ETHEREAL

Divide and Conquer Network Load Balancing in Large-Scale Distributed Training

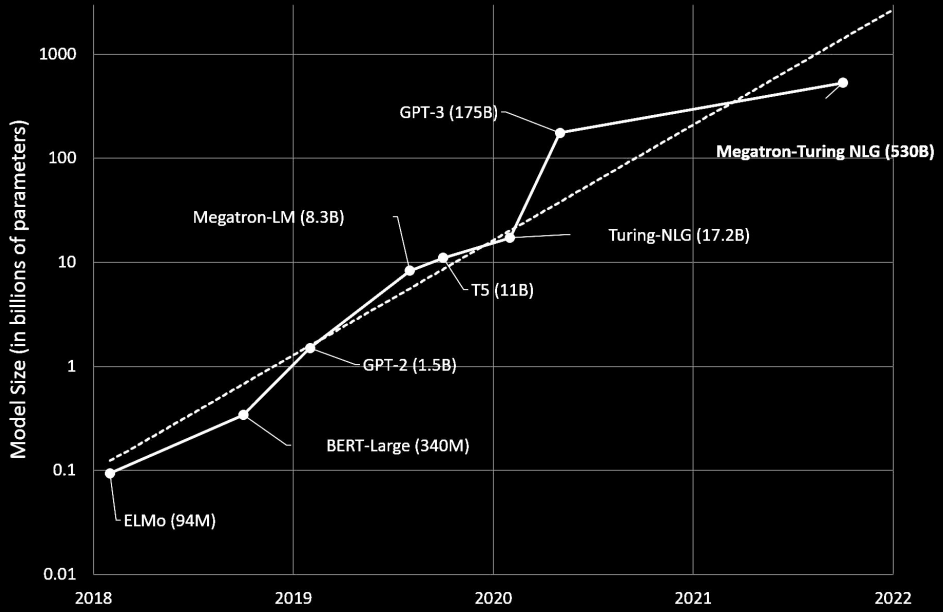
Vamsi Addanki, Prateesh Goyal, Ilias Marinos, Stefan Schmid

Technische Universität Berlin · Microsoft · NVIDIA · Technische Universität Berlin

Fantel Side Meeting, IETF 122
18 March, 2025

Motivation: Parallel Computing

- Exponentially growing compute requirements of language models

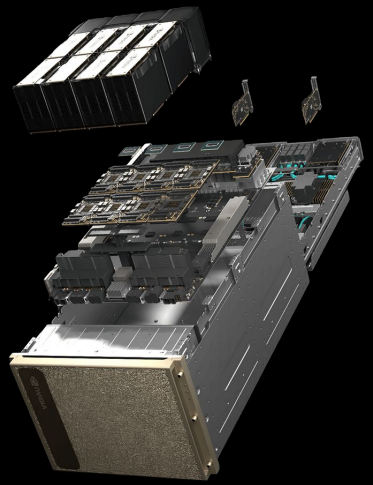


Trend of sizes of state-of-the-art NLP models over time

ETHEREAL

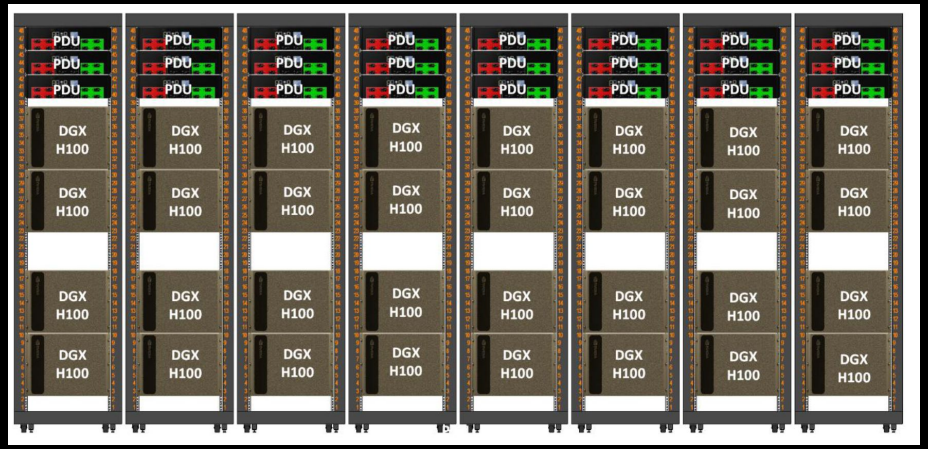
Motivation: Parallel Computing

- Enter parallelization



Motivation: Parallel Computing

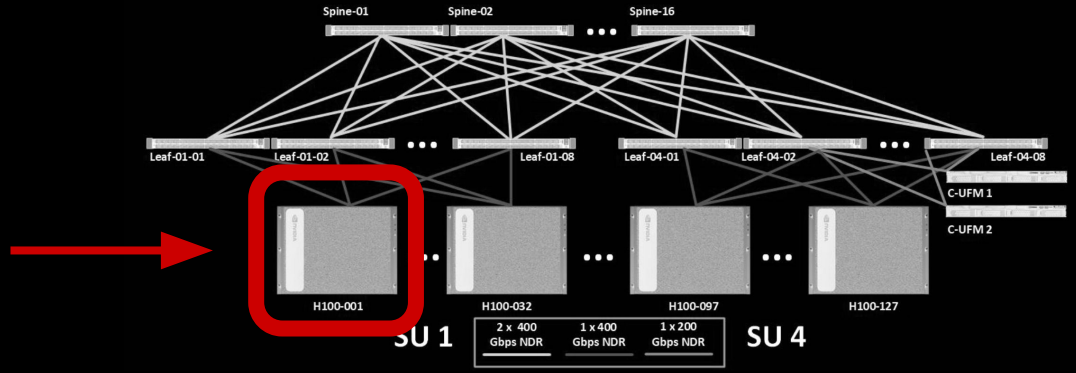
- Multiple multi-GPU servers can be connected together to create a “Super Pod”



Nvidia DGX Super Pod

Motivation: Parallel Computing

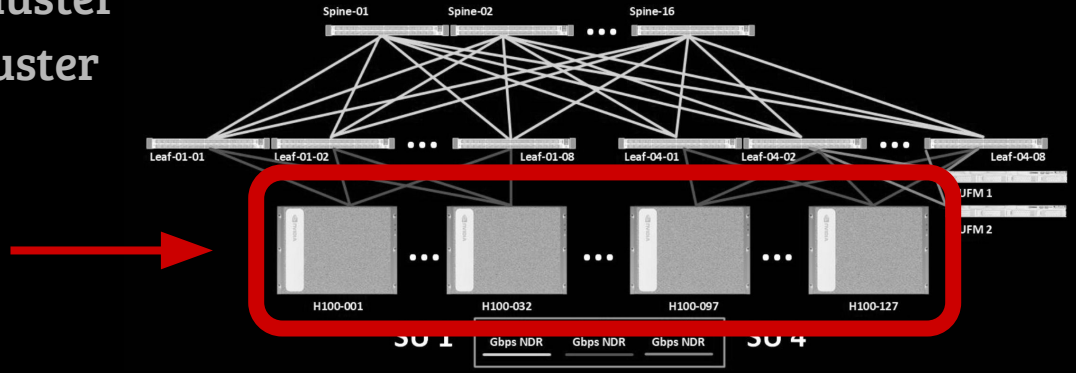
- Multiple multi-GPU servers can be connected together to create a “Super Pod”
- Each DGX server provides 8 GPUs for parallelization



Nvidia DGX Super Pod

Motivation: Parallel Computing

- Multiple multi-GPU servers can be connected together to create a “Super Pod”
- Each DGX server provides 8 GPUs for parallelization
- Interconnecting multiple servers allows for scaling to a large cluster
- e.g, GPT-4 was trained on a cluster of ~25000 GPUs



Nvidia DGX Super Pod

Motivation: Parallel Computing

- GPUs ***communicate*** during the forward and backward pass of training
  - e.g., GPUs compute local gradients and perform AllReduce operation
  - GPUs in a cluster exchange gradients in order to aggregate results

Load Balancing Problem under Link Failures

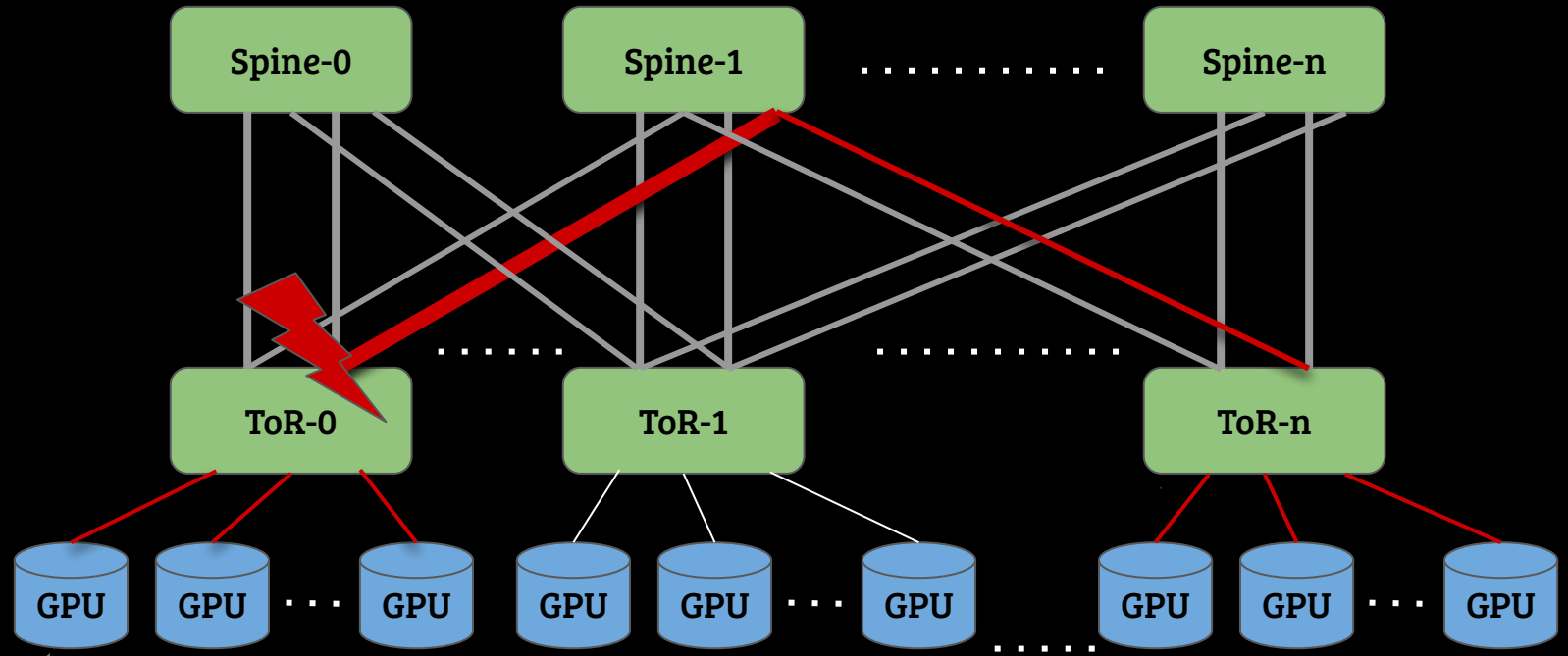
- Granularity
  - Per-packet
  - Per-flow
  - Sub-flow
- Path selection
  - Static
  - Adaptive
- Rerouting and fast notification
  - ECN
  - Timeout
  - Explicit failure notifications
  - …

Load Balancing Problem under Link Failures

- ~100 milliseconds just to identify link failure *locally*
  - A network switch continues to send packets to the failed port during the 100ms
  - At 400 Gbps, this leads to 5 GB dropped packets

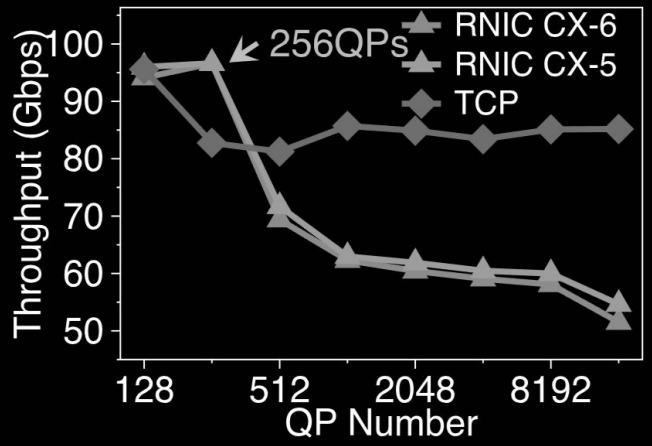
ECMP: Load Imbalance due to Low Entropy

- Hash collisions lead to load imbalance and congestion



MP-RDMA: Entropy at the cost of NIC Resources

- Split every flow k times
- Increased entropy to ECMP hash
- Better load balancing but at the cost of increased number of QPs at the NIC



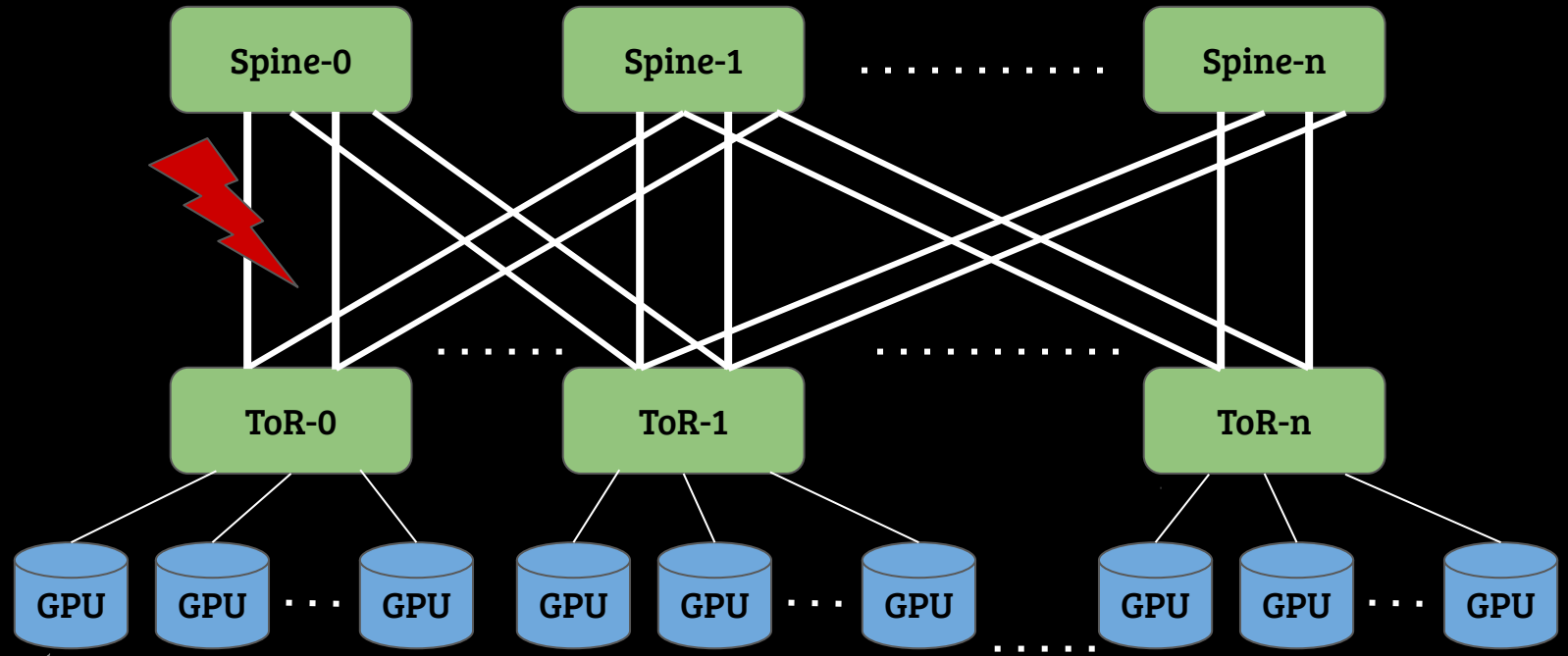
[3] SRNIC: A Scalable Architecture for RDMA NICs

Packet Spraying: Load Imbalance under Failures

- Optimal load balancing in a CLOS-based topology
- Suboptimal under asymmetry

Packet Spraying: Load Imbalance under Failures

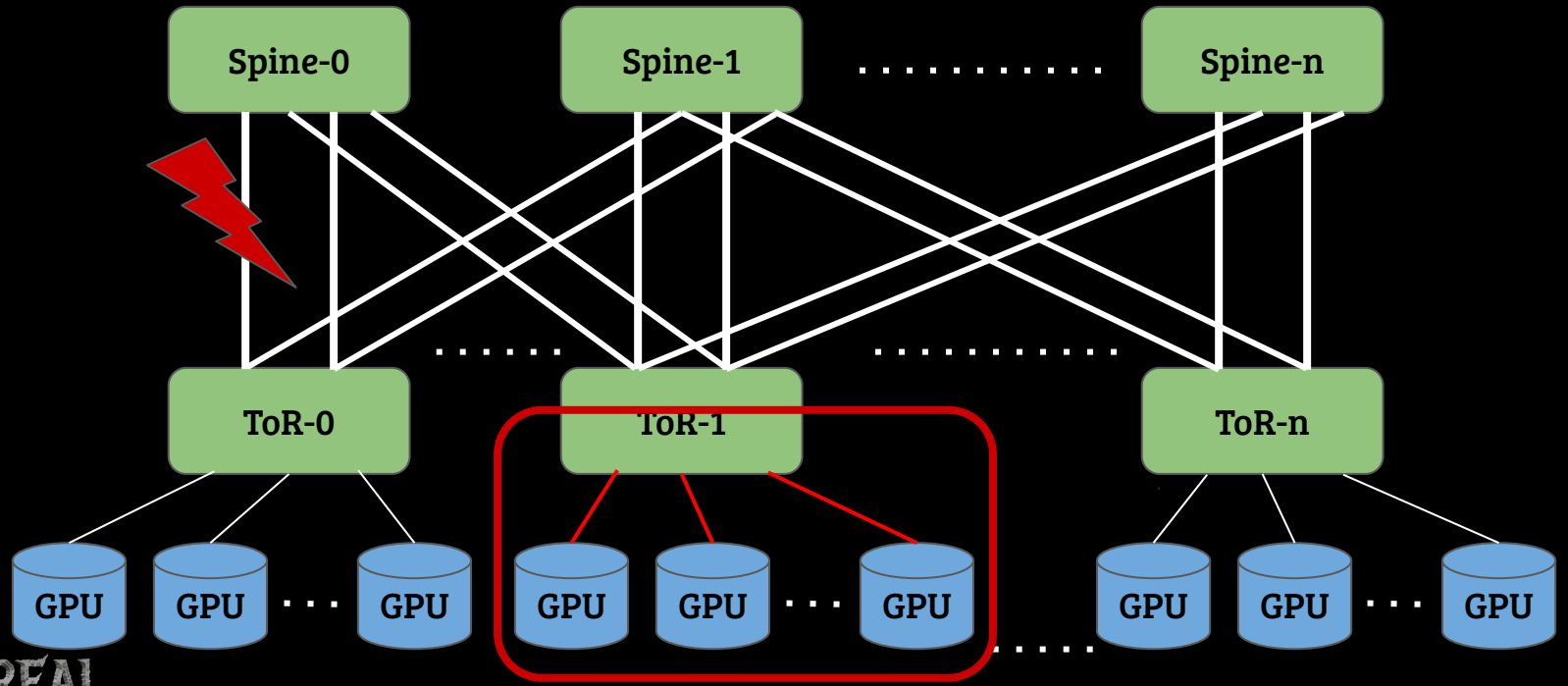
- Example: One link incident to ToR-0 fails



ETHEREAL

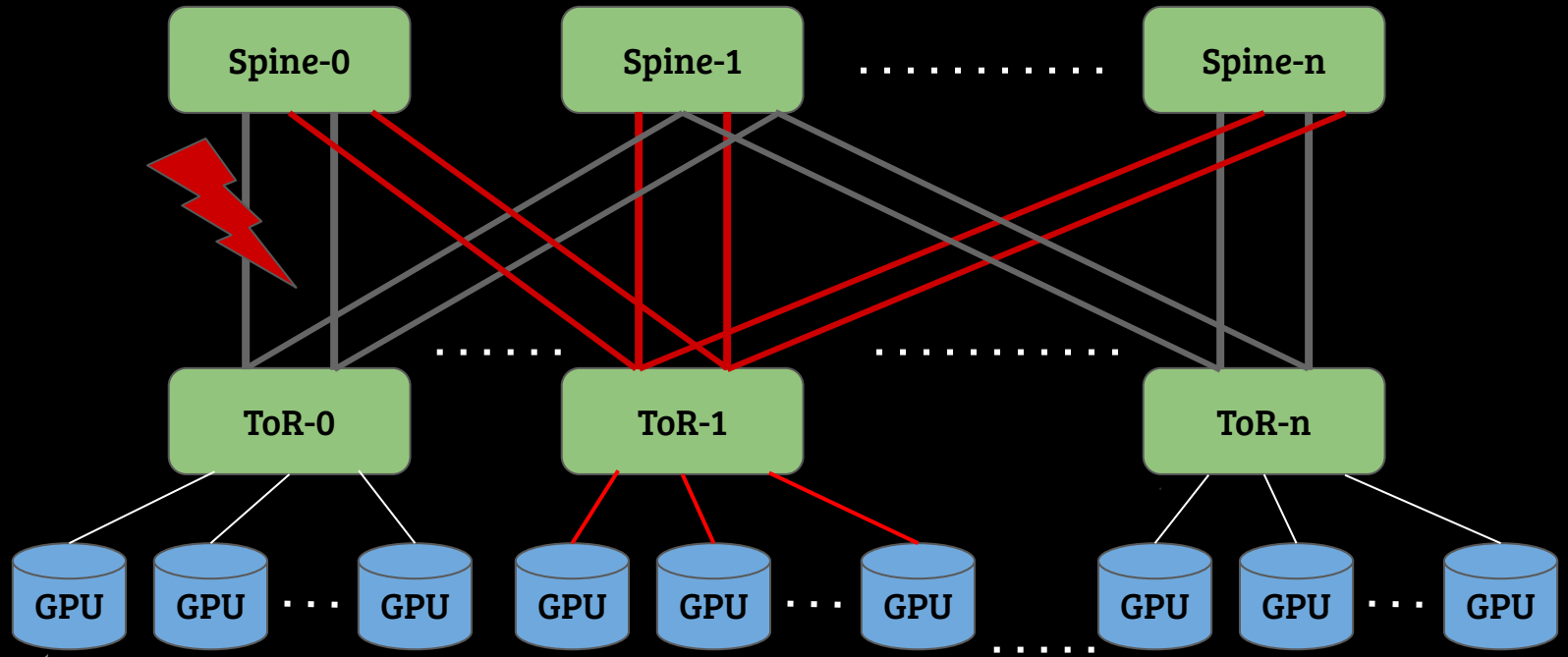
Packet Spraying: Load Imbalance under Failures

- Example: All-to-All Traffic from Tor-1 to ToR-0



Packet Spraying: Load Imbalance under Failures

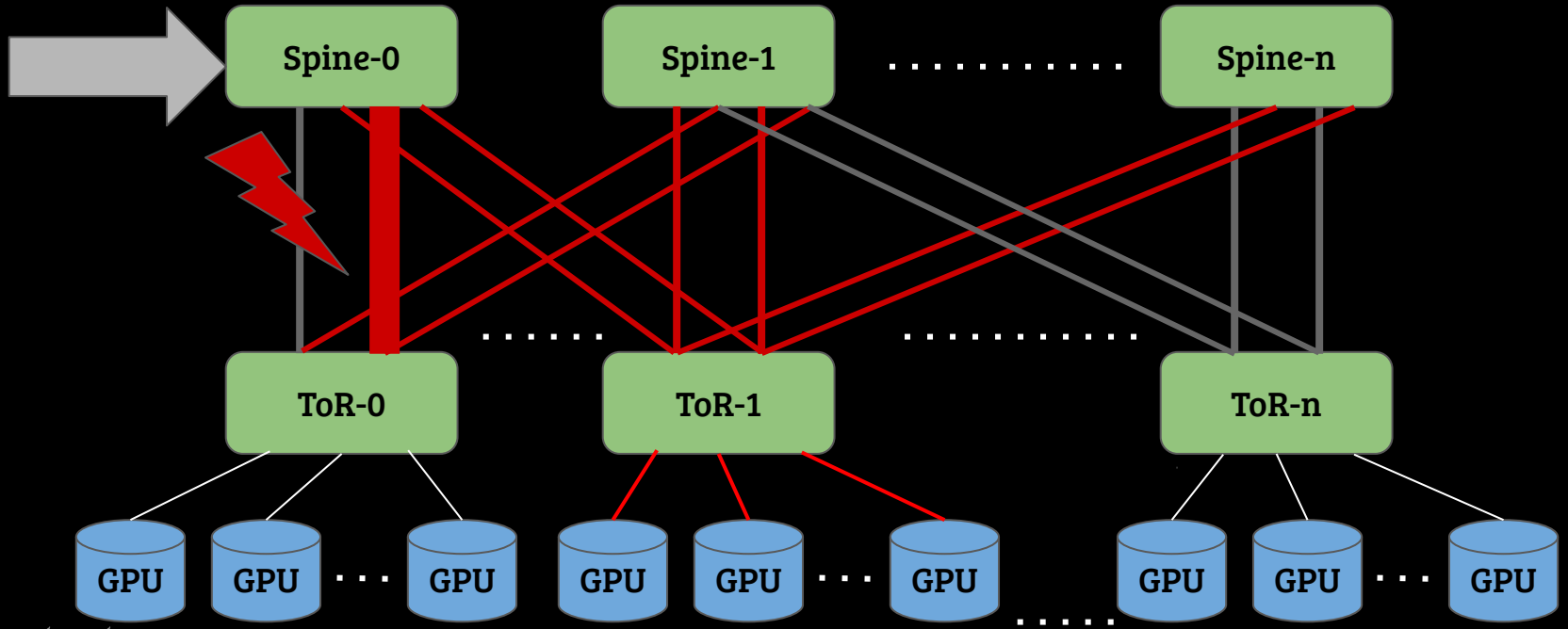
- Example: All-to-All Traffic from Tor-1 to ToR-0



ETHEREAL

Packet Spraying: Load Imbalance under Failures

- Congestion at Spine-0



Packet Spraying: Load Imbalance under Failures

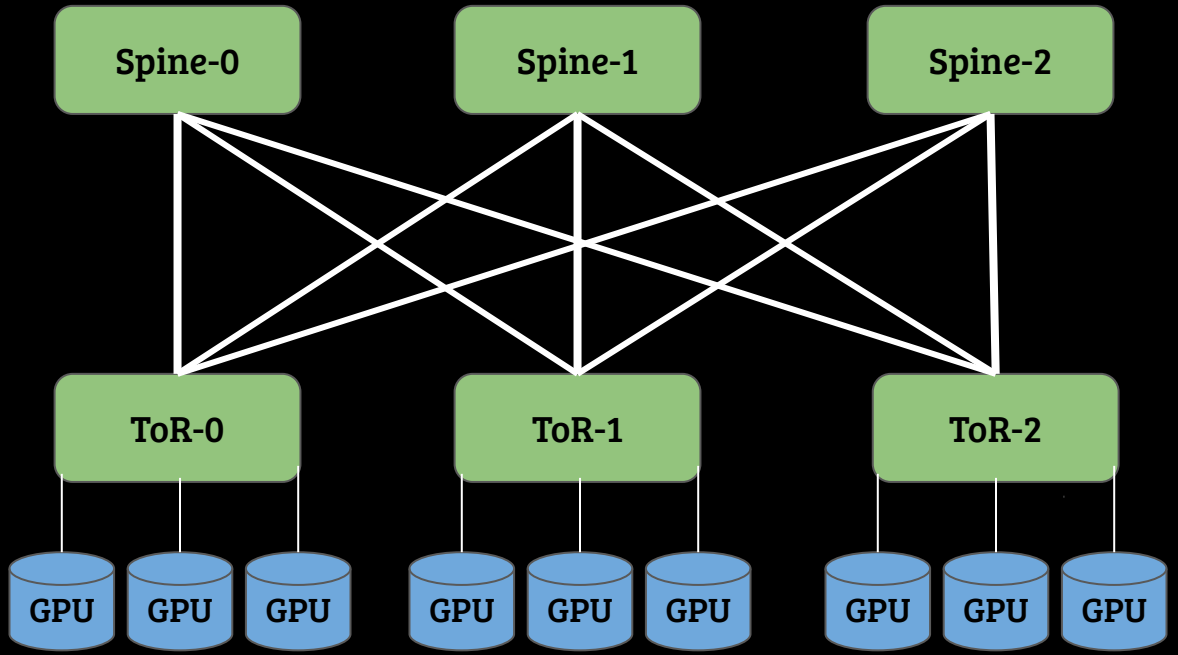
- Optimal load balancing in a CLOS-based topology
  - Novel transport and hardware requirements
  - Handling out-of-order packet delivery
  - Loss recovery mechanisms
  - ...
  - see Ultra Ethernet Consortium whitepaper
- Suboptimal under asymmetry
  - Congestion under failures
  - *Non-trivial to react to failures even with perfect failure notifications*

REPS: Path Flapping prevents Convergence

- A novel load balancing algorithm being discussed in industry and UEC
- Explore and Cache good paths
- Change path (reroute) upon receiving ECN
- Can lead to path flapping

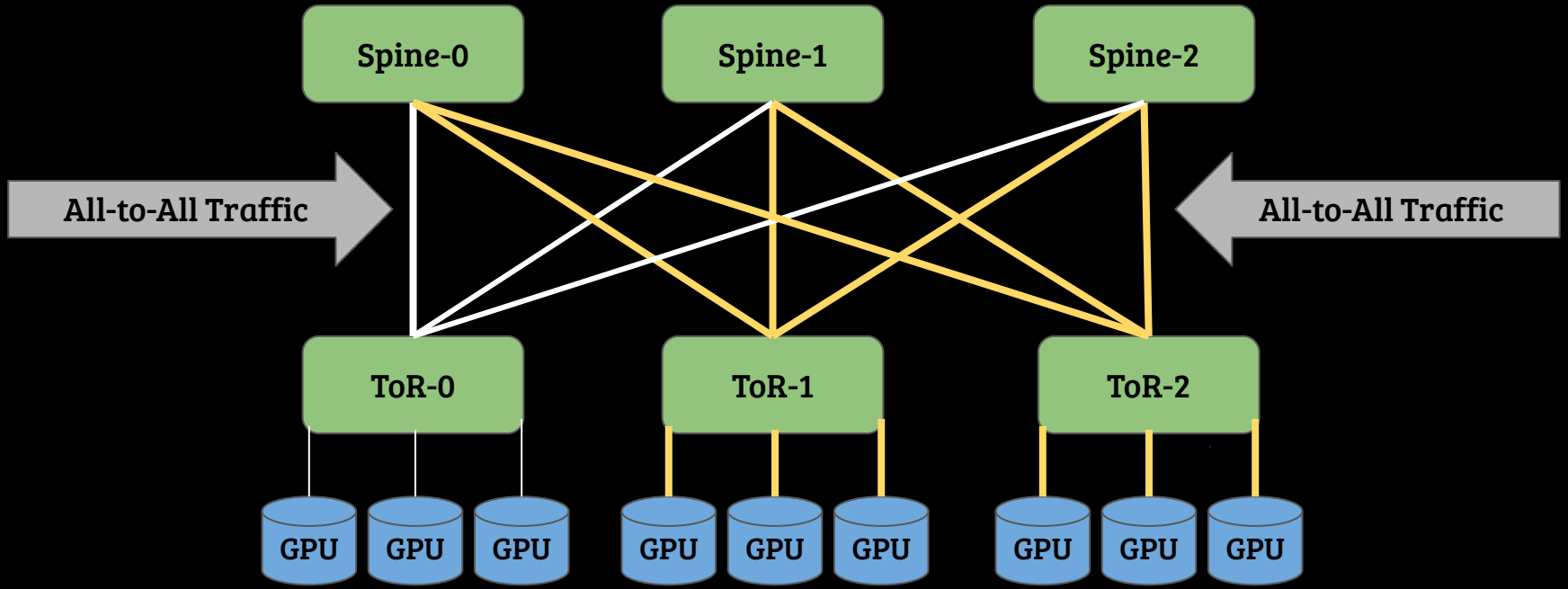
REPS: Path Flapping prevents Convergence

Example: All-to-All across GPU 3 to 8



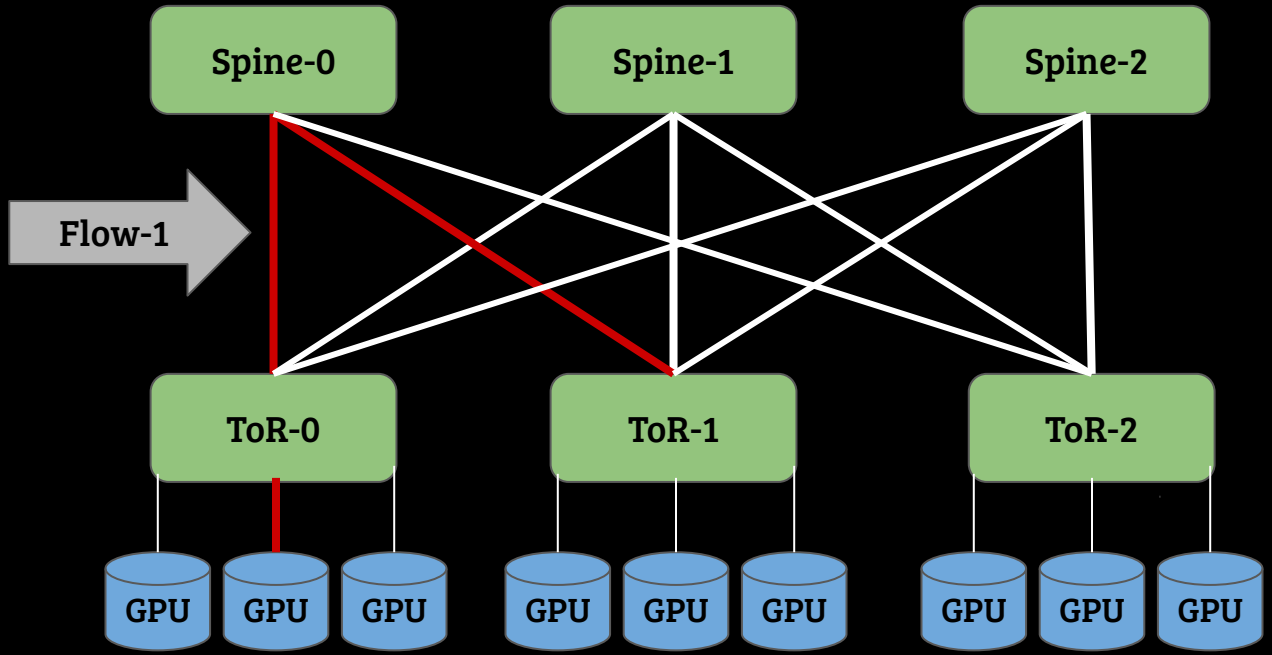
REPS: Path Flapping prevents Convergence

Example: All-to-All across GPU 3 to 8



REPS: Path Flapping prevents Convergence

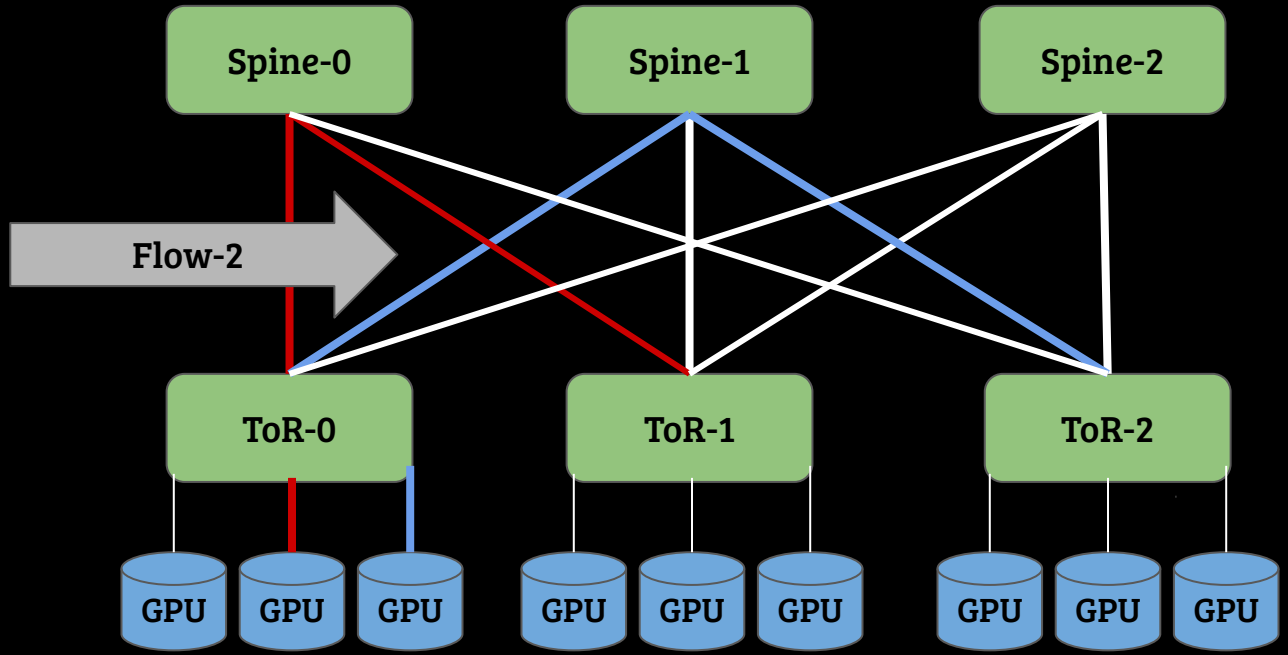
Example: All-to-All across GPU 3 to 8 & consider 4 flows and 3 paths



ETHEREAL

REPS: Path Flapping prevents Convergence

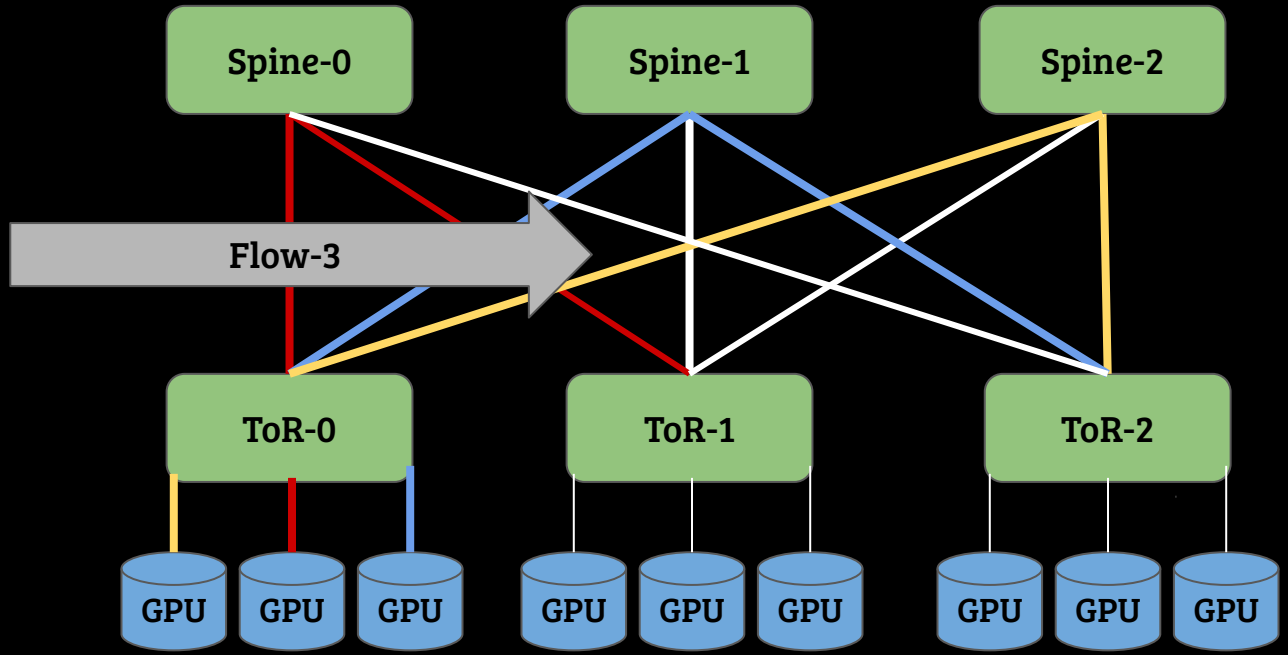
Example: All-to-All across GPU 3 to 8 & consider 4 flows and 3 paths



ETHEREAL

REPS: Path Flapping prevents Convergence

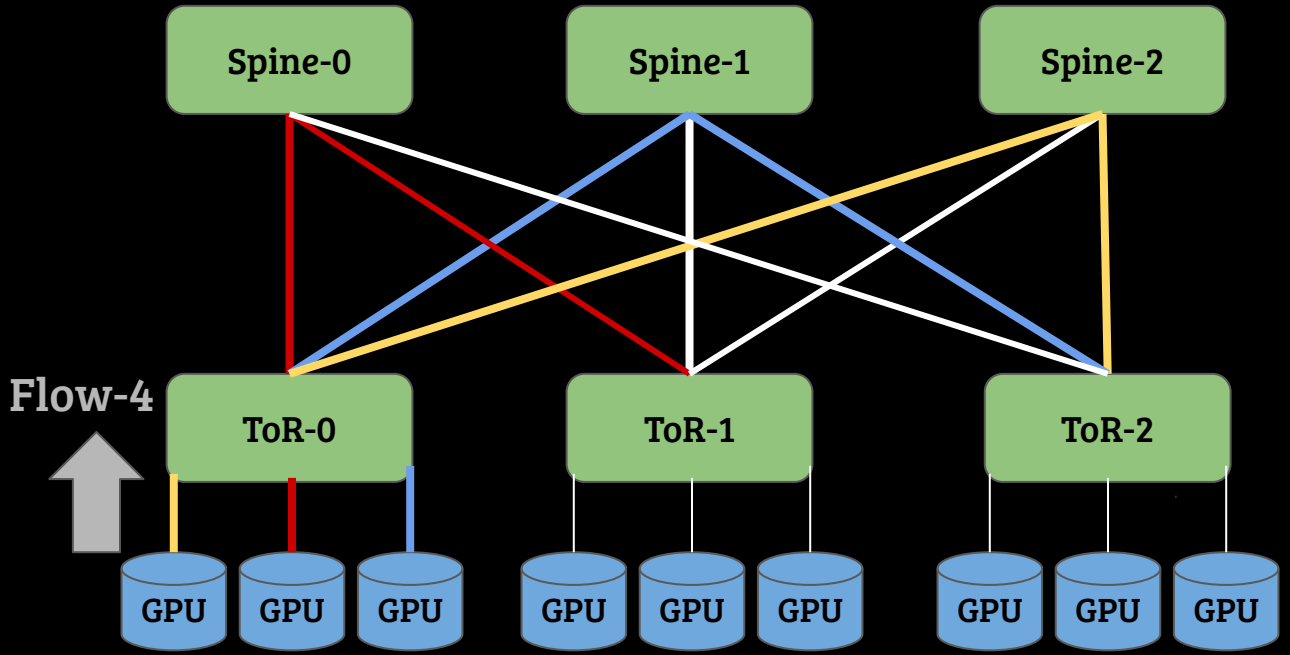
Example: All-to-All across GPU 3 to 8 & consider 4 flows and 3 paths



ETHEREAL

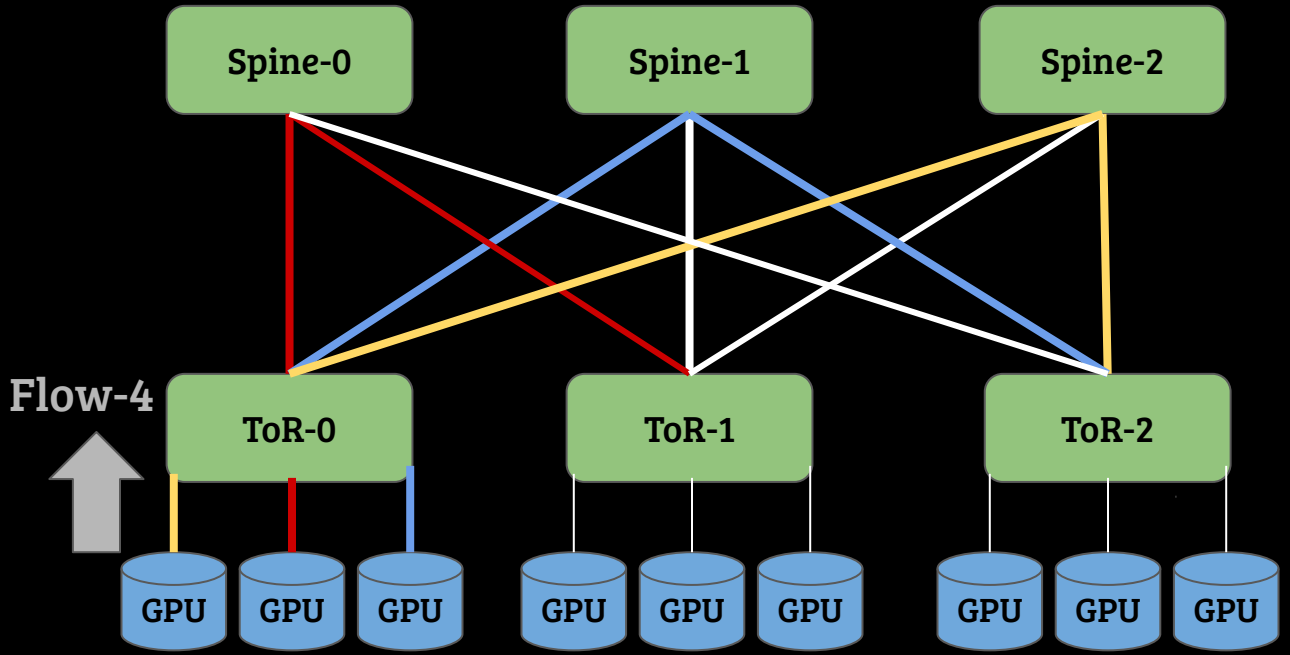
REPS: Path Flapping prevents Convergence

Example: All-to-All across GPU 3 to 8 & consider 4 flows and 3 paths



REPS: Path Flapping prevents Convergence

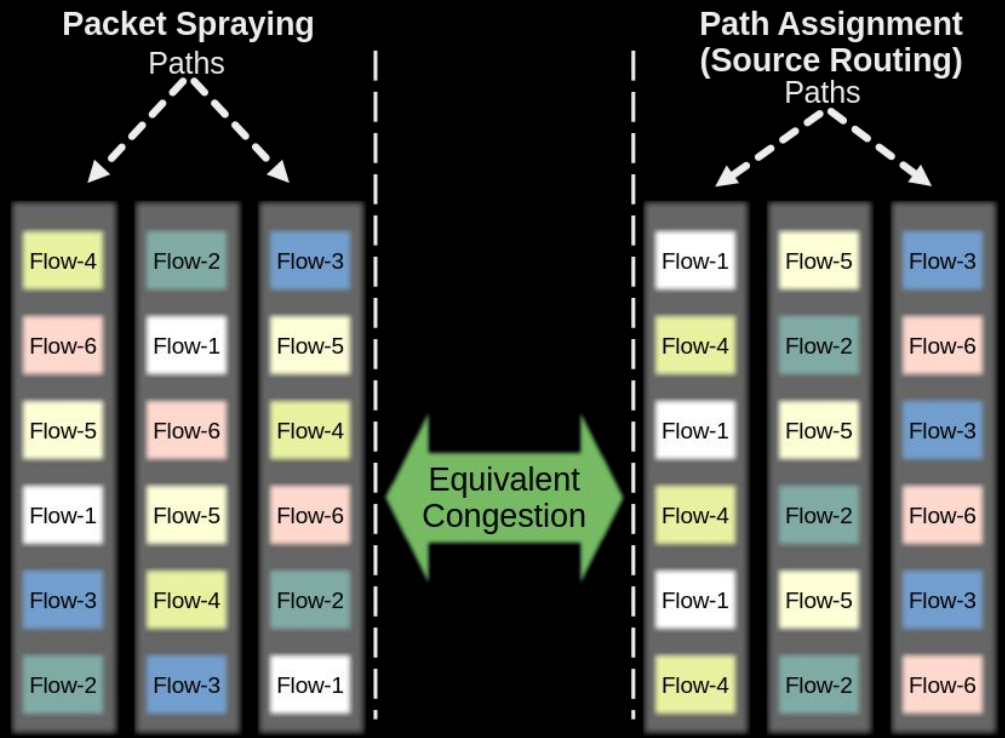
The 4th flow causes congestion (ECN) and triggers reroute



Summary and Design Goals

- Path context helps in identifying failures
- Number of flows fundamentally limits singlepath load balancing
- Goals:
  - Uniform load balancing, like Packet Spraying
  - Identify failed paths quickly, like REPS

Ethereal: Optimal Load Balancing w/wo Link Failures



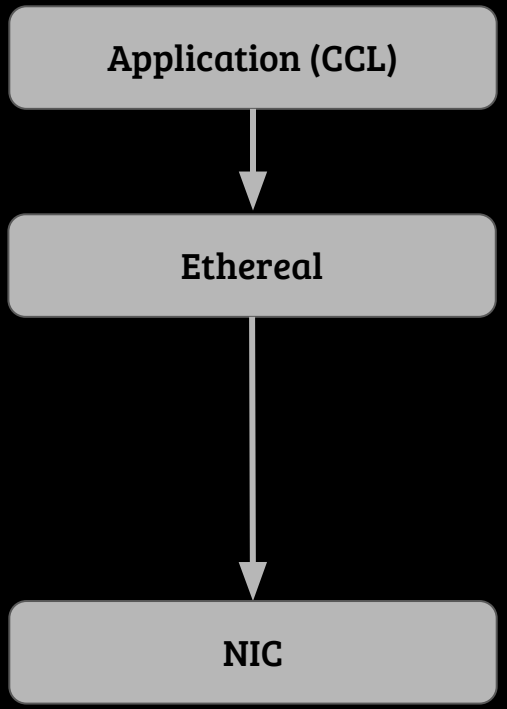
Ethereal: Optimal Load Balancing w/wo Link Failures

- Load balancing at the application layer (Communication Library)
- Source routing for path selection
- Minimal flow splitting
  - Key enabler: Flow size is known upon arrival, unlike traditional storage/search workloads
  - Prevents path flapping
  - Achieves optimal load balancing
- Flow timeout notification from NIC to application
- Full details: https://arxiv.org/pdf/2407.00550

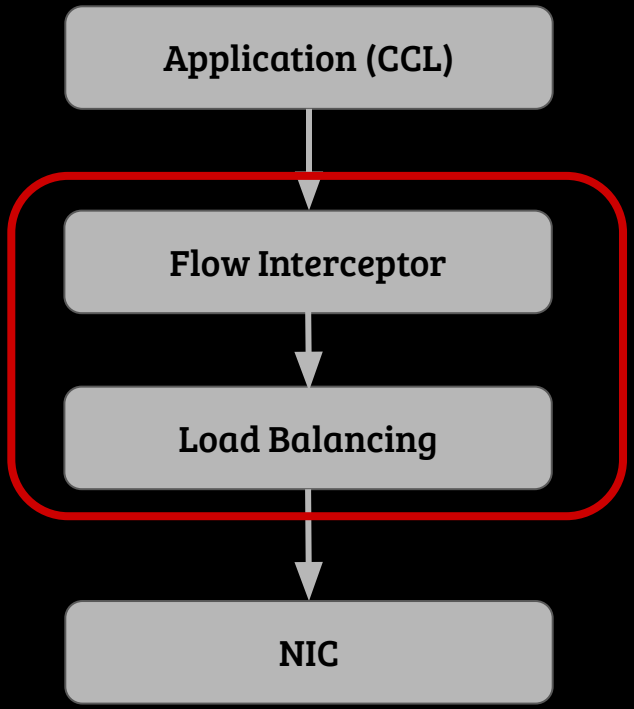
Ethereal: Optimal Load Balancing w/wo Link Failures

- Load balancing at the application layer (Communication Library)
  - Simple
  - Does not require new hardware
  - Challenge: RDMA NIC (hardware) handles all the network stack and flow transmission
    - How can link failures be notified to the application?

Ethereal: Optimal Load Balancing w/wo Link Failures

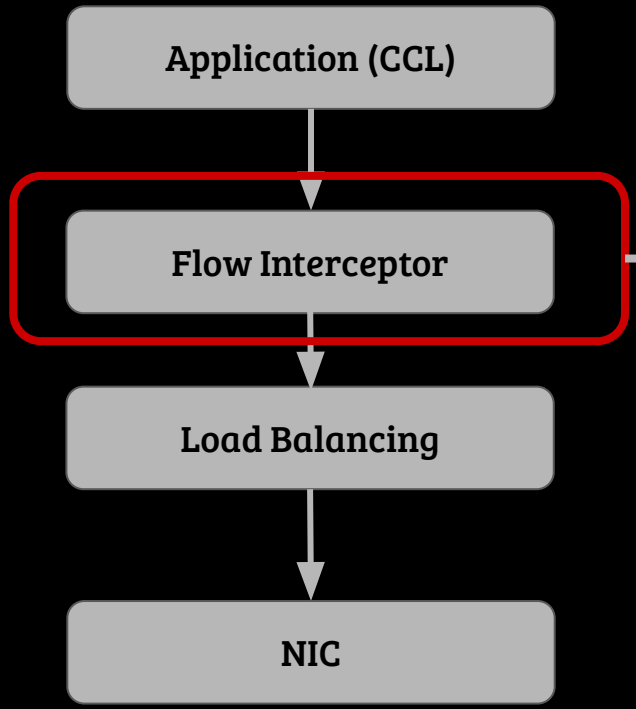


Ethereal: Optimal Load Balancing w/wo Link Failures



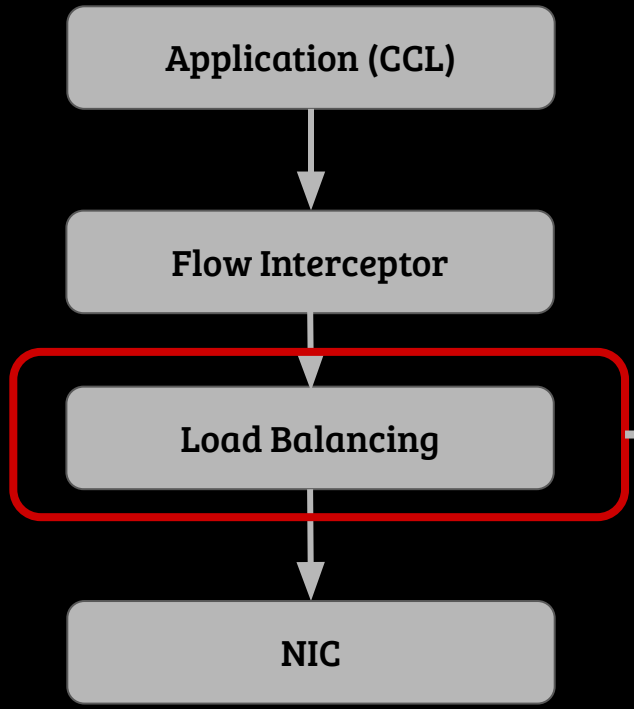
ETHEREAL

Ethereal: Optimal Load Balancing w/wo Link Failures



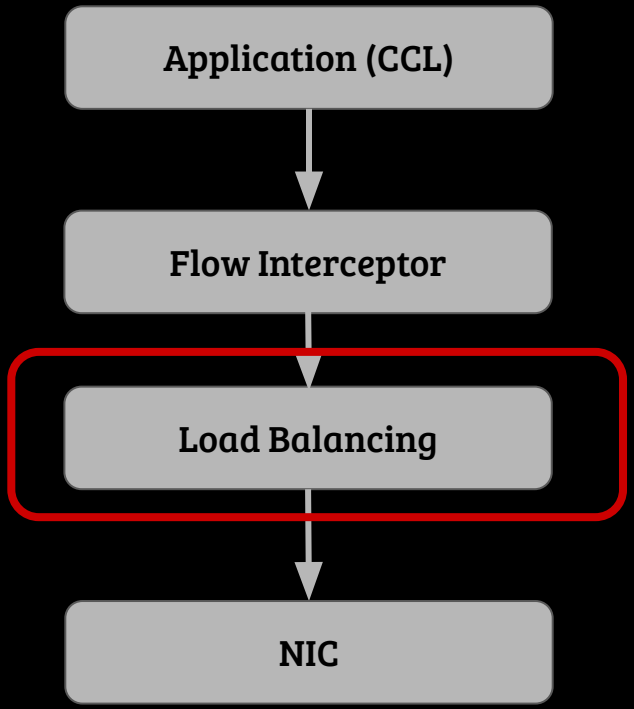
Flows within *each step* of a collective are likely to arrive in a batch

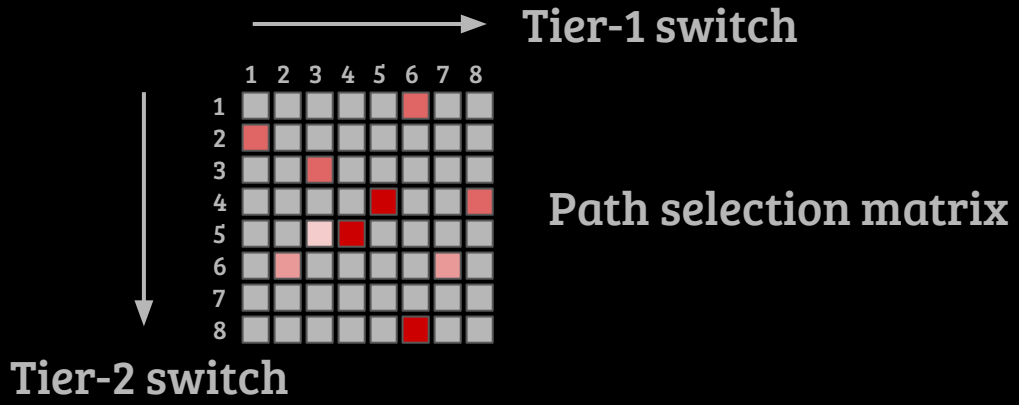
Ethereal: Optimal Load Balancing w/wo Link Failures



- Flow sizes are known
- Flow sizes are equal within a batch

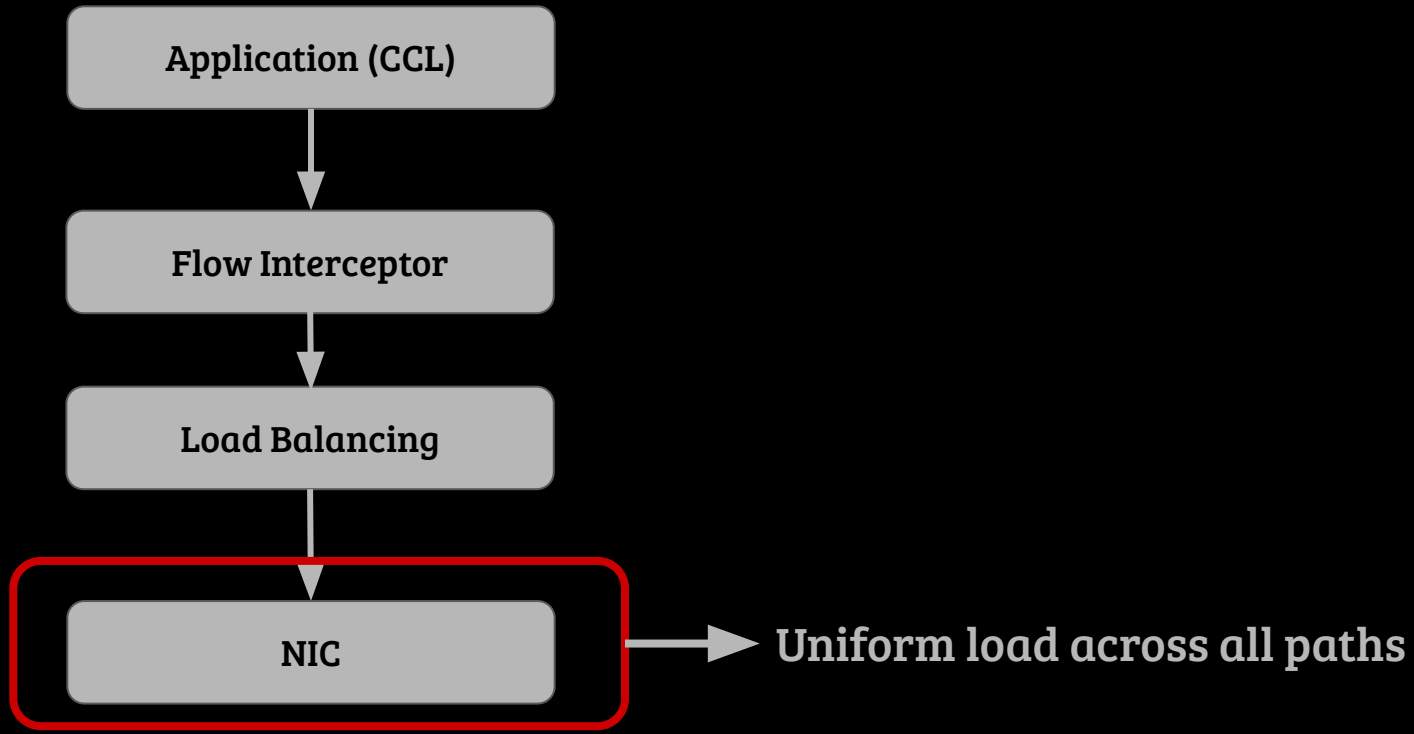
Ethereal: Optimal Load Balancing w/wo Link Failures





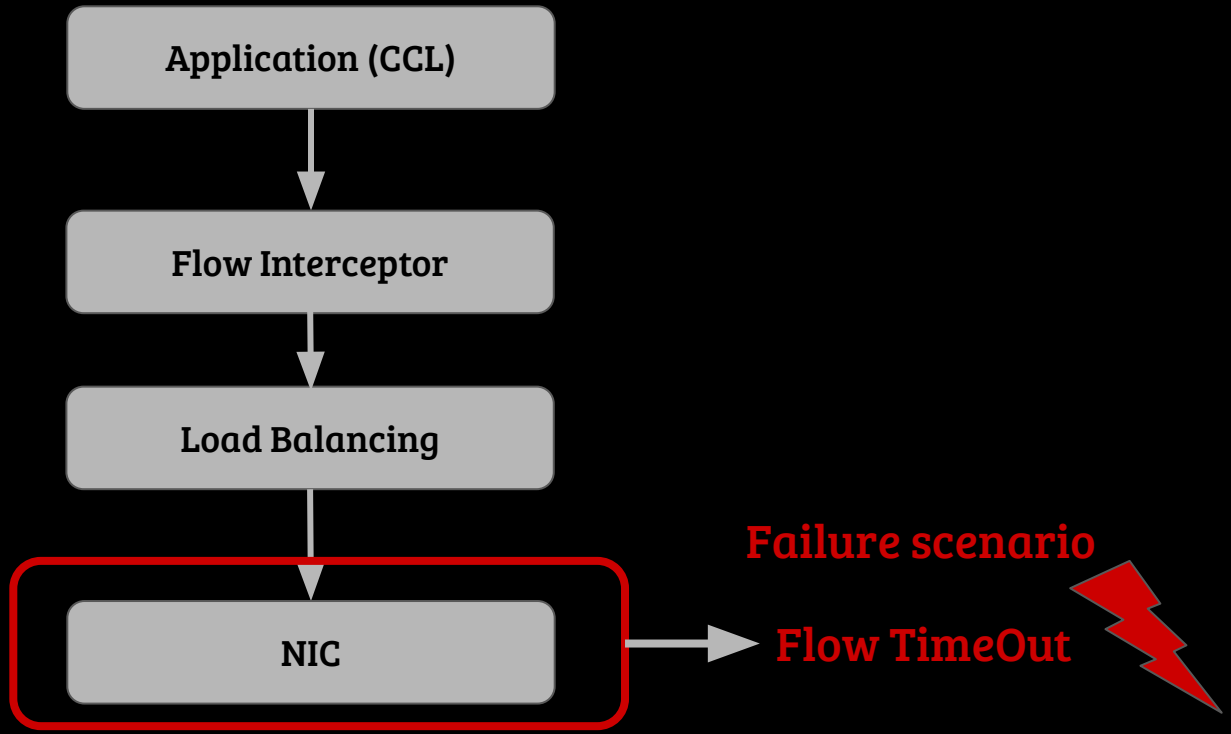
ETHEREAL

Ethereal: Optimal Load Balancing w/wo Link Failures

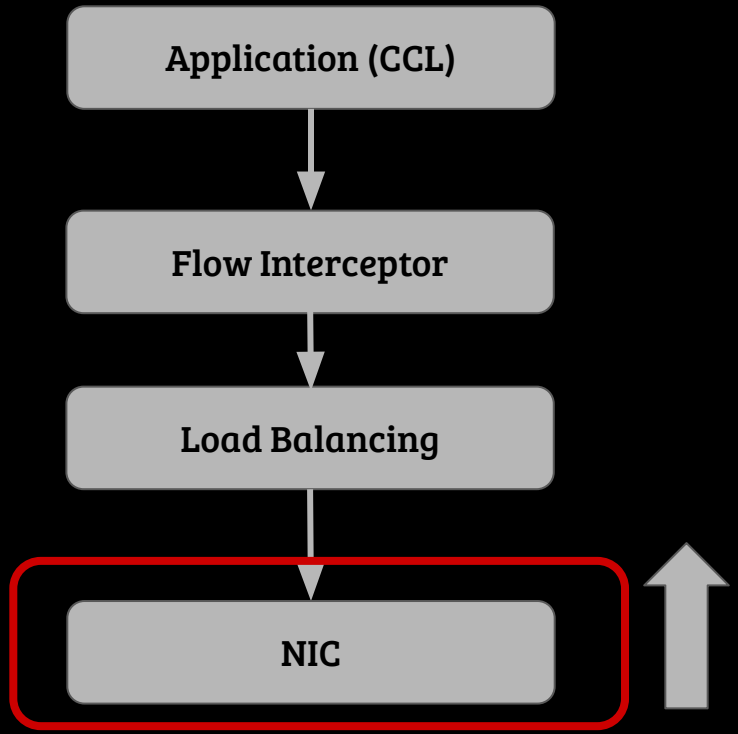


ETHEREAL

Ethereal: Optimal Load Balancing w/wo Link Failures



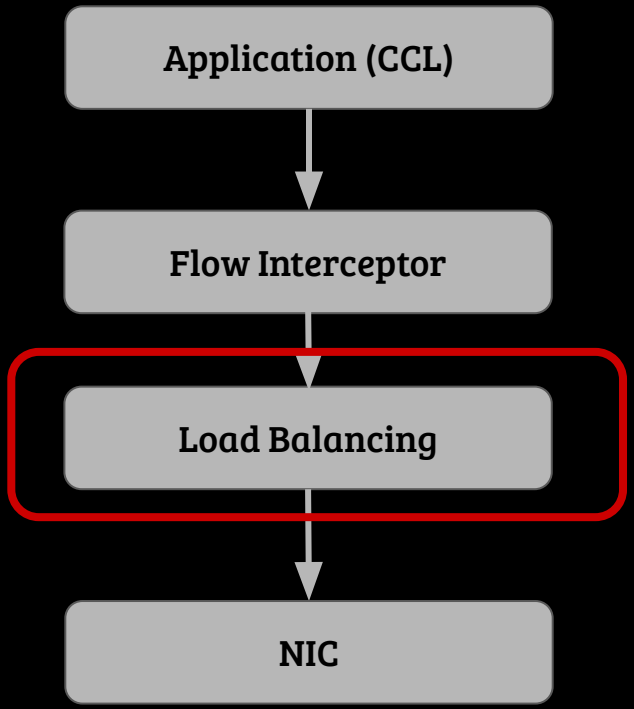
Ethereal: Optimal Load Balancing w/wo Link Failures

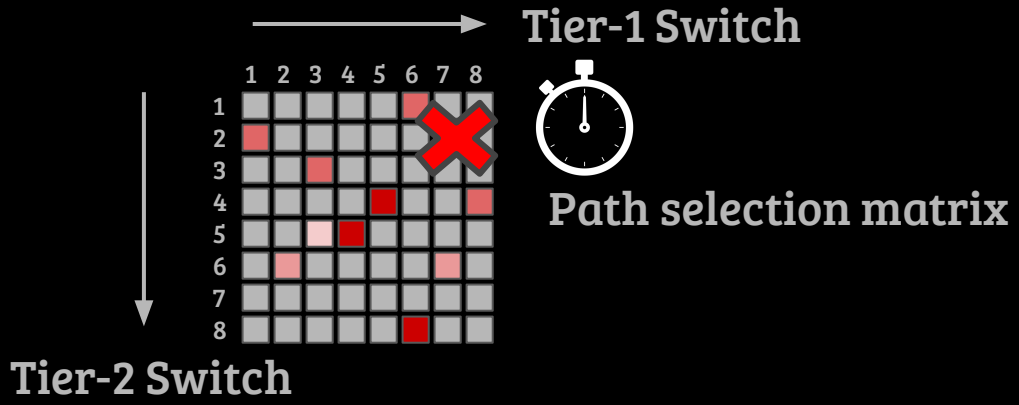


Insert failure notification to Completion Queue (CQ)

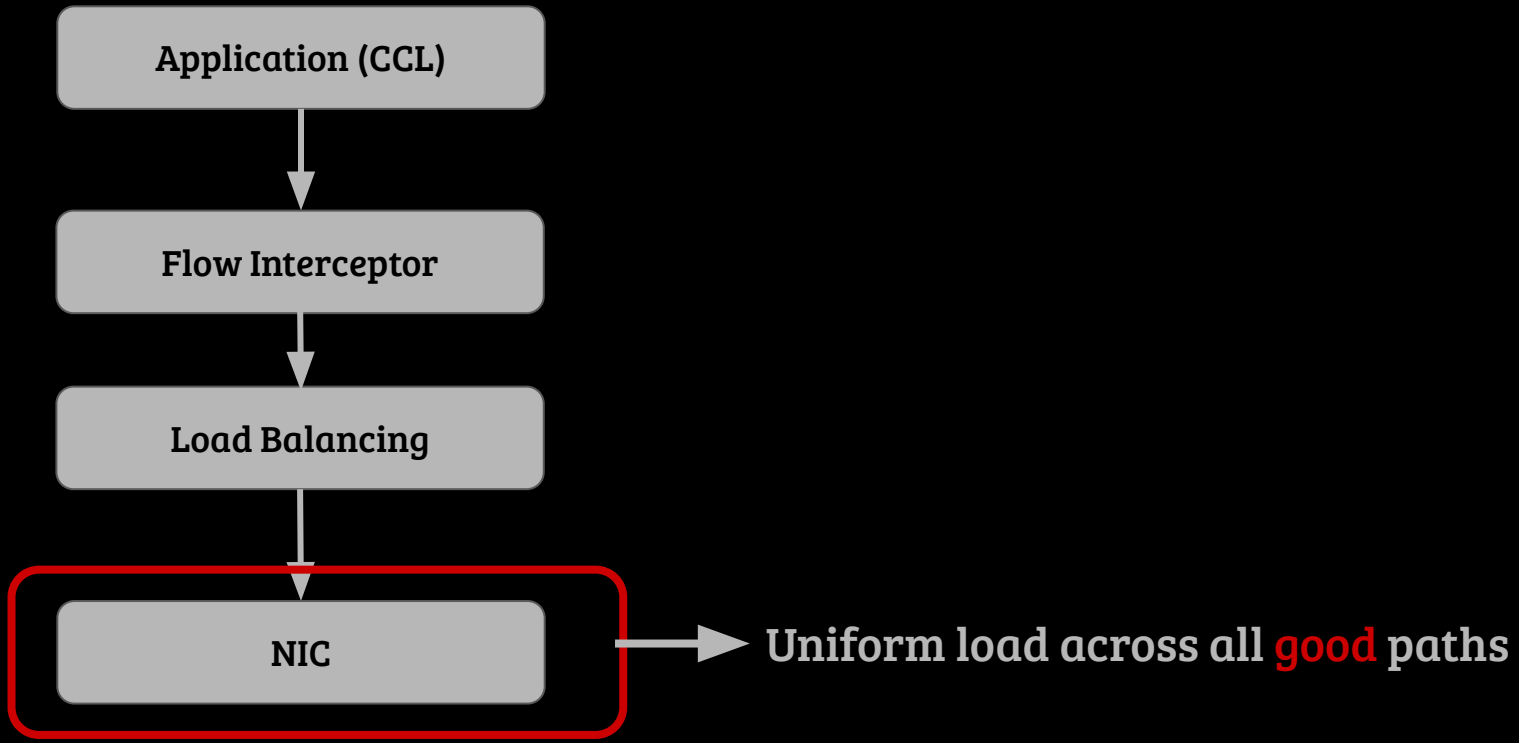
ETHEREAL

Ethereal: Optimal Load Balancing w/wo Link Failures





Ethereal: Optimal Load Balancing w/wo Link Failures



ETHEREAL

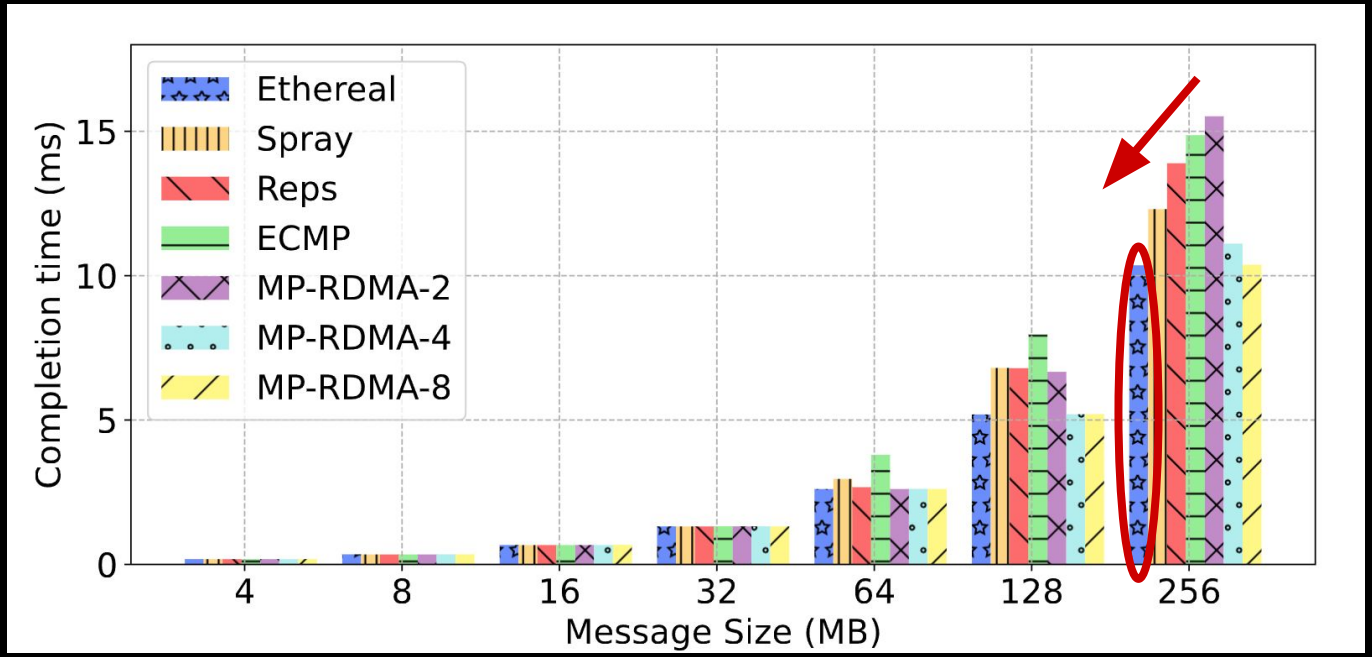
Failure Notifications

- NIC to Application notification
  - Currently relying on flow timeouts
  - Speed vs accuracy tradeoff
- Network to NIC notification: Open for discussion!
  - Source Quench?
  - Transceivers and drivers require fundamental changes to identify local failures rapidly!

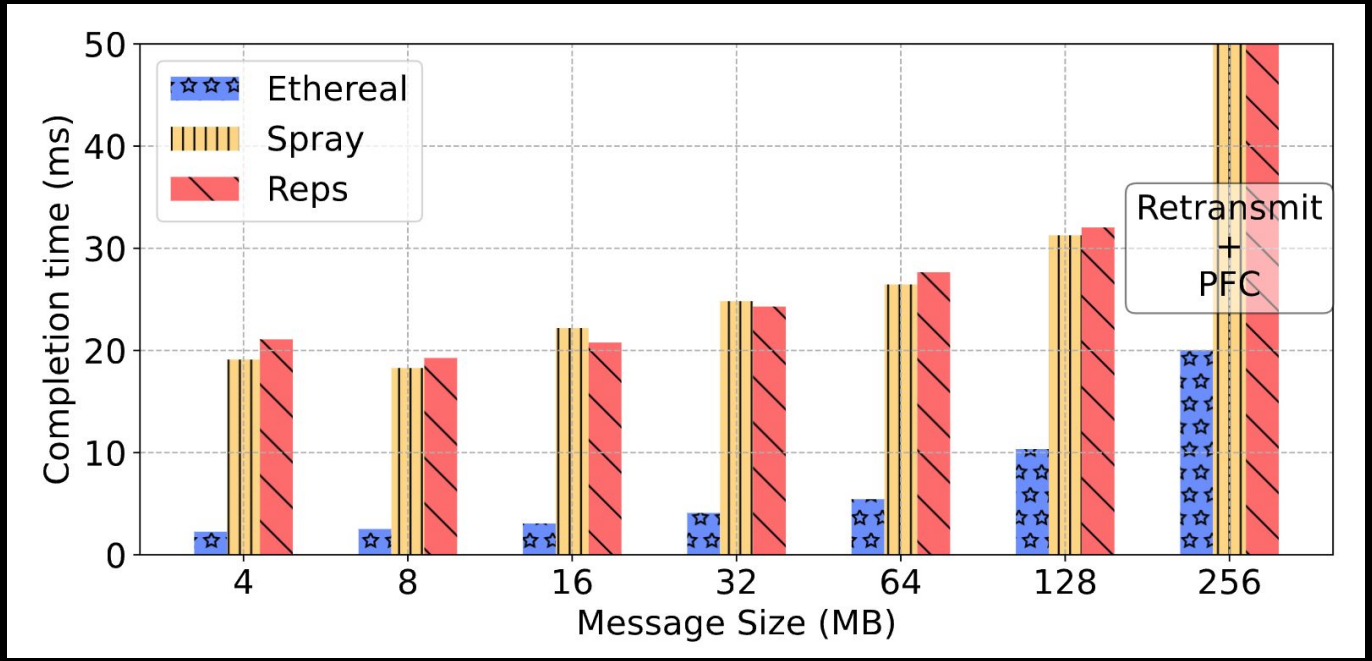
Evaluation

- Simulations: Astrasim & MLCommons Chakra
- Fat tree topology
  - 512 GPUs
  - 32 ToR
  - 32 Aggregation
  - 16 Core
- 400 Gbps links
- 64MB switch buffer size
- Timeout: 1 ms

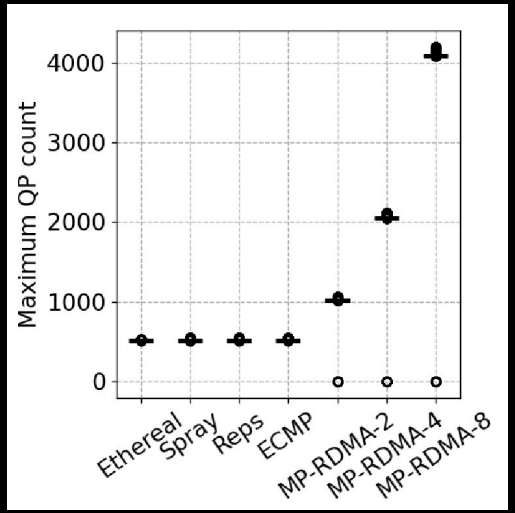
Ethereal Improves Collective Completion Times



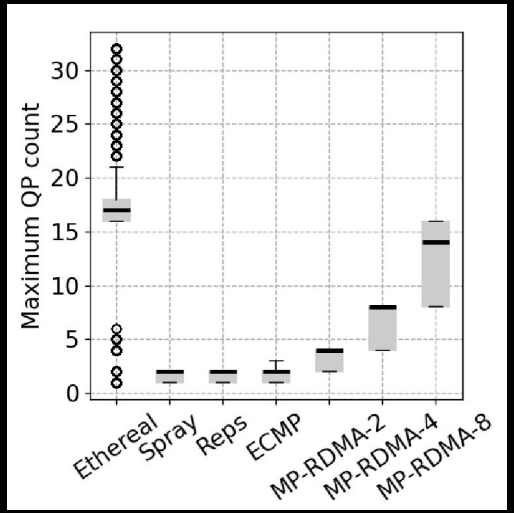
Ethereal Reacts to Failures Promptly



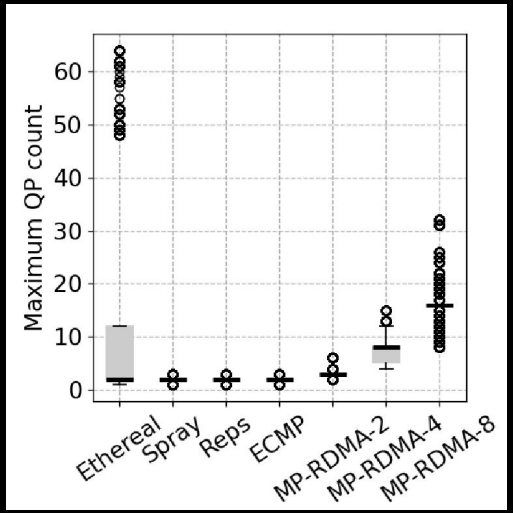
Ethereal Does not Significantly Increase QP requirements



All-to-All



Recursive Doubling



Ring

Conclusion

- Uniform load balancing in CLOS topologies does not require significant hardware changes to NIC
- Ethereal achieves optimal load balancing using singlepath transport
  - Source routing
  - Minimal flow splitting
- Failure notifications from the network can further improve performance
- Source code available online
  - On request currently, please send an email or ping me on LinkedIn

Thank You

Vamsi Addanki
vamsiaddanki.net
vamsi@inet.tu-berlin.de
@Vamsi_DT