ASPLOS 2025

Artifacts Available acm V1.1

Artifacts Evaluated acm Functional V1.1

Results Reproduced acm V1.1

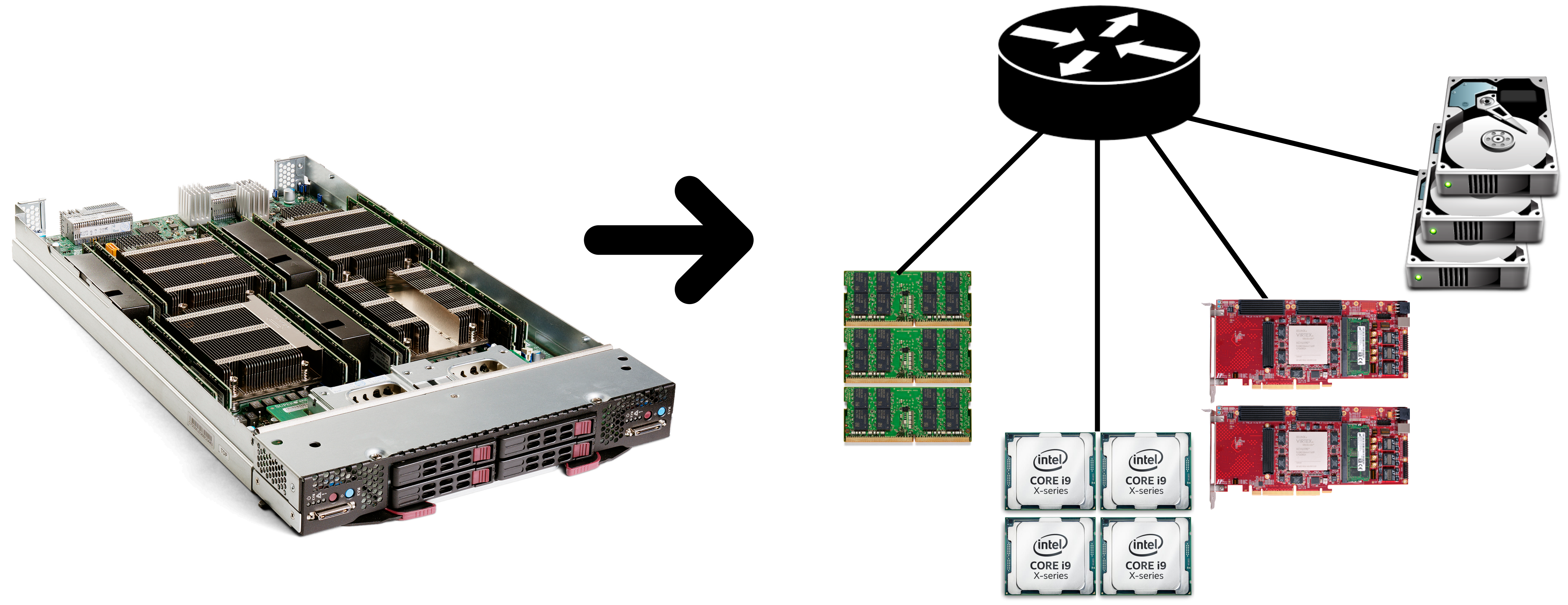
EDM: An Ultra-Low Latency Ethernet Fabric for Memory Disaggregation

Weigao Su, Vishal Shrivastav

PURDUE UNIVERSITY®

Why memory disaggregation?

- The need for memory is surging
- Constraints of individual servers
- Fine-grained pooling, elastic scaling



Why is Ethernet promising?

- Dominant datacenter network fabric
- High bandwidth (Terabit Ethernet link)
- Low management cost, distance scaling…

OSDI ’18 — **LegoOS** OS for disaggregated rack

NSDI ’19 — **Shoal** Optical switching for disaggregation

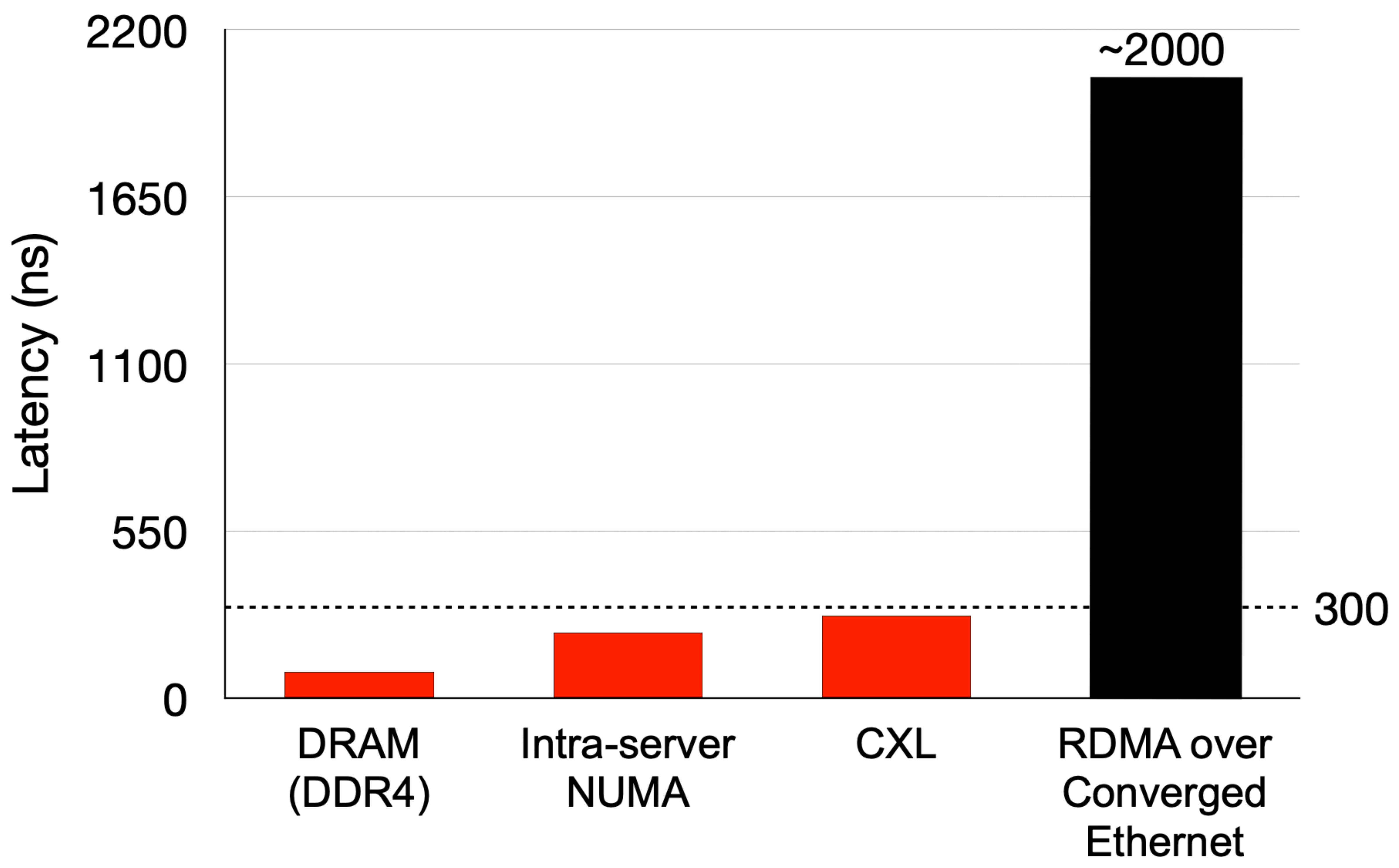
OSDI ’20 — **AIFM** Application-level far memory integration

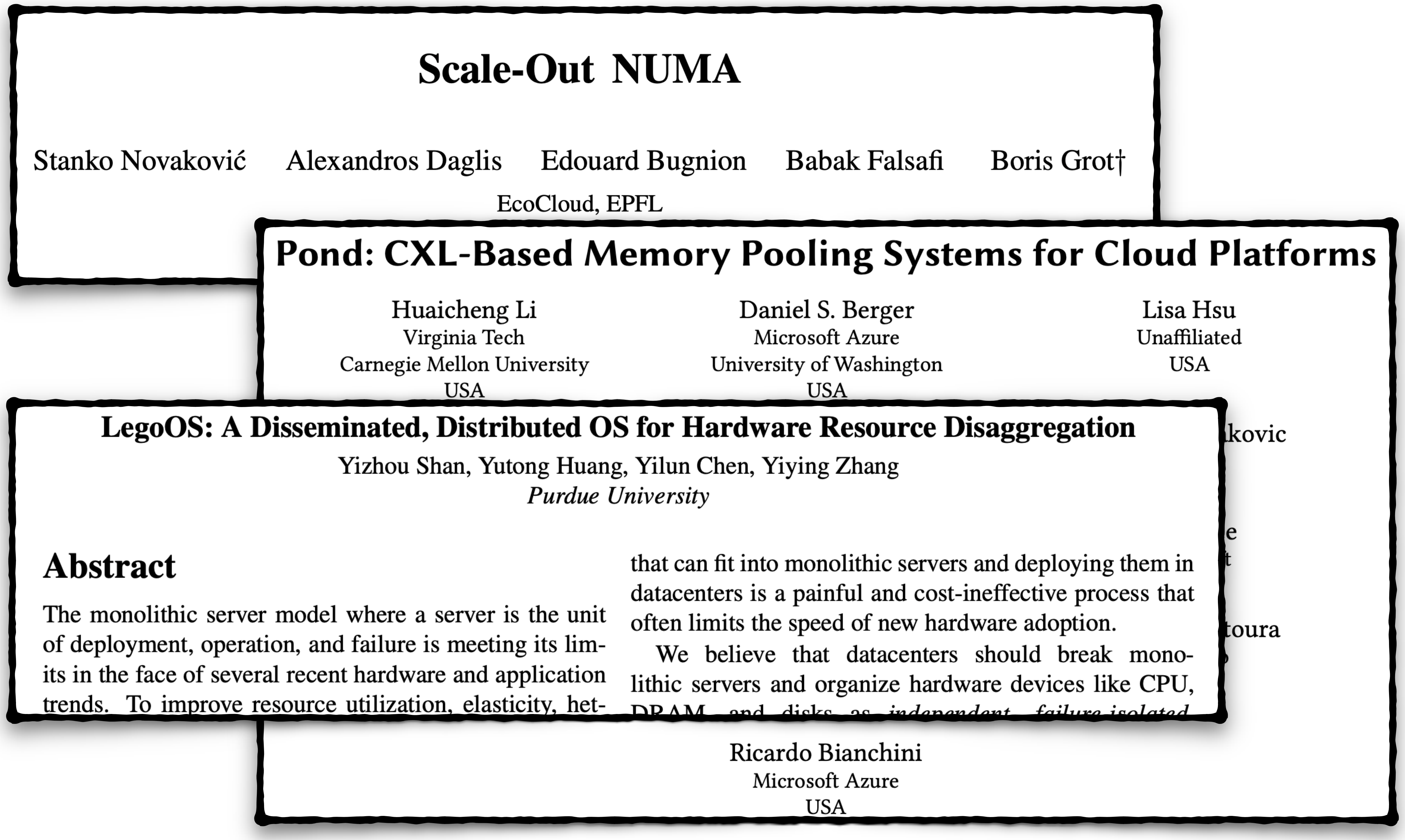
SOSP ’21 — **Mind** In-network memory management

SIGCOMM ’23 — **Cowbird** ToR-offloaded far memory management

However, the latency in Ethernet is prohibitive, prompting proposals of *separate* fabric to carry memory traffic

Custom processor interconnect, PCIe, Infiniband, etc.




Scale-Out NUMA

Stanko Novaković Alexandros Daglis Edouard Bugnion Babak Falsafi Boris Grot†
EcoCloud, EPFL

Pond: CXL-Based Memory Pooling Systems for Cloud Platforms

Huaicheng Li, Virginia Tech, Carnegie Mellon University, USA
Daniel S. Berger, Microsoft Azure, University of Washington, USA
Lisa Hsu, Unaffiliated, USA

kovic

toura

Ricardo Bianchini, Microsoft Azure, USA

LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation

Yizhou Shan, Yutong Huang, Yilun Chen, Yiying Zhang
Purdue University

Abstract

The monolithic server model where a server is the unit of deployment, operation, and failure is meeting its limits in the face of several recent hardware and application trends. To improve resource utilization, elasticity, het- that can fit into monolithic servers and deploying them in datacenters is a painful and cost-ineffective process that often limits the speed of new hardware adoption.

We believe that datacenters should break monolithic servers and organize hardware devices like CPU, DRAM, and disks as *independent, failure-isolated*

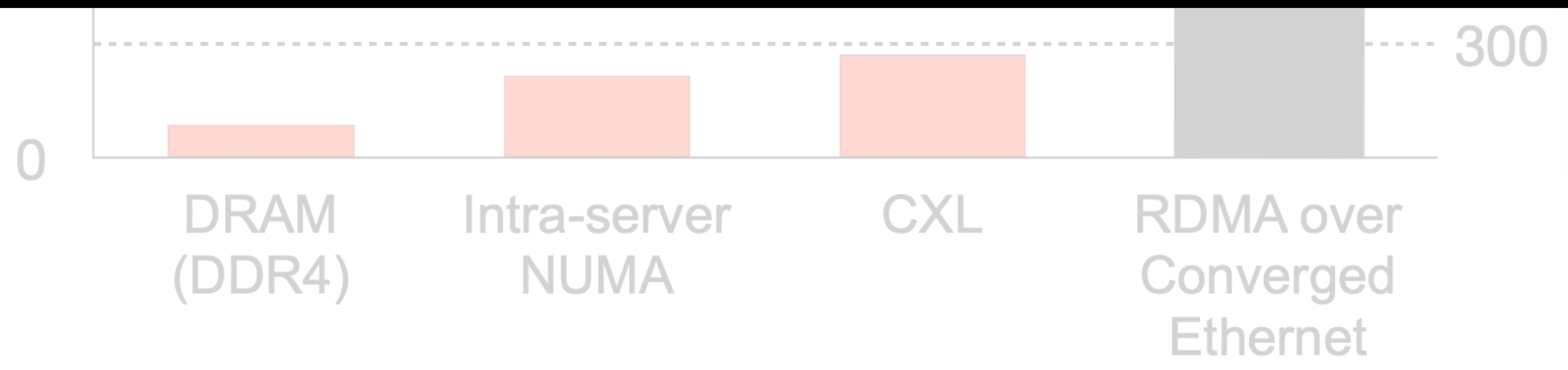
But, separate fabrics for different traffic makes the network **costly** and **harder to manage**

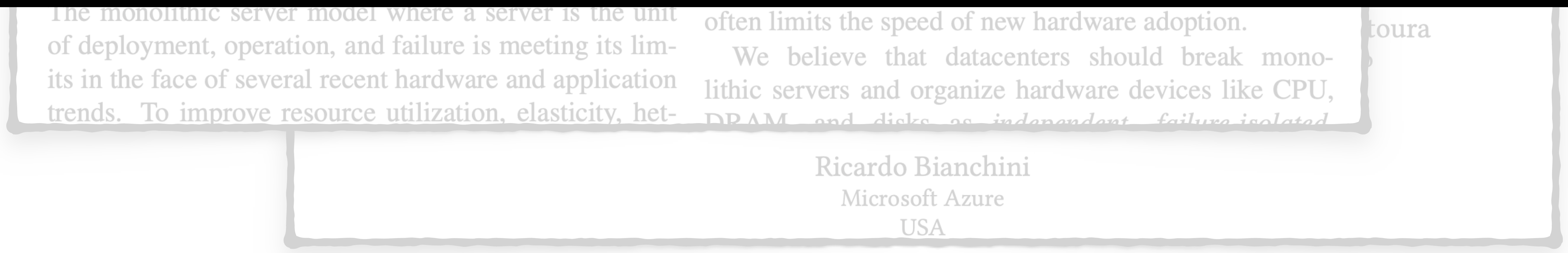
However, the latency in Ethernet is prohibitive, prompting proposals of *separate* fabric to carry memory traffic

Custom processor interconnect, PCIe, Infiniband, etc.

A low latency Ethernet fabric would allow us to have a single unified network fabric to carry all kinds of traffic (memory, storage, IP, …)

… easier to manage, lower cost, statistical bandwidth multiplexing




The monolithic server model where a server is the unit of deployment, operation, and failure is meeting its limits in the face of several recent hardware and application trends. To improve resource utilization, elasticity, het-

often limits the speed of new hardware adoption.

We believe that datacenters should break monolithic servers and organize hardware devices like CPU, DRAM, and disks as *independent, failure-isolated*

Ricardo Bianchini
Microsoft Azure
USA

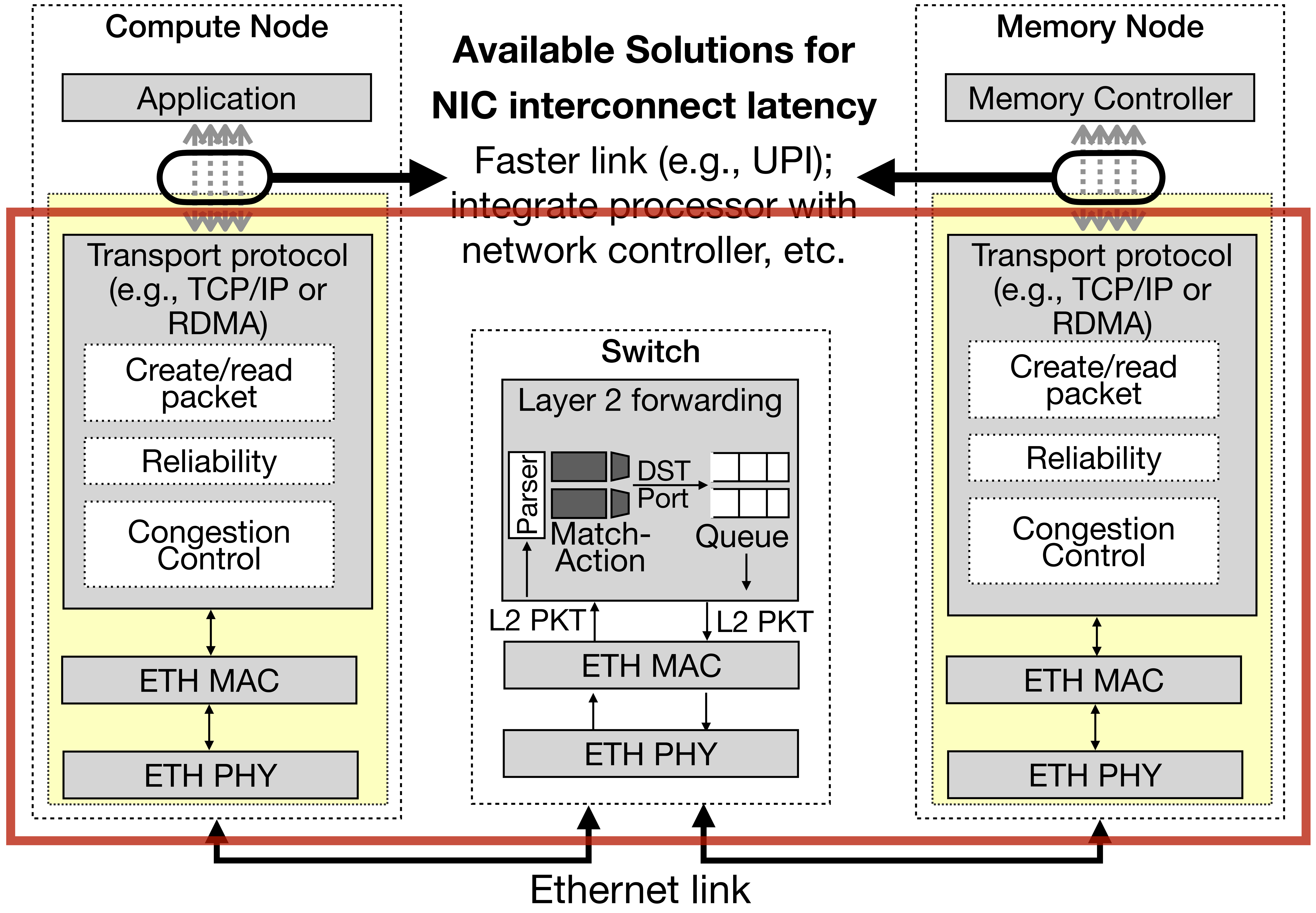
But, separate fabrics for different traffic makes the network costly and harder to manage

Research goal

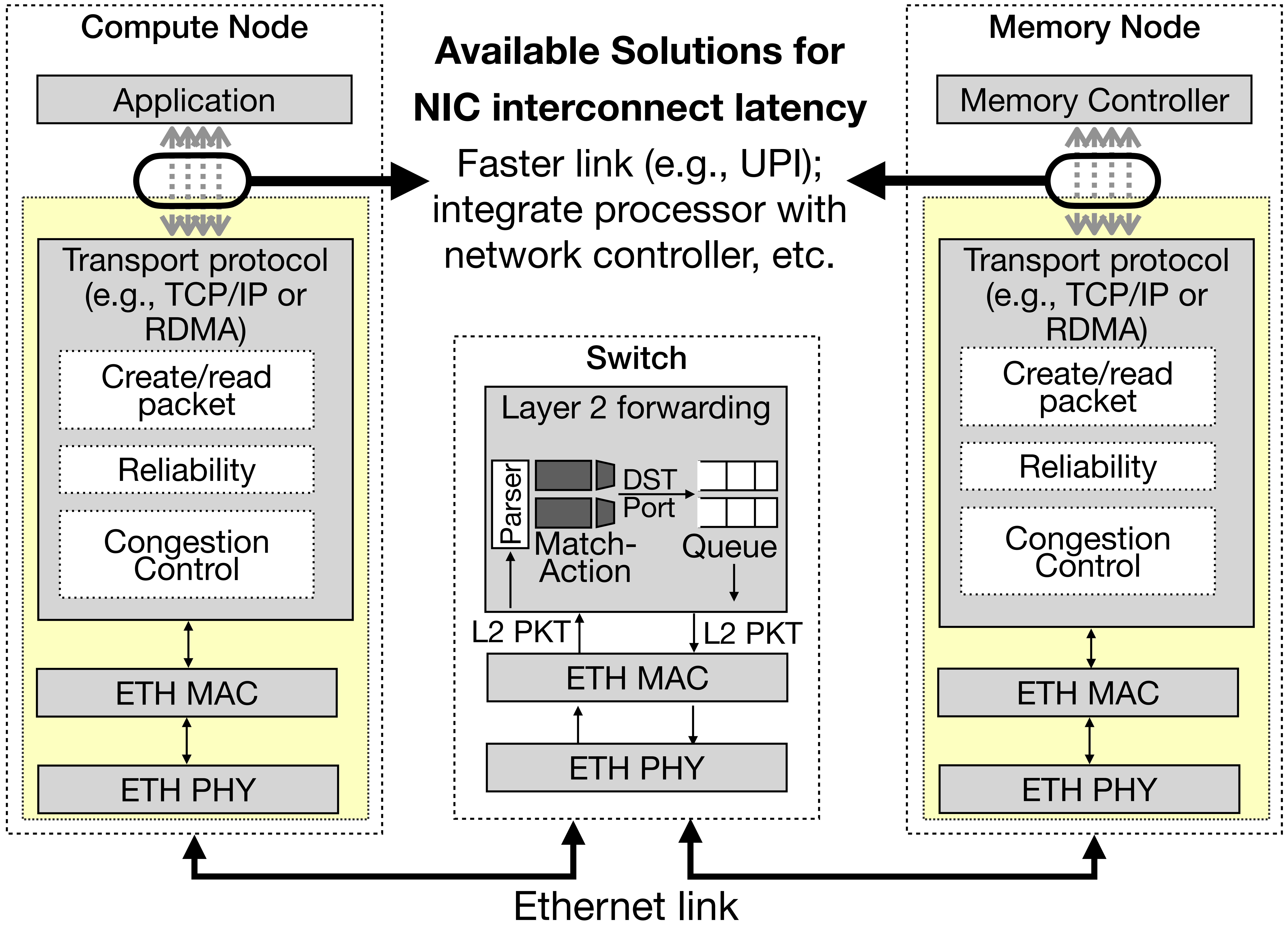
Achieving near **intra-server memory access latency** over rack-scale **Ethernet**

(while maintaining high bandwidth utilization)

Memory Disaggregation over Ethernet

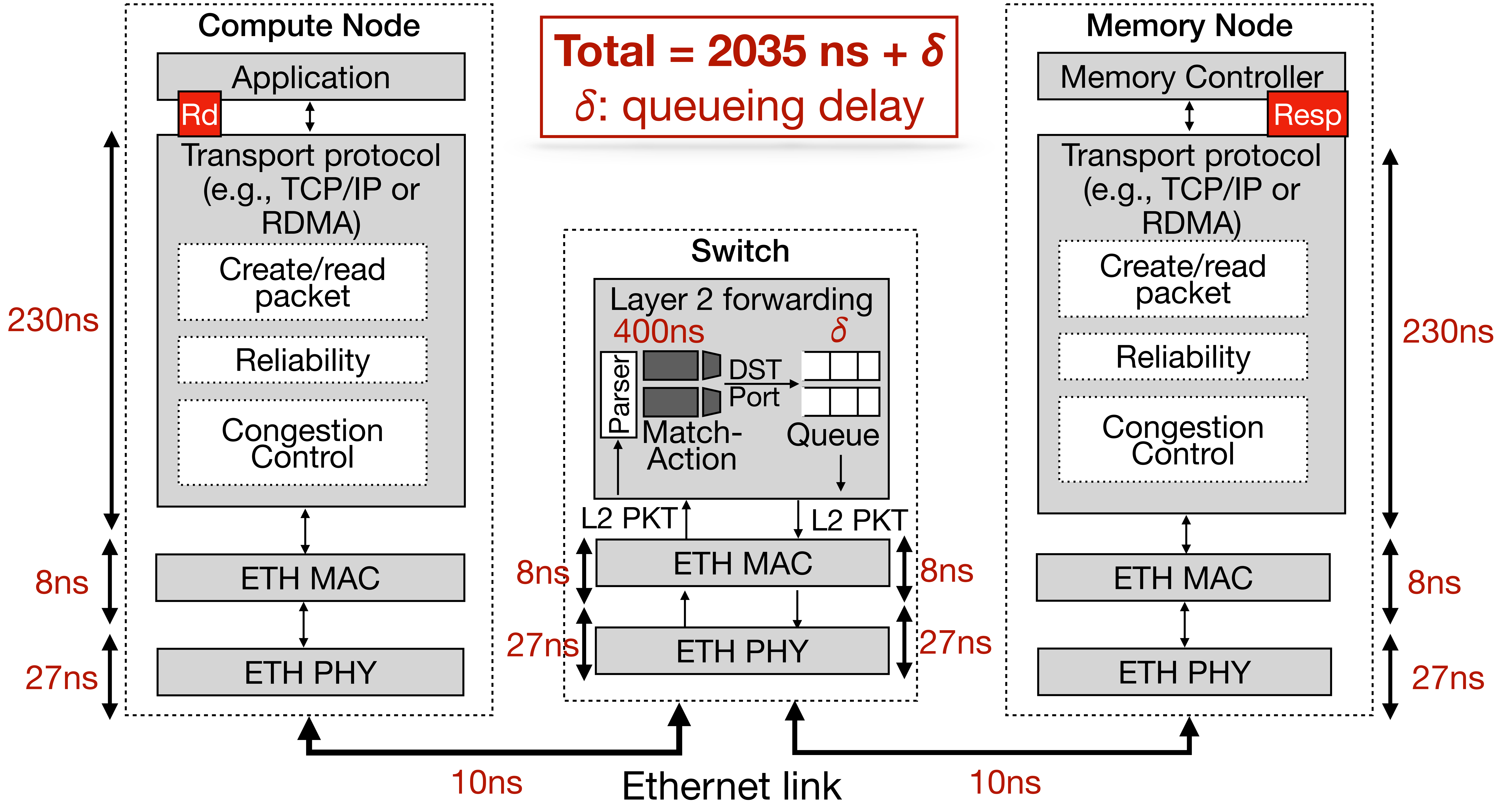


Memory Disaggregation over Ethernet

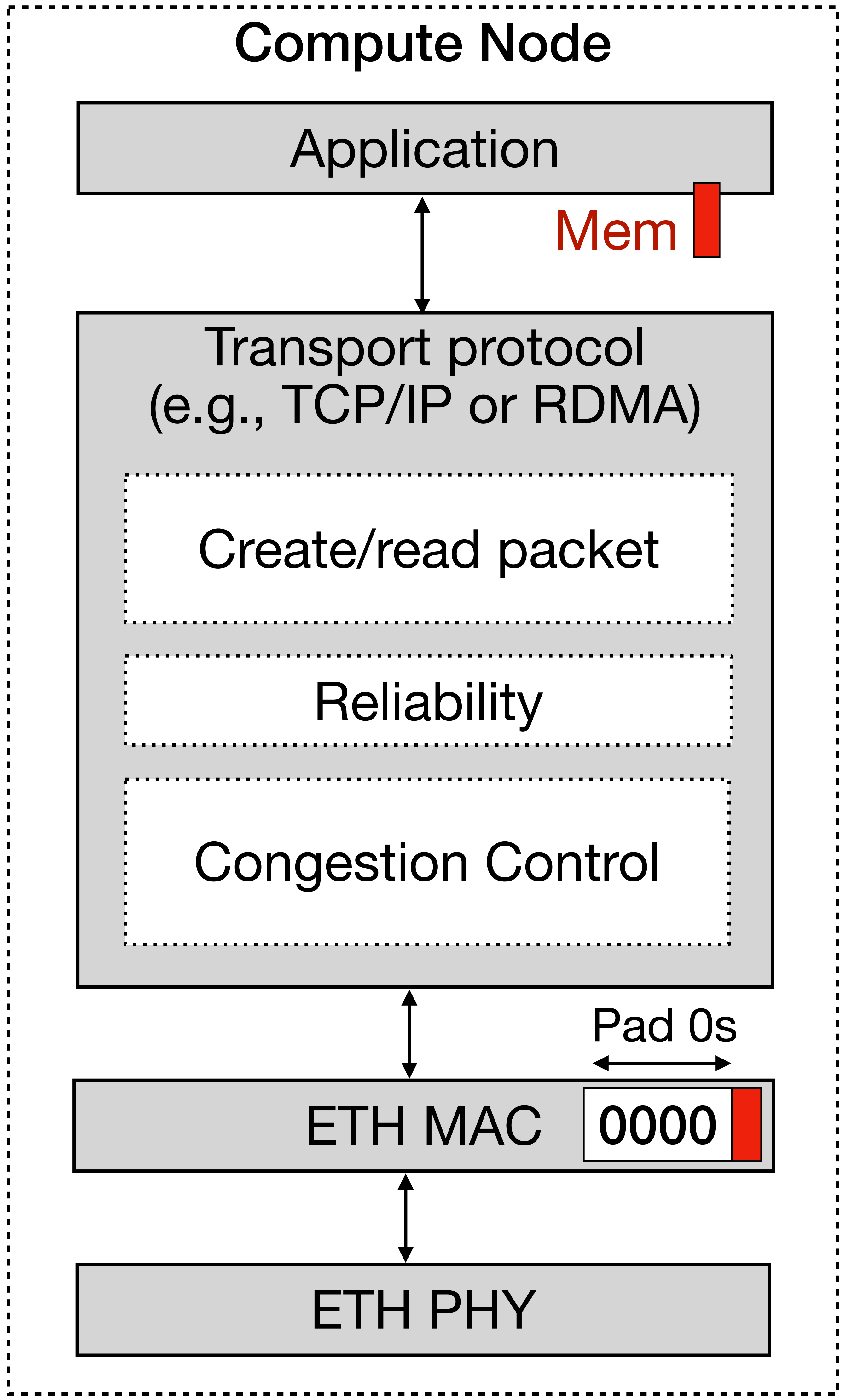


Latency Overheads of Existing Memory Disaggregation over Ethernet

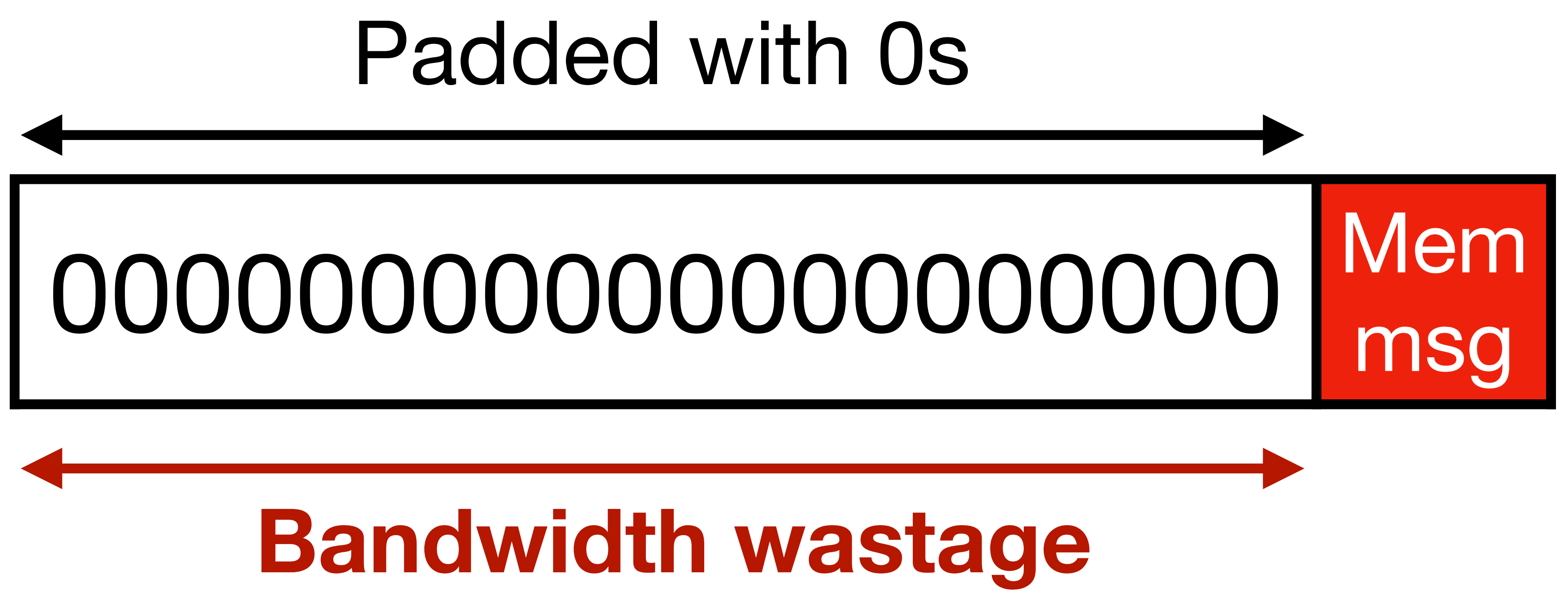
An example of remote read request over RDMA



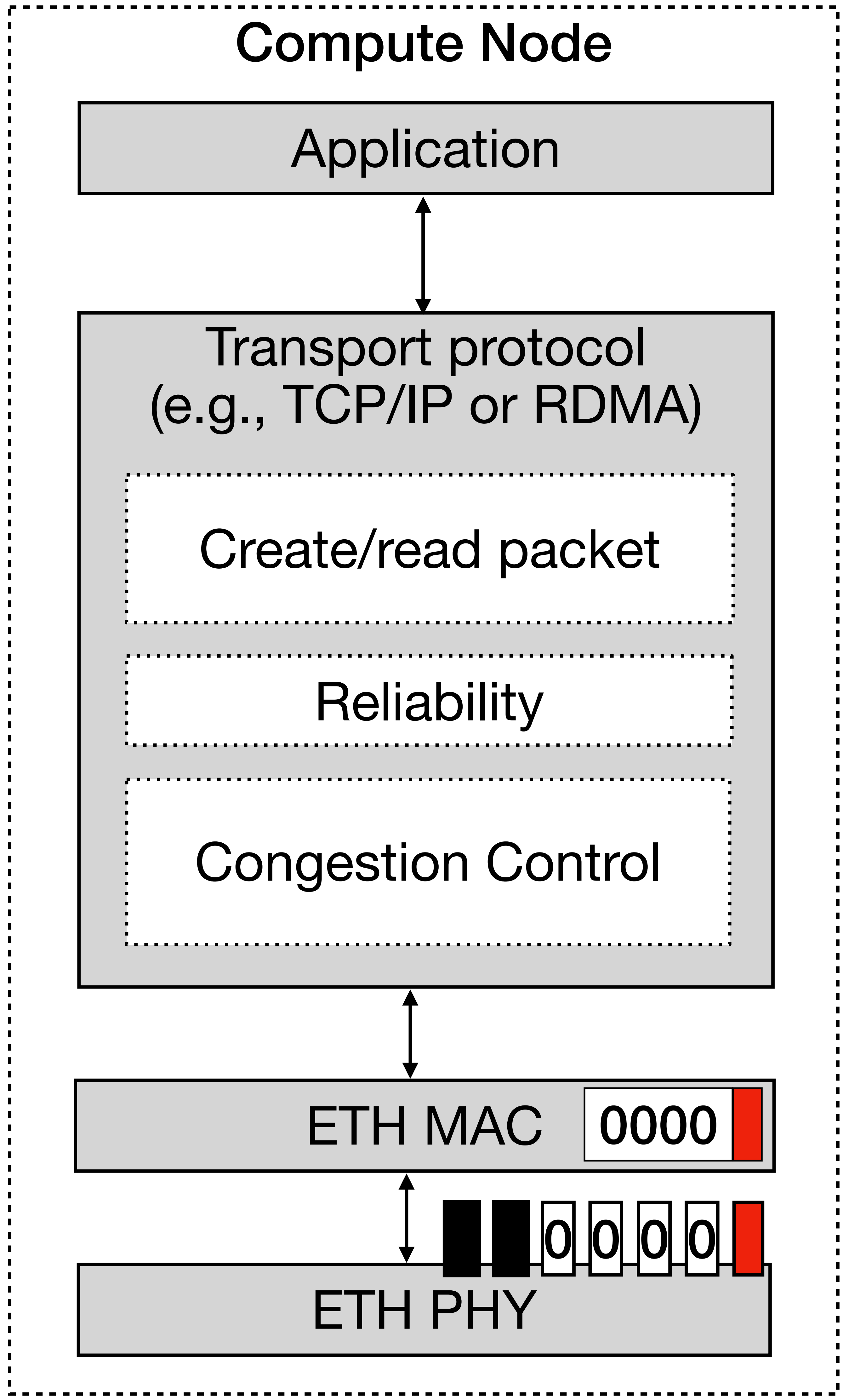
Other Overheads



1. **Ethernet MAC enforces minimum 64B frame**
   … but memory messages can be much smaller (e.g., read requests are typically 8-16B)

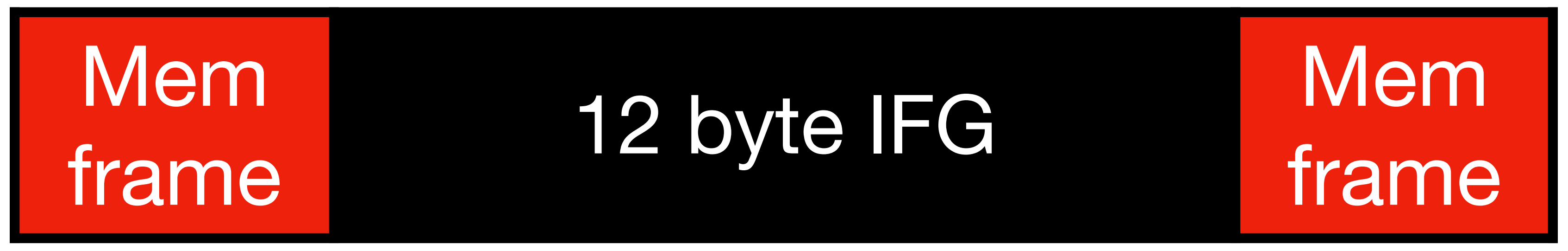


Other Overheads

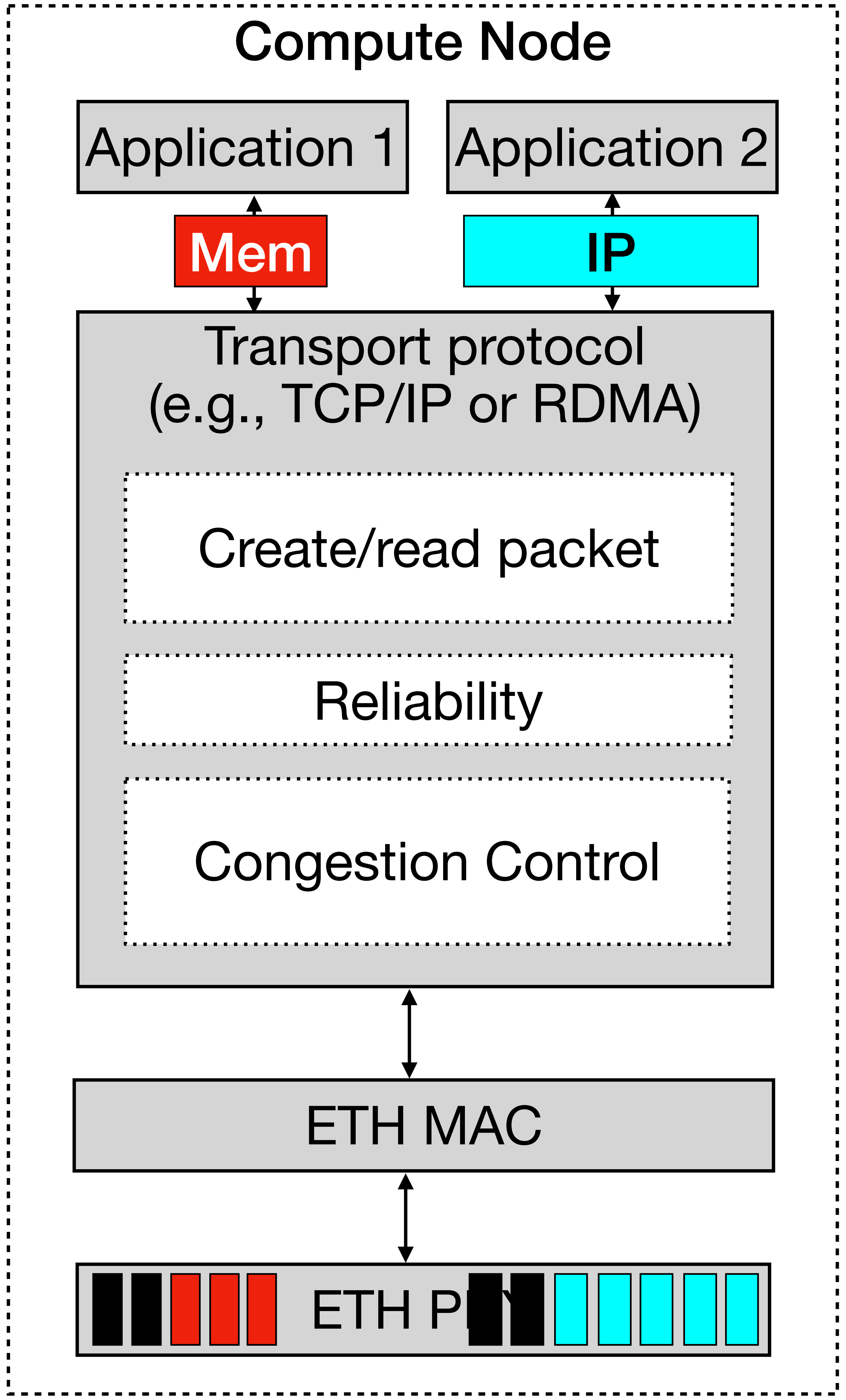


1. **Ethernet MAC enforces minimum 64B frame**
   … but memory messages can be much smaller (e.g., read requests are typically 8-16B)

2. **Ethernet MAC enforces minimum of 12 bytes Inter-frame gap (IFG)**
   … high overhead for small memory messages



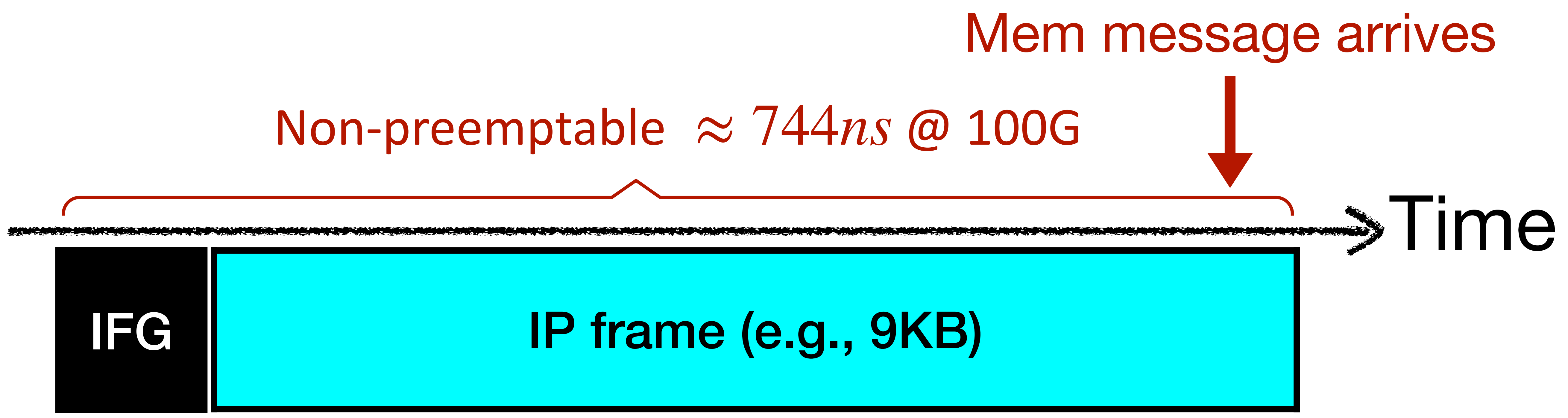
Other Overheads



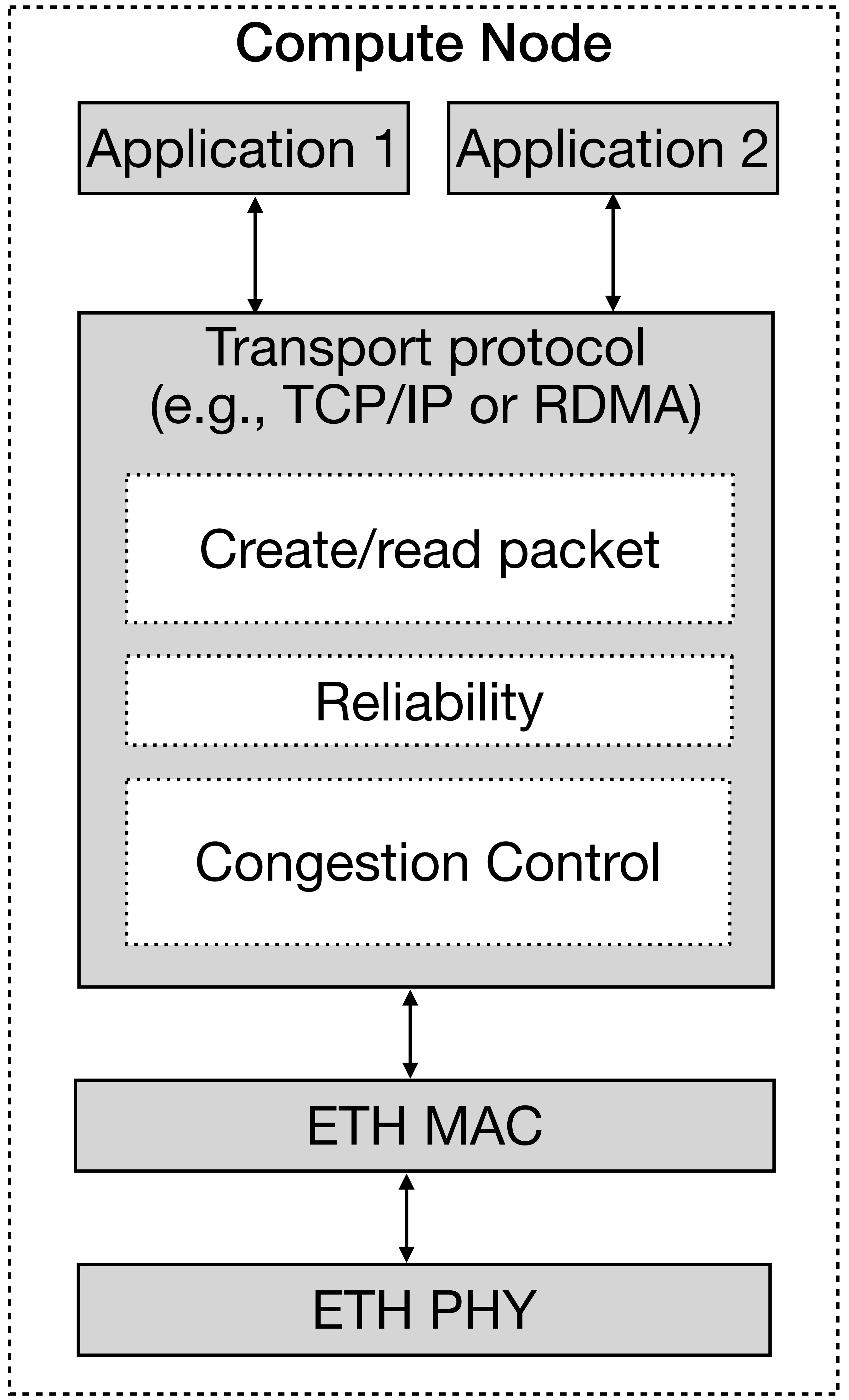
1. **Ethernet MAC enforces minimum 64B frame**
   … but memory messages can be much smaller (e.g., read requests are typically 8-16B)

2. **Ethernet MAC enforces minimum of 12 bytes Inter-frame gap (IFG)**
   … high overhead for small memory messages

3. **Ethernet MAC does not allow intra-frame preemption**
   … a large non-memory frame may block the transmission of a small memory message



Other Overheads



1. **Ethernet MAC enforces minimum 64B frame**
   … but memory messages can be much smaller (e.g., read requests are typically 8-16B)

2. **Ethernet MAC enforces minimum of 12 bytes Inter-frame gap (IFG)**
   … high overhead for small memory messages

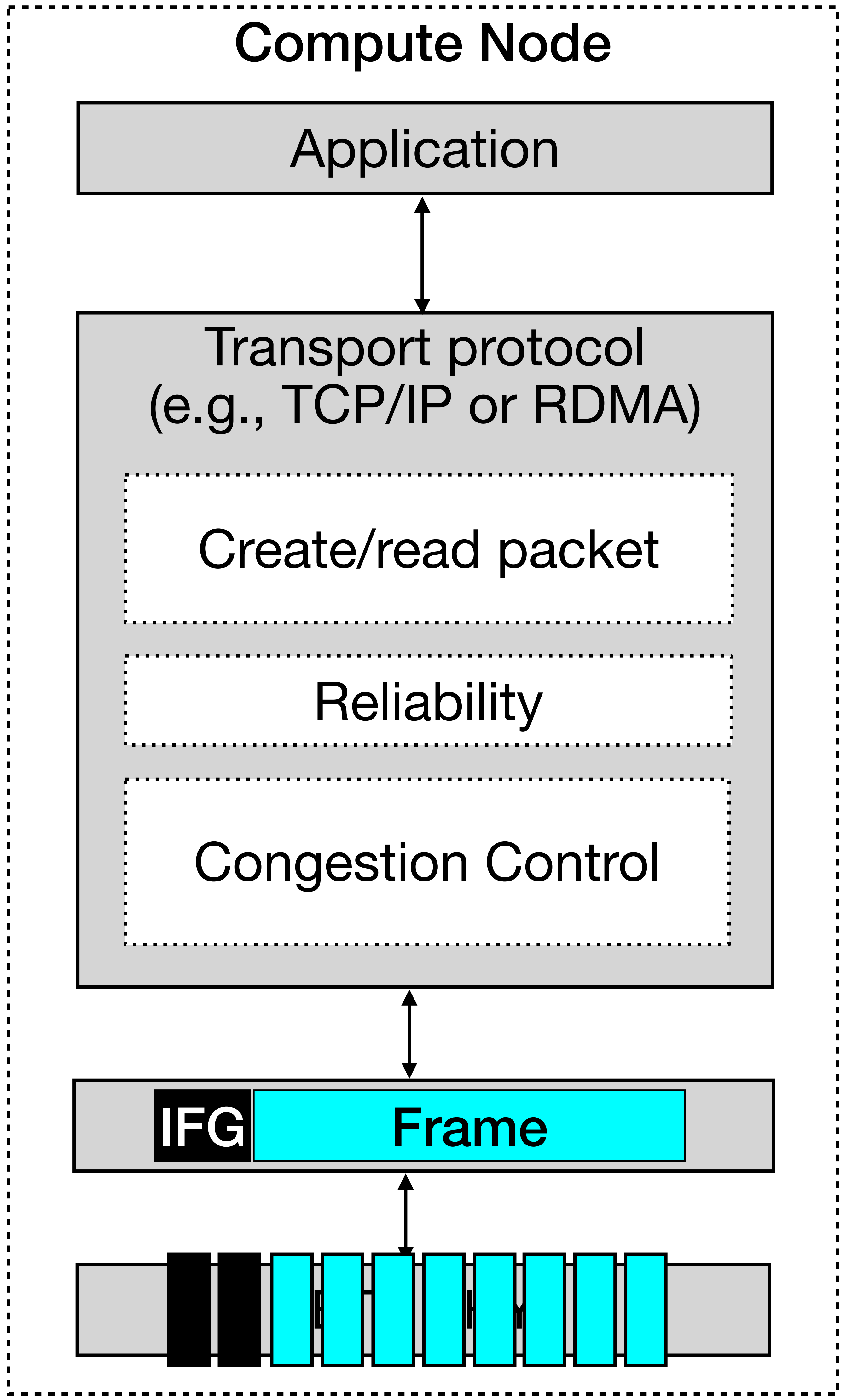
3. **Ethernet MAC does not allow intra-frame preemption**
   … a large non-memory frame may block the transmission of a small memory message

Root cause: MAC layer processing

Design Choice # 1:

Implement the entire
protocol for remote memory access
within Ethernet's **Physical layer**

Rationale for Remote Memory Protocol in PHY

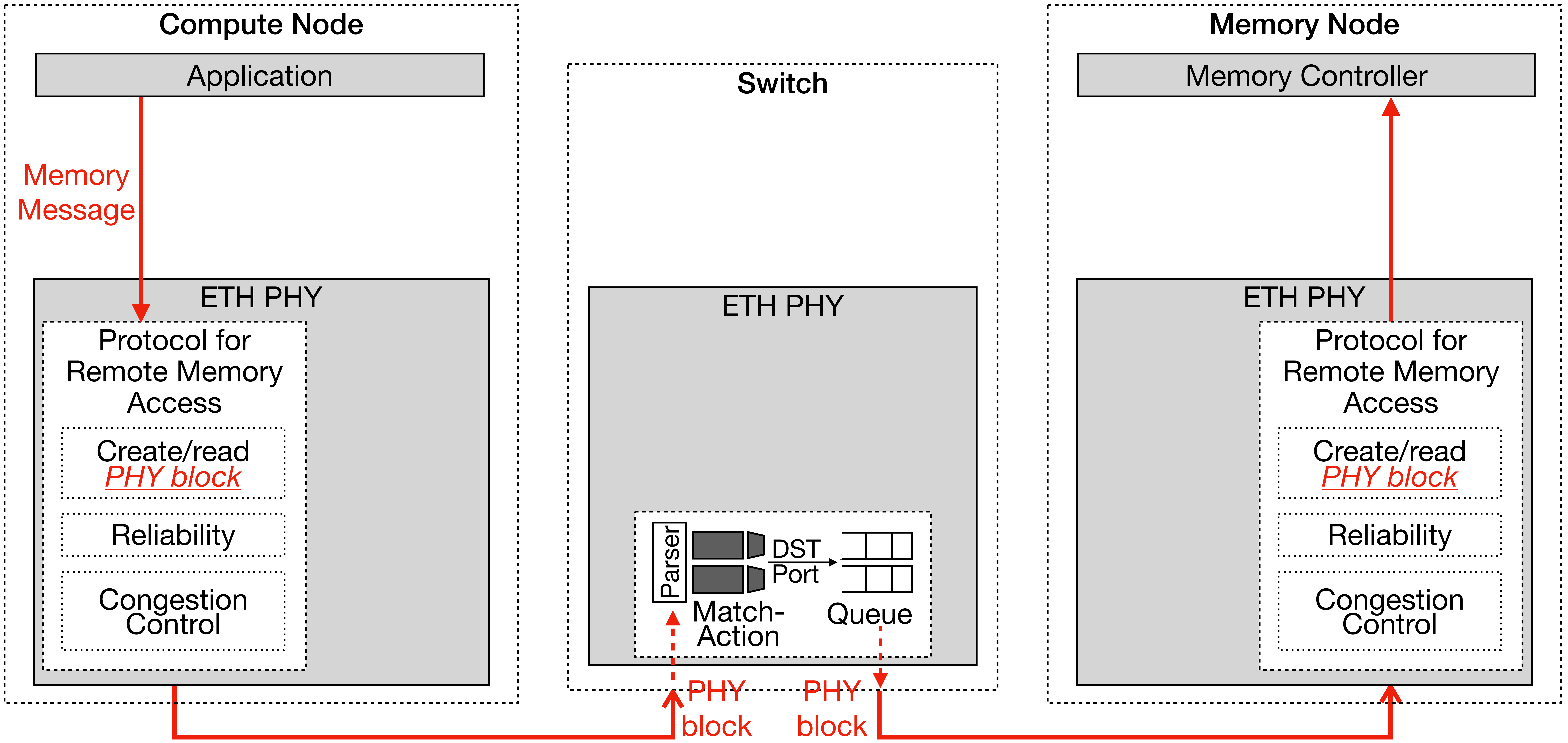


Ethernet PHY already reformats a MAC layer frame into a series of 66-bit PHY blocks

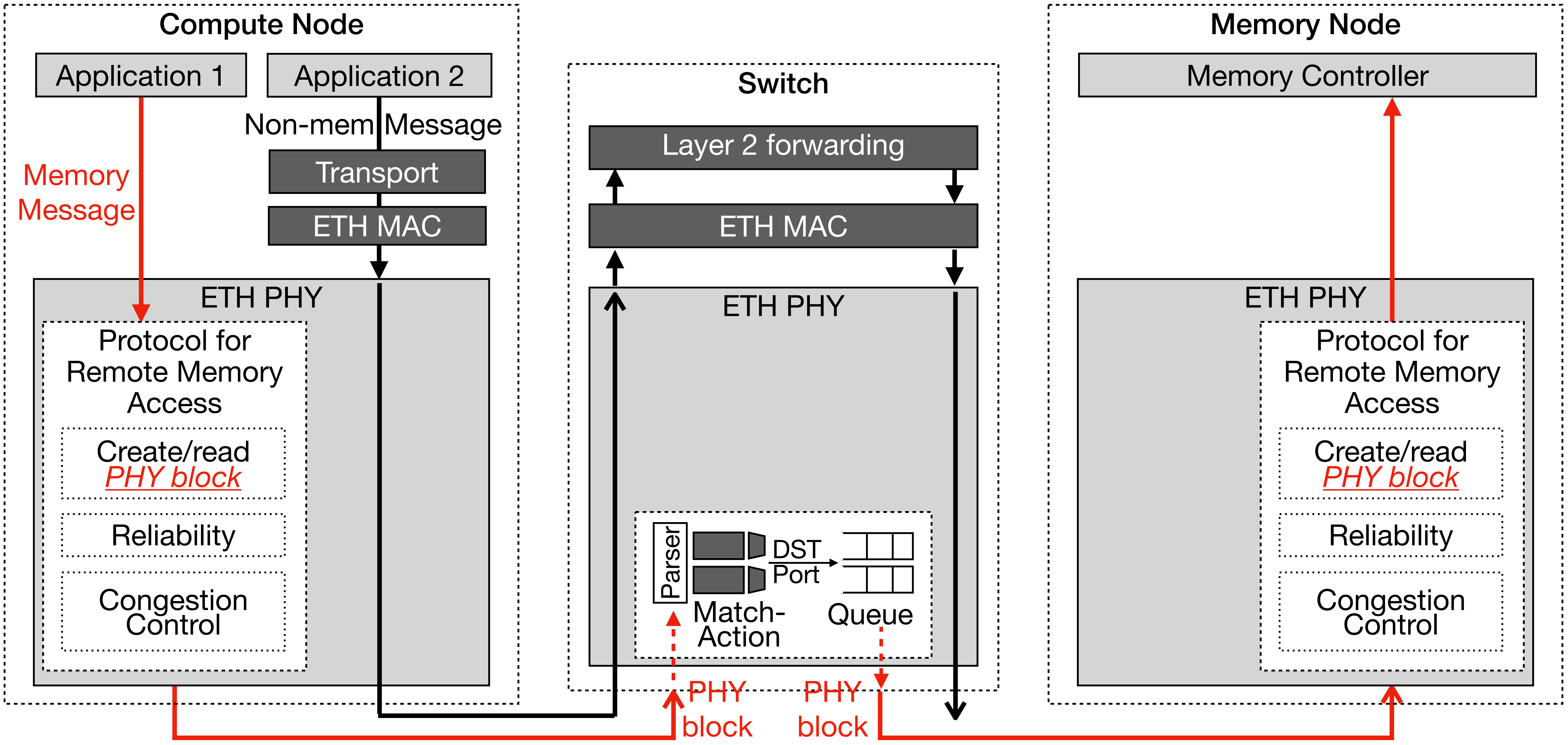
… thus, unlike the MAC layer that works at a **frame** granularity, PHY works at fine-grained **block** granularity

- 66 bit PHY block vs. 64 byte minimum MAC frame size
- PHY also has access to IFG blocks
- Message interleaving can be done at block granularity in PHY rather than at frame granularity in MAC

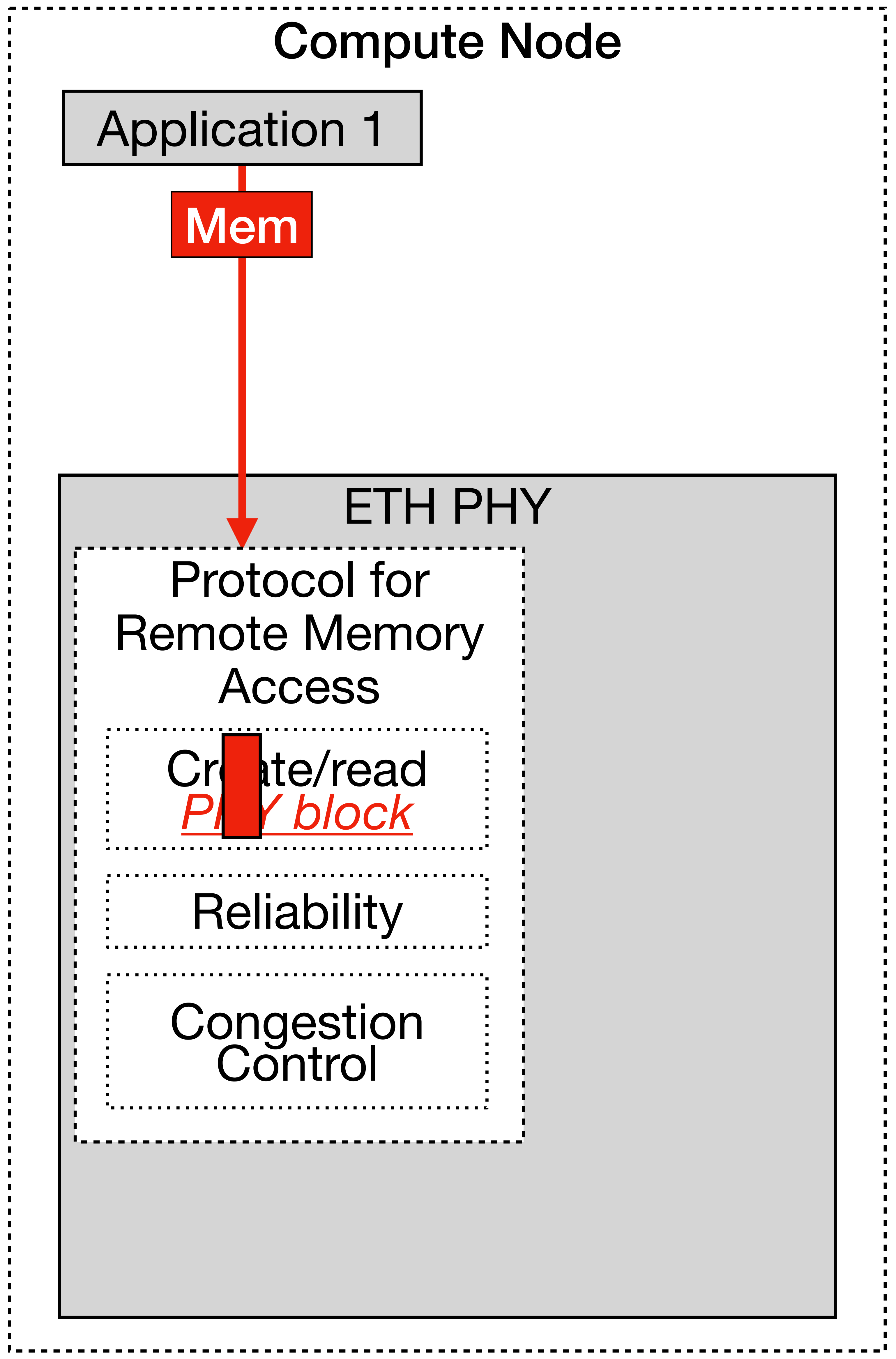
Architecture of Remote Memory Protocol in the PHY



Architecture of Remote Memory Protocol in the PHY

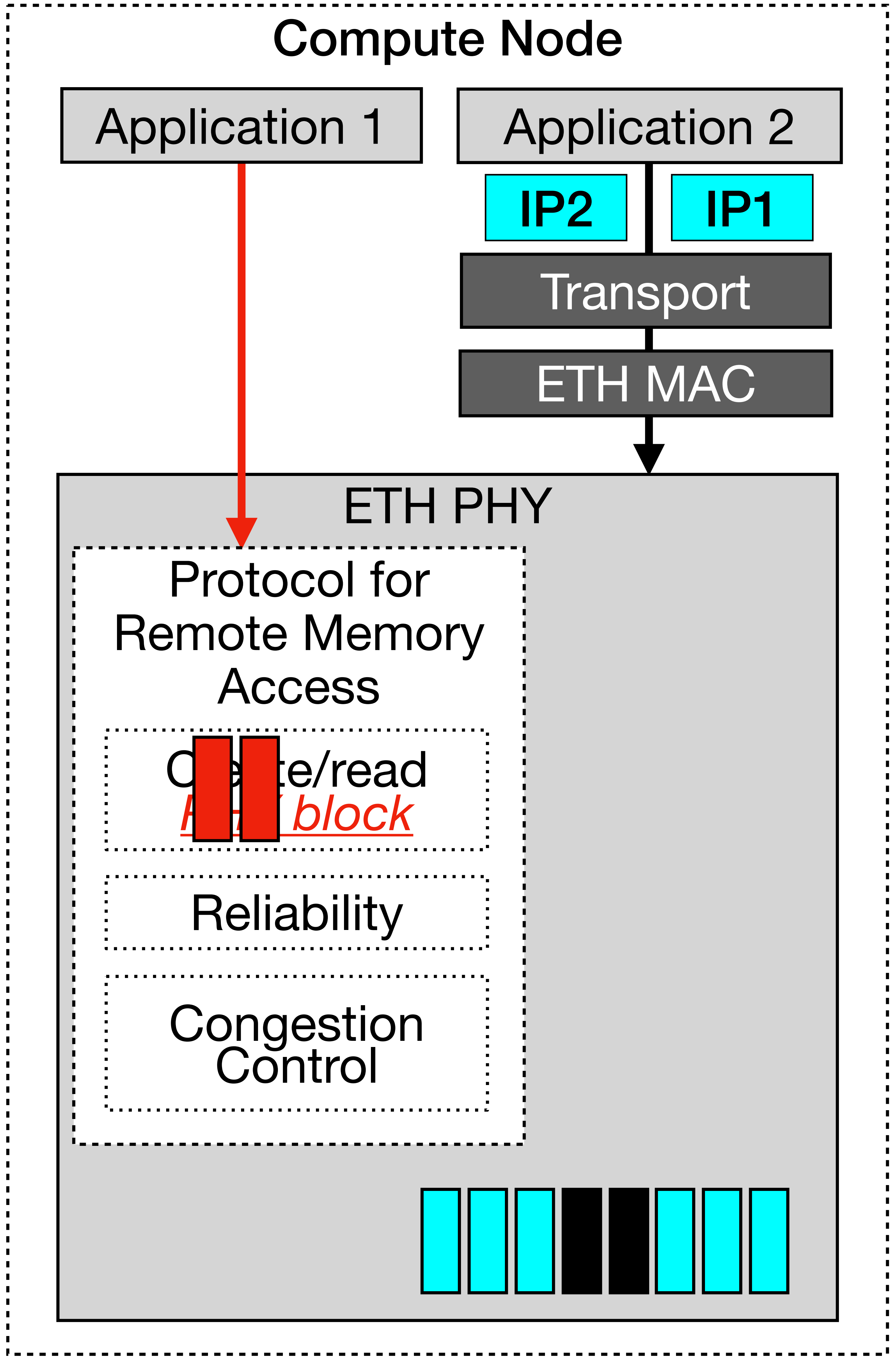


Benefits of Remote Memory Protocol in the PHY



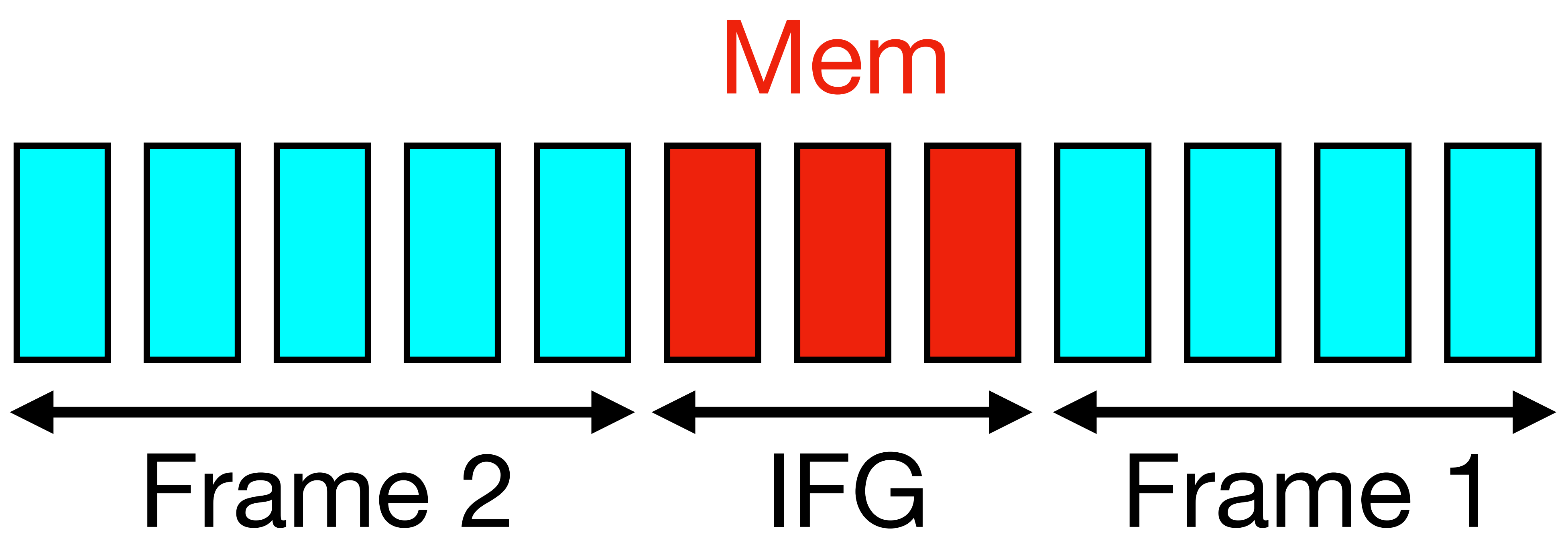
1. **Ethernet PHY operates at a fine data granularity of 66-bit PHY blocks.**
   Avoids bandwidth wastage for small memory messages.

Benefits of Remote Memory Protocol in the PHY

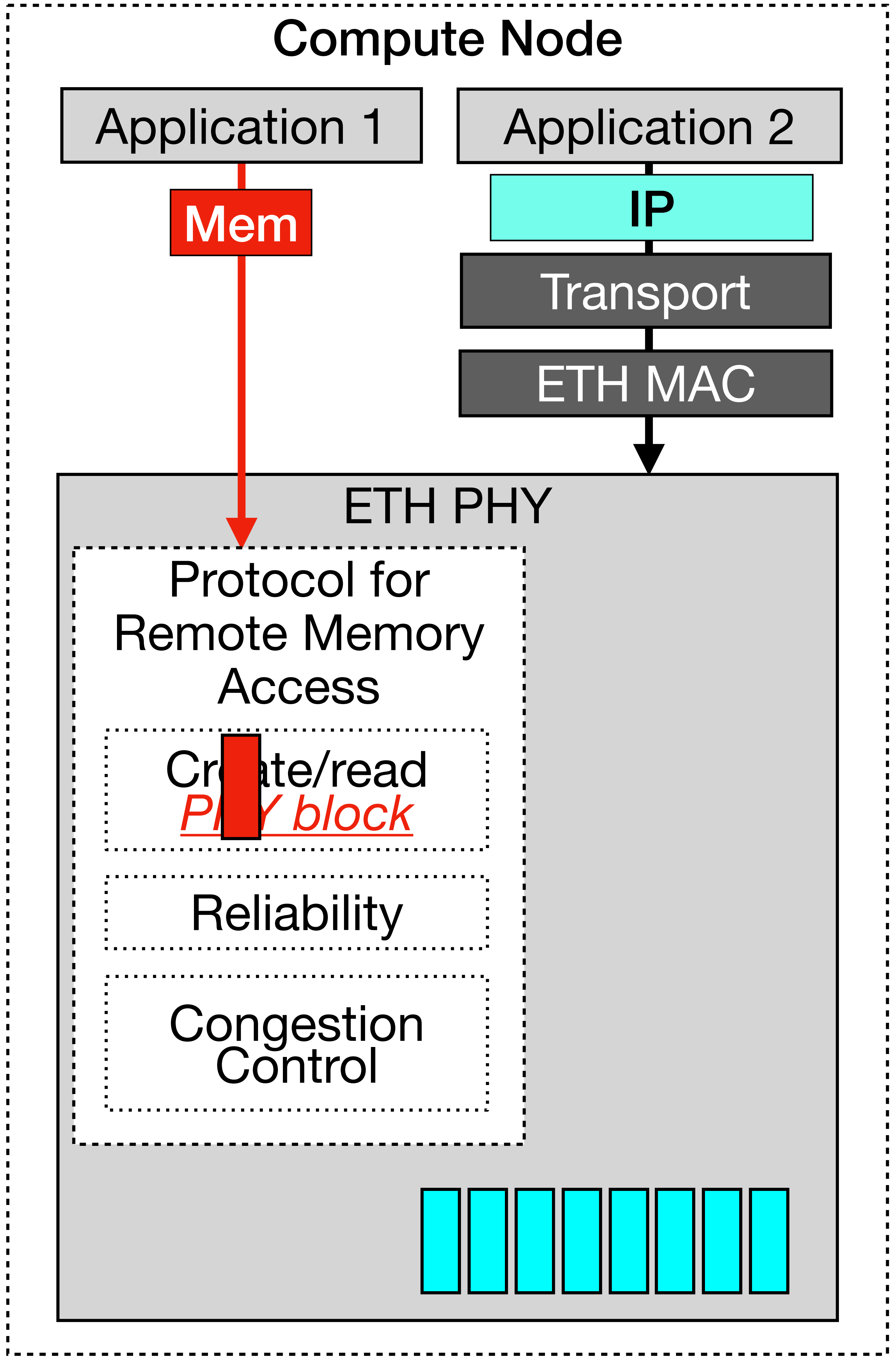


1. **Ethernet PHY operates at a fine data granularity of 66-bit PHY blocks.**
   Avoids bandwidth wastage for small memory messages.

2. **Ethernet PHY has access to IFG bits.**
   Can repurpose IFG to carry memory messages.



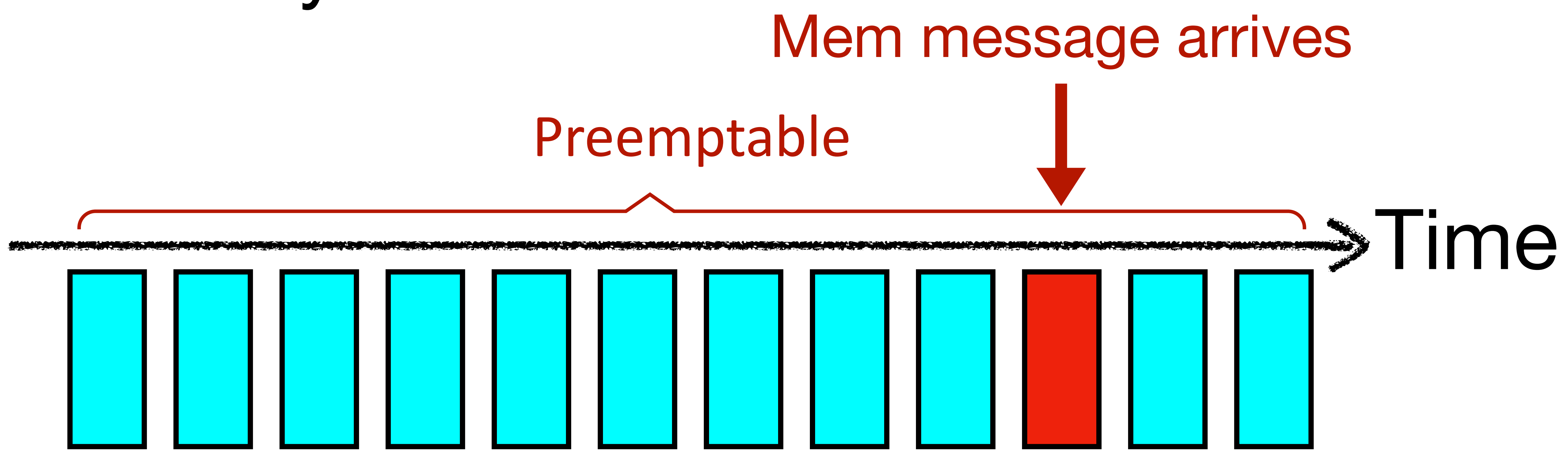
Benefits of Remote Memory Protocol in the PHY



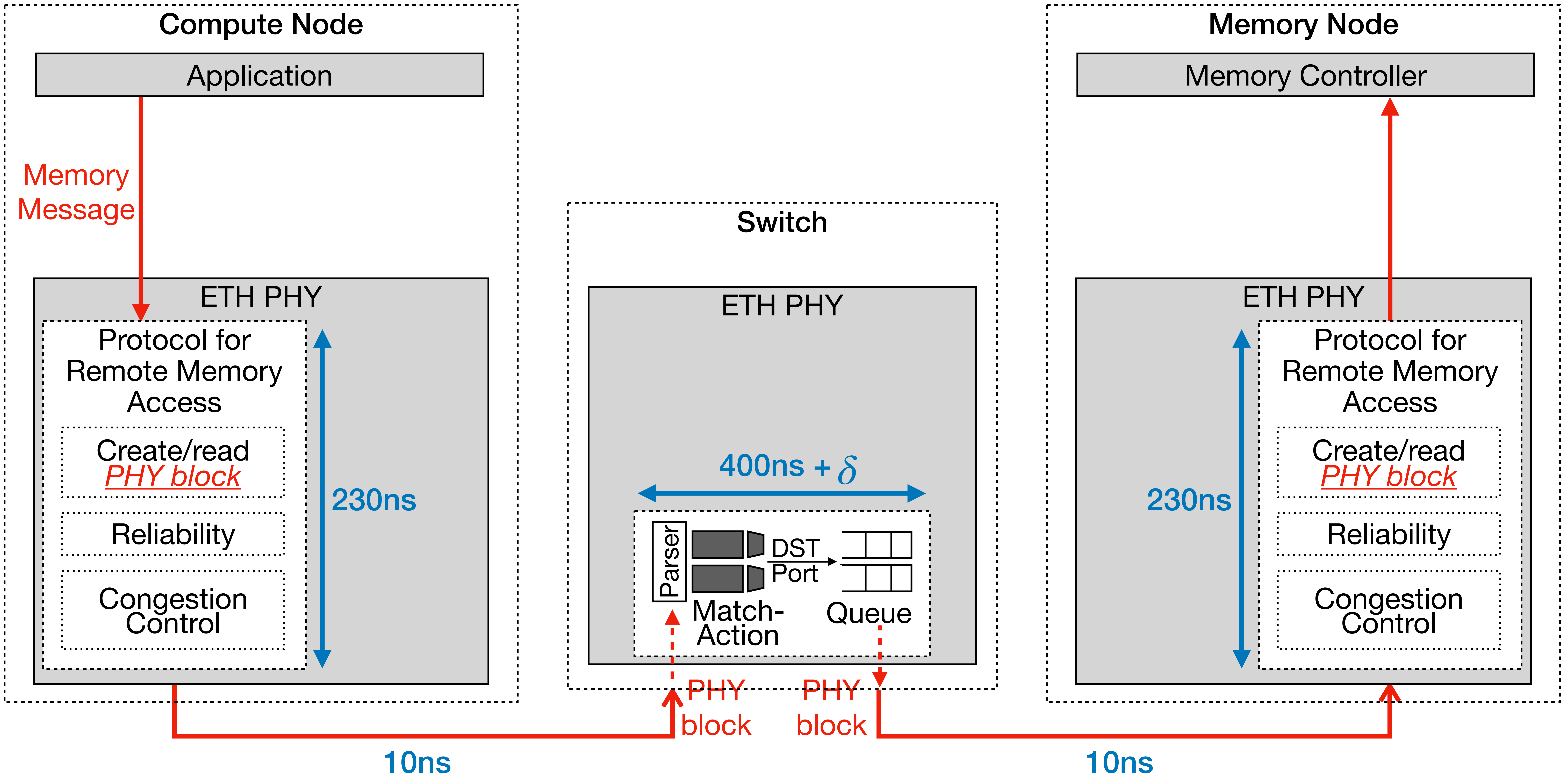
1. **Ethernet PHY operates at a fine data granularity of 66-bit PHY blocks.**
   Avoids bandwidth wastage for small memory messages.

2. **Ethernet PHY has access to IFG bits.**
   Can repurpose IFG to carry memory messages.

3. **Ethernet PHY enables intra-frame preemption**
   Avoids blocking of small memory message by a large non-memory frame.



Remote Memory Protocol in the PHY : What about latency?



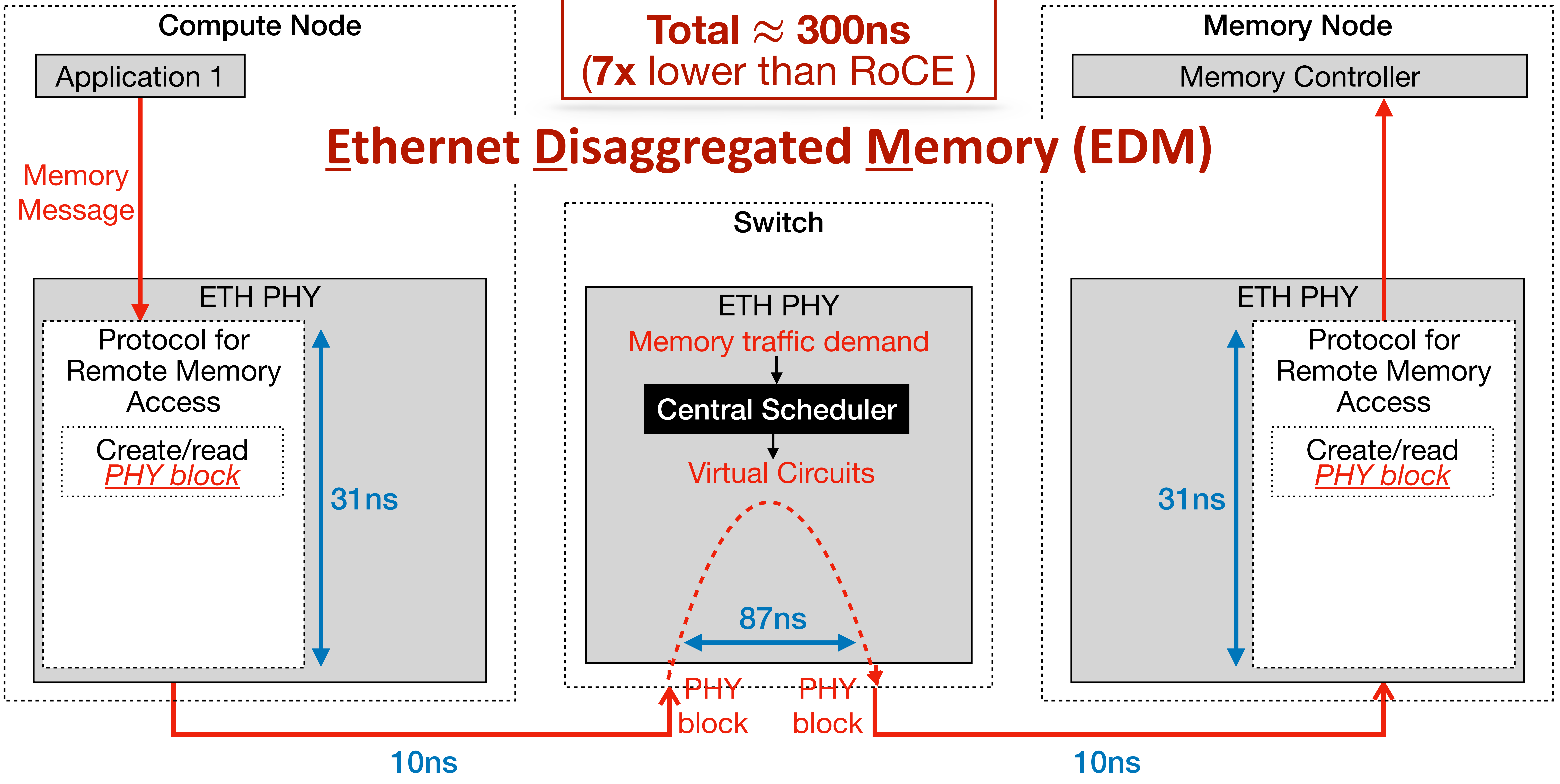
Design Choice # 2:

Implement a
centralized memory traffic scheduler
in the PHY of the switch

Central Scheduler in the Switch PHY

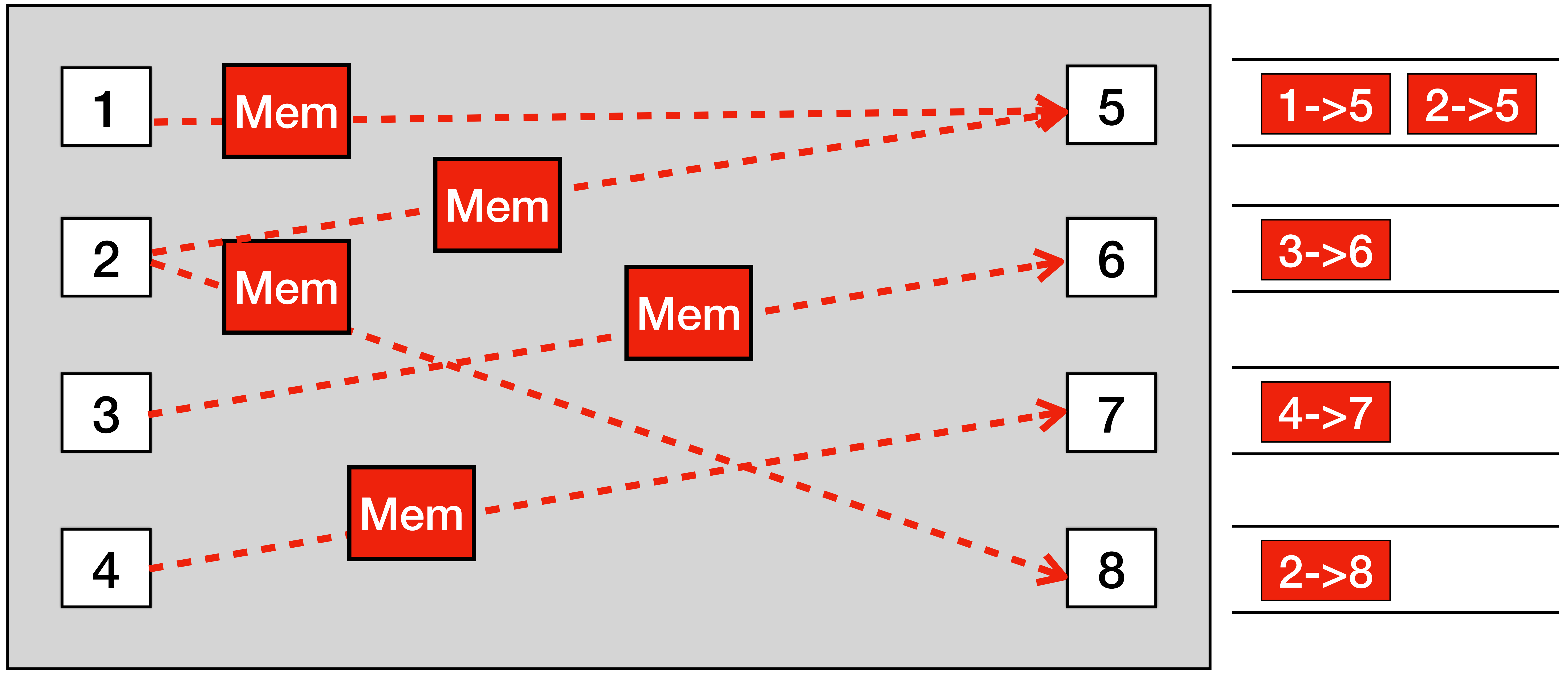
Total ≈ 300ns
(**7x** lower than RoCE)

Ethernet Disaggregated Memory (EDM)



Overview of Central Scheduler

- **Step 1:** Nodes send their memory message demands {src->dst} to the switch scheduler
- **Step 2:** Switch scheduler creates virtual circuits by forming a Matching based on demand
    - Naive maximal matching ~O(N); EDM uses **Parallel Iterative Matching (PIM)**[1] ~O(log(N))
- **Step 3:** Nodes exchange memory messages over established circuits



[1] Anderson et al. “High Speed switch scheduling for Local Area Networks”. TOCS 1993.

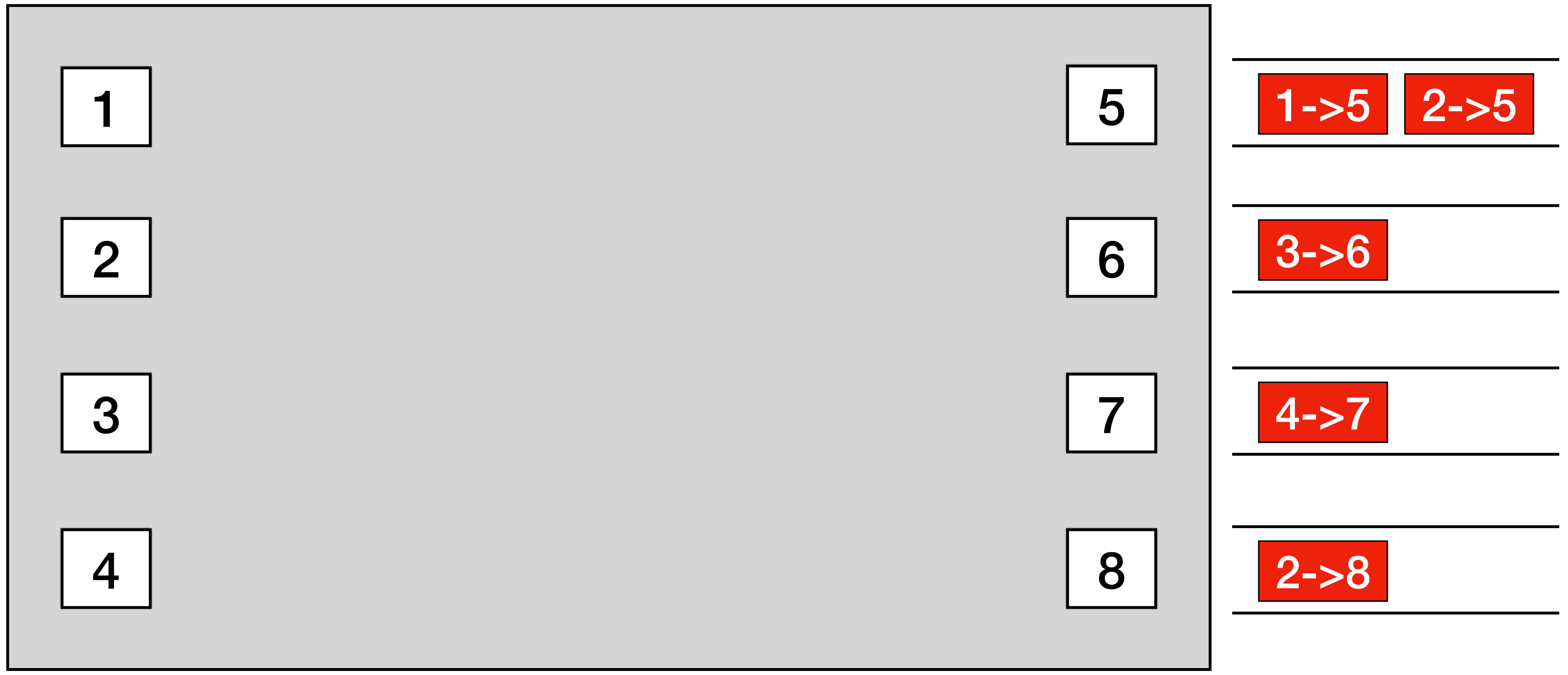
Practical Central Scheduler

- **Challenge 1:** Low latency for memory messages under bandwidth contention
- **Challenge 2:** Accurate, low overhead memory traffic demand estimation
- **Challenge 3:** Line rate, low latency scheduling hardware pipeline

Challenge # 1: Achieve low latency under bandwidth contention

Solution: Augment PIM with priority scheduling

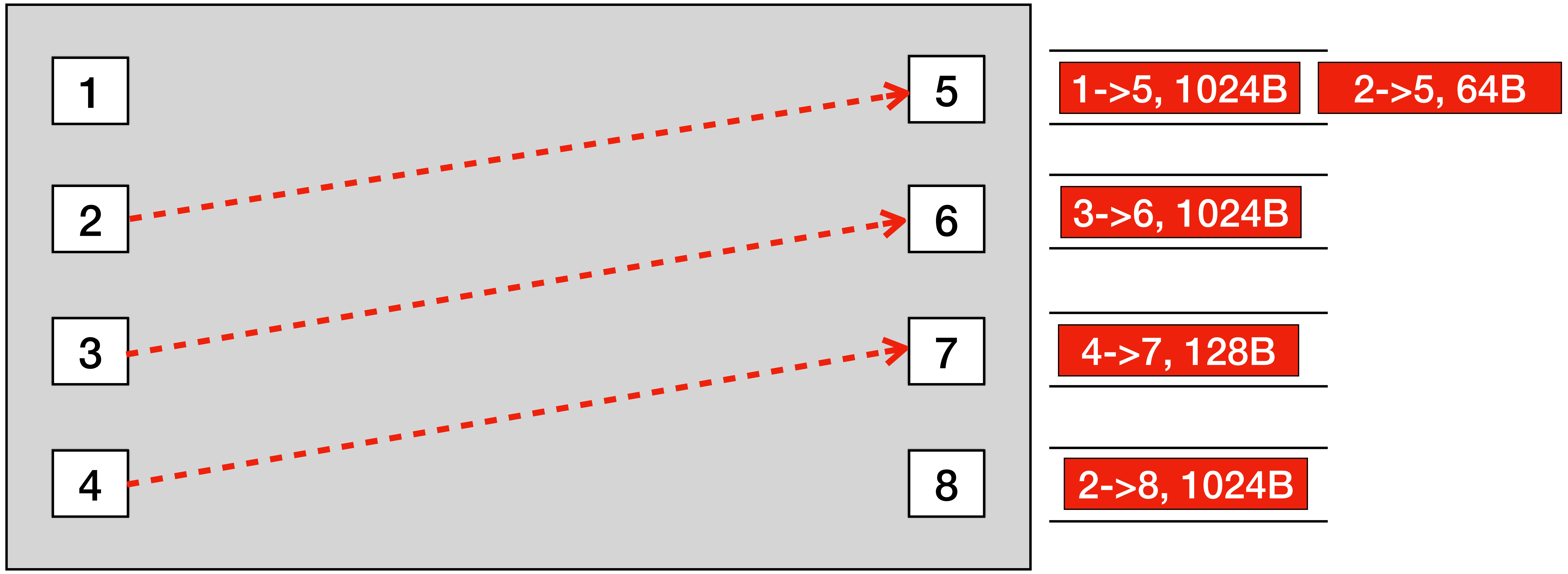
- First Come First Serve (FCFS) for *light-tailed* traffic distribution
- Shortest Remaining Processing First (SRPT) for *heavy-tailed* traffic distribution



Challenge # 1: Achieve low latency under bandwidth contention

Solution: Augment PIM with priority scheduling

- First Come First Serve (FCFS) for *light-tailed* traffic distribution
- Shortest Remaining Processing First (SRPT) for *heavy-tailed* traffic distribution

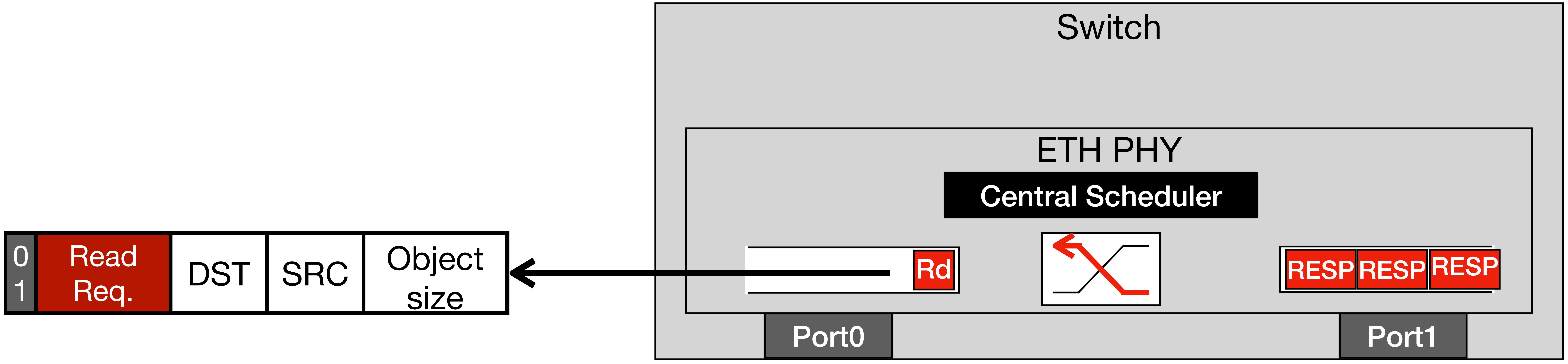


- For SRPT, each demand message from the nodes also contains the **size of message**
- Demand messages per node are processed in the **increasing order of remaining bytes**
- Matching contention -> **prioritize** demand messages with **smaller remaining bytes**

Challenge # 2: Acquire accurate, low overhead memory traffic demand matrix

Solution: Leverage the nature of memory access interface that specifies amount of data to be read or written

- For **reads**, read request implicity contains demand for read reply
  - Zero bandwidth and latency overhead
- For **writes**, send an explicit demand message to switch
  - Small bandwidth overhead (notifications are small)
  - Latency (~RTT/2) is small within a rack



Challenge # 3: Design line-rate, low latency scheduling hardware pipeline

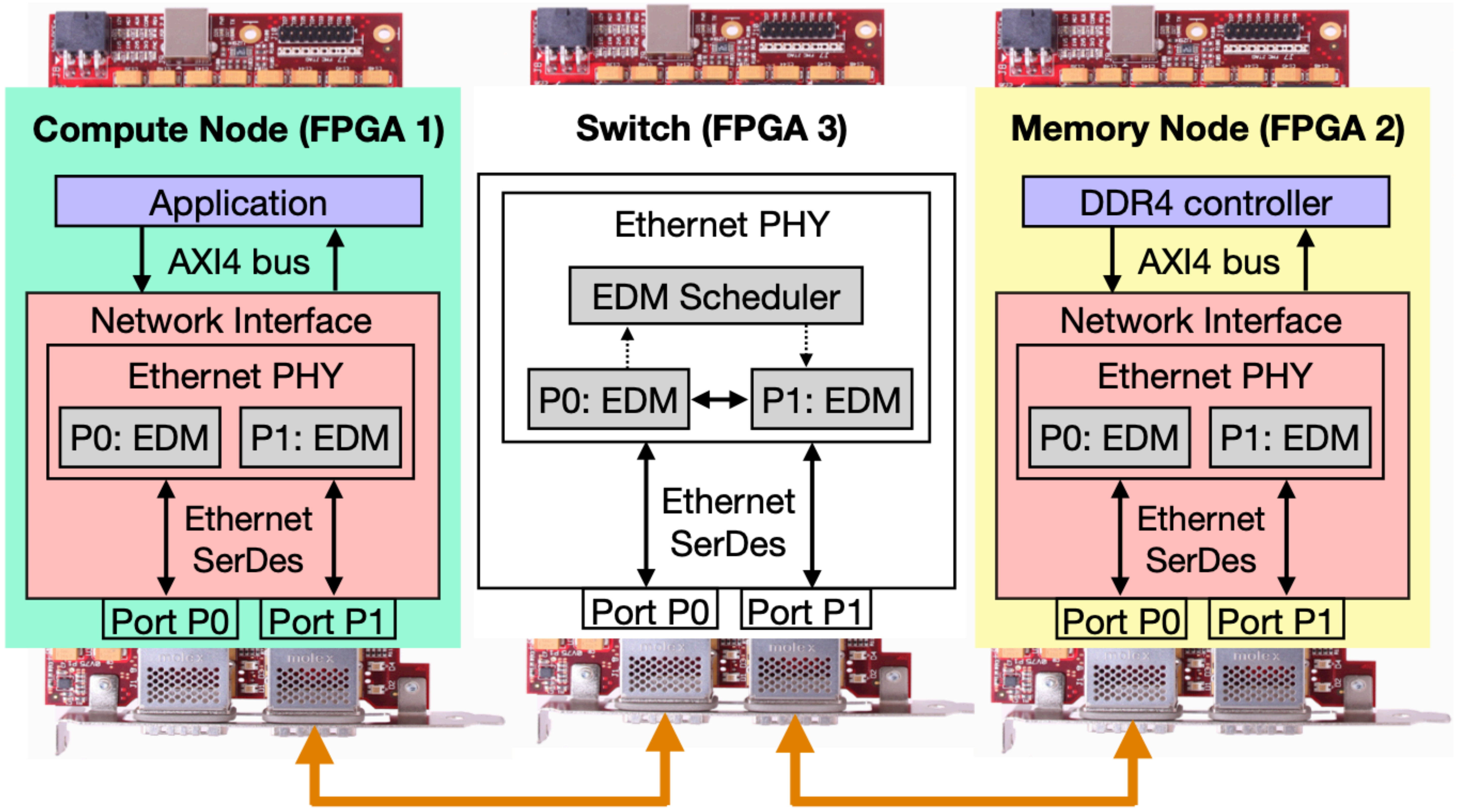
Naive implementation of **priority-based PIM** would take **O(log(N)) cycles** per PIM iteration (N: number of demand messages)

Solution: Leverage hardware parallelism to intelligently trade-off hardware resources for time

- Use combination of constant-time ordered list data structure with a fast priority encoder to implement priority-based PIM

EDM can implement each iteration of PIM in exactly 3 clock cycles

Implementation



Hardware Testbed

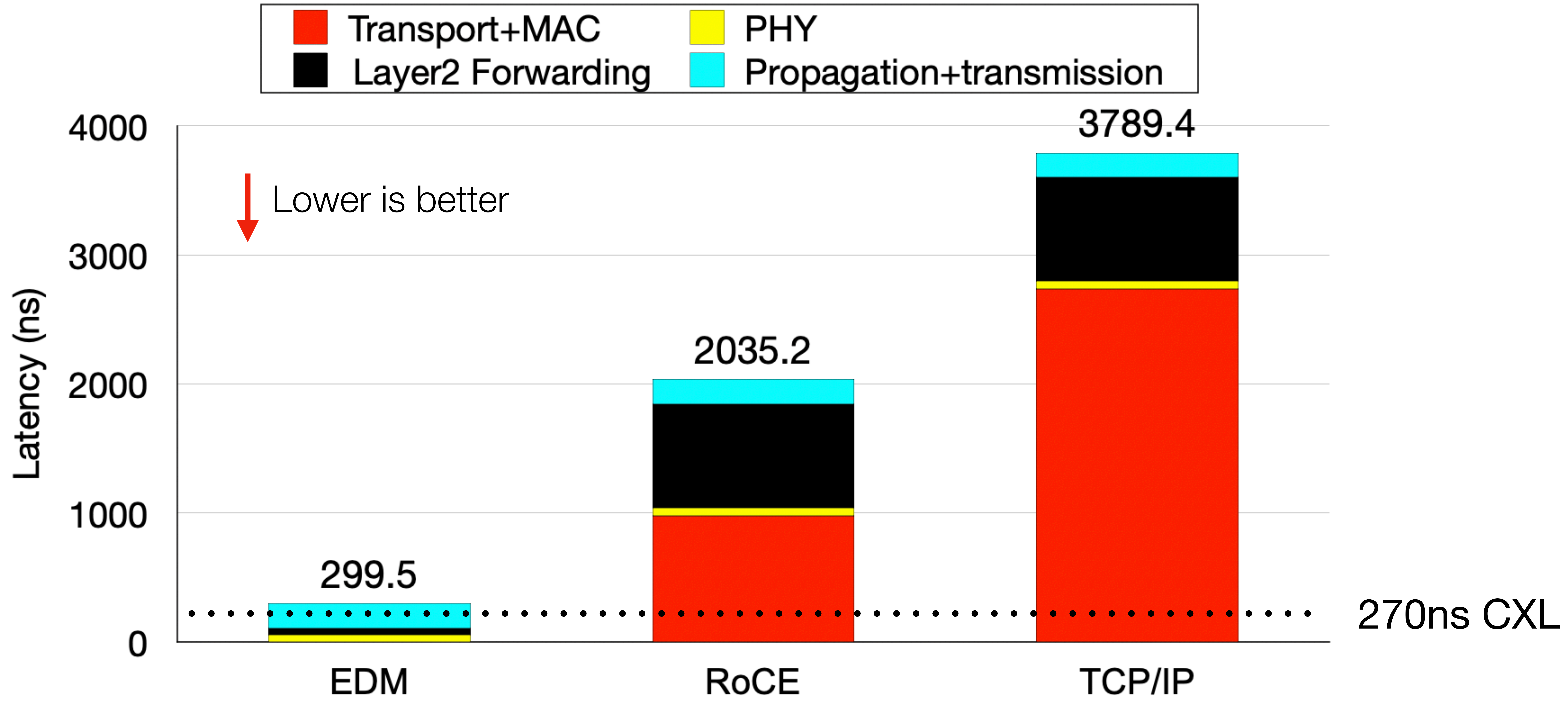
- Three Xilinx Alveo U200 FPGAs
- Open-source 25GbE (Corundum)
- Synopsys ASIC RTL compiler

Network Simulator

- A single rack with 144 nodes
- Fed with real-world traces
- Compare against 6 classes of scheduling / congestion control

Evaluation

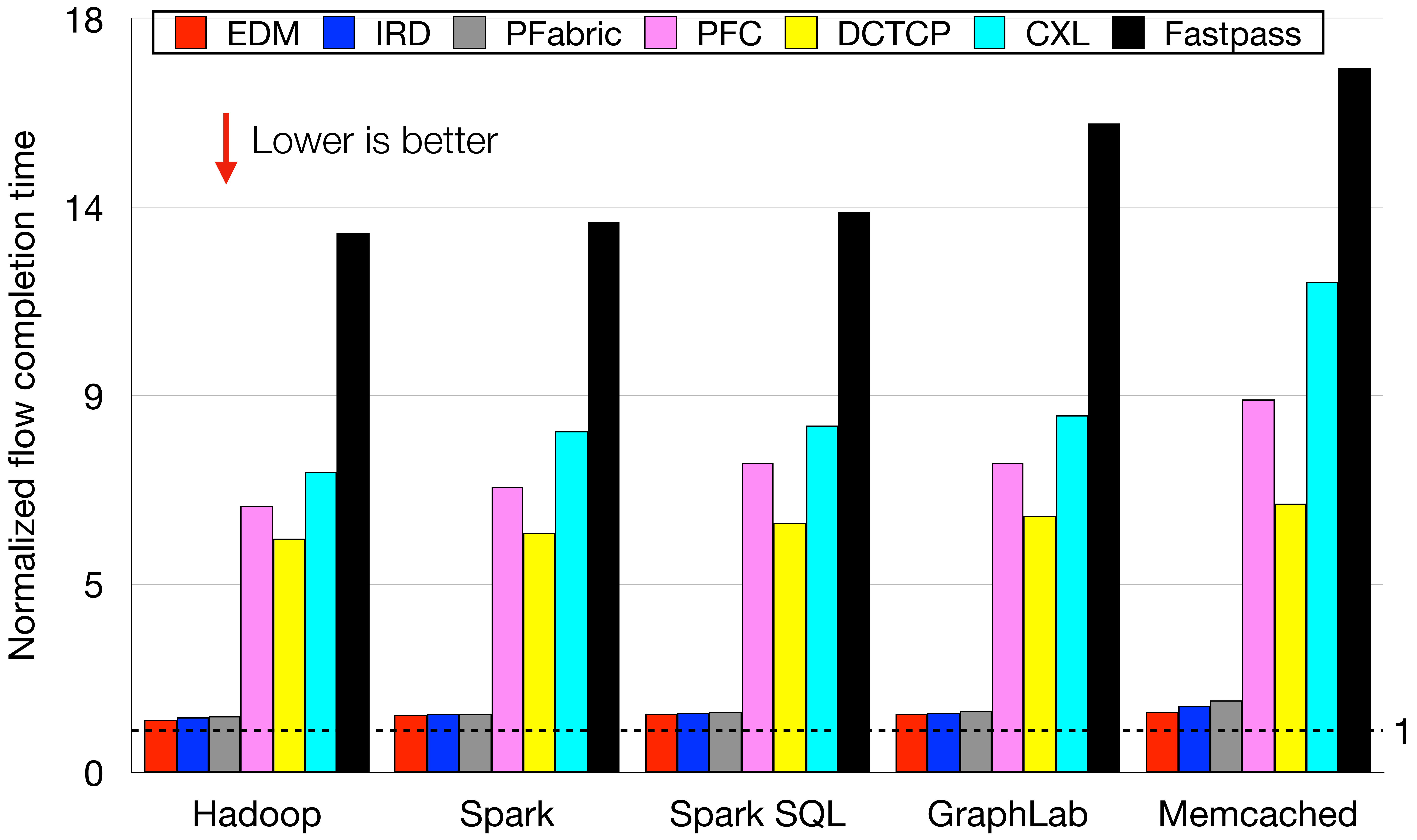
- End-to-end unloaded latency



Evaluation

- Disaggregated workloads in a loaded network

Experiment name	Dataset
Hadoop, Spark	Generator @sortbenchmark.org
Spark SQL	Big Data Benchmark@Berkeley
GraphLab	Movie rating data @Netflix
Memcached	KV-store@YCSB



Summary

- EDM is a low latency Ethernet fabric for memory disaggregation.
- EDM uses two ideas for low latency w/ high bandwidth utilization:
  - EDM implements the **protocol for remote memory access** entirely in the **Ethernet PHY.**
  - EDM implements a **fast, centralized memory traffic scheduler** in the switch's PHY.
- EDM incurs a latency of **~300ns (7x lower than RoCE)** in an unloaded network, and **< 1.3x** its unloaded latency under heavy network loads.

Thank you !

Code: https://github.com/wegul/EDM