Realizing RotorNet: Toward Practical Microsecond Scale Optical Networking

Max Mellette, Alex Forencich, Rukshani Athapathu,
Alex C. Snoeren, George Papen, George Porter

SIGCOMM 2024

…A ten-year expedition

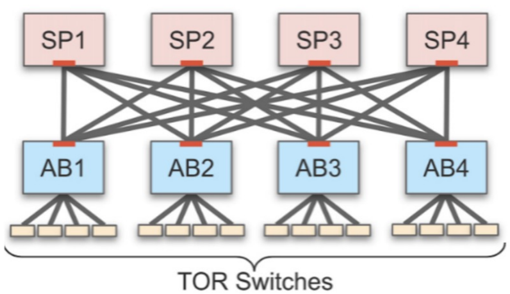
inFocus NETWORKS

UC San Diego

Goal: Transition to optical switching

Conventional Clos

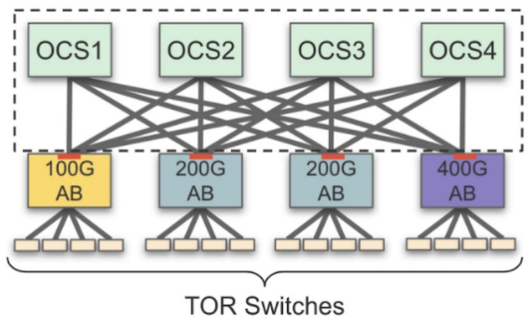
(electronic packet switches)



- High cost and power

Google Apollo

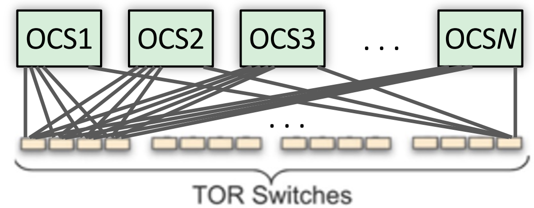
(introduced OCS spine)



- Leverage aggregation at AB:
  - Limit number of OCS ports
  - Traffic stability
- OCS:
  - 100's of ports
  - ~ ms switching
- Cost & power savings, but limited to spine layer

RotorNet

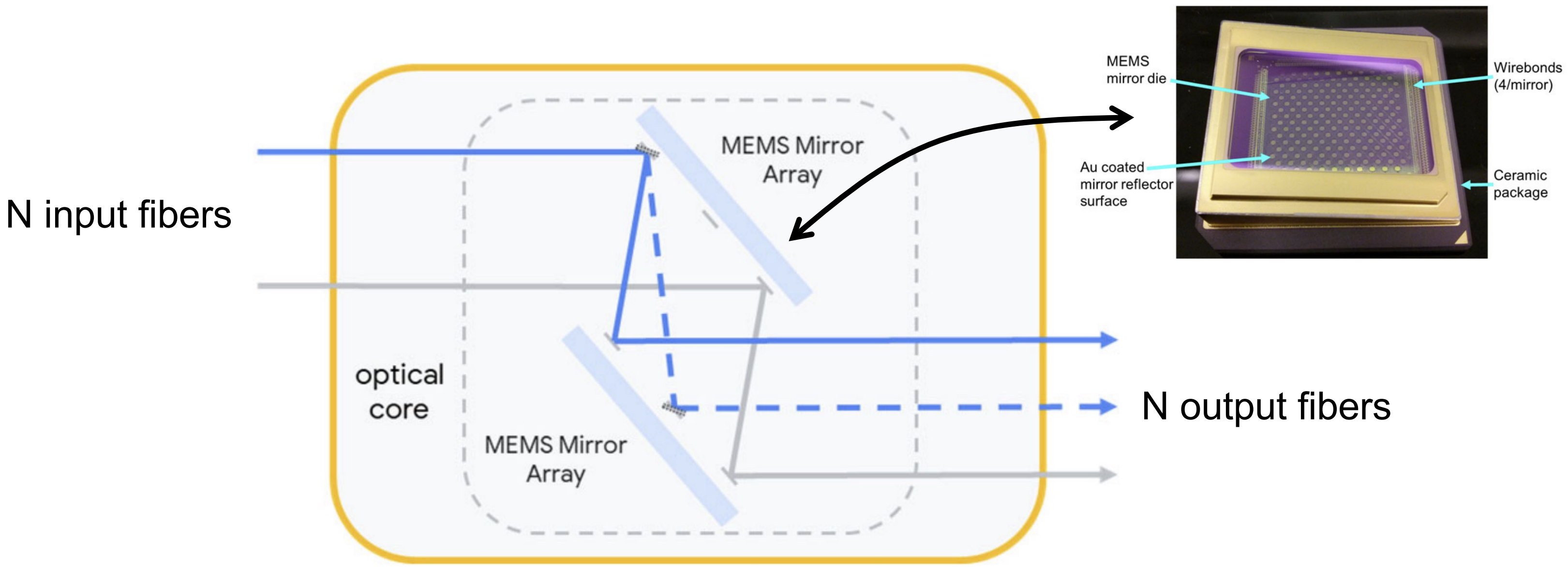
(scale up OCS radix and speed to flatten)



- Connect racks directly with OCS:
  - More OCS ports
  - Dynamic traffic
- OCS:
  - 1,000+ ports
  - ~ µs switching
- Additional cost & power savings

How to scale OCS speed and radix?

Operation of a MEMS OCS:



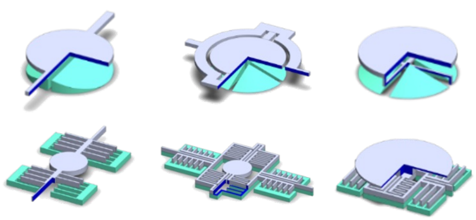
How to scale OCS speed and radix?

Question: **Can we make MEMS faster in existing OCS designs?**

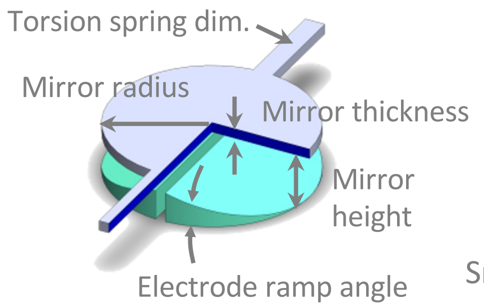
MEMS device optimization study:

Plate & Comb drives

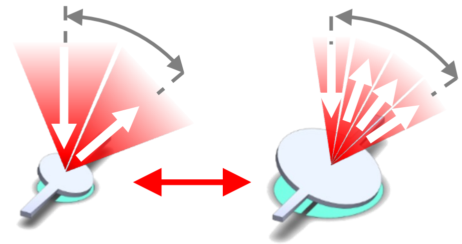
(not to scale)



Actuator geometries



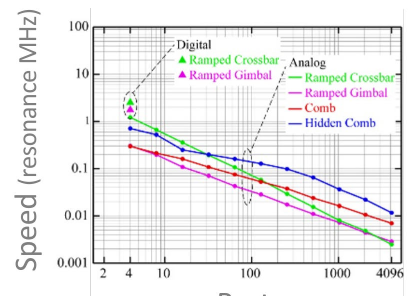
Switch-level performance



Small, fast mirror
Few ports

Large, slow mirror
Many ports

Speed vs Ports



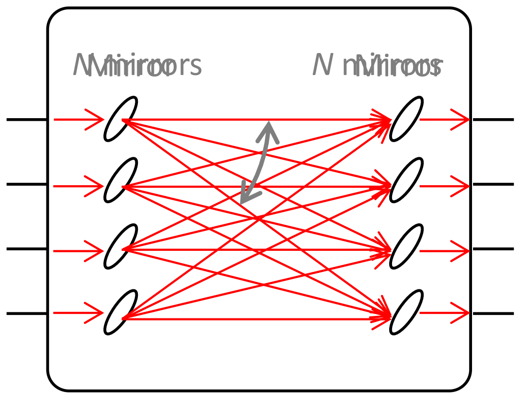
Mellette et al., "Scaling limits of MEMS beam-steering switches for data center networks"

Challenging MEMS fab and device/optical tolerances…
… and fundamental tradeoff between speed and ports

Idea #1: Partial connectivity

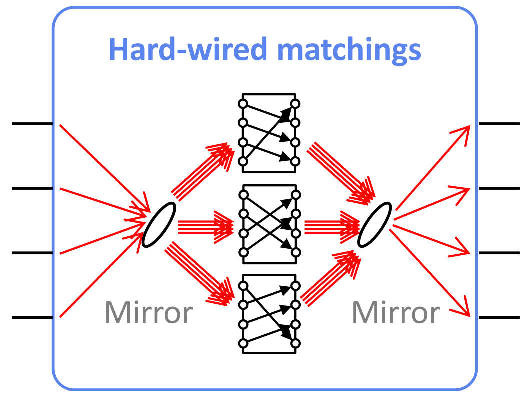
Conventional OCS:

N input ports



N output ports

Optical Selector switch:



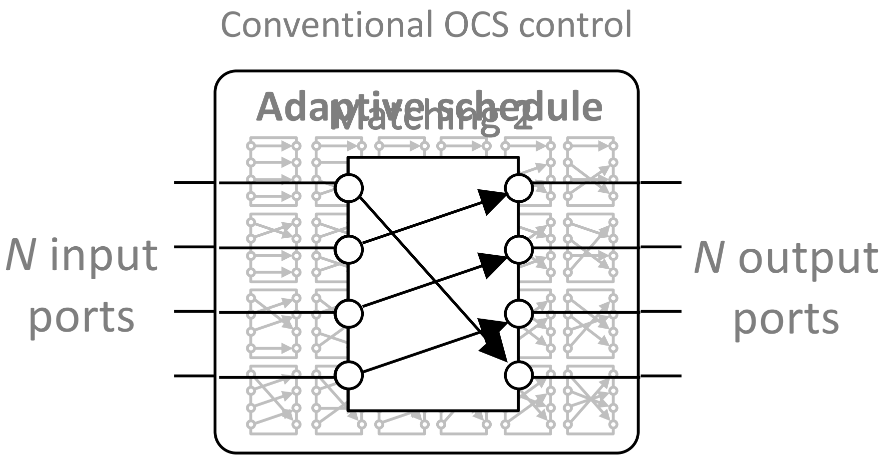
Cost & complexity	# ports	# matchings
Speed	~ 1 / (# ports)	~ 1 / (# matchings)

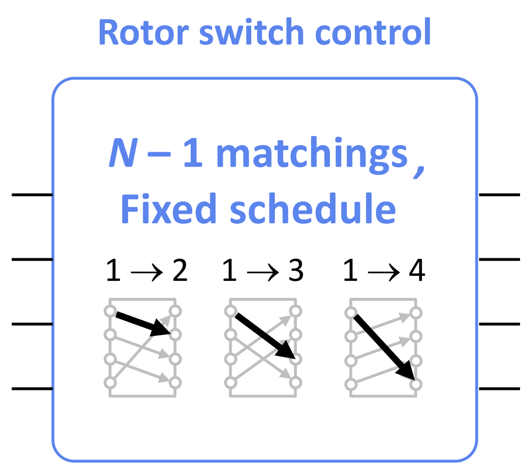
** Topology enables matchings << ports per switch*

Limiting OCS connectivity can increase speed and/or radix

Idea #2: Oblivious routing

Mellette et al., "Rotornet: A scalable, low-complexity, optical datacenter network"

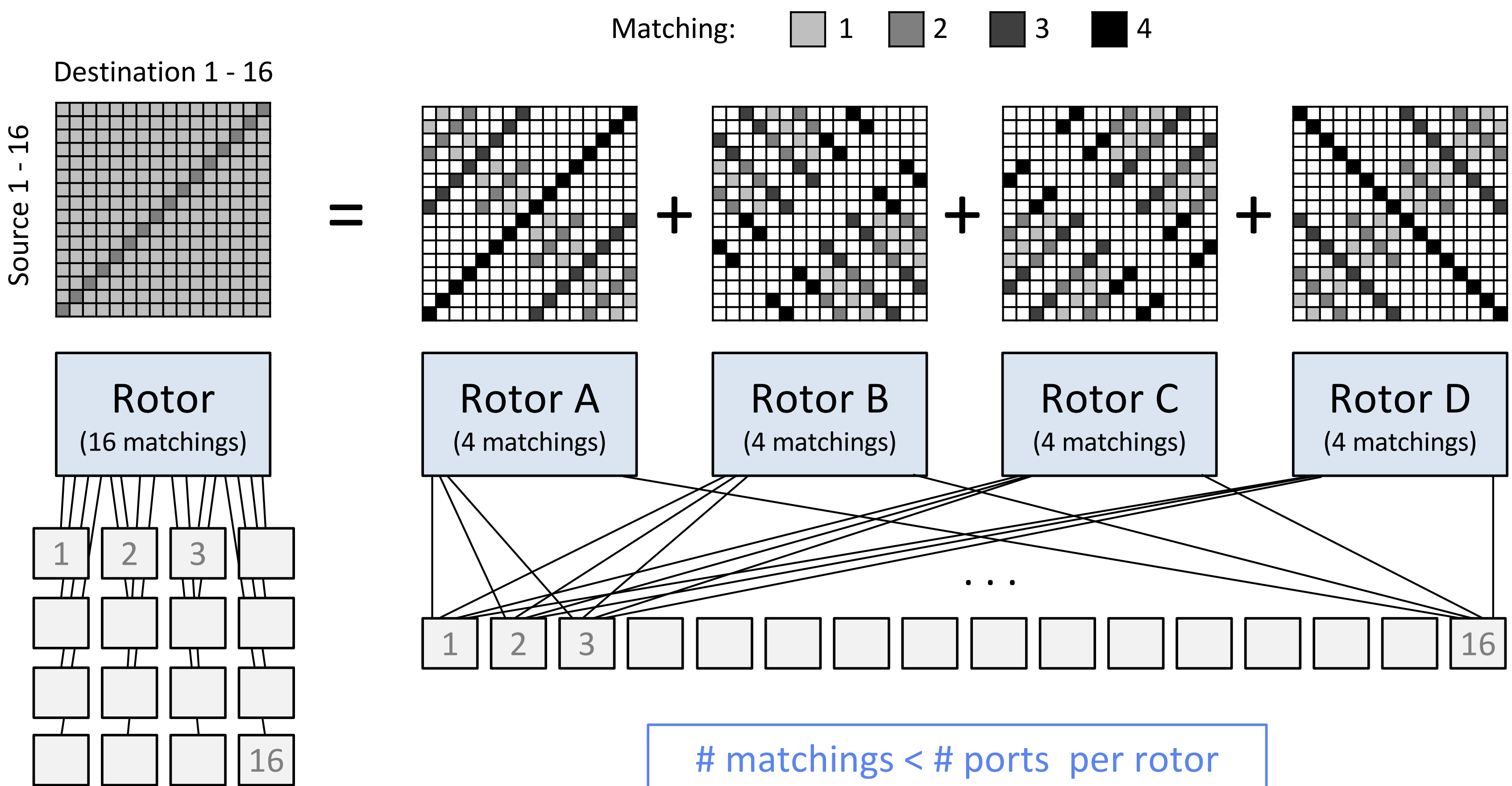




Traffic throughput	~ 100% (if switch & ctl can keep up…)	50 – 100%
Control complexity	High (match traffic in real time)	Low ("open loop")

Simplifies hardware and control for <2x overall throughput hit (Valiant load balancing)

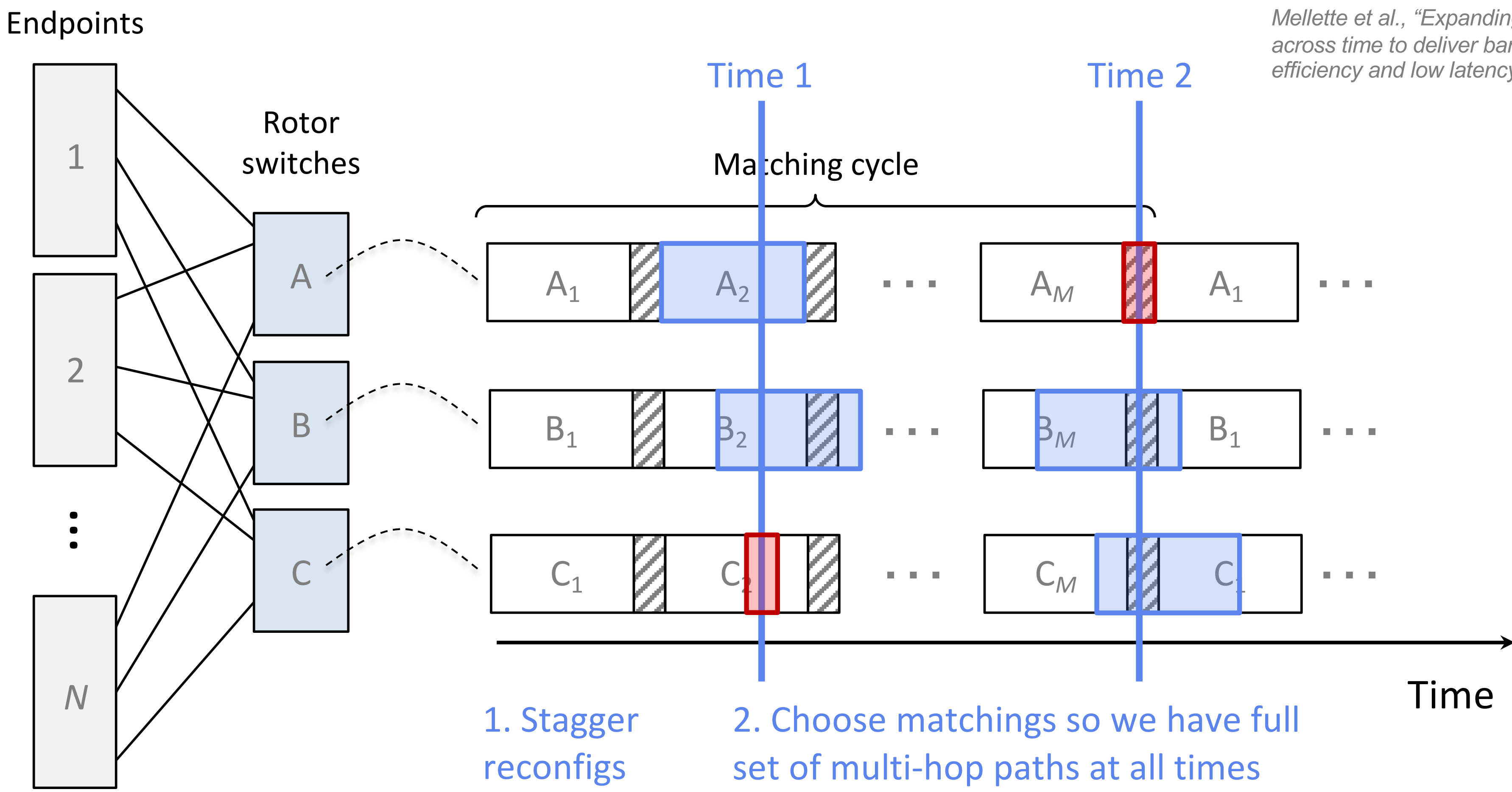
Idea #3a: Parallel rotors…



matchings < # ports per rotor

Idea #3b: …to construct expander graphs

Mellette et al., "Expanding across time to deliver bandwidth efficiency and low latency"



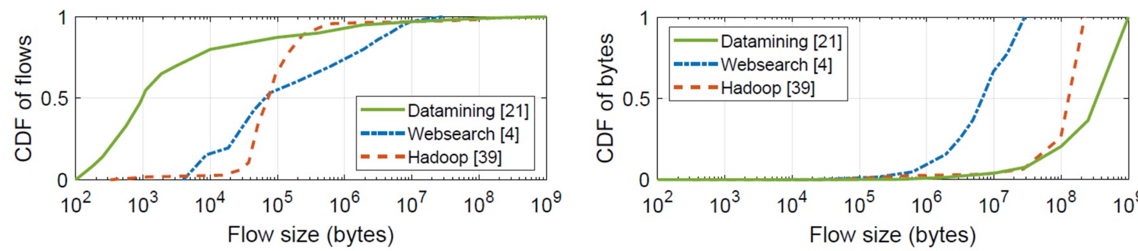
1. Stagger reconfigs

2. Choose matchings so we have full set of multi-hop paths at all times

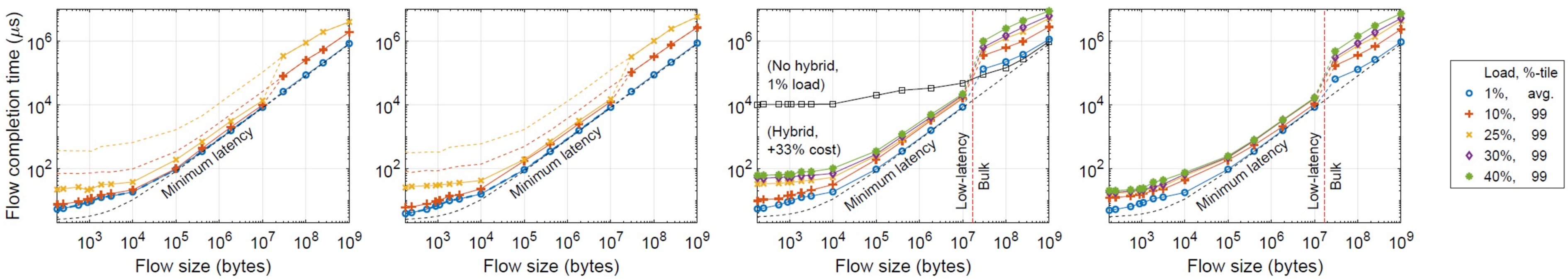
Idea #4: Latency-sensitive routing

Cloud DC workloads with a mixture of flow sizes:

Mellette et al., "Expanding across time to deliver bandwidth efficiency and low latency"

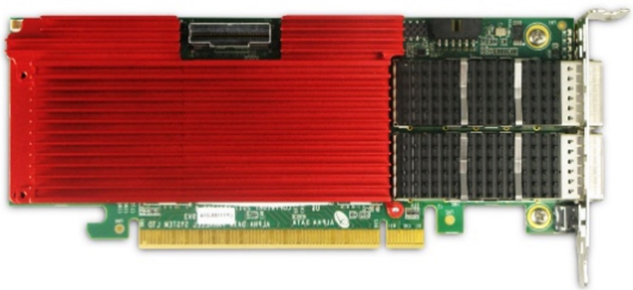


3:1 Clos | Expander Graph | RotorNet | RotorNet w/ low latency

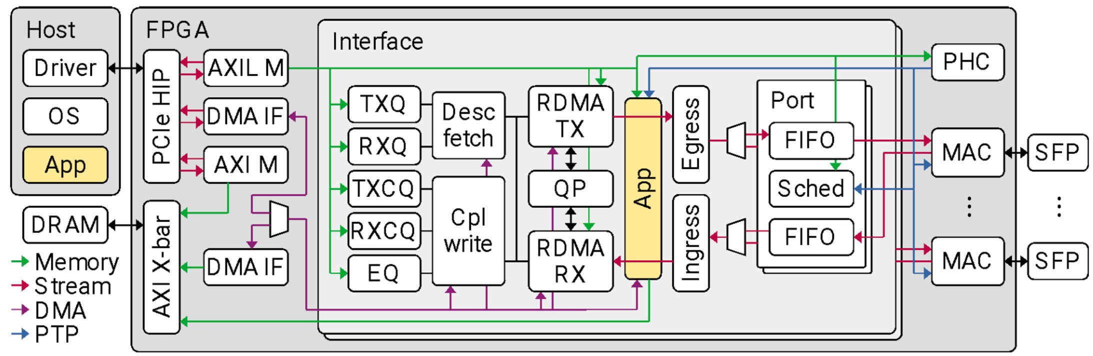


RotorNet can improve system performance/cost ratio for realistic datacenter workloads with mixture of short and long flows

Artifact #1: Corundum NIC platform

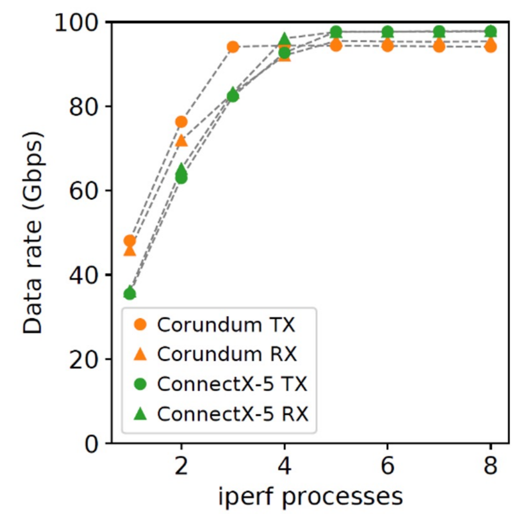


(Multiple HW targets)



100 Gb/s platform

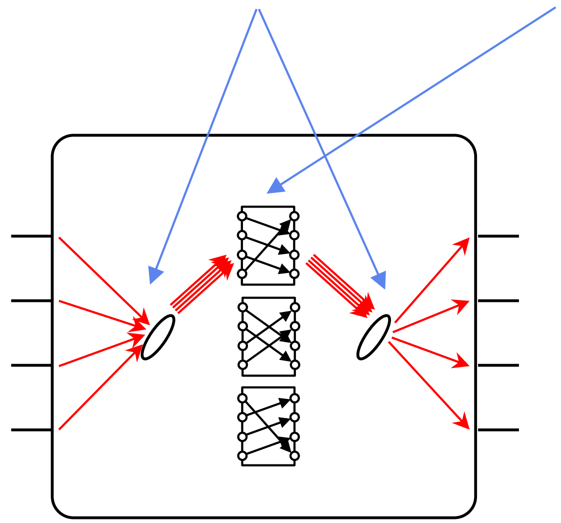
- IEEE 1588 PTP time sync
- Precisely timed packet admission
- Custom NIC logic (schedulers, routing, store & forward, cut through)
- 1,000s of hardware queues
- Open source



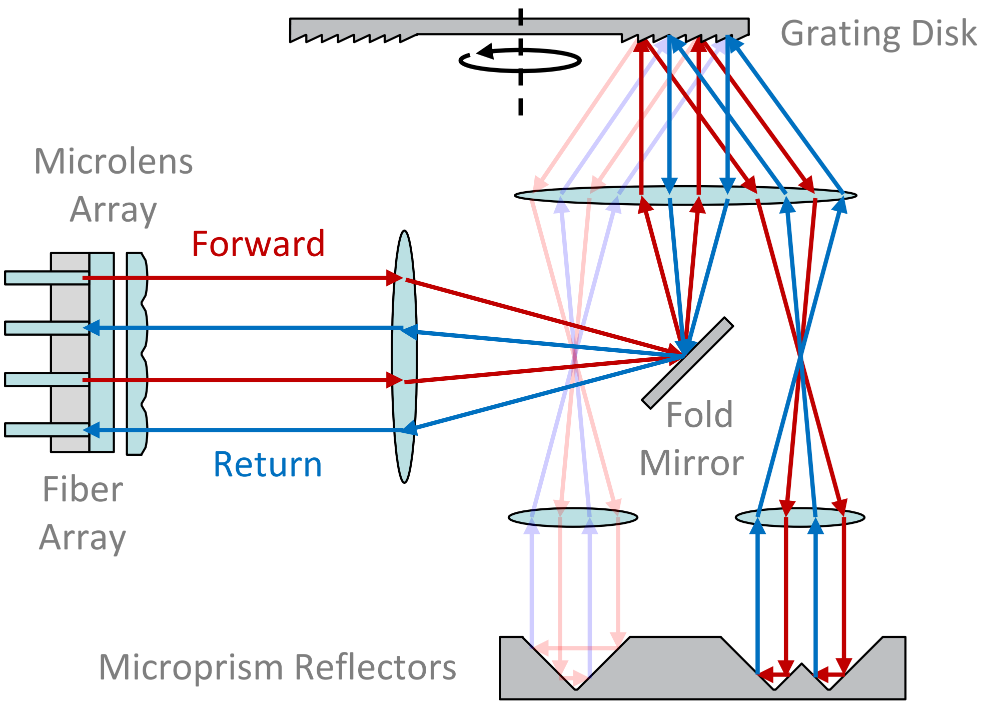
Forencich et al., "Corundum: An Open-Source 100-Gbps Nic"

Artifact #2: Rotor switch

Key idea: separate *switching* from *routing*

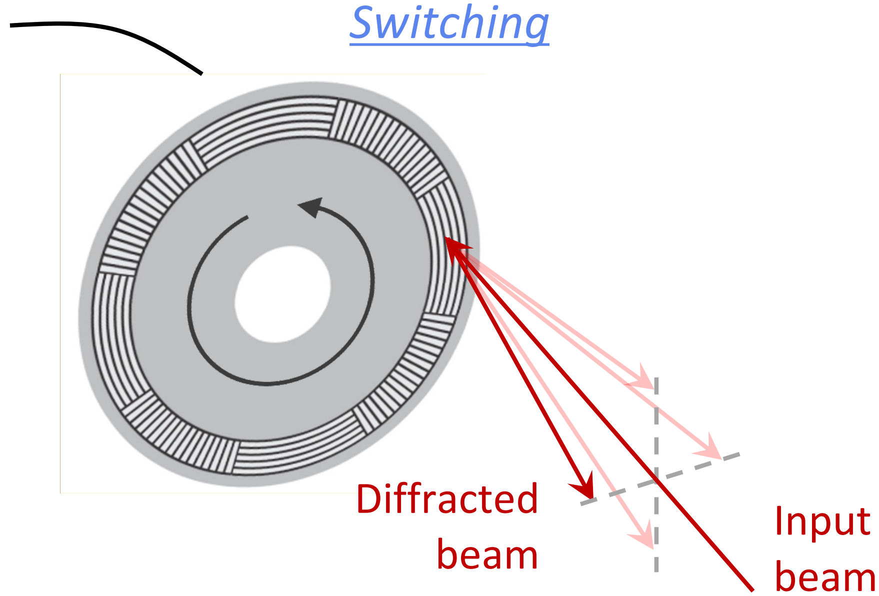


Rotor switch – principle of operation

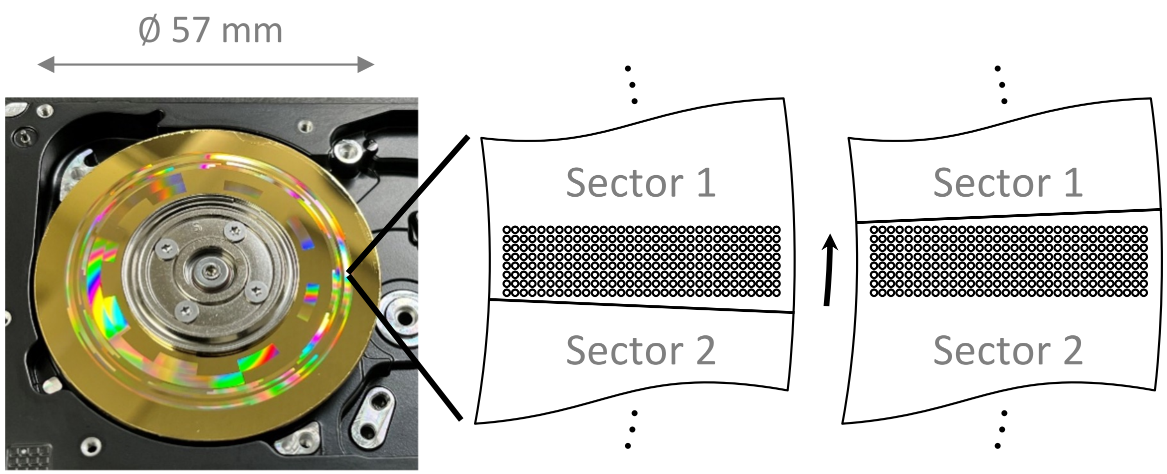


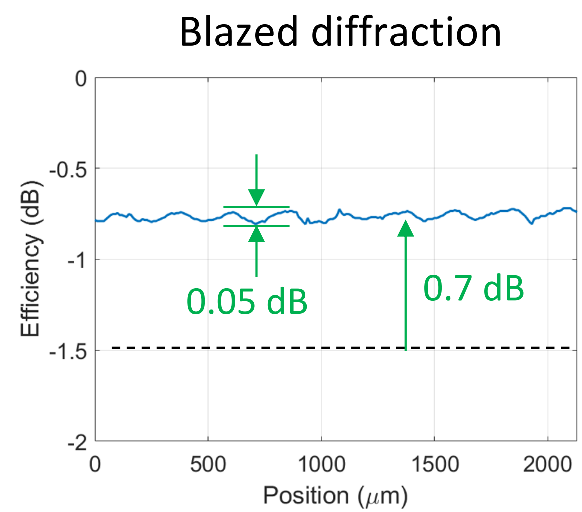
Routing

Switching



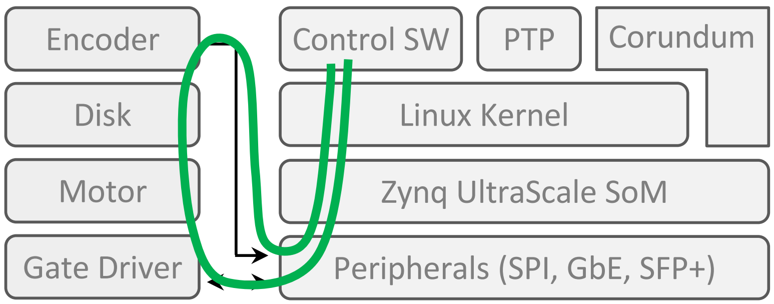
Subsystem: Grating disk





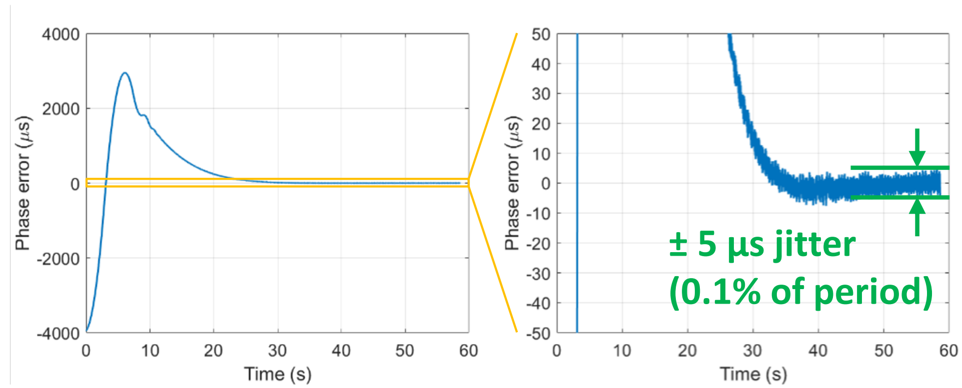
- **Many DOF for increasing switching speed**
- Low cost fab:
  - Greyscale litho master
  - NIL submaster + sub-submaster stamping
- Additional room for improvement with fab process

Subsystem: Rotor phase control

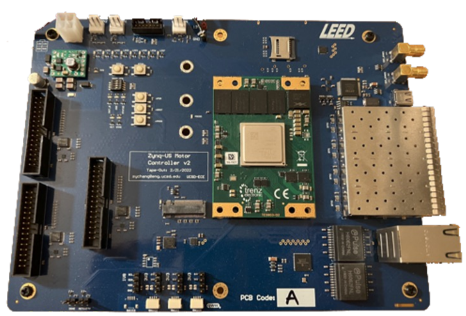


Control loop

Disk phase error, 15k rpm:



Encoded
Rotor disk

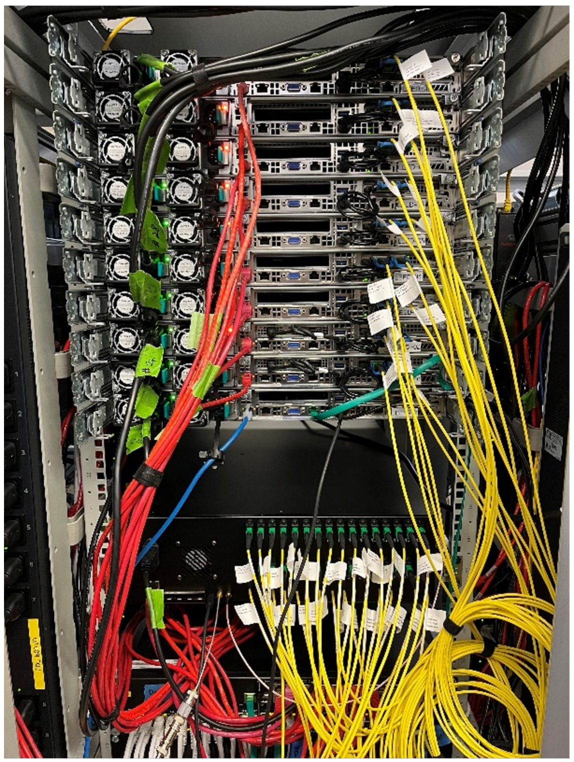


Custom control board

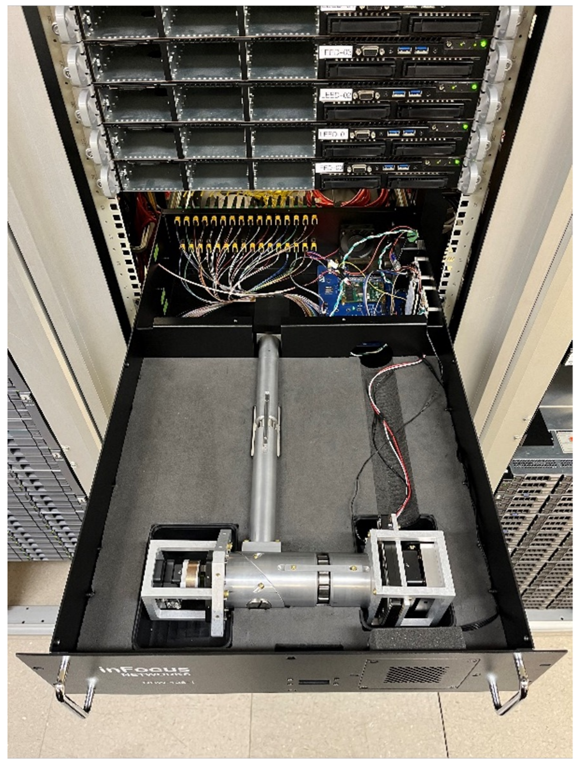
- **Phase jitter < switching time**
- Control board runs Linux
- ssh access to manage switch over GbE

16-node testbed integration (128 optical links total)

Hot Aisle

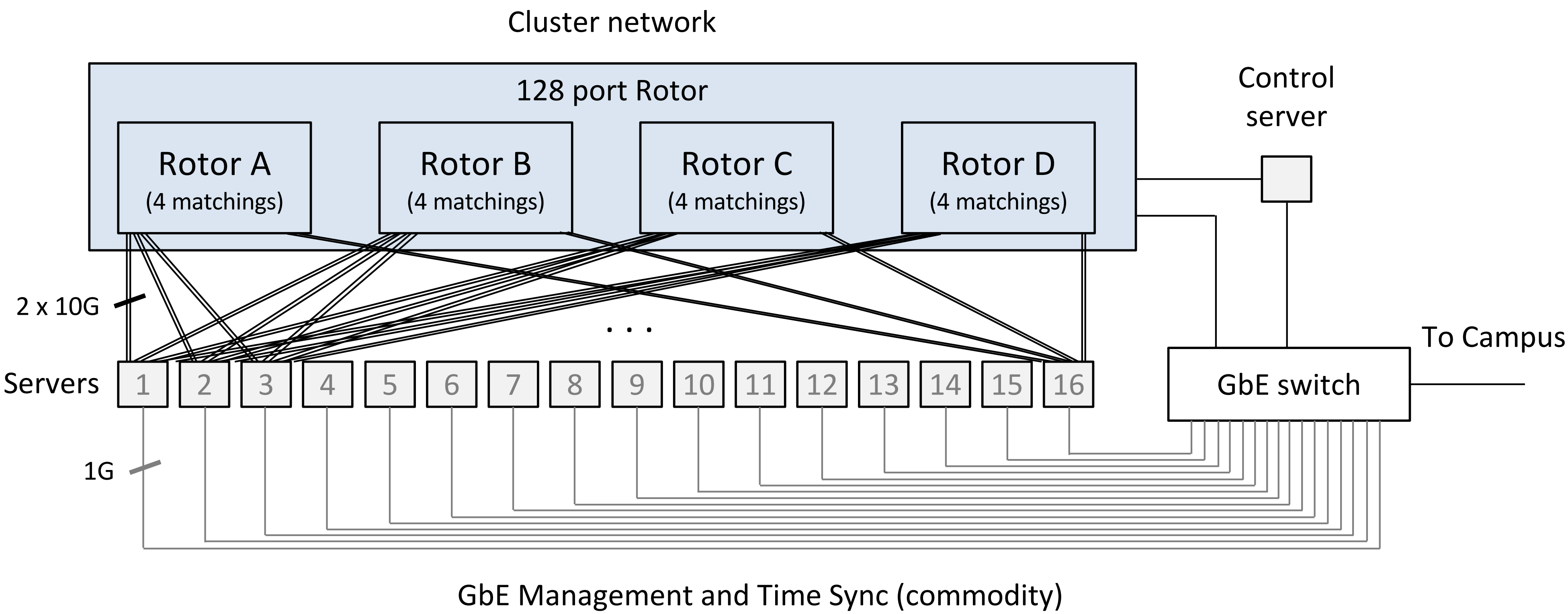


Cold Aisle (lid removed)

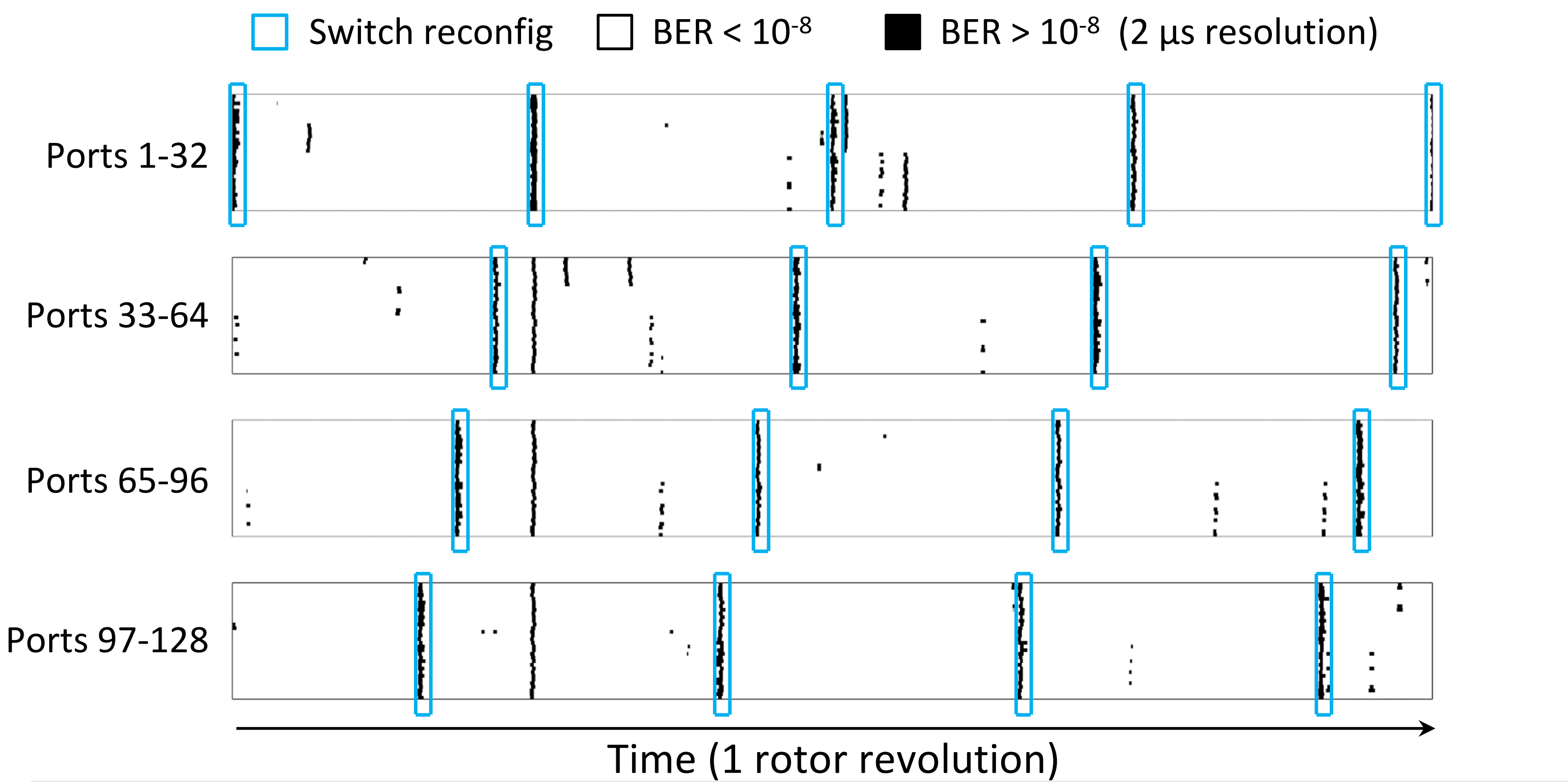


- University “production” machine room with hot & cold aisles
- Standard 19” rack housing servers, packet switches, storage, & rotor switch
- 1 cluster control server
- 16 cluster compute servers:
  - 80 Gb/s per server optical, through rotor switch
  - 1 Gb/s management + IEEE 1588 PTP type sync

16-node testbed integration (128 optical links total)

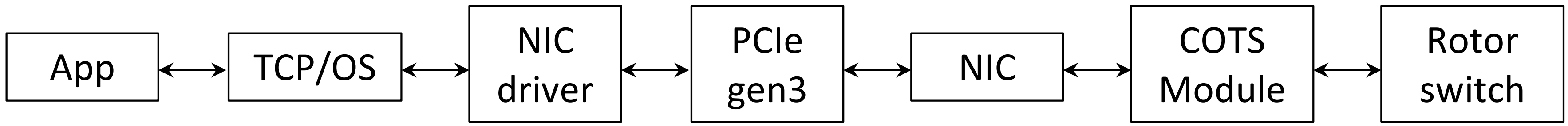


Idea #5: Deterministically mask rotor imperfections

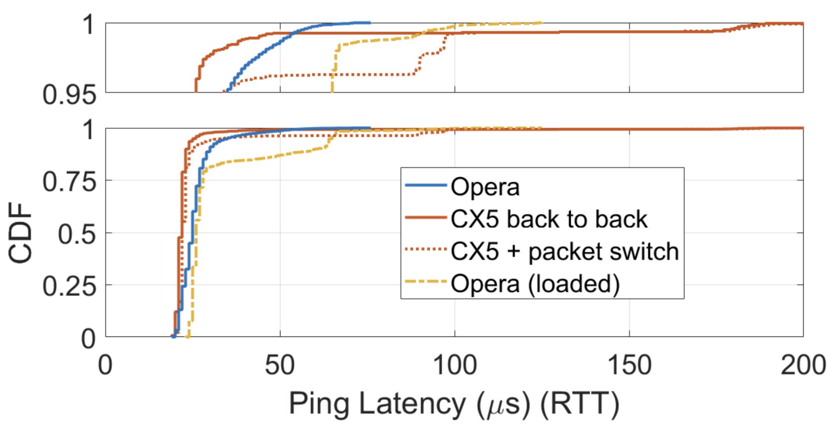


Point defects are masked at the NIC. < 1% throughput overhead.

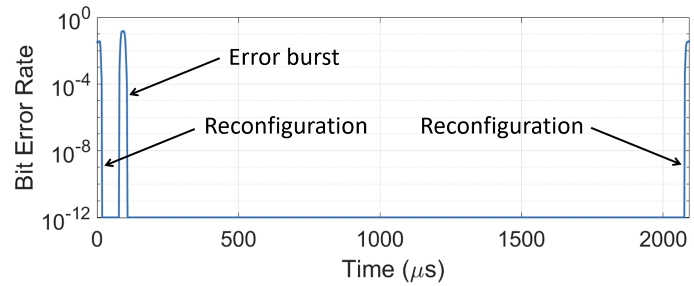
End-to-end system level characterization



Latency



Throughput

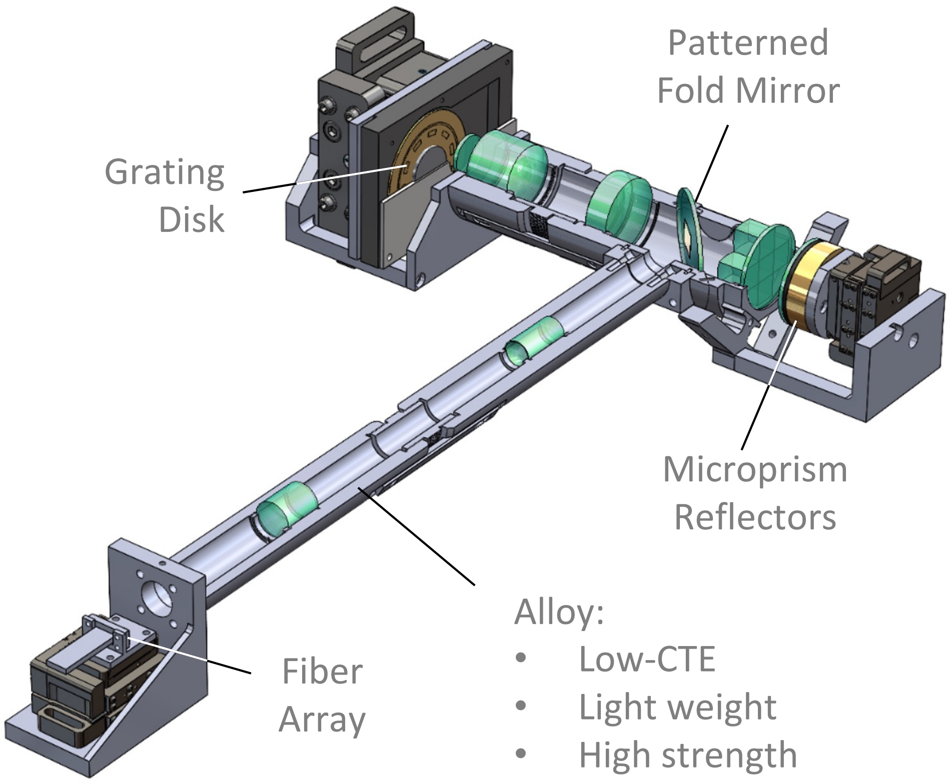


	w/o mask	w/ mask
TCP Retransmissions	6780	0
TCP CWND	70 kB	1.2 MB
`iperf3` relative throughput	0.893	0.995

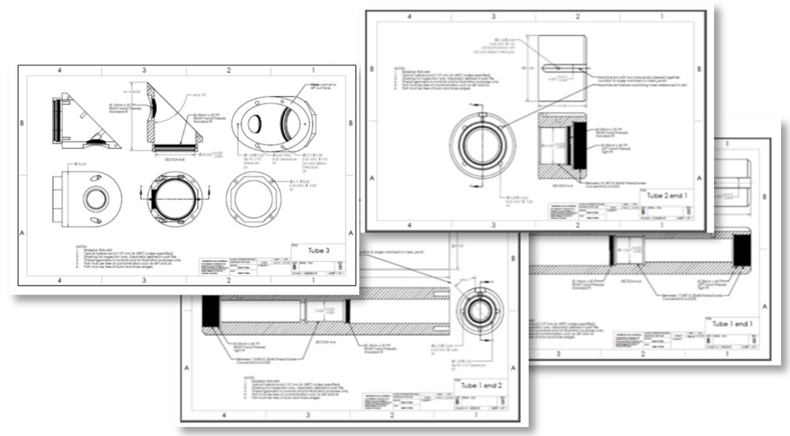
Mellette et al., "Realizing RotorNet: Toward Practical Microsecond Scale Optical Networking"

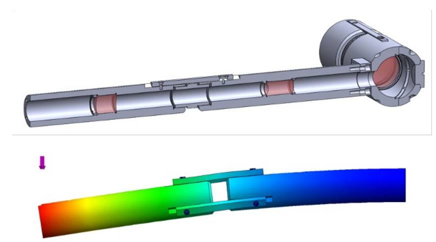
Full stack optical networking with performance almost indistinguishable from packet switching

RotorNet Realized!



Alloy:
- Low-CTE
- Light weight
- High strength
- Machinable





Future directions

This work assumed a cloud datacenter context – no knowledge of traffic

$\rightarrow$ Need to provision all N matchings

New context: machine learning – collective communications are known and repetitive

$\rightarrow$ May only need a small number of matchings
$\rightarrow$ Cyclic switching may be a good fit