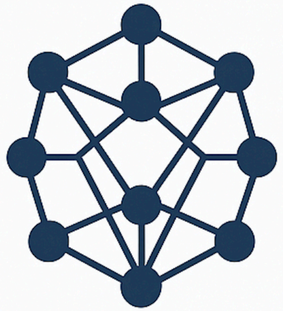
2025 Workshop on



RECONFIGURABLE NETWORKS

Remote Memory ~ Local Memory over Reconfigurable Ethernet Fabric [1]

Vishal Shrivastav

Joint work with Weigao Su

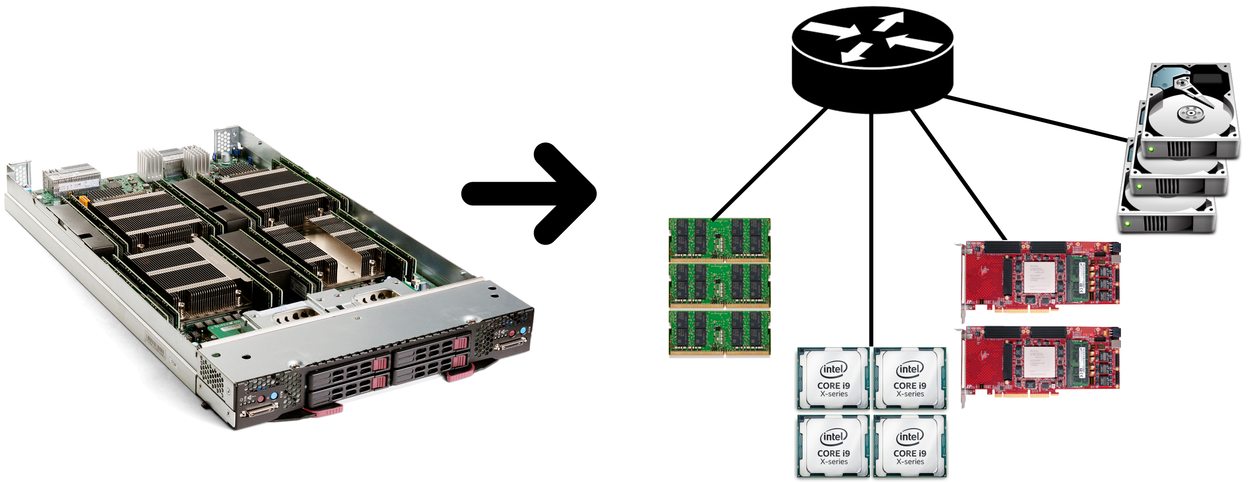
PURDUE UNIVERSITY

Reference:

[1] “EDM: An Ultra-Low Latency Ethernet Fabric for Memory Disaggregation”. *Weigao Su and Vishal Shrivastav*. ASPLOS 2025

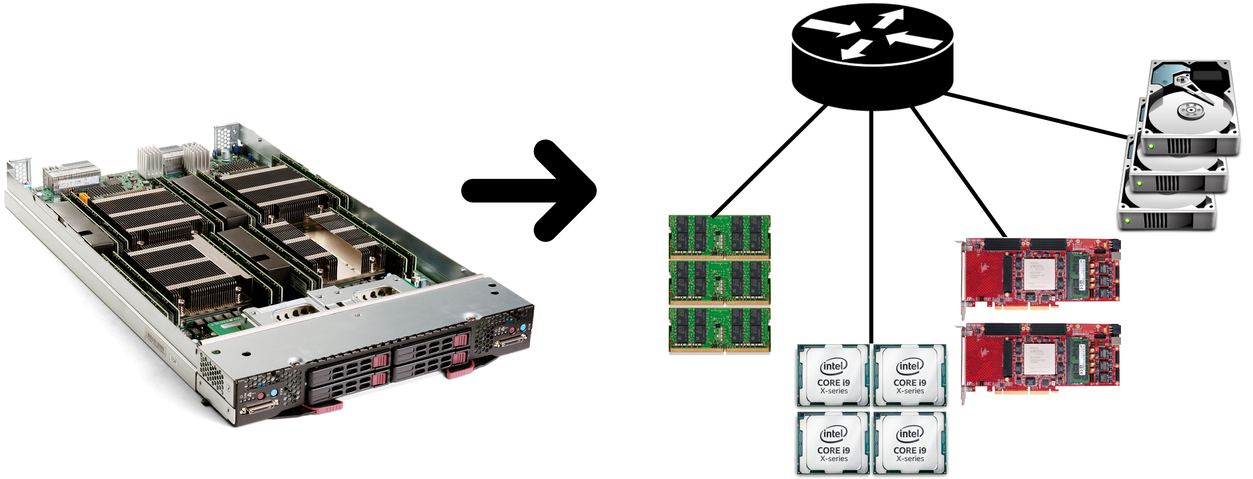
Why memory disaggregation?

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



Why memory disaggregation?

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

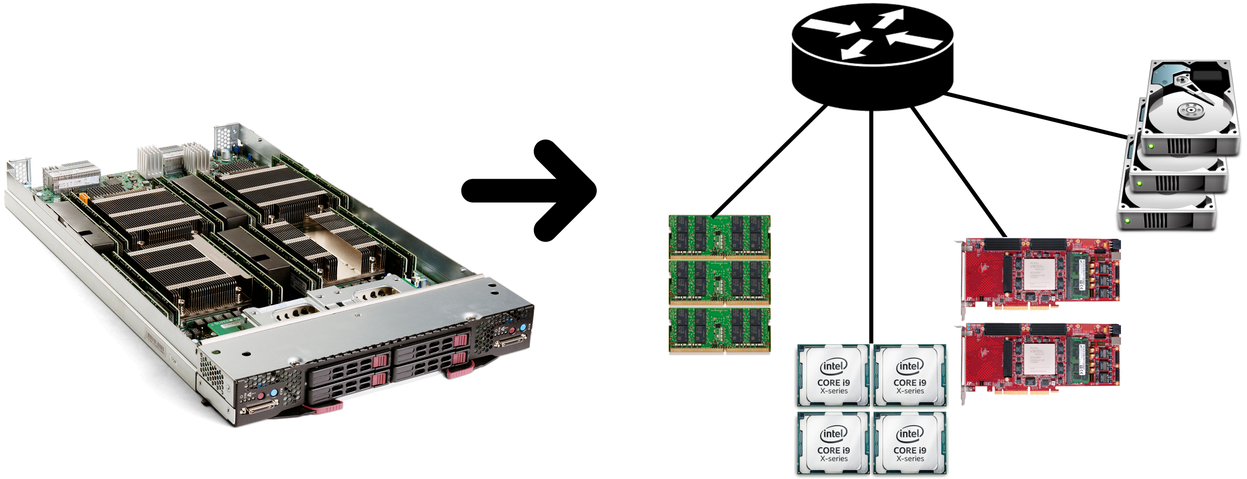


Why is Ethernet promising?

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

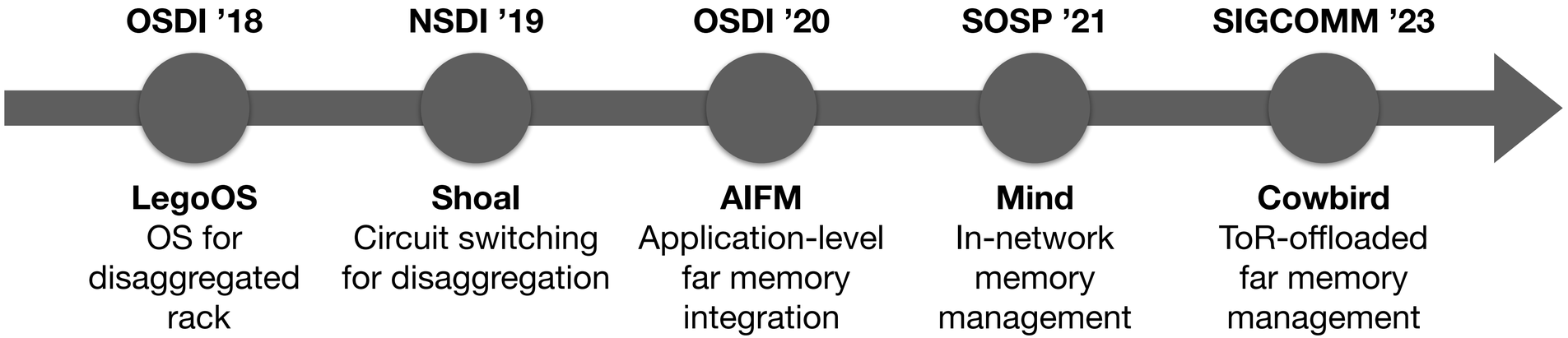
Why memory disaggregation?

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

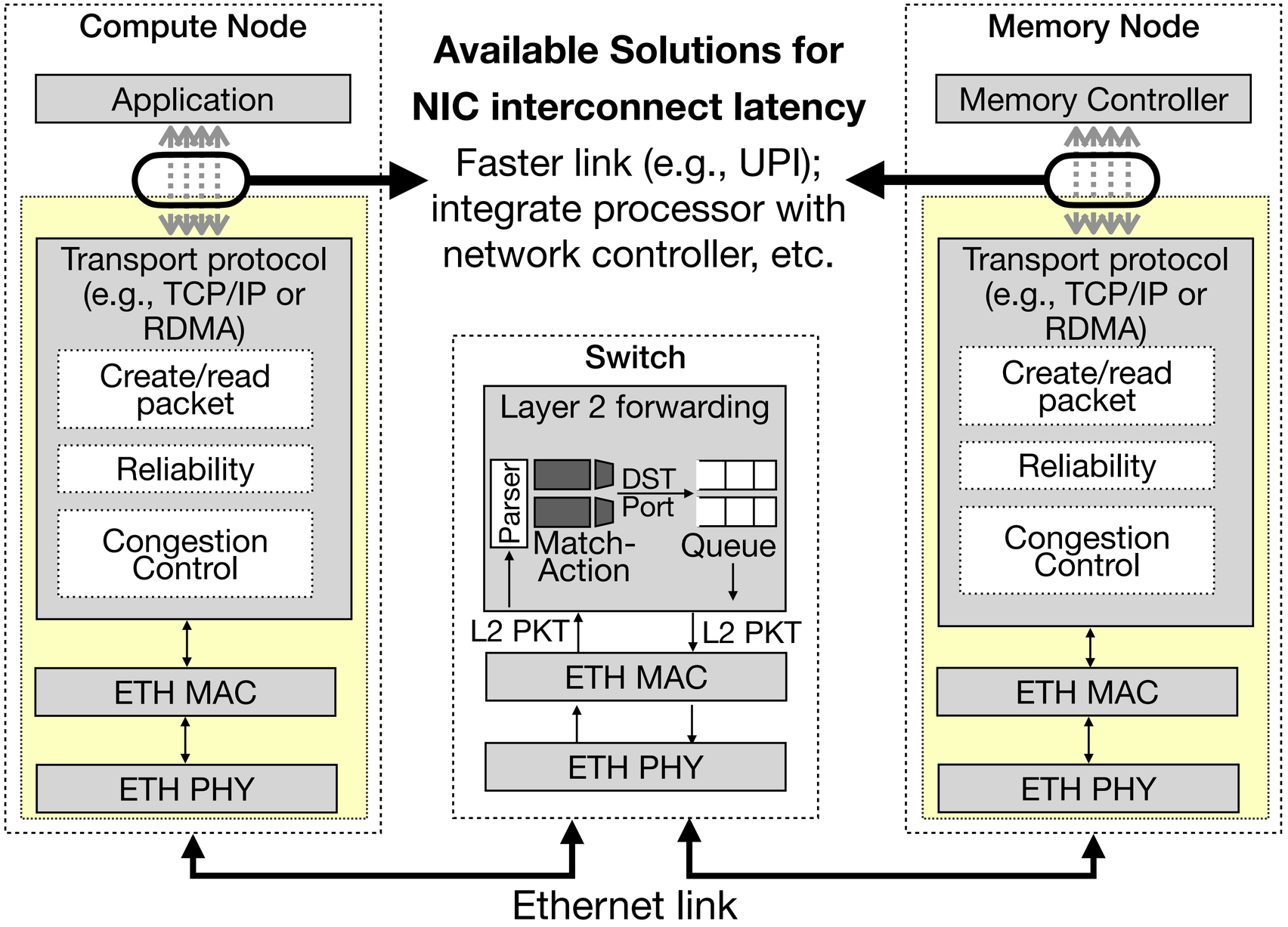


Why is Ethernet promising?

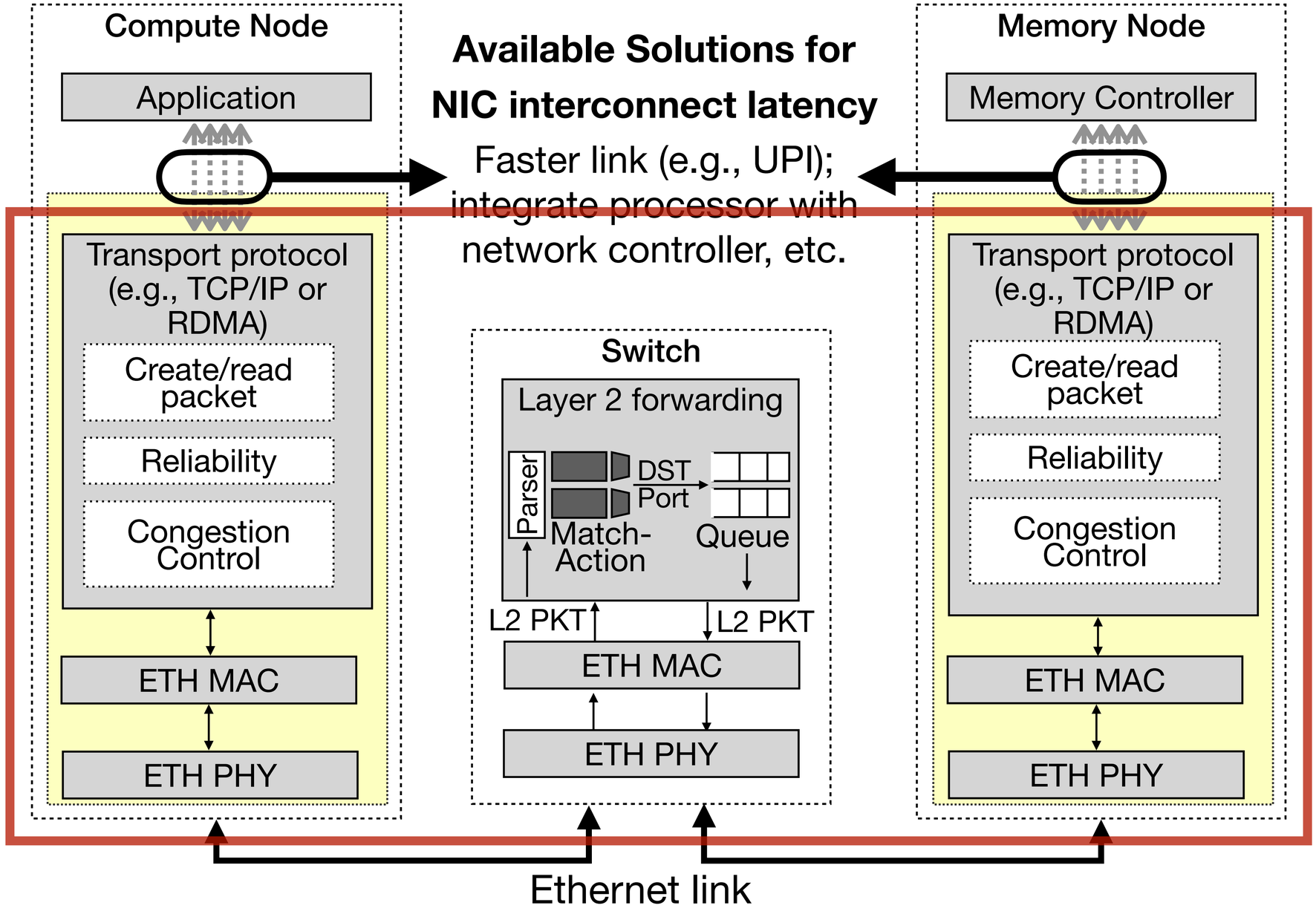
- Dominant datacenter network fabric
  - Low management cost, distance scaling…
- High bandwidth (Terabit Ethernet link)



Memory Disaggregation over Ethernet

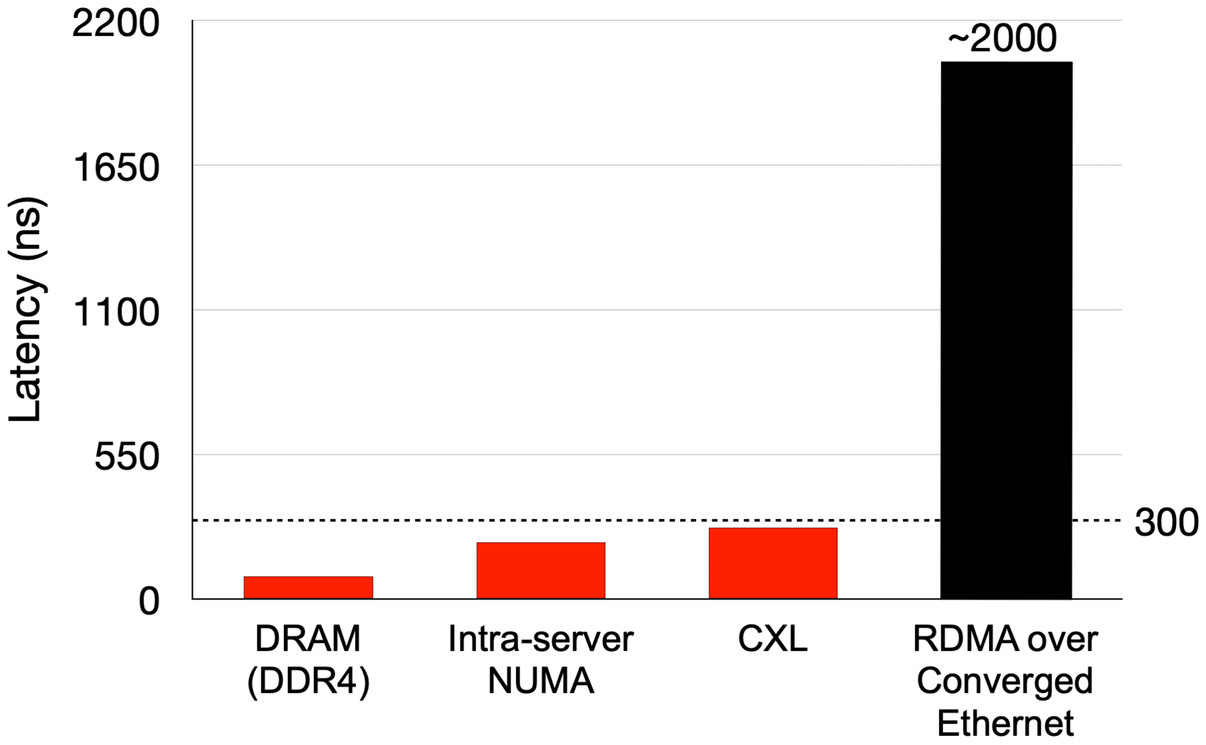


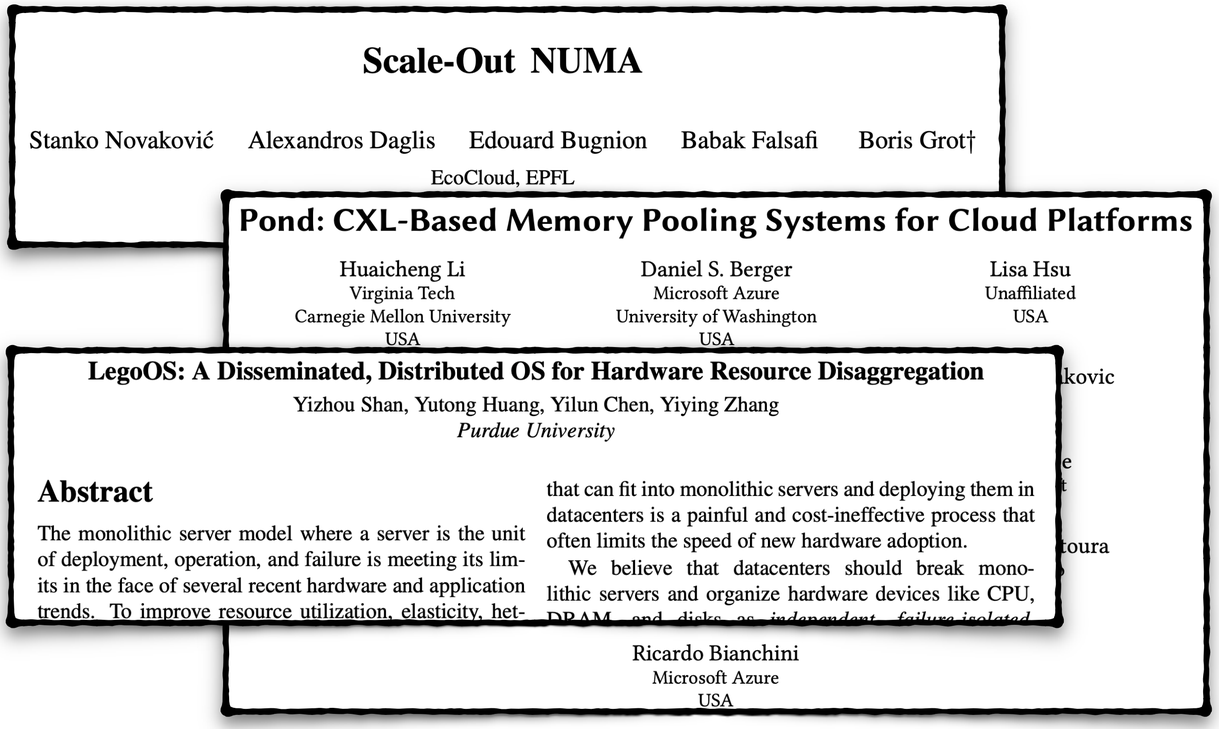
Memory Disaggregation over Ethernet



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

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-

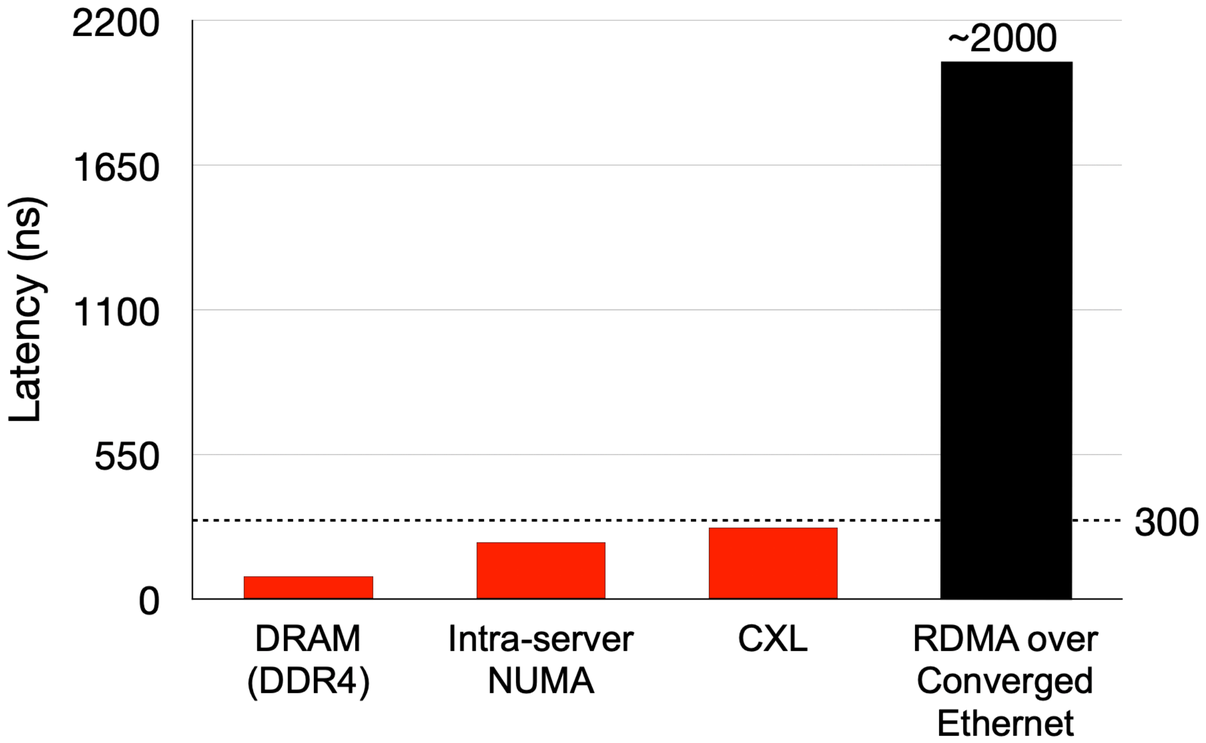
that can fit into monolithic servers and deploying them in datacenters is a painful and cost-ineffective process that often limits the speed of new hardware adoption.

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

Ricardo Bianchini
Microsoft Azure
USA

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

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*

Ricardo Bianchini
Microsoft Azure
USA

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.

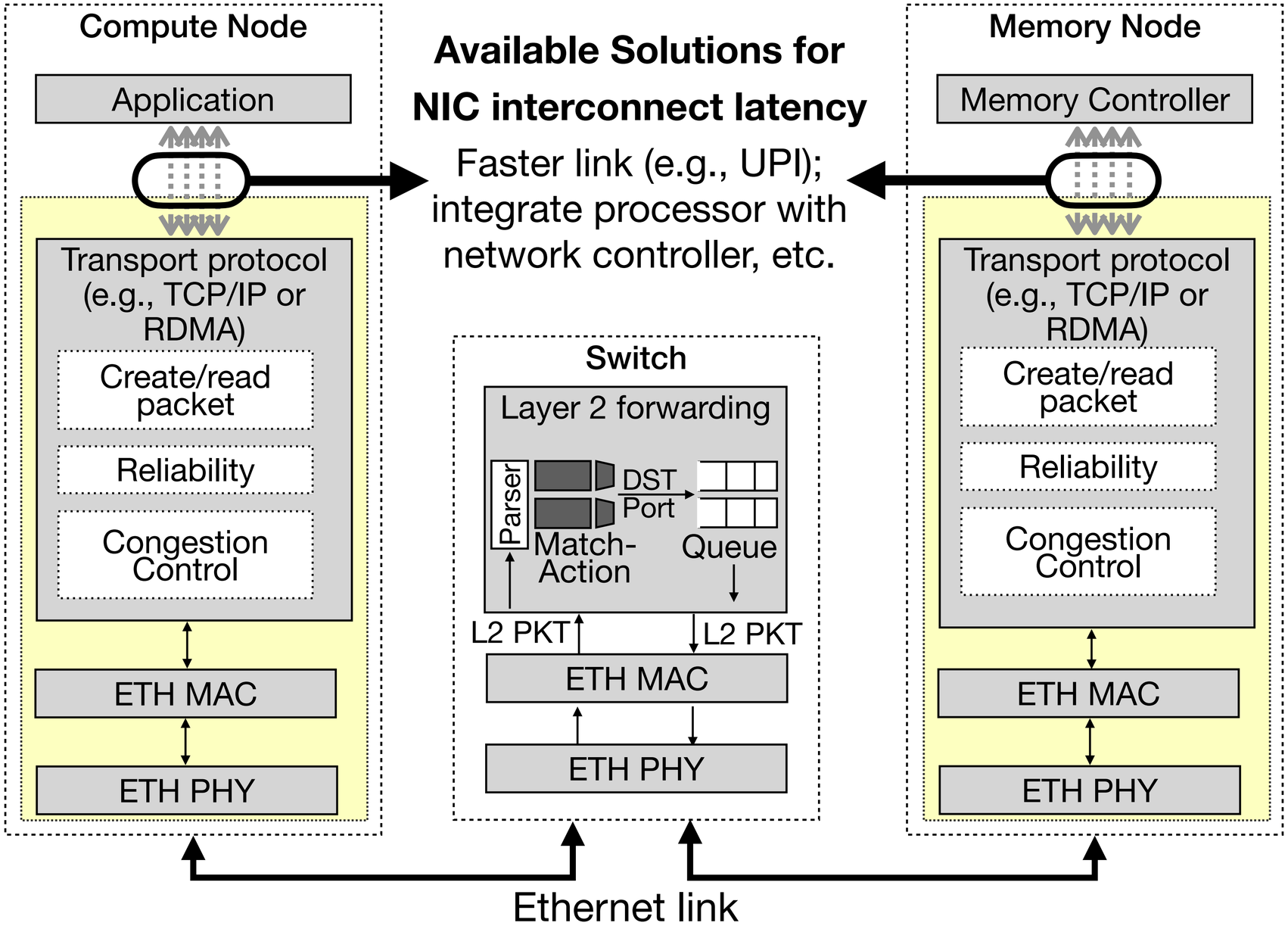
We believe that datacenters should break monolithic servers and organize hardware devices like CPU, DRAM, and disks as *independent, failure isolated*

toura

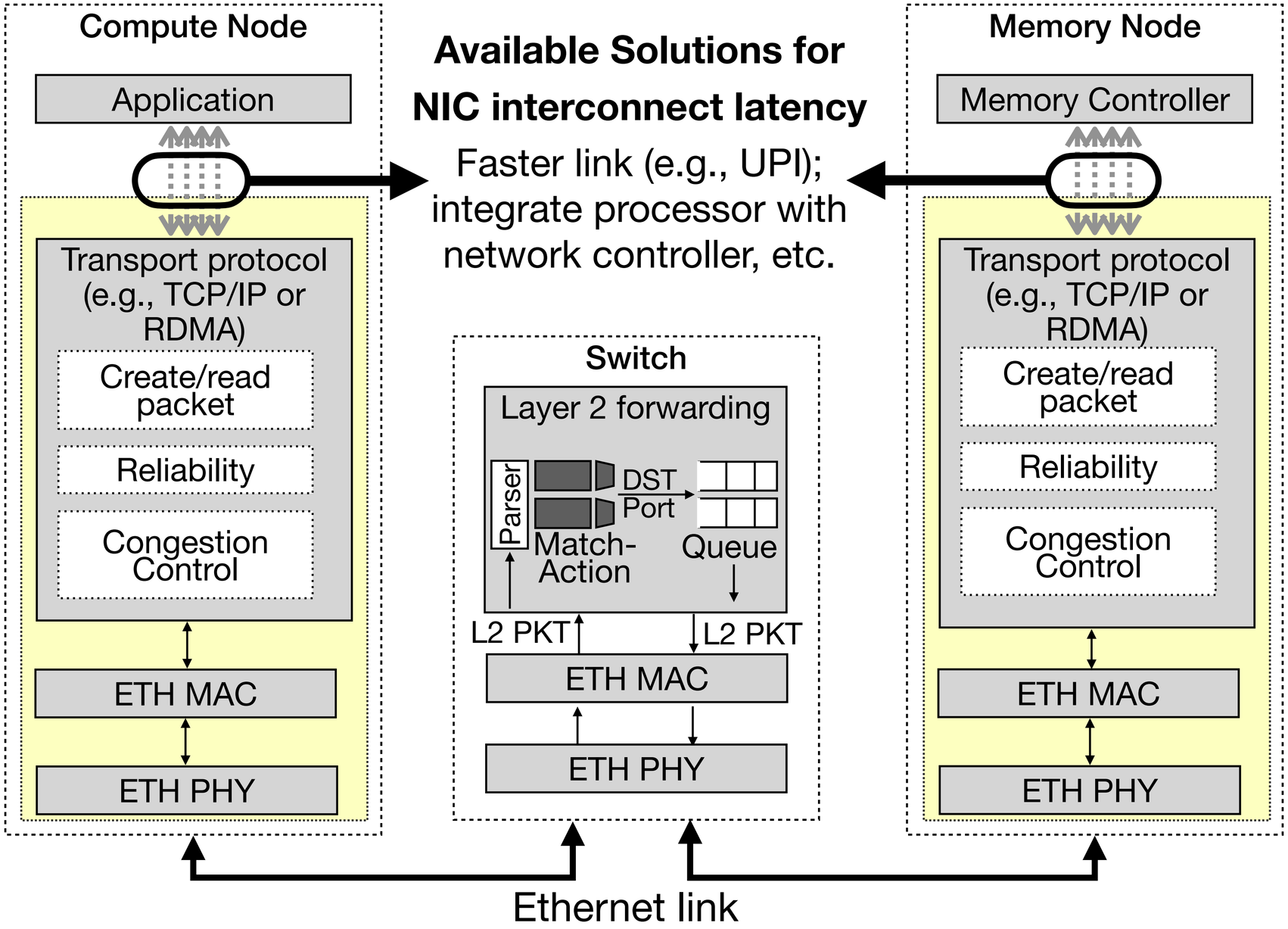
Ricardo Bianchini
Microsoft Azure
USA

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

Memory Disaggregation over Ethernet



Memory Disaggregation over Ethernet



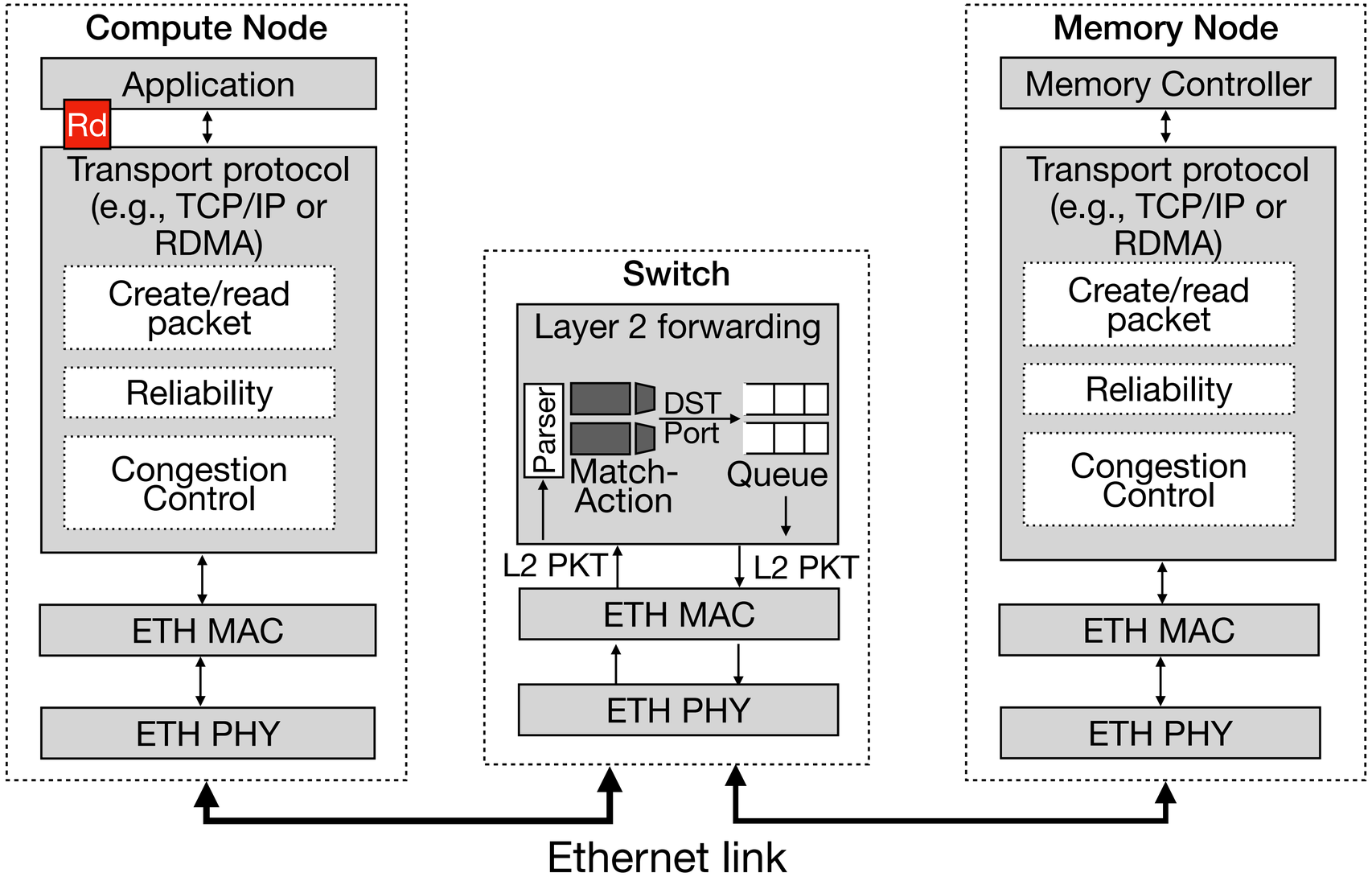
Research goal

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

(while maintaining high bandwidth utilization)

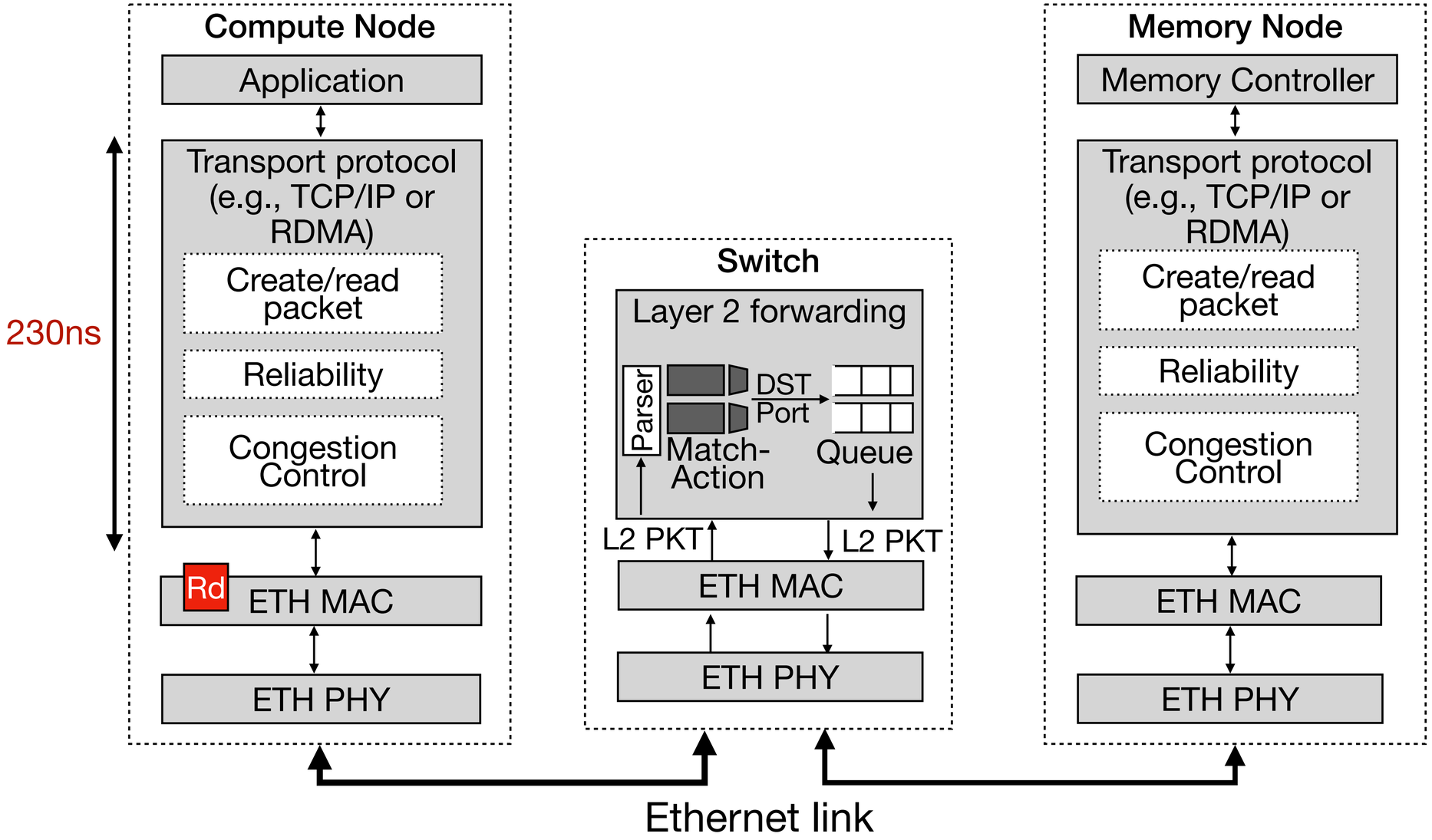
Latency Overheads of Existing Memory Disaggregation over Ethernet

An example of remote read request over RDMA



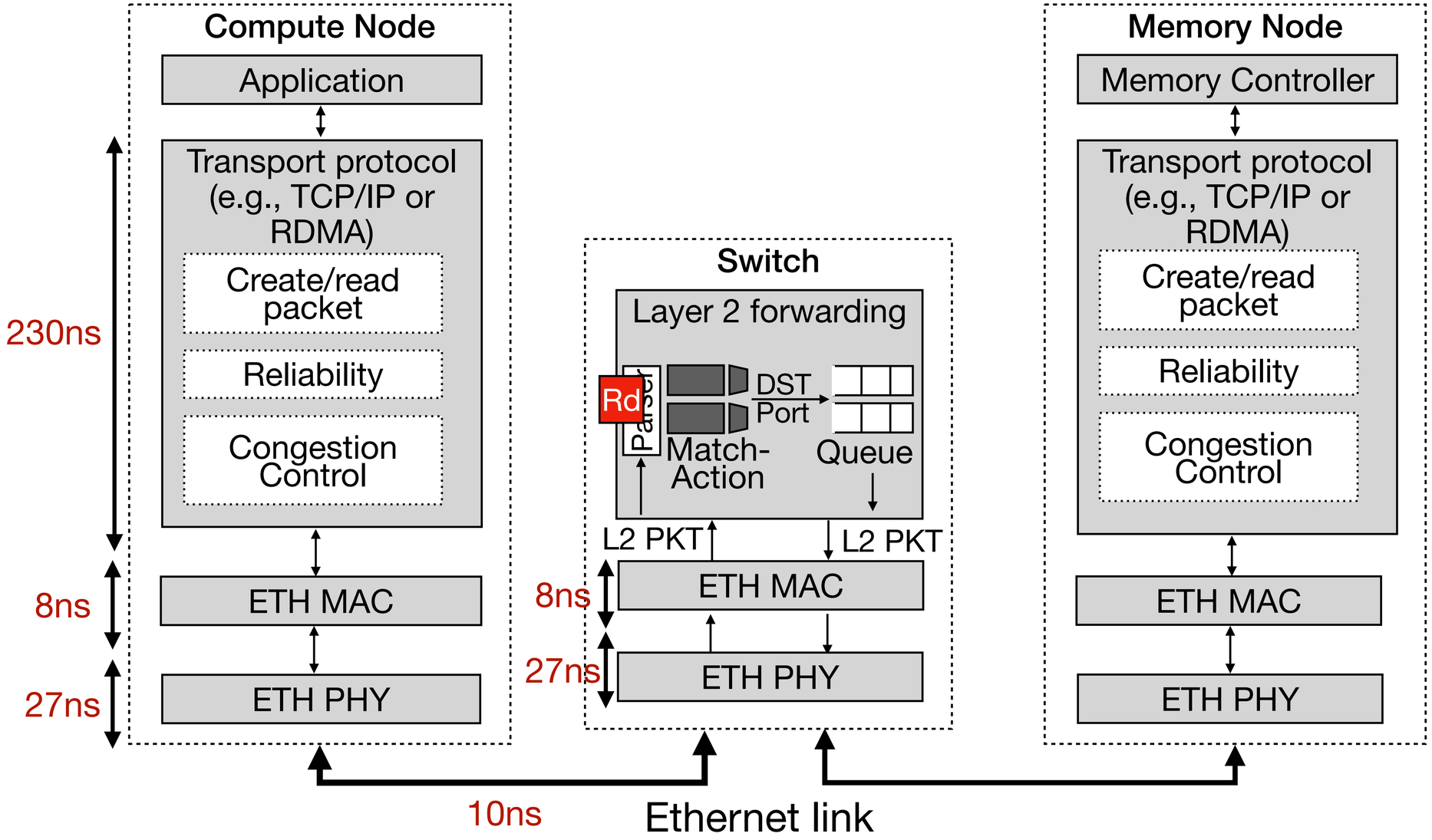
Latency Overheads of Existing Memory Disaggregation over Ethernet

An example of remote read request over RDMA



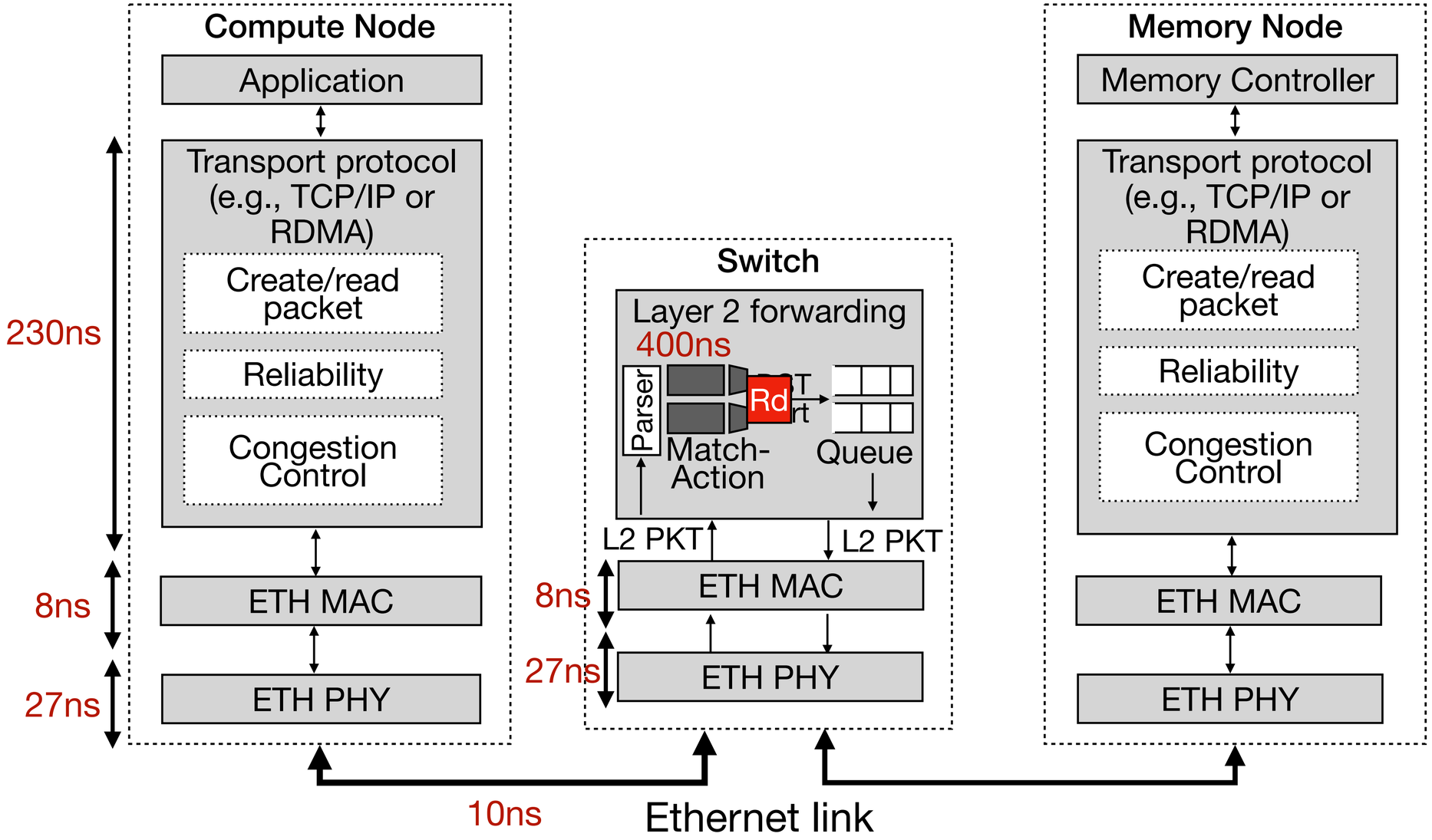
Latency Overheads of Existing Memory Disaggregation over Ethernet

An example of remote read request over RDMA



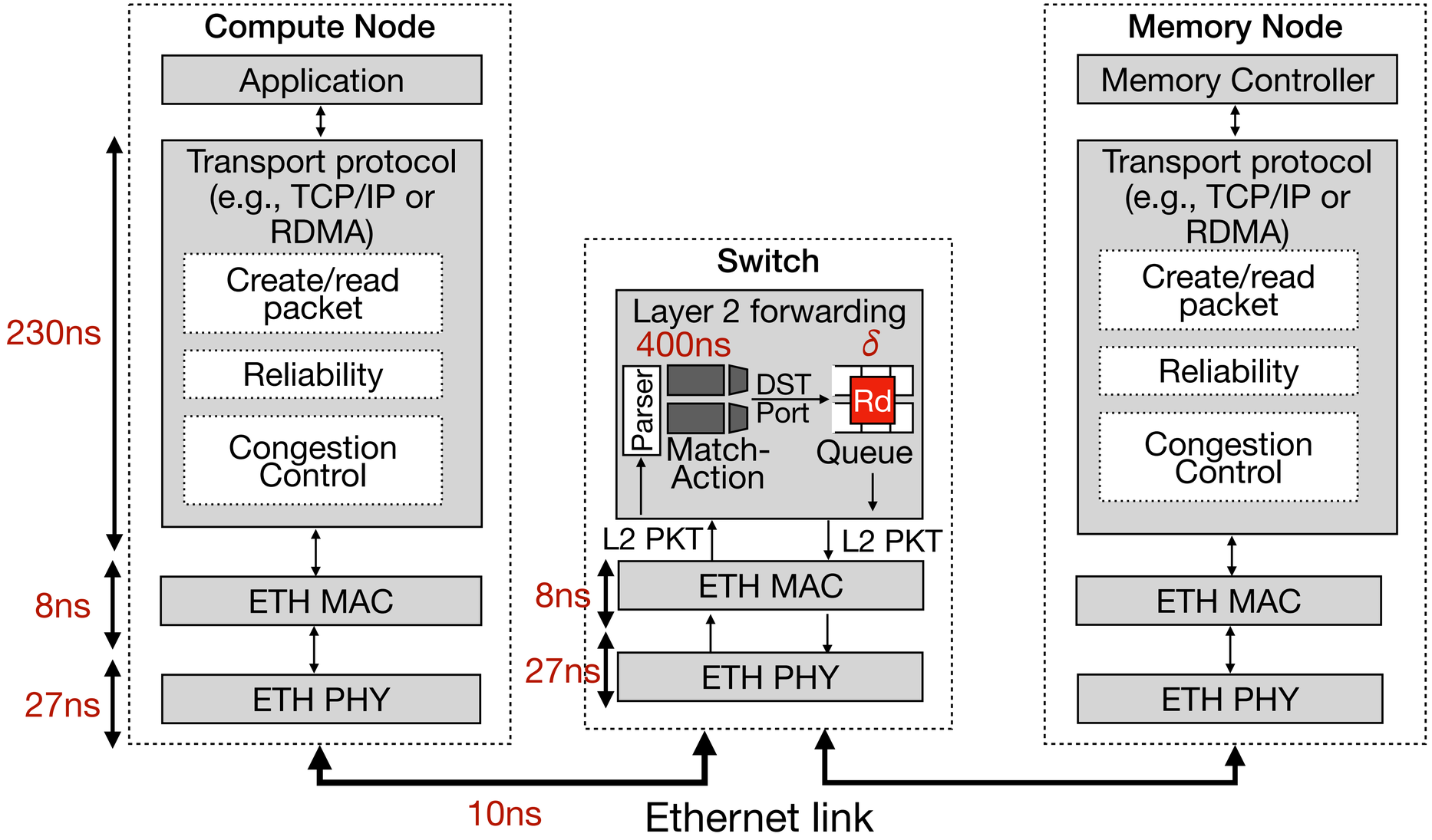
Latency Overheads of Existing Memory Disaggregation over Ethernet

An example of remote read request over RDMA



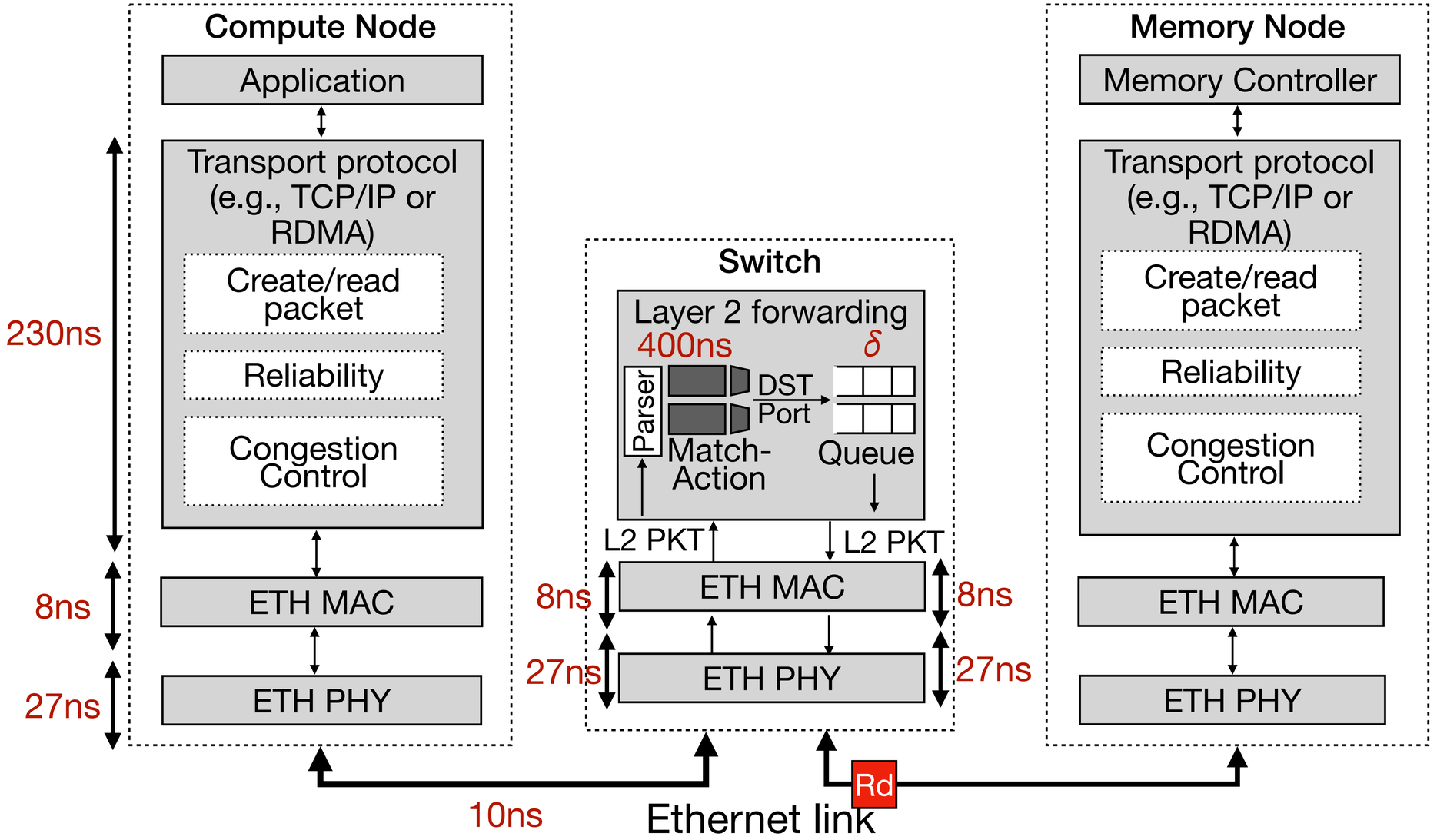
Latency Overheads of Existing Memory Disaggregation over Ethernet

An example of remote read request over RDMA



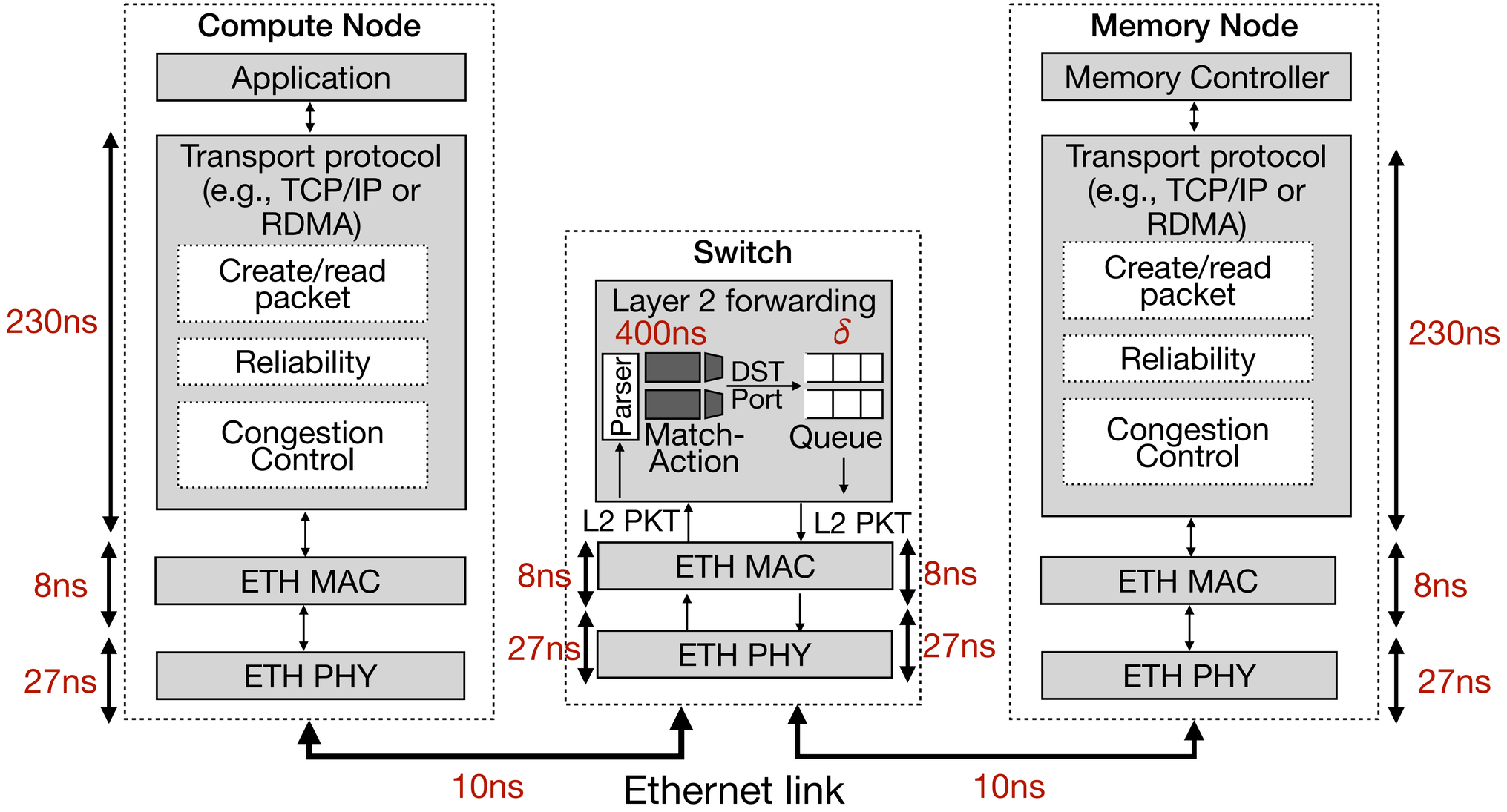
Latency Overheads of Existing Memory Disaggregation over Ethernet

An example of remote read request over RDMA



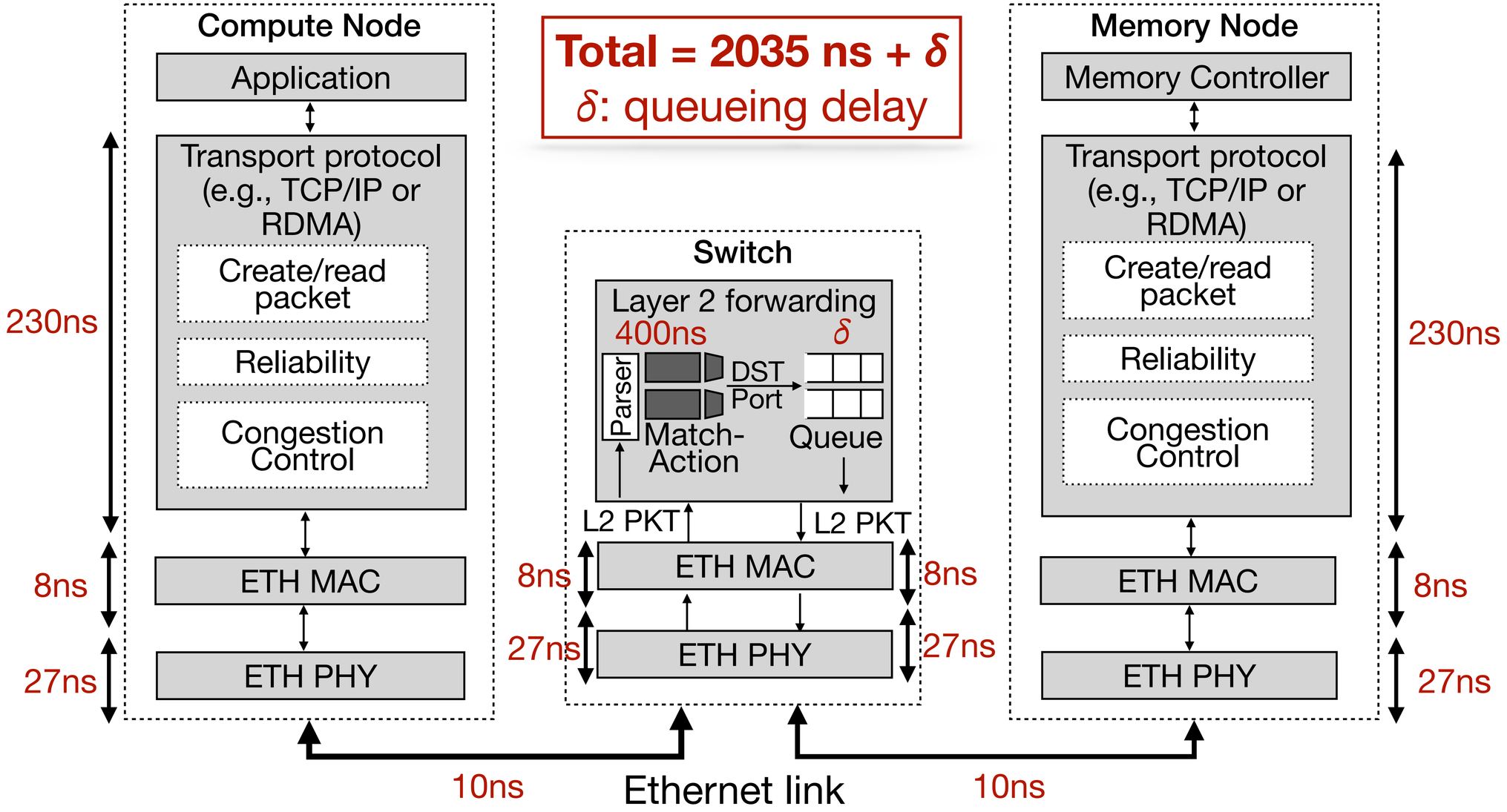
Latency Overheads of Existing Memory Disaggregation over Ethernet

An example of remote read request over RDMA

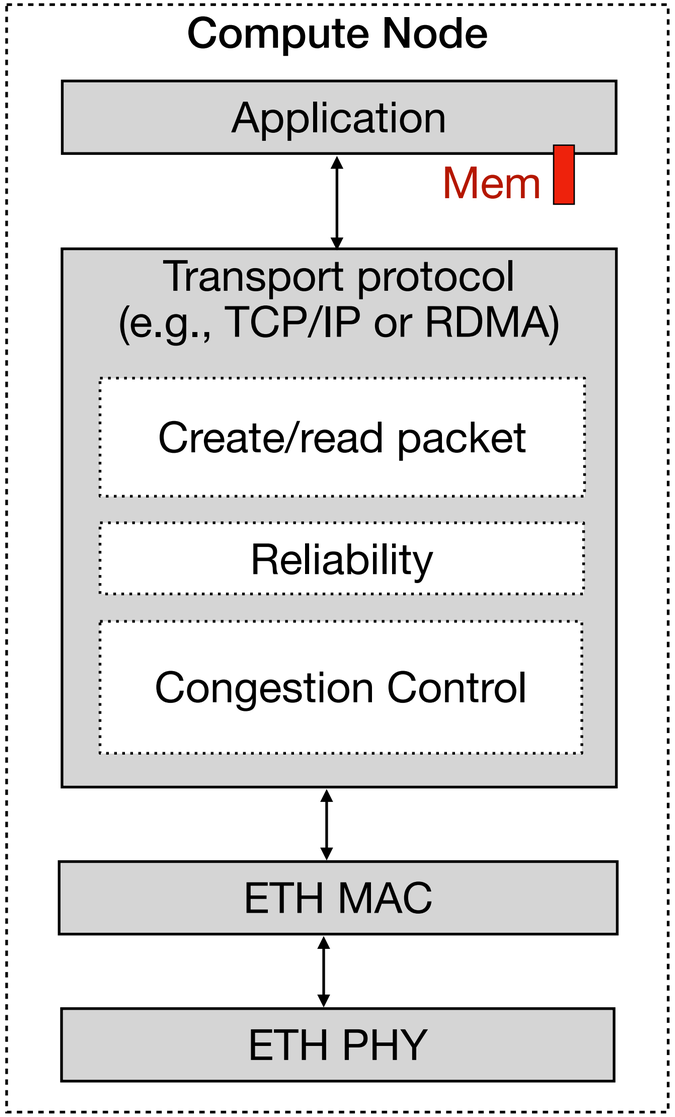


Latency Overheads of Existing Memory Disaggregation over Ethernet

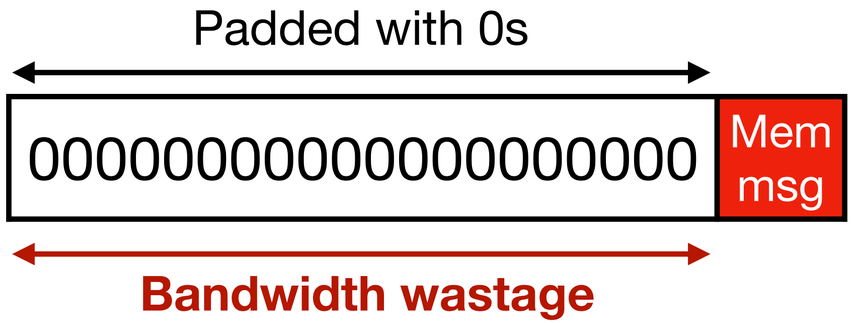
An example of remote read request over RDMA



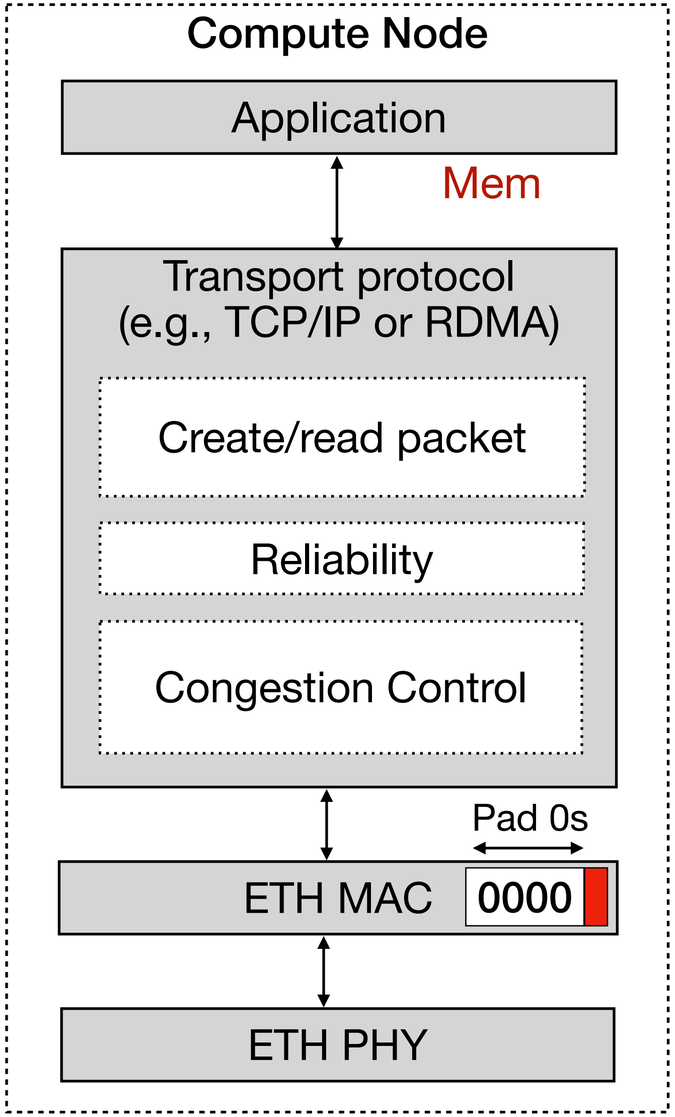
Other Overheads



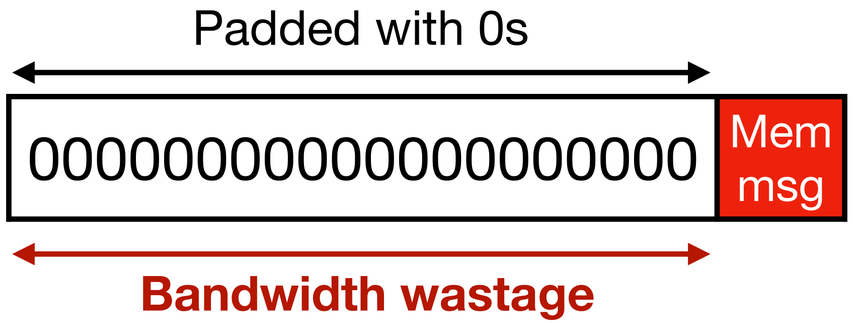
1. **Ethernet MAC enforces minimum 64B frame**
   … but memory messages can be much smaller (e.g., read requests are typically 8-16B)



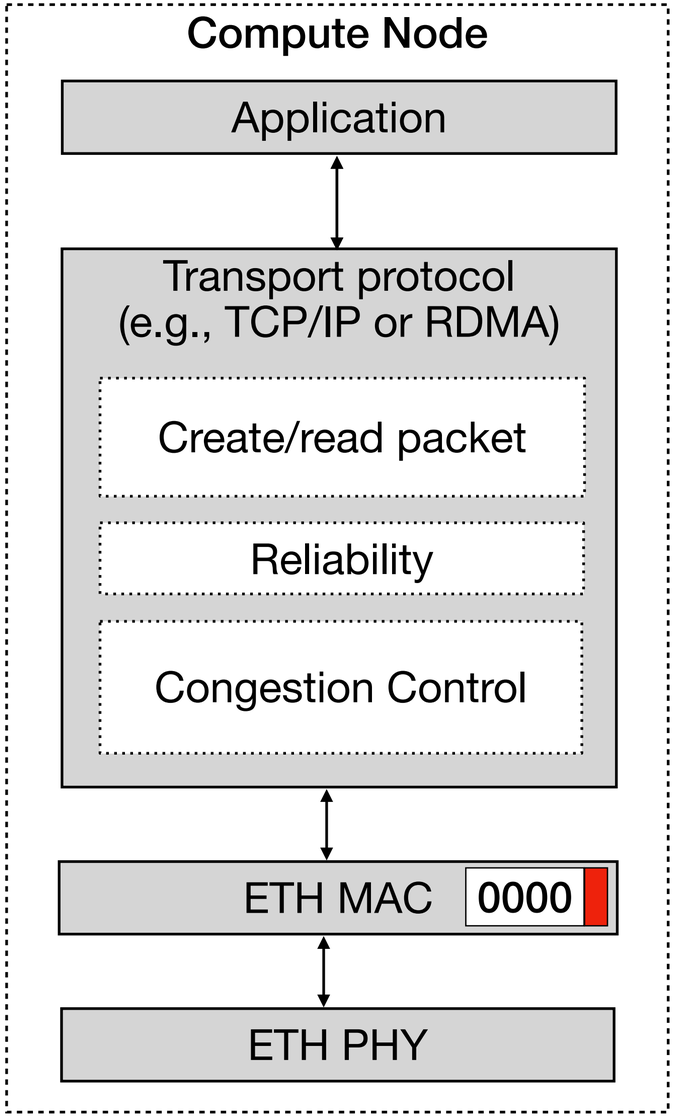
Other Overheads



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



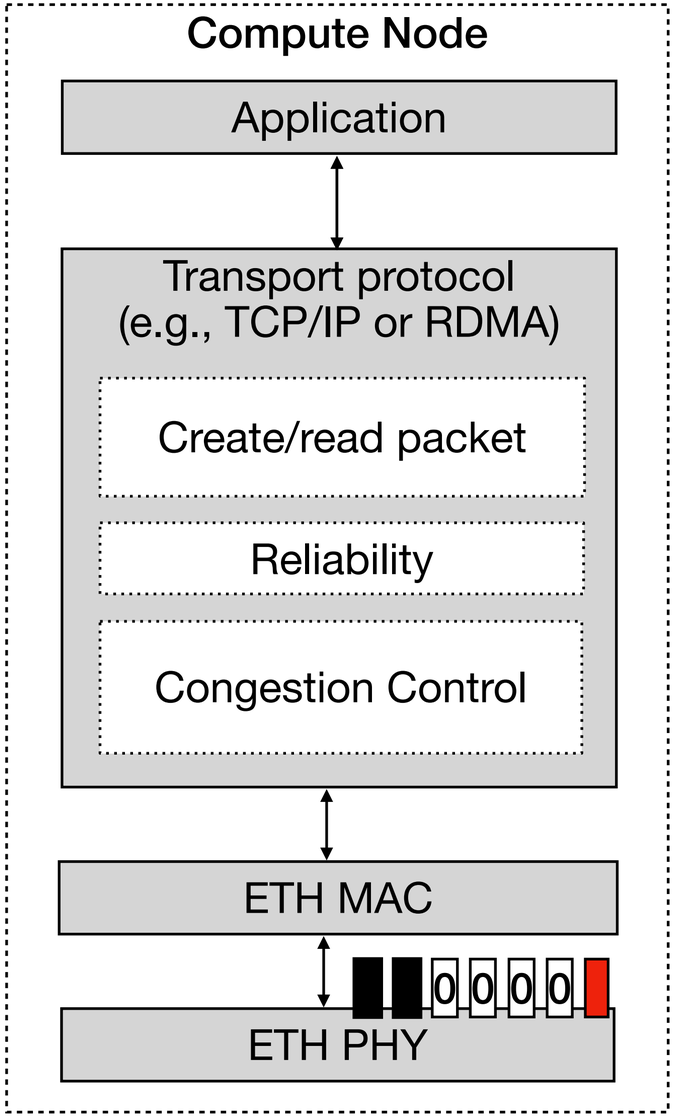
Other Overheads



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

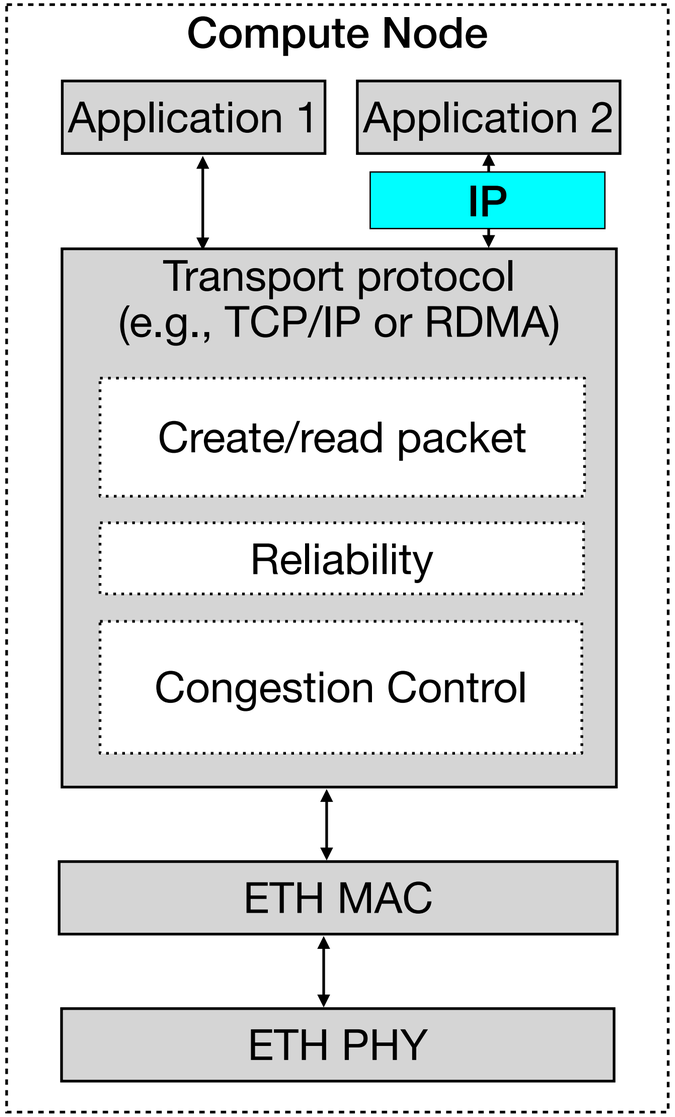
Other Overheads



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

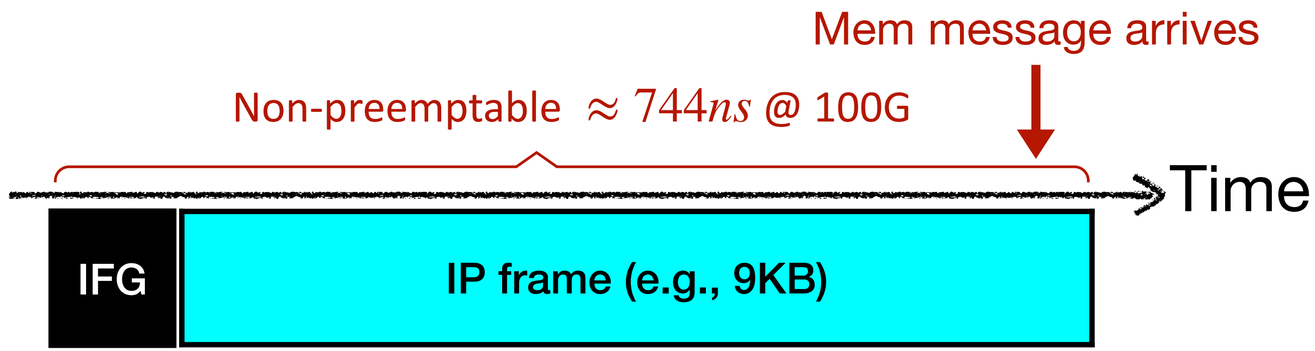
Other Overheads



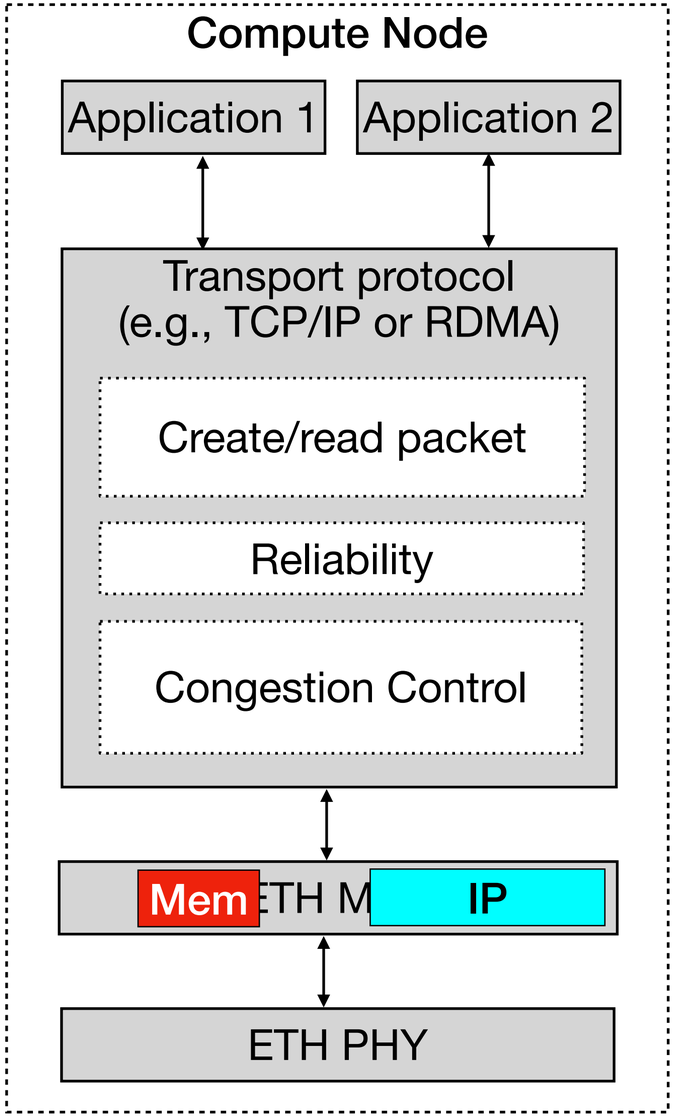
1. **Ethernet MAC enforces minimum 64B frame**
   … but memory messages can be much smaller (e.g., read requests are typically 8-16B)

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



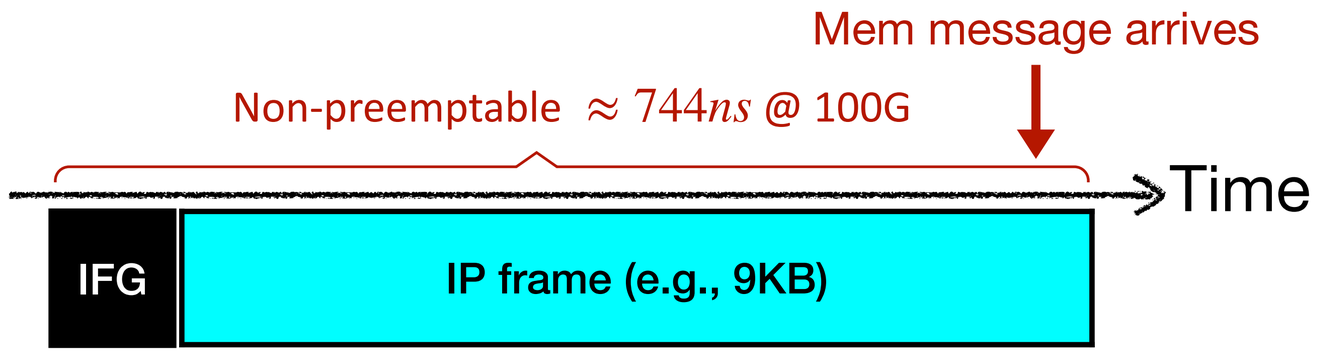
Other Overheads



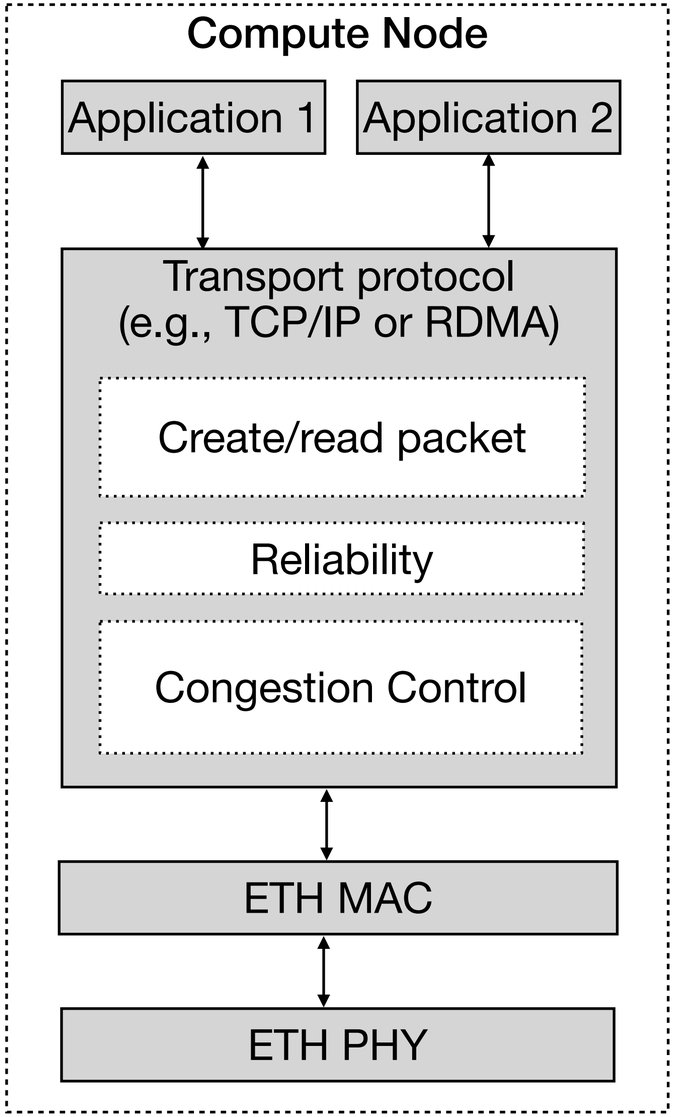
1. **Ethernet MAC enforces minimum 64B frame**
   … but memory messages can be much smaller (e.g., read requests are typically 8-16B)

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



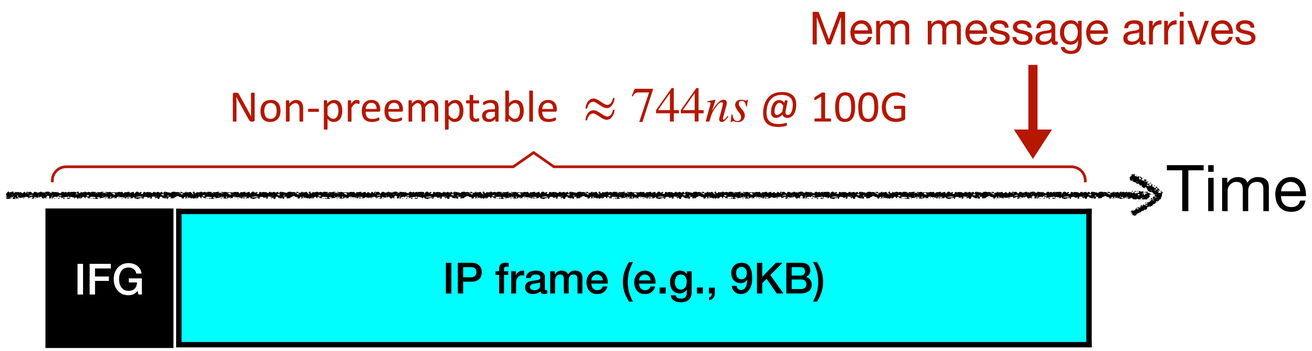
Other Overheads



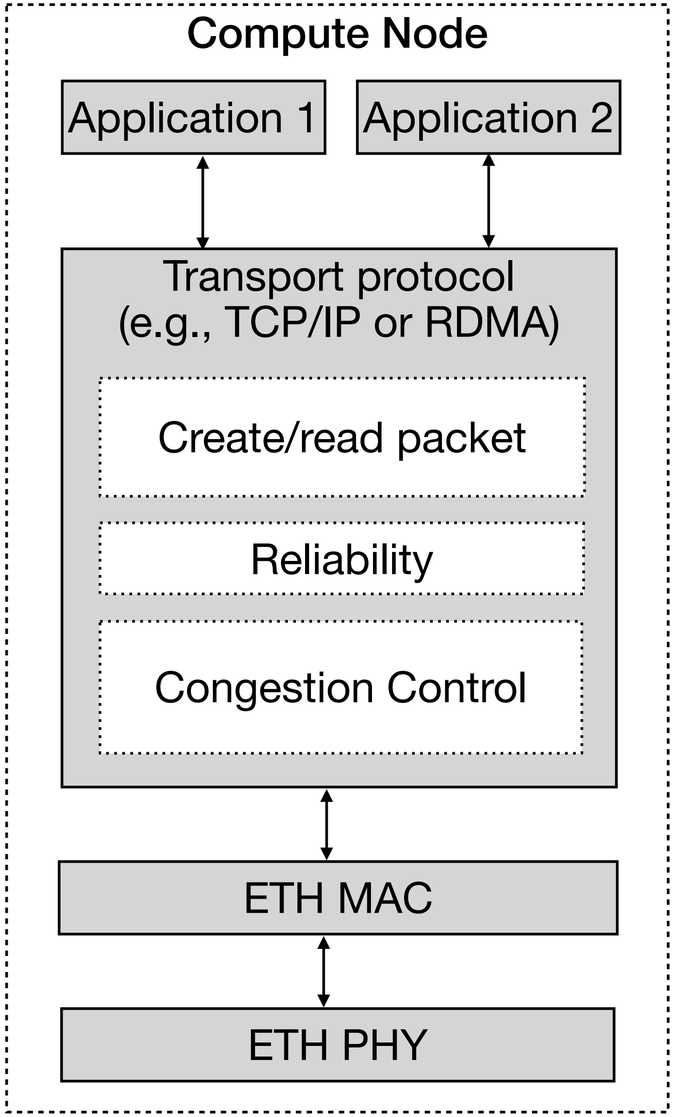
1. **Ethernet MAC enforces minimum 64B frame**
   … but memory messages can be much smaller (e.g., read requests are typically 8-16B)

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

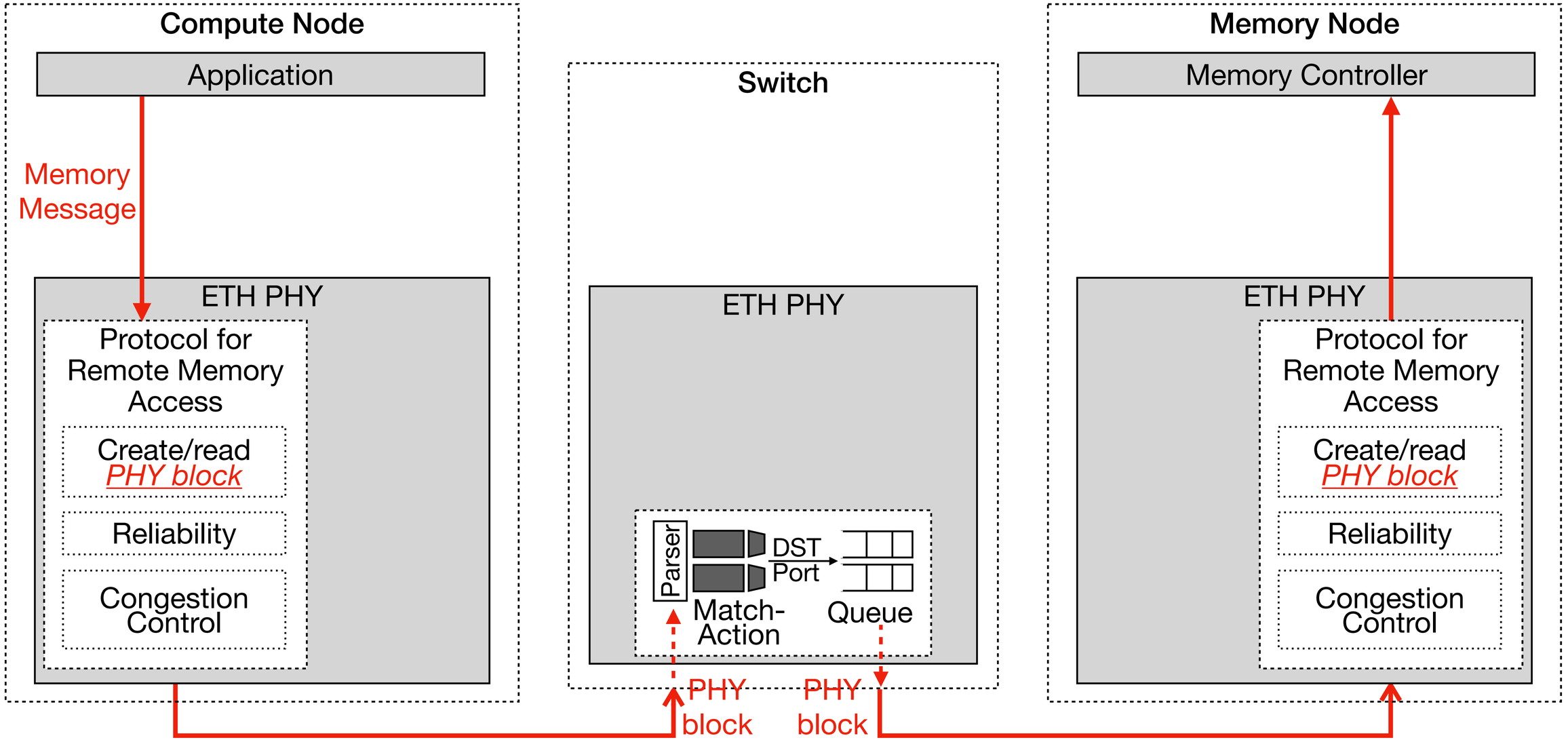
3. **Ethernet MAC does not allow intra-frame preemption**
   … a large non-memory frame may block the transmission of a small memory message

Root cause: MAC layer processing

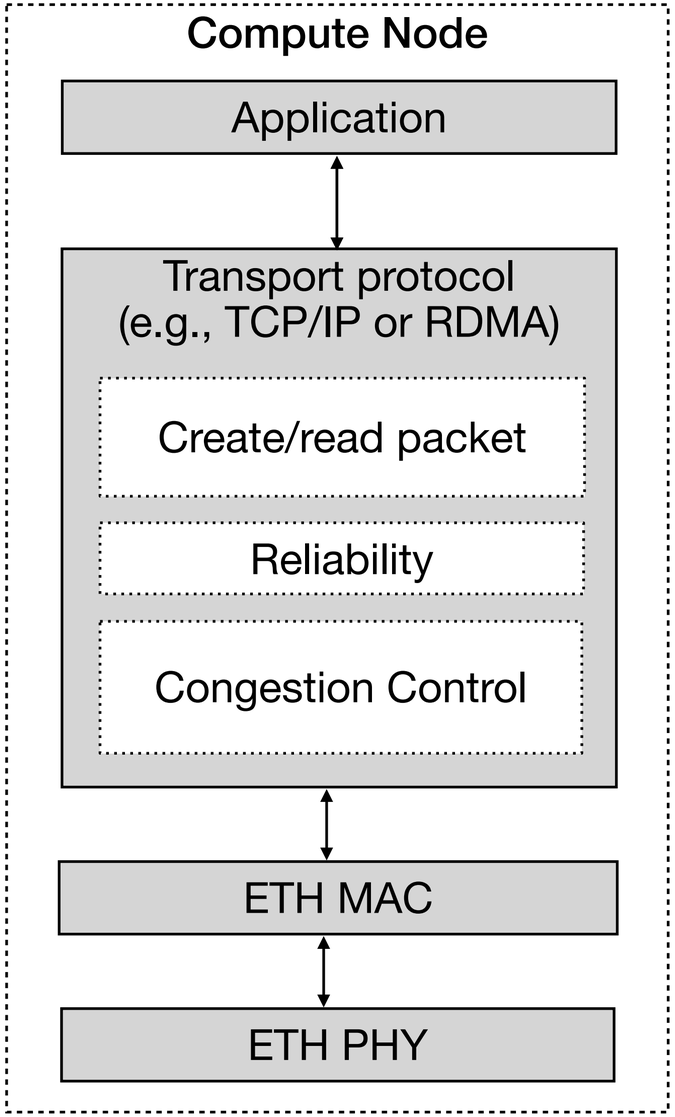
Design Choice # 1:

Implement the entire
protocol for remote memory access
within Ethernet's **Physical Layer (PHY)**

Architecture of Remote Memory Protocol in the PHY



Rationale for Remote Memory Protocol in PHY

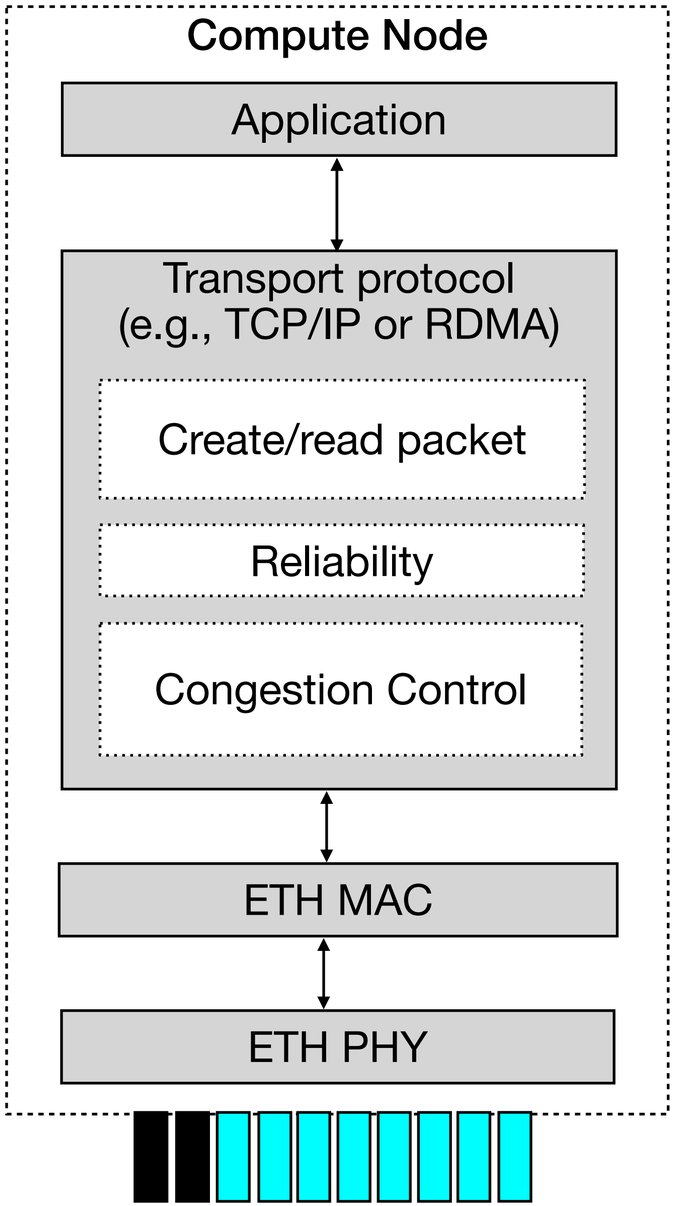


Ethernet PHY already reformats a MAC layer frame into a series of 66-bit PHY blocks

… 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
- Message interleaving can be done at block granularity in PHY rather than at frame granularity in MAC
- PHY also has access to IFG blocks

Rationale for Remote Memory Protocol in PHY

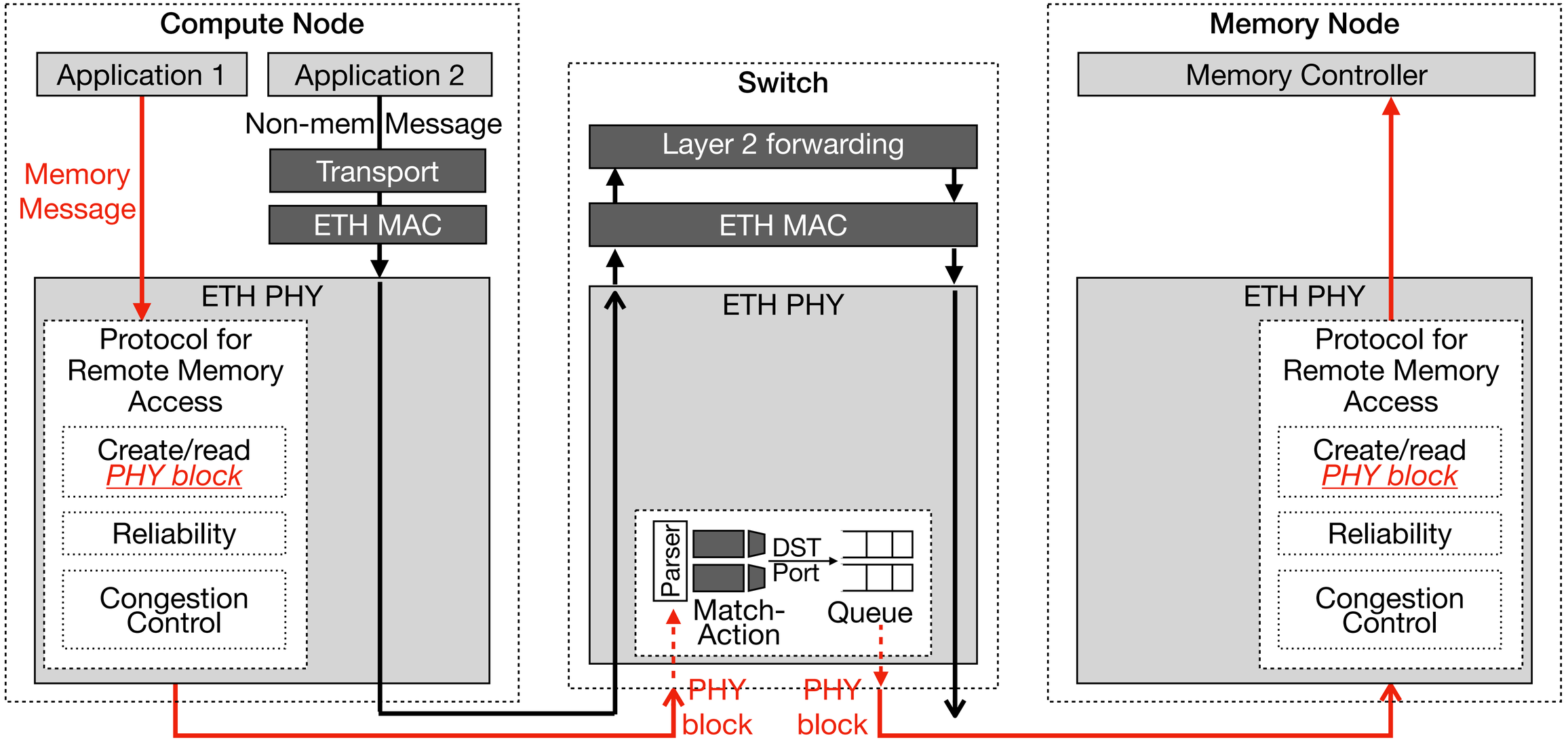


Ethernet PHY already reformats a MAC layer frame into a series of 66-bit PHY blocks

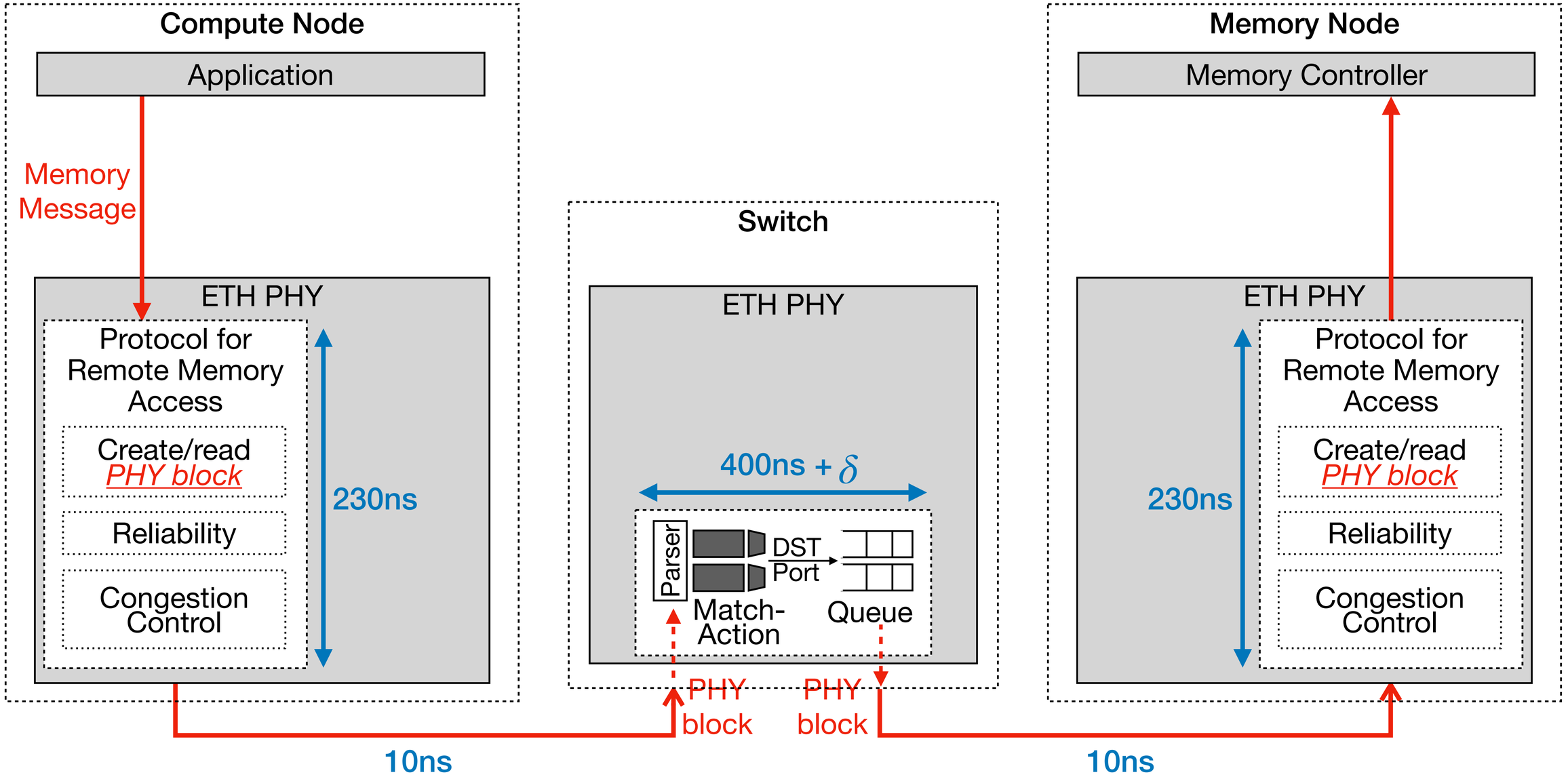
… thus, unlike the MAC layer that works at a **frame** granularity, PHY works at fine-grained **block** granularity

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

Architecture of Remote Memory Protocol in the PHY



Remote Memory Protocol in the PHY : What about latency?

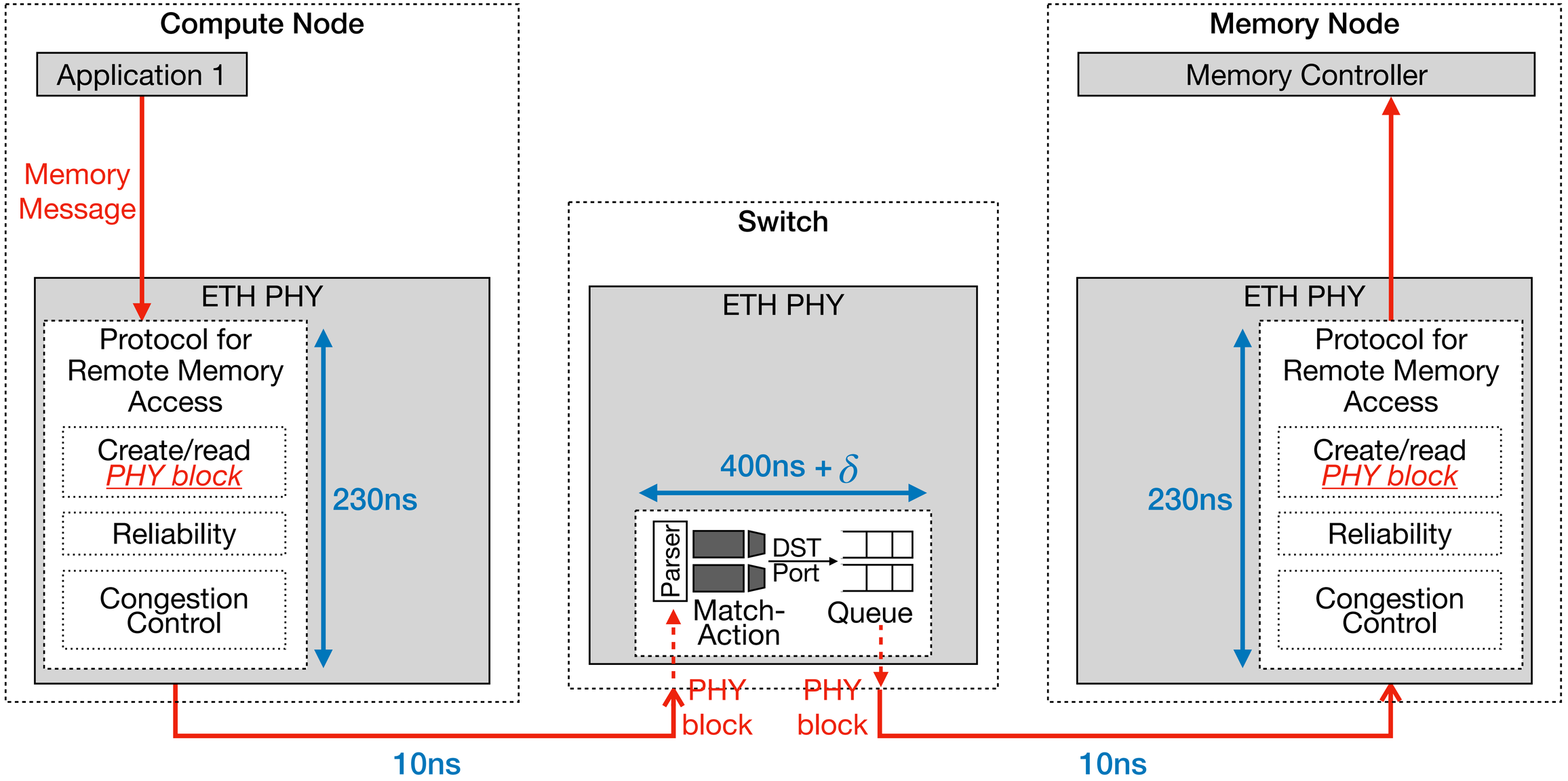


Design Choice # 2:

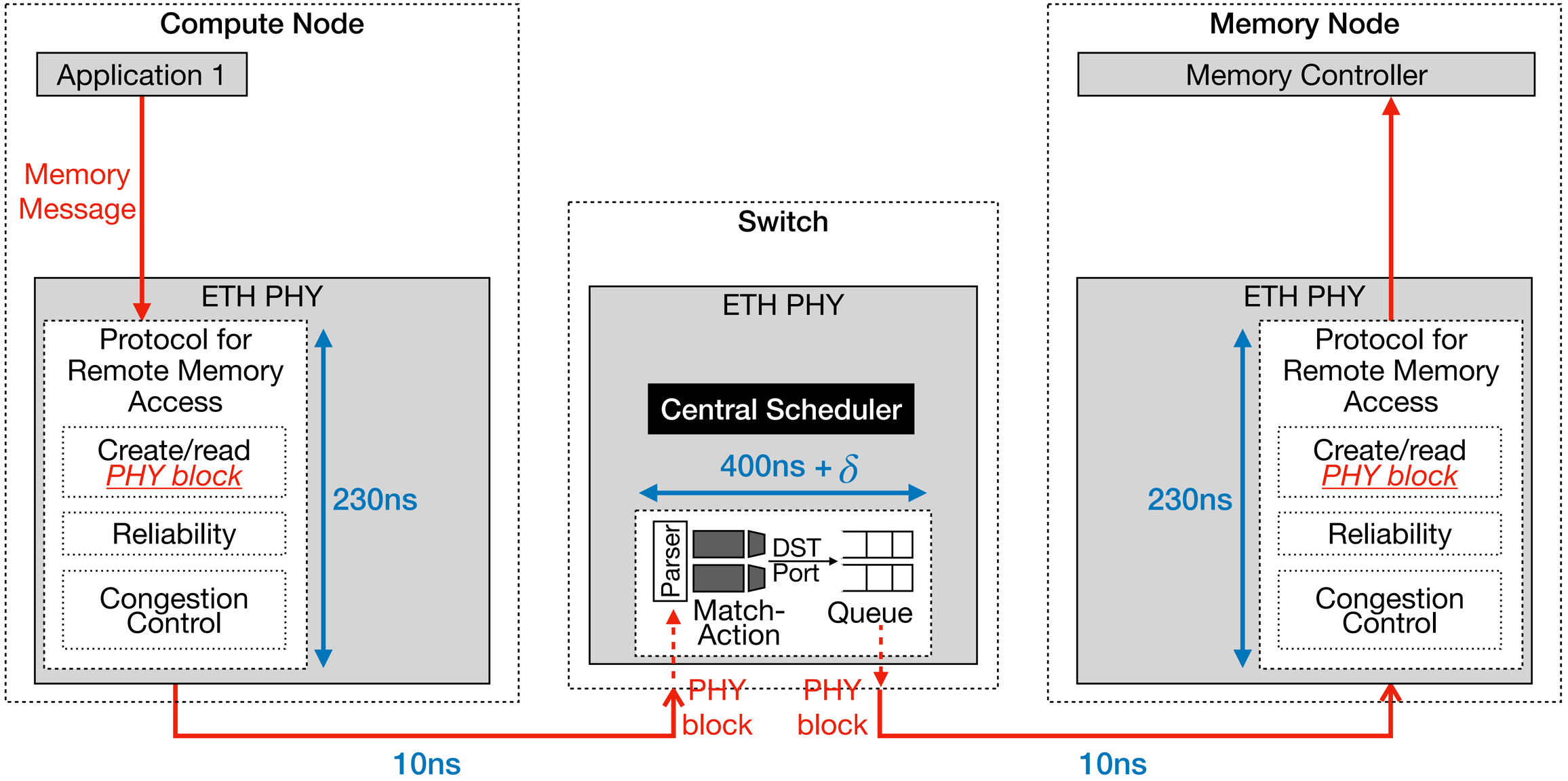
Packet Switching ➡ Reconfigurable (Circuit) Switching

Using a centralized memory traffic scheduler implemented in the PHY of the Ethernet switch

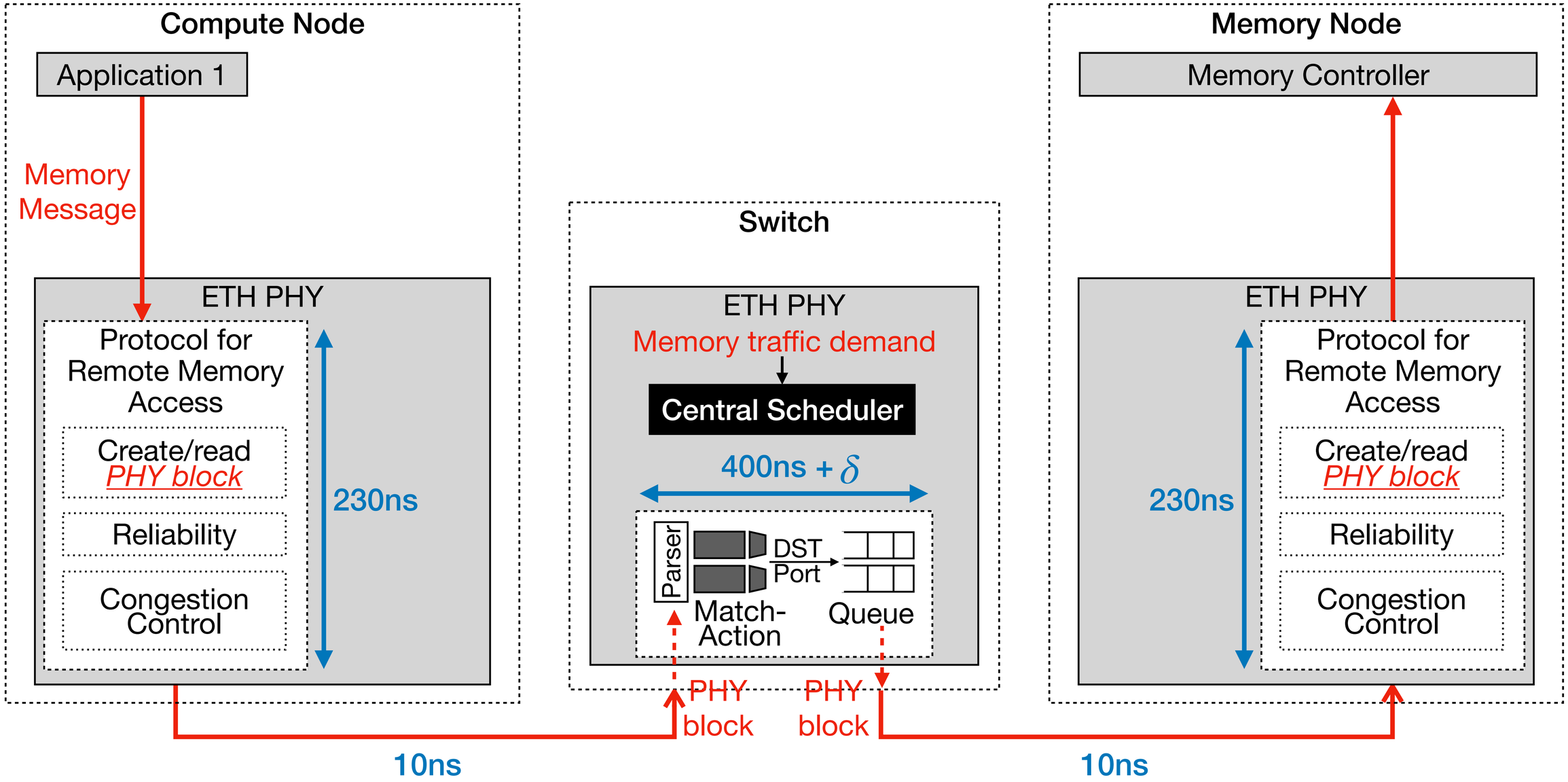
Central Scheduler in the Switch PHY



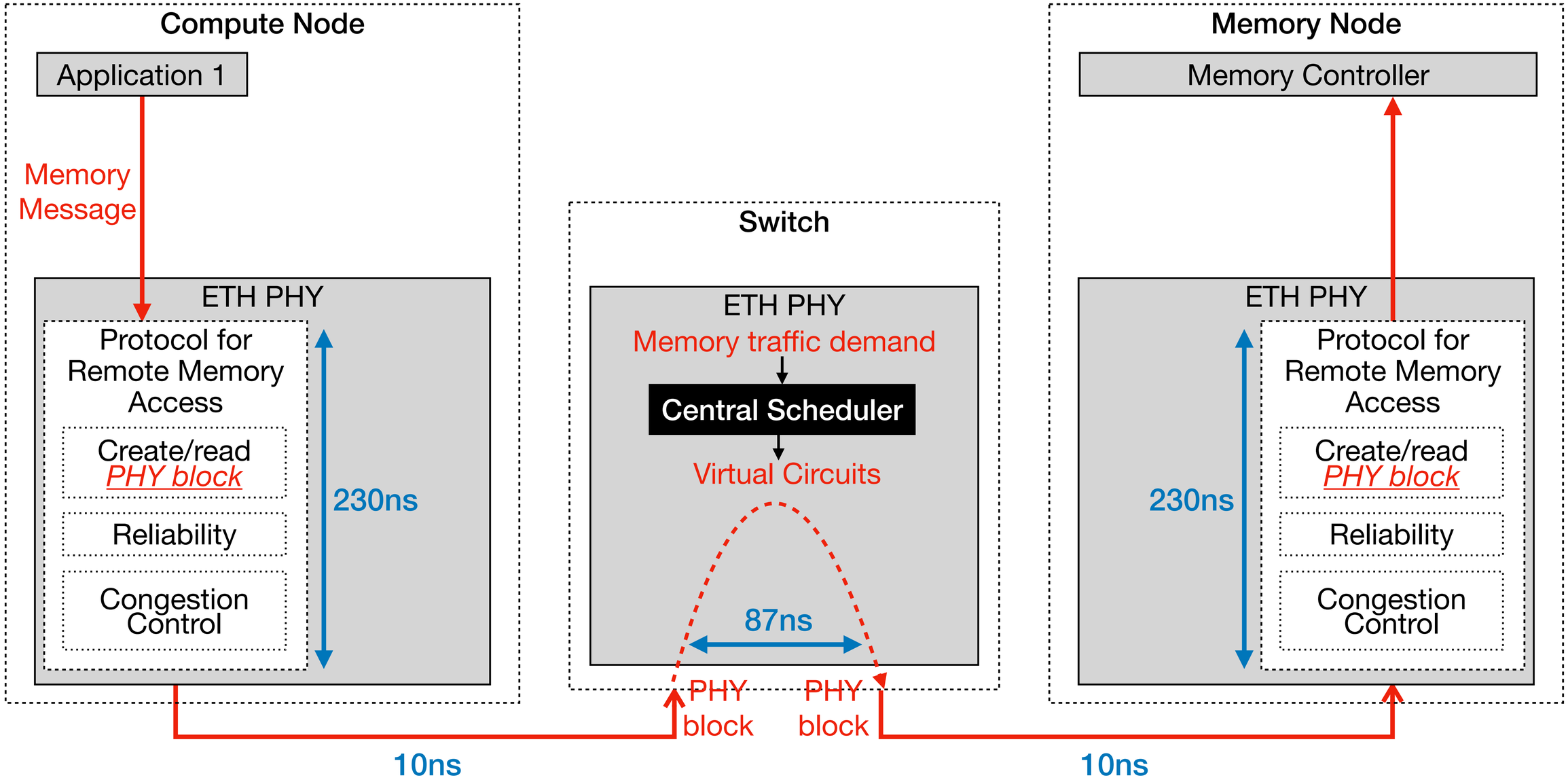
Central Scheduler in the Switch PHY



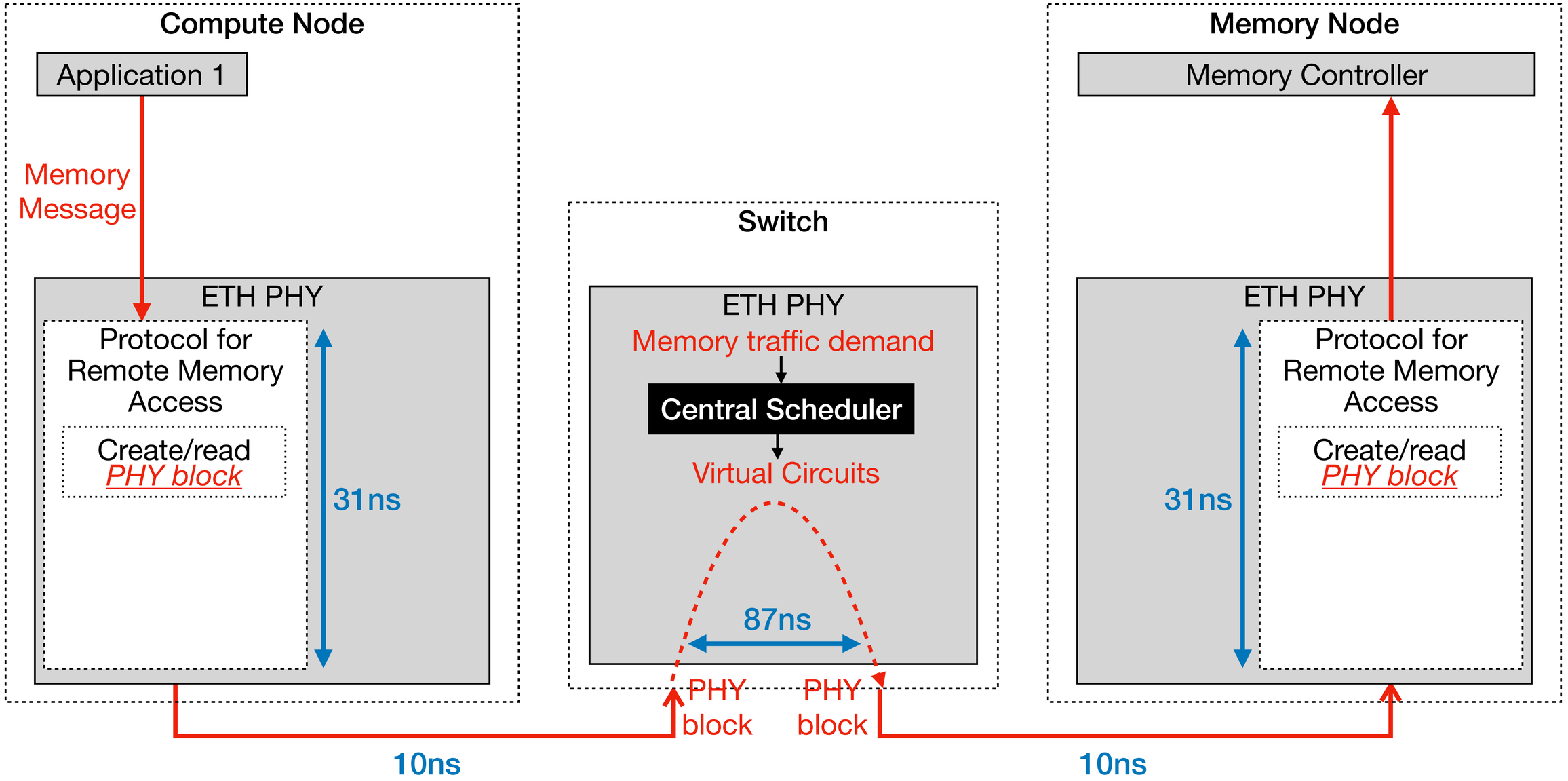
Central Scheduler in the Switch PHY



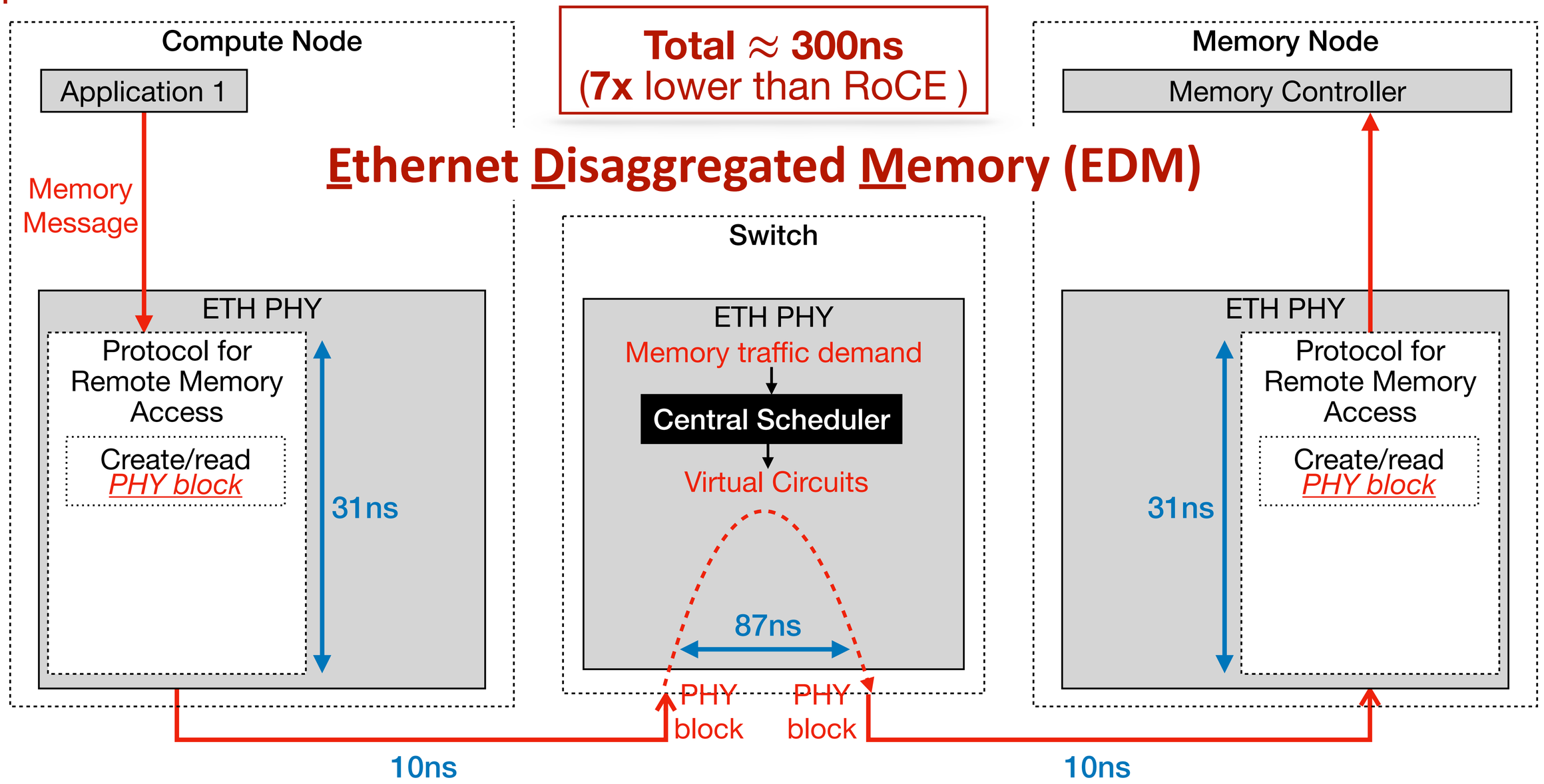
Central Scheduler in the Switch PHY



Central Scheduler in the Switch PHY



Central Scheduler in the Switch PHY



Challenges w/ Central Scheduler

- **Challenge 1:** Accurate traffic demand estimation
- **Challenge 2:** Send demands to the switch with low bandwidth, latency overhead
- **Challenge 3:** Line rate, low latency scheduler pipeline

Challenges w/ Central Scheduler

- **Challenge 1:** Accurate traffic demand estimation
  - Solution: Leverage the interface to memory controller
- **Challenge 2:** Send demands to the switch with low bandwidth, latency overhead
- **Challenge 3:** Line rate, low latency scheduler pipeline

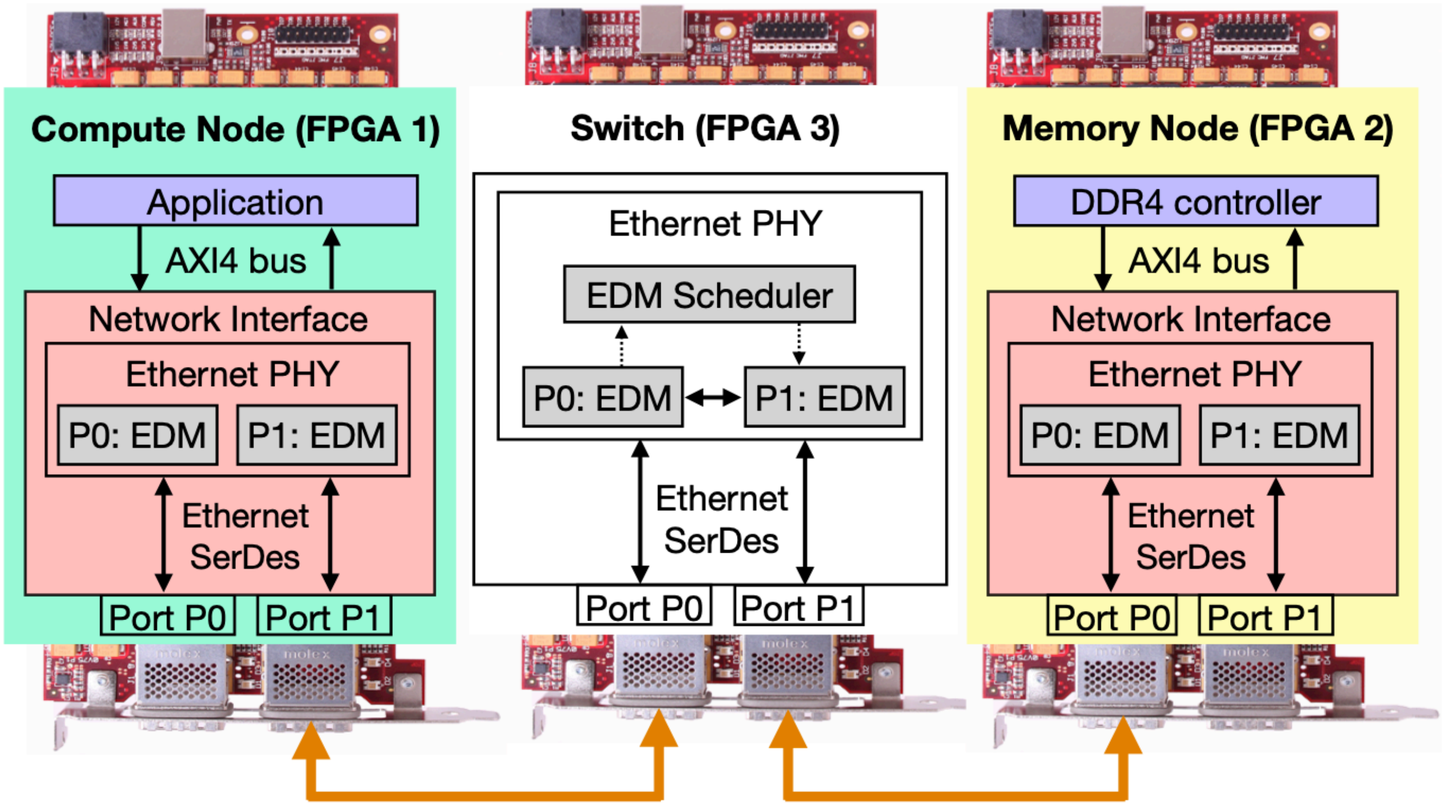
Challenges w/ Central Scheduler

- **Challenge 1:** Accurate traffic demand estimation
  - Solution: Leverage the interface to memory controller
- **Challenge 2:** Send demands to the switch with low bandwidth, latency overhead
  - Solution: Leverage request-reply nature of memory access
- **Challenge 3:** Line rate, low latency scheduler pipeline

Challenges w/ Central Scheduler

- **Challenge 1:** Accurate traffic demand estimation
  - Solution: Leverage the interface to memory controller
- **Challenge 2:** Send demands to the switch with low bandwidth, latency overhead
  - Solution: Leverage request-reply nature of memory access
- **Challenge 3:** Line rate, low latency scheduler pipeline
  - Solution: Leverage hardware parallelism in switch's PHY

Implementation & Evaluation

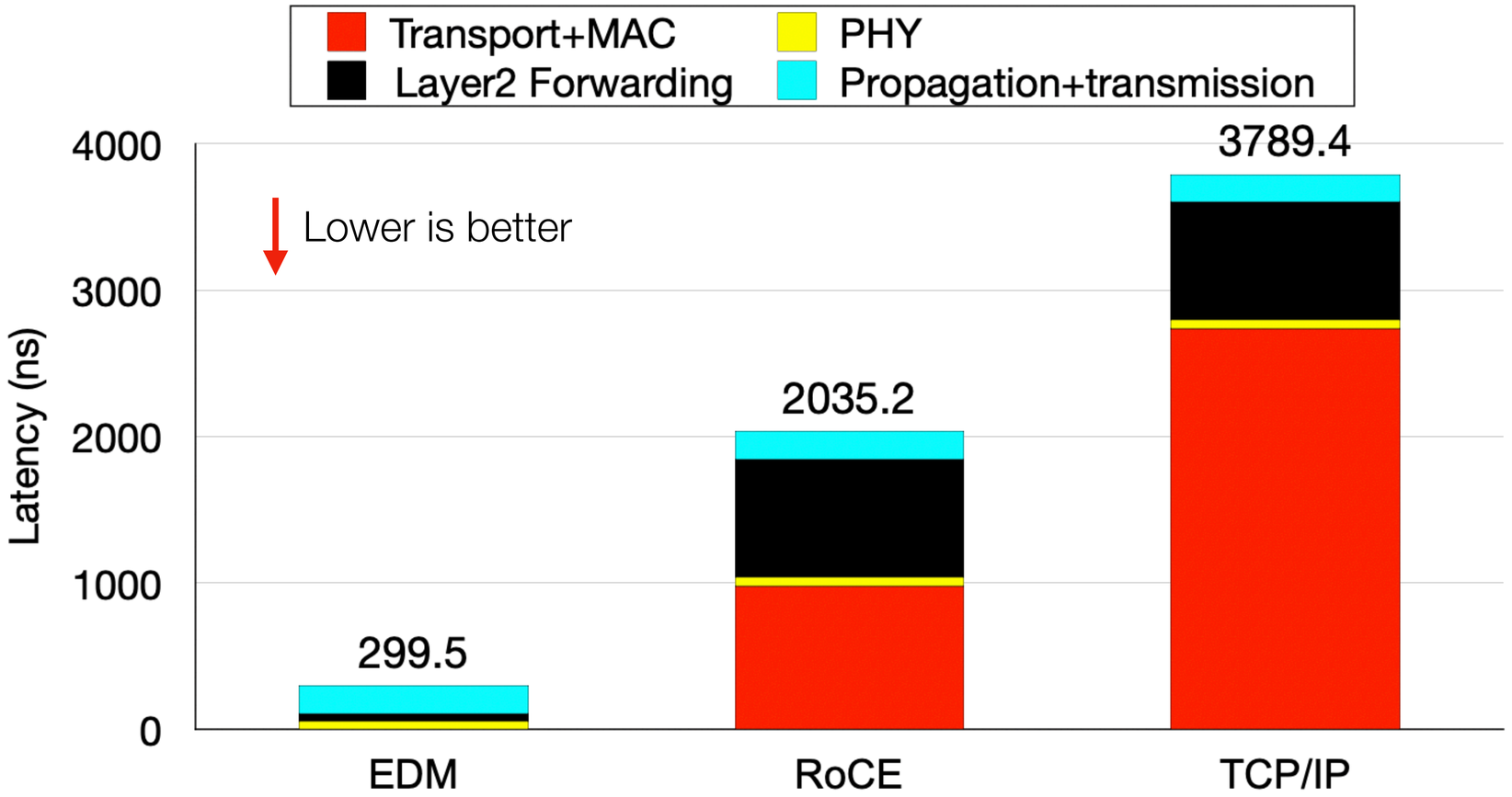


Hardware Testbed

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

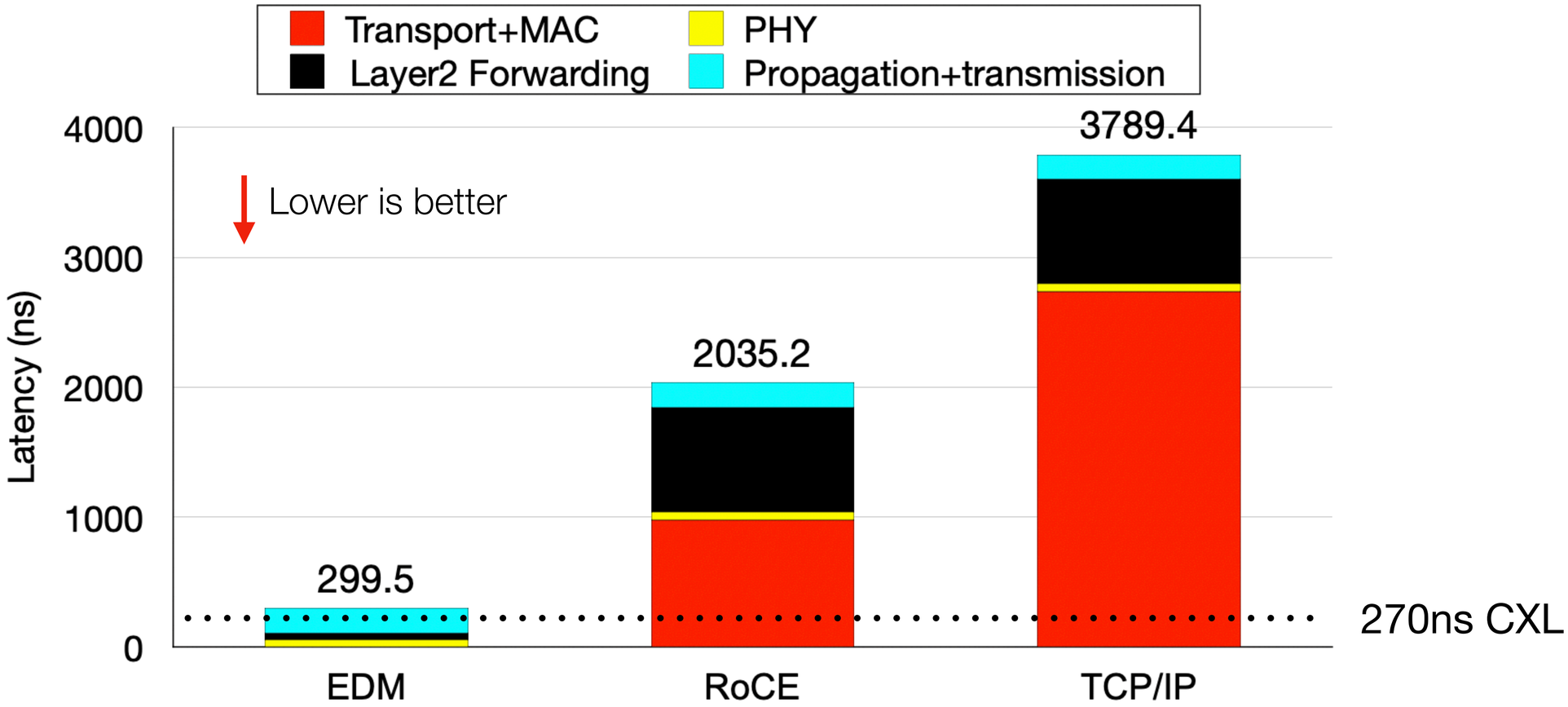
Evaluation Result

- End-to-end unloaded latency

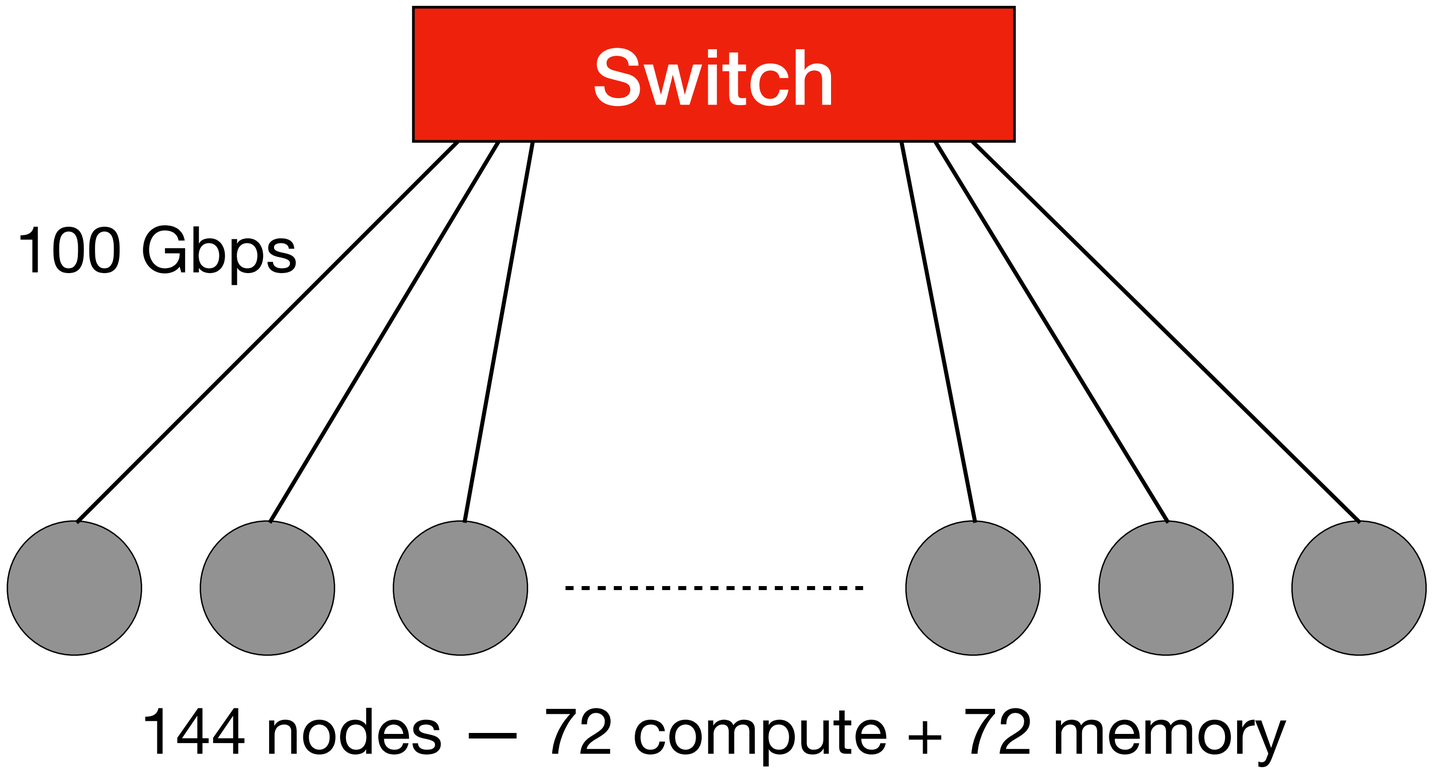


Evaluation Result

- End-to-end unloaded latency



Implementation & Evaluation



Hardware Testbed

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

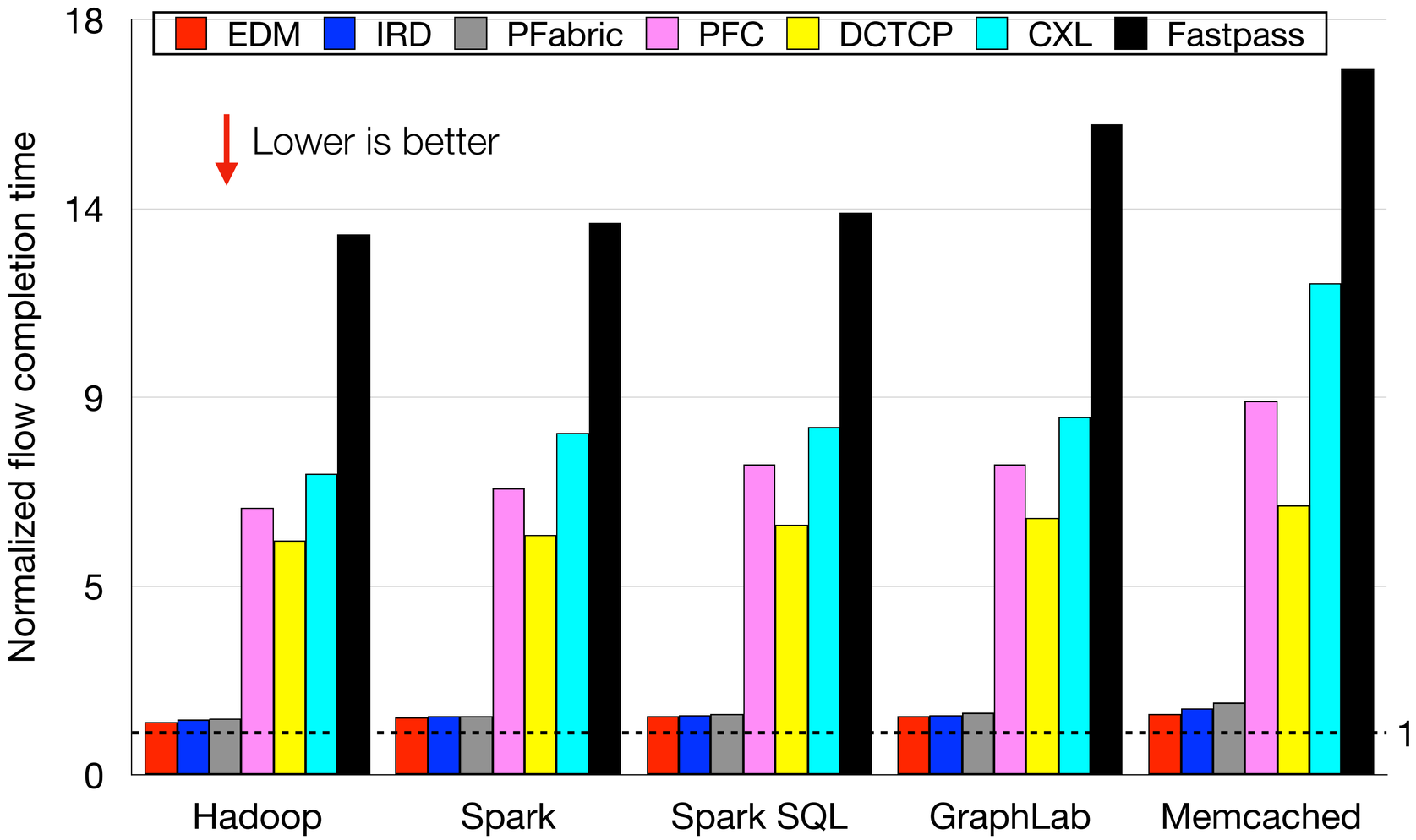
Network Simulator

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

Evaluation Result

- 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