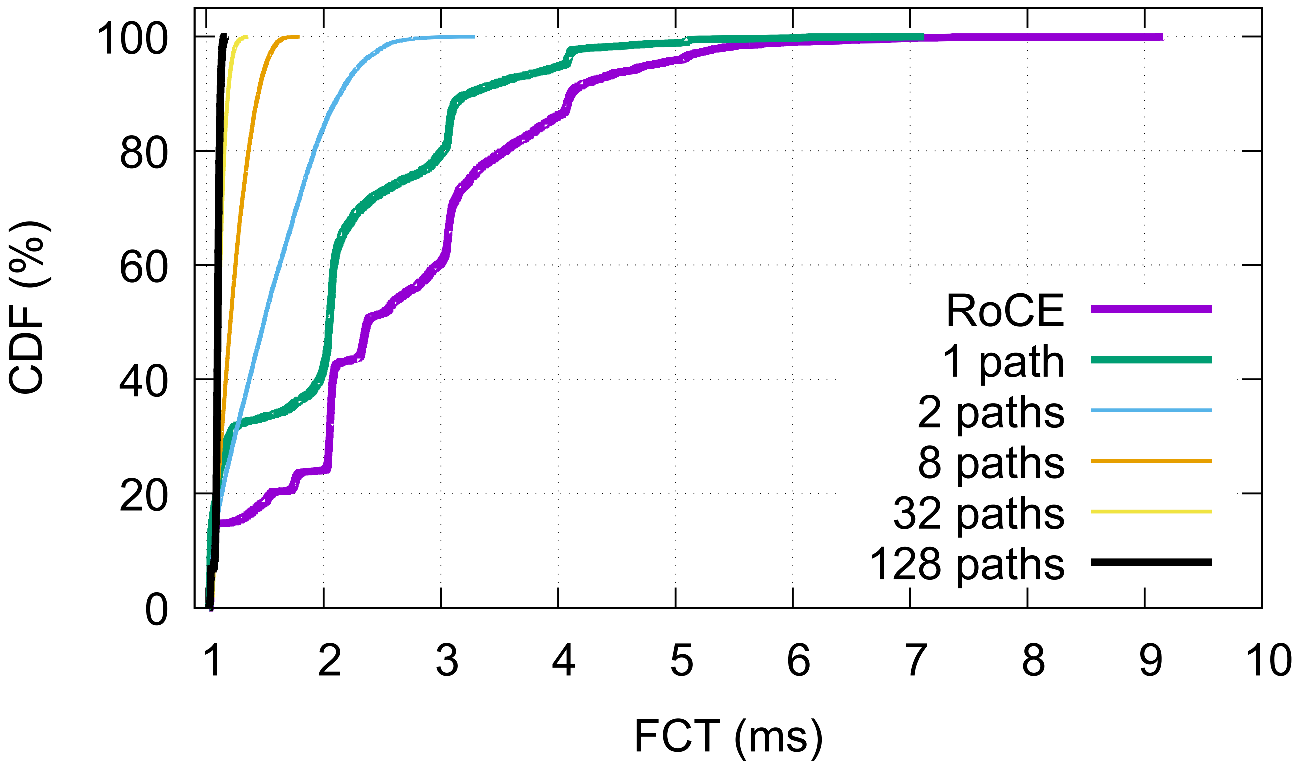


Flow collisions are the main issue for single path transport

Collisions and PFC head of line blocking the main issue for RoCEv2

Ultra Ethernet Transport: a multipath replacement to RoCEv2.

Need multipath for pipeline parallelism



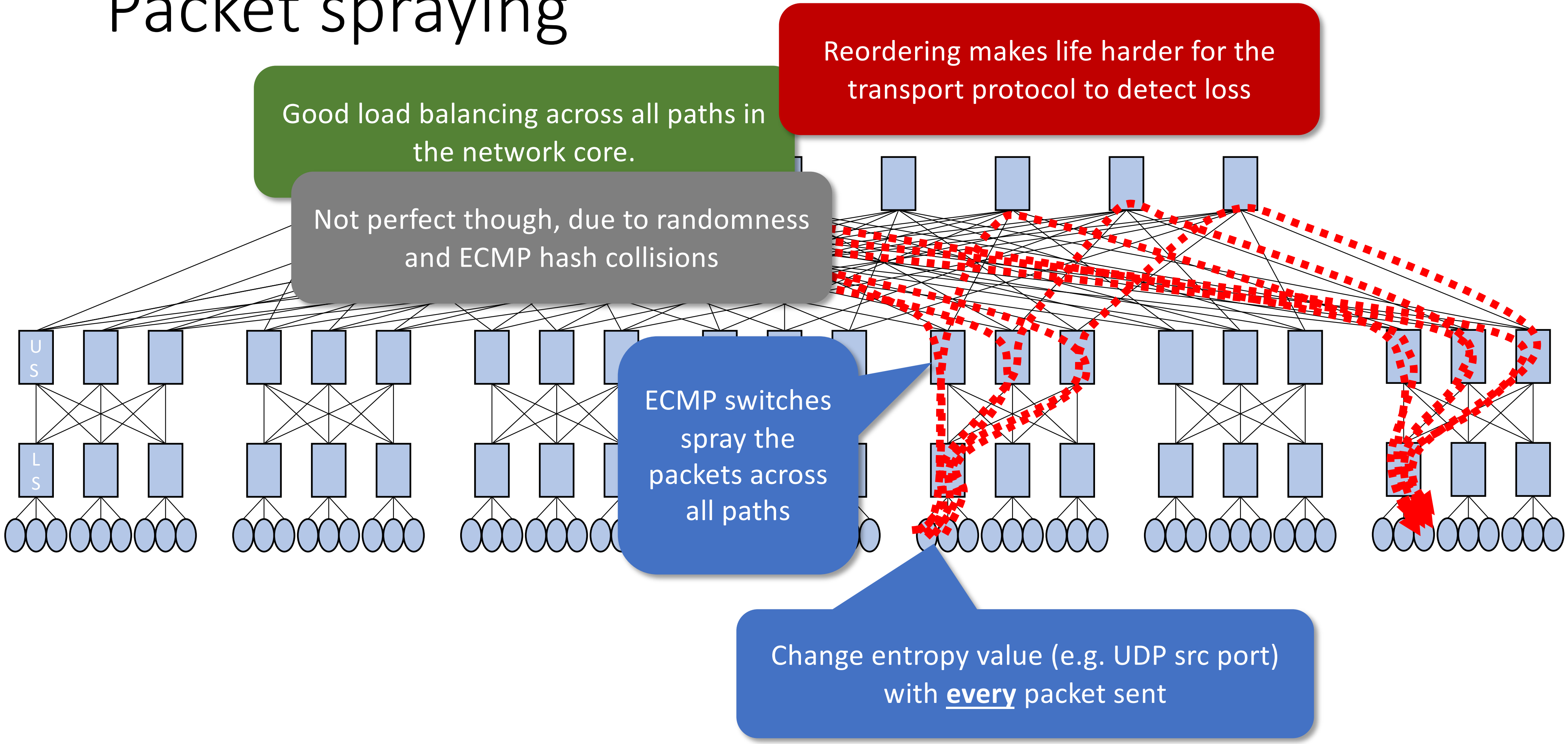
Takeaway: multipath is key to reduce tail FCT for ML training.

Flows start at line rate, but don't like PFC. Packet trimming being standardized.

Ultra Ethernet Transport

- Network-wide RTT and linkspeed – known to transport.
- Global window limits total amount of traffic.
  - Upper bounded to 1.5BDPs.
- Zero-RTT connection setup.
- Packet spraying:
  - Sender adds an entropy value (EV) to each packet (e.g. UDP source port).
  - Sender chooses different EVs for different packets.
  - Switches hash EV & 5 tuple = packets take probabilistically different paths.
- Load balancing to spread traffic across available paths.
  - How should we choose EV for outgoing packets?

Packet spraying



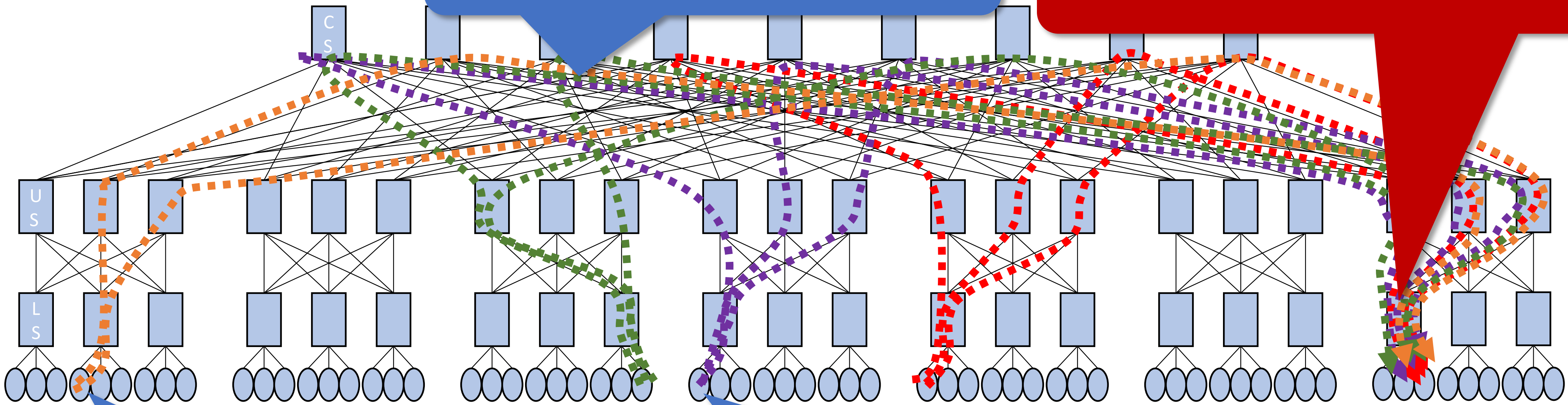
Ultra Ethernet Transport: reliability

- Sender sends packets on different paths.
  - Packets have Ack Request bit requesting immediate ACK.
- Selective Acknowledgements:
  - Carry cumulative ACK.
  - Reference ACK + bitmap.
- SACK generation algorithm:
  - When AR bit is set.
  - When packet is ECN marked.
  - When a fixed number of packets or bytes where received (e.g. 16KB).
  - Bitmap logic: cover newly arrived packets with oldest PSN.

Flow start up

Spraying load-balances the core for all flows

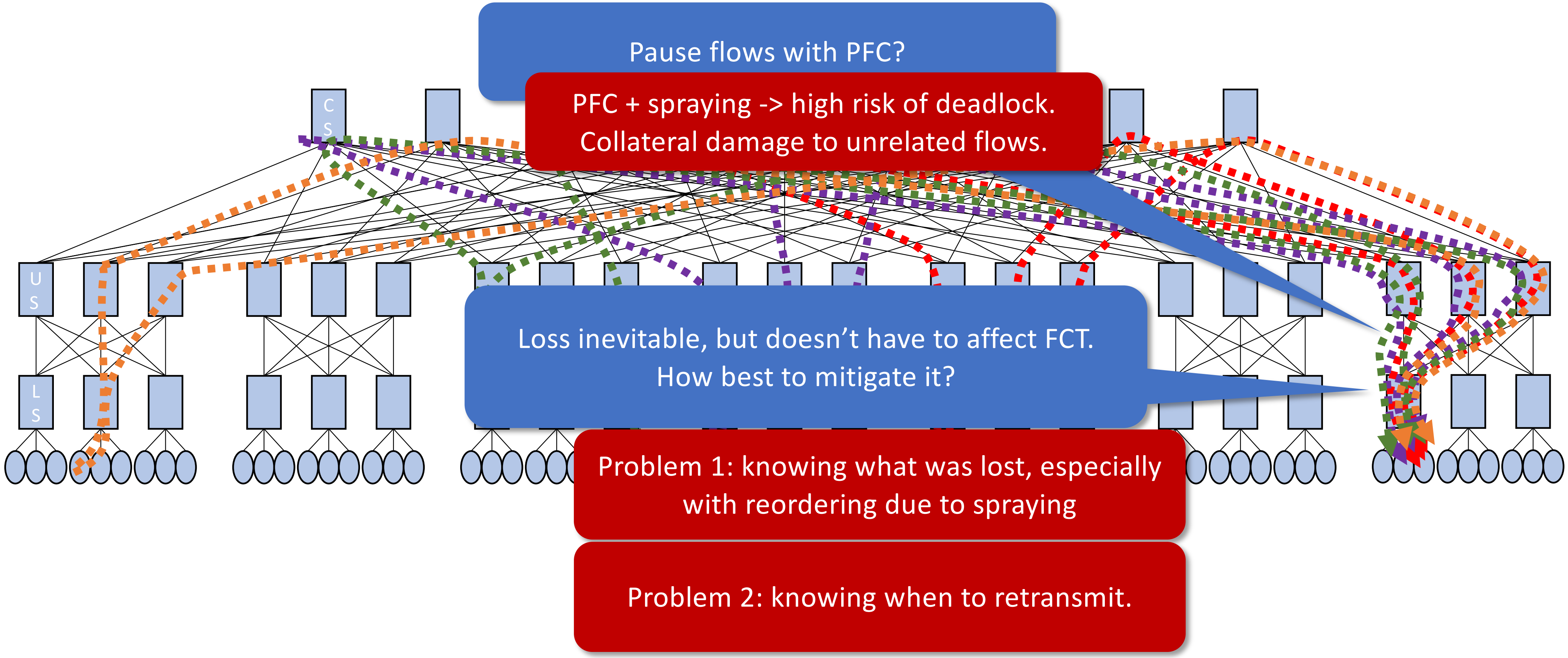
Incast close to the receiver is a problem



To minimize flow completion time, flows need to start at line rate in first RTT

Synchronized collectives mean flows start simultaneously

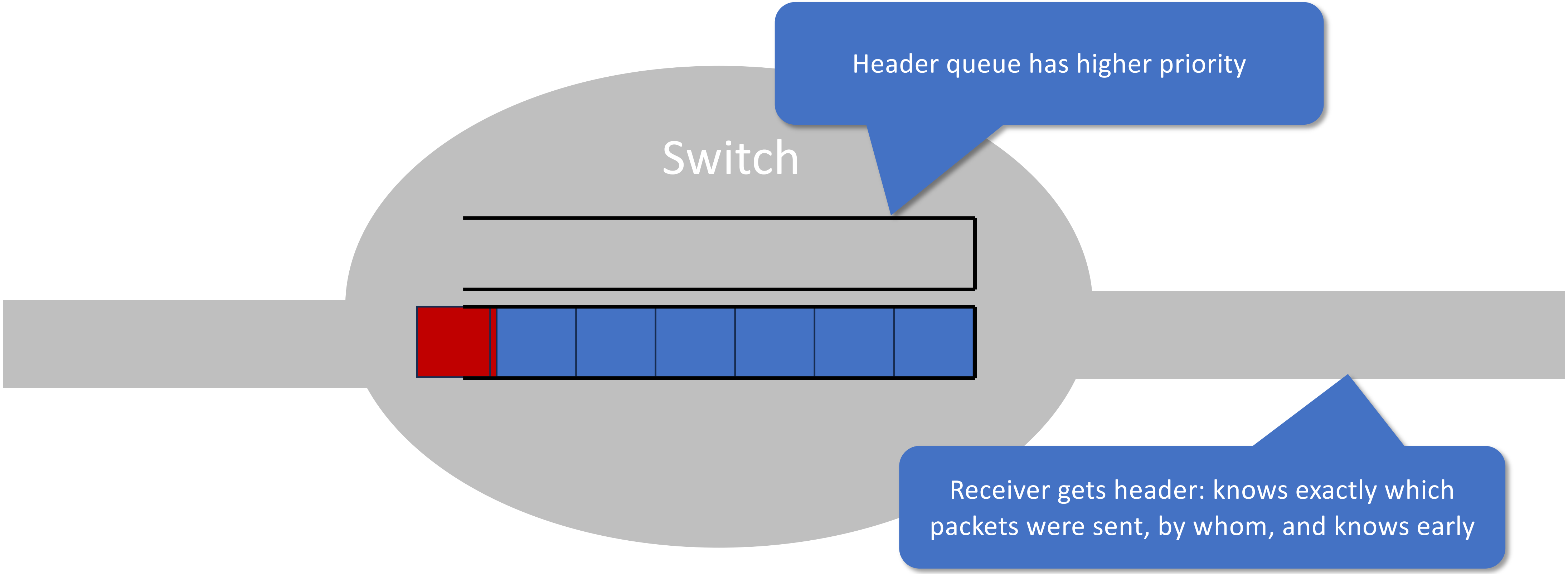
Flow start up



Ultra Ethernet Transport: loss detection

- Packet spraying makes loss detection harder.
- Senders detect lost packets and retransmit them with priority.
- Three methods of detection:
  - **Packet trimming + receiver NACKs.**
  - Receiver out of order count.
  - Keep per EV state to detect lost packets.
  - **Retransmit timeouts, per packet.**

Packet trimming



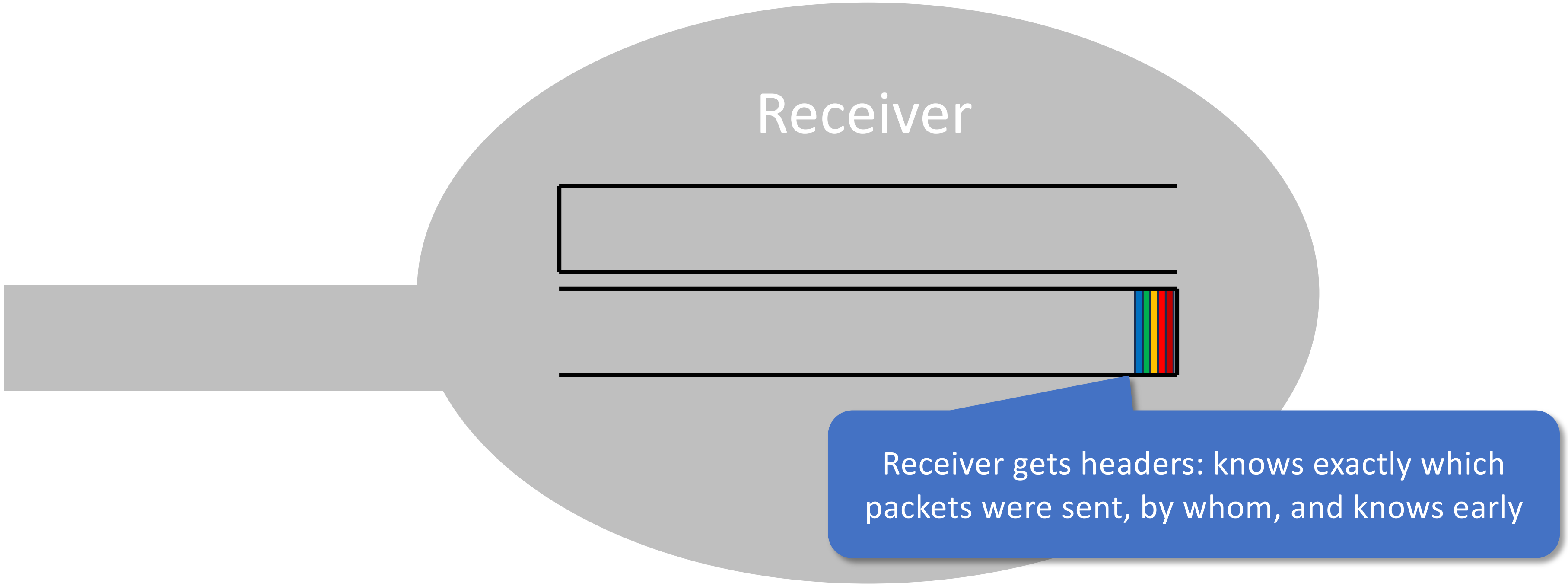
What about congestion control?

- Always start at line rate, after 1st RTT congestion control kicks in.
- Trims at receiver lead to NACKs which notify sender of lost packets.

- NSCC: Sender-driven congestion control
  - Targets sub-BDP standing queue at the bottleneck.
  - Use ECN and delay simultaneously.
    - Aggressive increase when queue ~ 0. Gentler increase otherwise.
    - Multiplicative decrease when ECN & delay above threshold.
  - Average delay across all paths.
  - Handles both incast and oversubscription.

- Receiver-driven control – see next slide (e.g. EQDS).

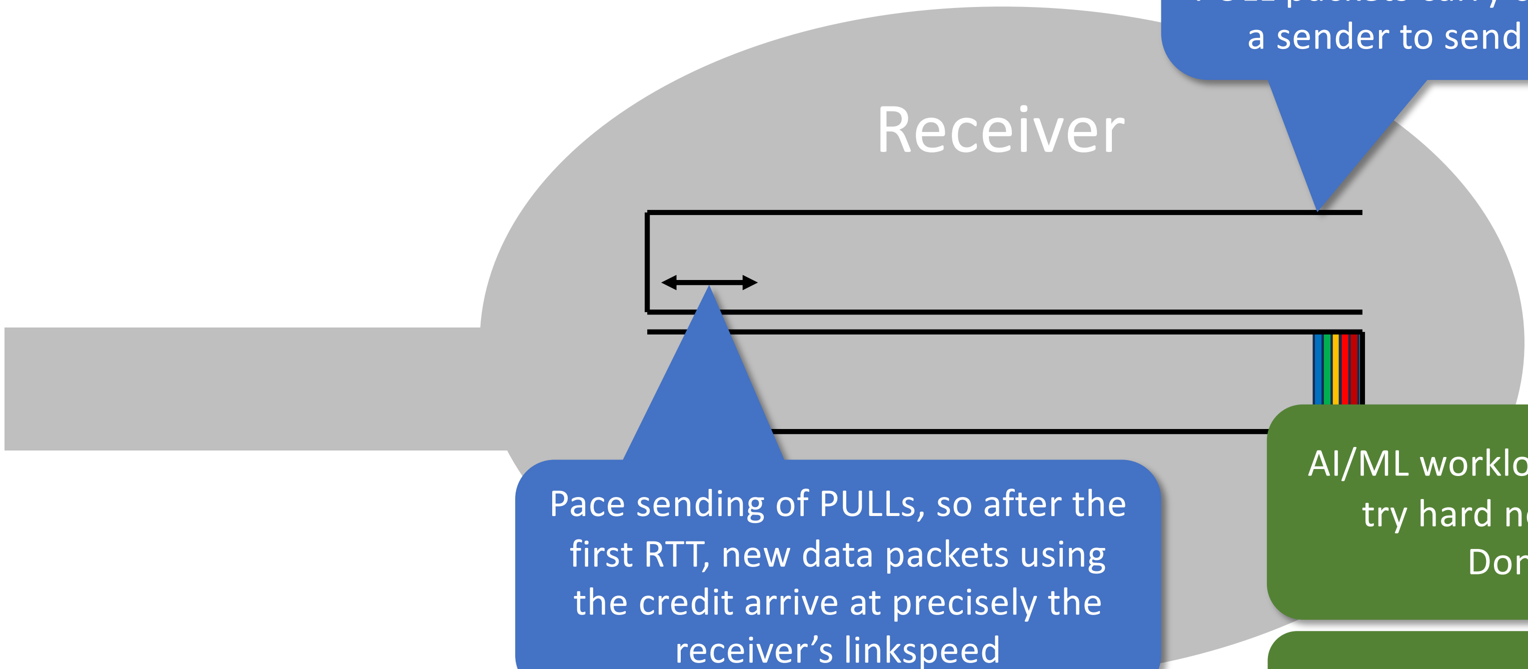
Receiver credit pacing



Receiver credit pacing

For each unique sender, place a PULL packet in the pull queue.

PULL packets carry credit, allowing a sender to send more data.



Pace sending of PULLs, so after the first RTT, new data packets using the credit arrive at precisely the receiver’s linkspeed

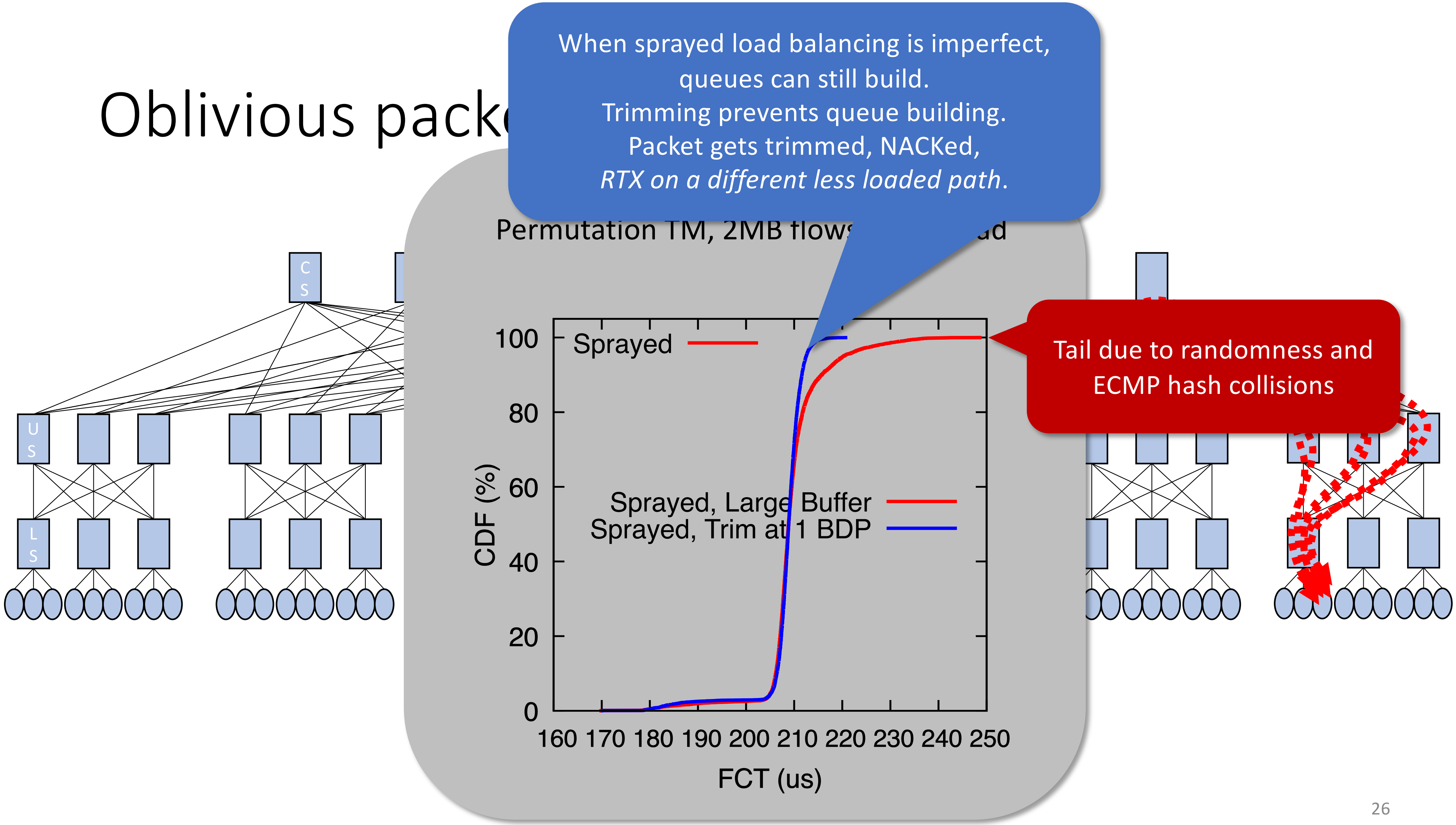
AI/ML workloads (and collective libraries) try hard not to create large incasts. Don’t always succeed!

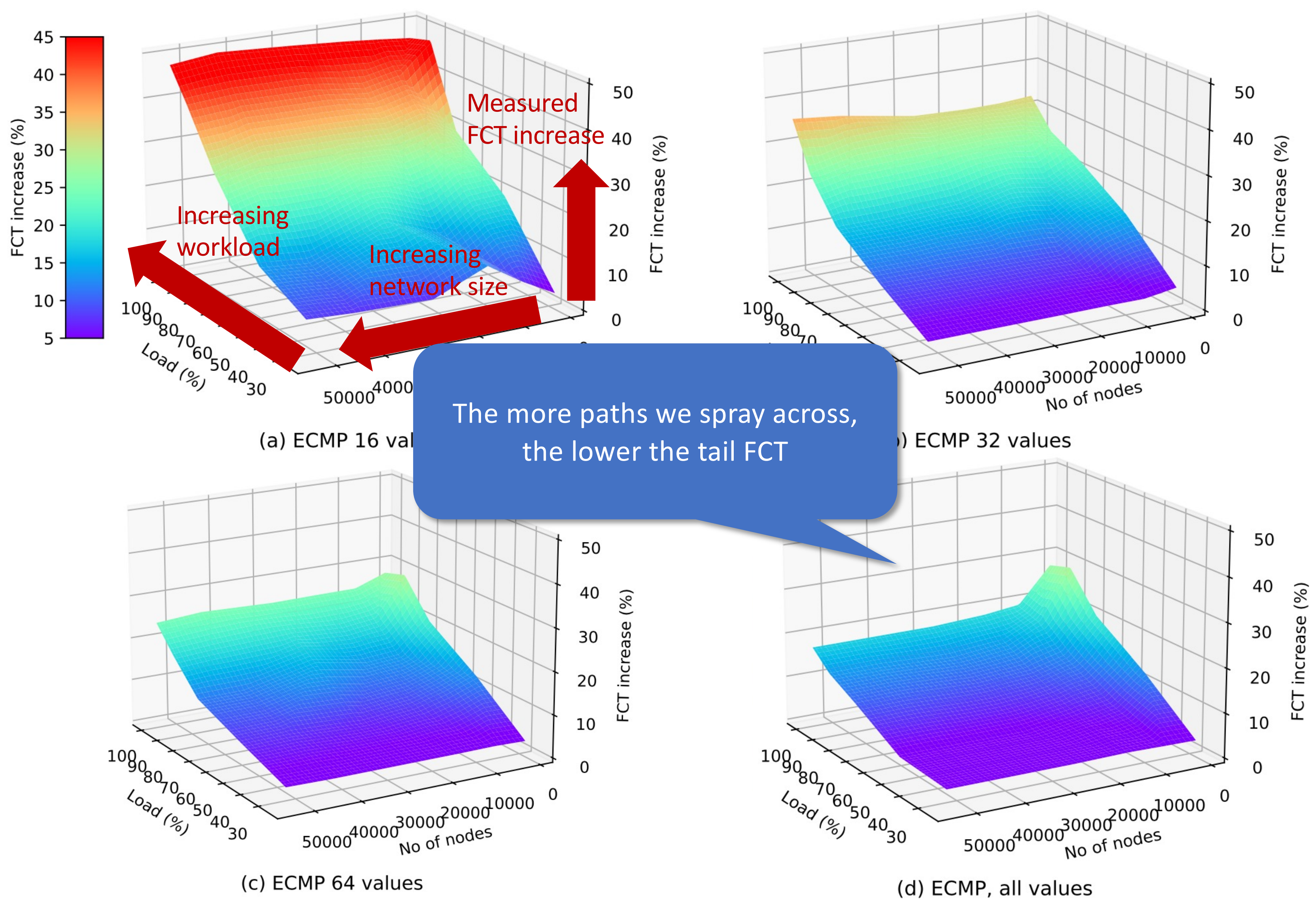
Storage servers can see large incasts. Receiver credit controls incasts really well.

What is the best way to spray packets?

- **Simplest: oblivious load balancing**
- Pick a random EV for each packet.
- Works very well if network capacity is uniform.

Oblivious packe





(a) ECMP 16 values

(b) ECMP 32 values

(c) ECMP 64 values

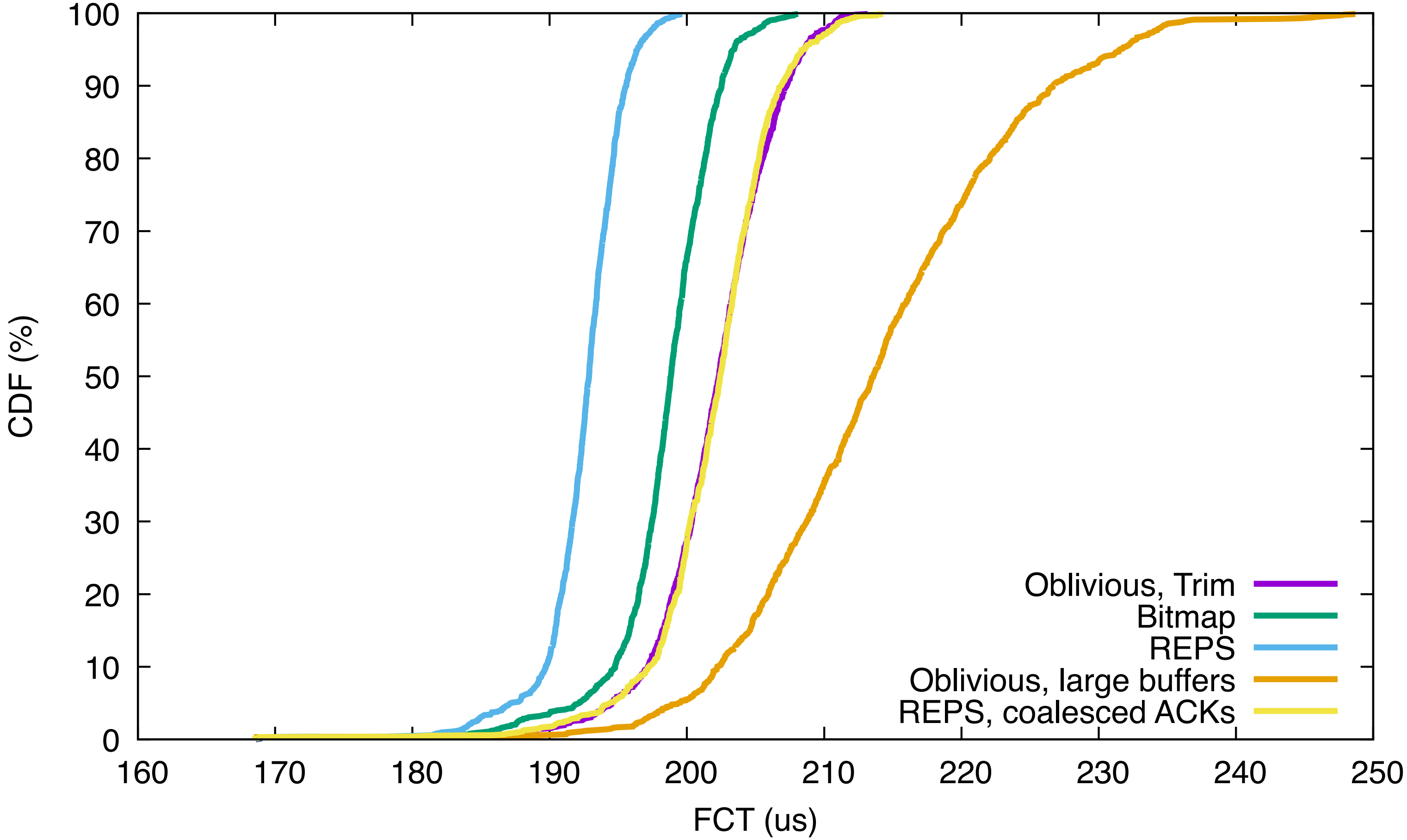
(d) ECMP, all values

Keeping per path state

- Bitmap load balancing (e.g. Strack)
  - Per EV state - one or a few bits.
  - When ACK indicates ECN mark, increment EV state.
  - When EV is next to be picked but non-zero state, decrement state, skip.

- Recycled entropies (REPS):
  - Keep EV cache for which we got an ACK without ECN set.
  - Path selection: pick EV from cache if non-empty. Otherwise pick random EV.

Load balancing algorithms comparison



Summary

- Ultra Ethernet Transport uses packet spraying and window based congestion control.
- Networks run in best-effort mode, packet trimming recommended.
- Highly scalable design with:
  - Shared receive queue.
  - Dynamically initiated Packet Delivery Contexts.
  - Single context per host-pair.
- Specification published in June 2025.
  - Development started in late 2022.
  - Many companies contributed.

Backup slides

Multipath TCP
RFC 8684, 6356

- Open multiple subflows between the source and destination.
  - ECMP hashes each subflow to a (probabilistically) different path.
  - MPTCP connection setup: 2RTTs.
- Multipath congestion control
  - One window per path ($cwnd_j$), each with its own ACK clock, sequence space.
    - Subflow acts like a TCP connection from the network point of view.
    - But its cwnd is not independent.
  - Per path window depends on that window at all other windows.
    - On each ACK for subflow j, $cwnd_j$ += *a* / total_cwnd
    - On each loss for subflow j, $cwnd_j = cwnd_j / 2$

Why not use MPTCP for AIML networks?

- Connection setup increases FCT by at least 1RTT.
- Load balancing works well for long flows (hundreds of RTTs)
  - Not so well for shorter flows.
- Need to use many paths.
- But minimum MPTCP window depends on #paths.
  - E.g. 256 paths means min 256 packet window.
  - (This equals BDP at 800Gbps).
  - Congestion collapse in incast.

Packet Delivery Highlights

- Four delivery modes

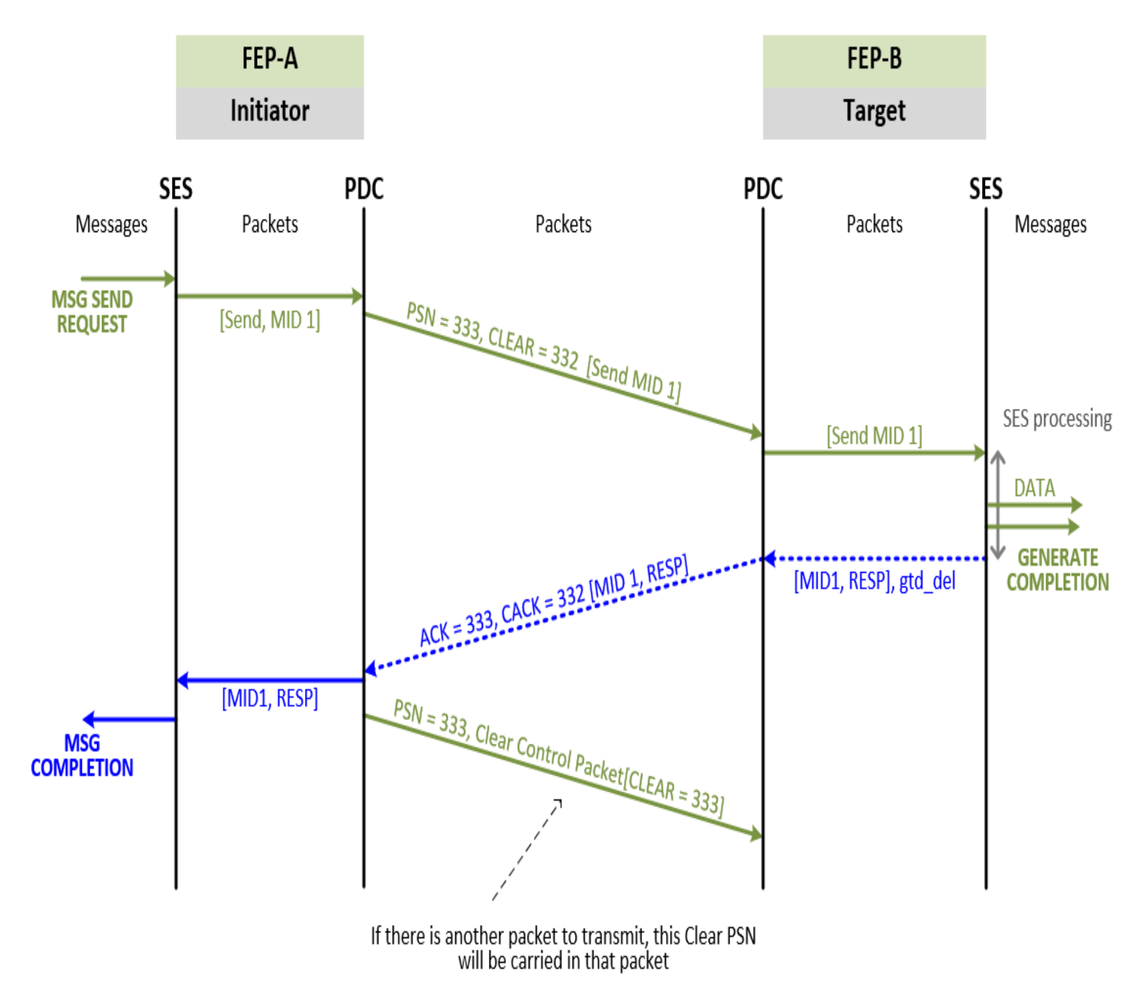
Abbr.	Name	Req?	Description
RUD	Reliable, Unordered	Yes	Ephemeral connection using Packet Sequence Numbers (PSN)
ROD	Reliable, Ordered	Yes	Ephemeral connection using Packet Sequence Numbers
RUDI	RUD for Idempotent ops	HPC	Connectionless using packet numbers
UUD	Unreliable, Unordered	Yes	Connectionless datagram service, no acknowledgements

- Dynamic, ephemeral connections for RUD/ROD
  - Minimize persistent state to maximize scalability
  - Zero start up time
  - Guaranteed delivery of both request and response/ACK
- Out of order data placement and selective retransmit
  - ROD uses GoBackN to maintain order

Packet Delivery Context (PDC) – RUD/ROD

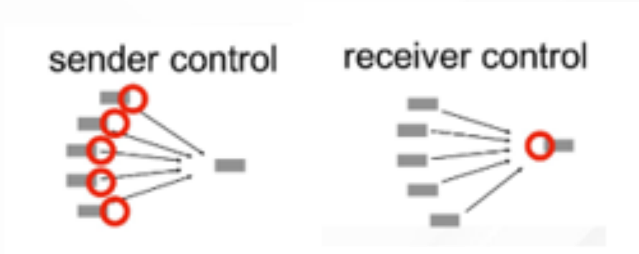
- Limited number of contexts
  - No fixed destination ala RoCE QP
- Based on transmit request, re-use existing PDC or allocate a new one
  - Zero start up – first packet carries data
  - One RTT close

- Transmit a PDS Request
- Receiver generates an ACK
  - ACK carries response … usually ‘OK’
- If ACK carries any state, then ACK is ‘cleared’ (~acknowledge the ACK)
  - E.g., if error event or other state



Congestion Management Highlights

- **NSCC**: sender-based algorithm using RTT and ECN to control window size
  - Uses delay as primary measure of congestion
  - Required
- **RCCC**: receiver-based algorithm using credit
  - Receiver allocates credit to sender
  - Optional



- **Multipath** (aka packet spraying) for RUD, RUDI, UUD
  - Method is not defined, a few examples provided (REPS, based on ECN, oblivious)
- **Trimming** – rather than drop a packet when a buffer is congested, trim the packet and forward at high priority to the destination
  - Destination has information to request retransmission – send a NACK
  - Required support on FEPs (NICs), optional support in switches

Semantic Layer Highlights

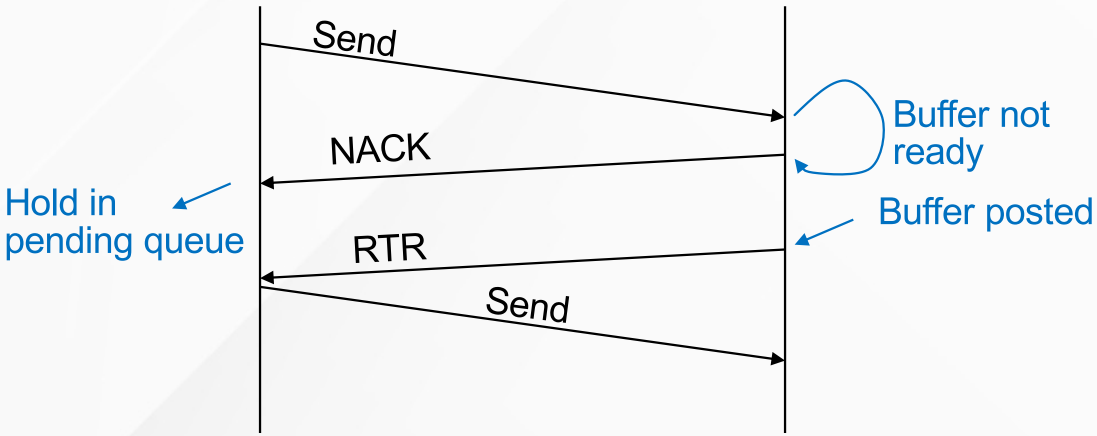
- Addressing: {Fabric Address, PIDonFEP, Resource Index}
  - Fabric Address selects the FEP = fabric endpoint (e.g., NIC); this is an IP address
  - PIDonFEP selects a process (e.g., rank) on the FEP
  - Resource Index selects a service (e.g., MPI, CCL) and the asso. queue or RMA memory region
- JobID is used to isolate resources across different jobs or clients
- All messages have a corresponding completion with error/success indication
- Types of operations
  - Ordered or unordered
  - Optional 64-bit header data (aka immediate)

Operations
Send – tagged and untagged
Write
Read
Atomic – fetching & non-fetching
Rendezvous Send
Deferred Send

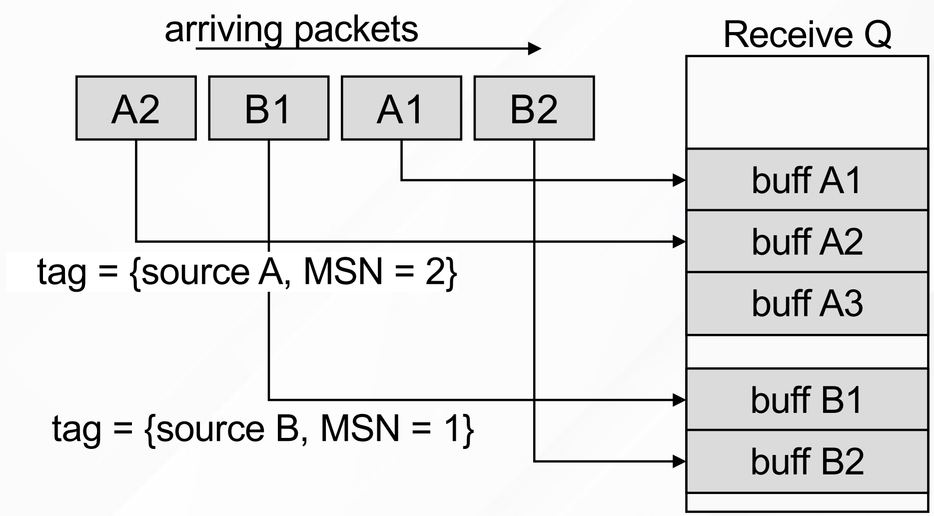
Responses
Default
Response – with state, poss. Err
Response with data
No response

Semantics – Deferrable Send and Exact Match Send

- Deferrable Send is an improvement compared to RoCE RNR
  - If the target buffer is not available, the received NACKs the packet(s)
  - When the buffer is posted, the receiver sends an RTR which indicates to the sender the buffer is now available – resend the msg



- Exact Match Send provides a method to identify specific buffers within a shared receive queue
  - UET uses dynamic connection to save state so there is no RQ per sender
  - RQ per sender can be emulated using a simple tagging scheme – e.g., use a tag like {source rank, MSN}



UET Profiles – Summary of RQMTs for NICs

Feature	AI Base	AI Full	HPC
Send	1 MTU	REQ	REQ
Tagged Send – Exact Match		REQ	REQ
Tagged Send - Wildcard			REQ
RMA Write	REQ	REQ	REQ
RMA Read		REQ	REQ
Atomics		REQ	REQ + tags
Deferrable Send	REQ*	REQ	
Rendezvous Send			REQ
RUD, ROD, UUD	REQ	REQ	REQ
RUDI			REQ
NSCC	REQ	REQ	REQ
RCCC			
Multipath & Trimming	REQ	REQ	REQ
Security			

UEC 1.1 Specification – Current SDRs

- Four SDR are approved for the next phase of UEC work in TR WG
- The are likely to be clarifications and improvements to UET
- Additional SDRs may be proposed

SDR	Topic	Leader
TR3006	Congestion control for Lossless networks	Costin Raiciu
TR3007	Programmable congestion control (PCC)	Torsten Hoefler
TR3009	Congestion Signaling (CSIG)	Jai Kumar & Brad Karp
TR3010	UltraEthernet for Local Networks (ULN)	Karen Schramm