TORSTEN HOEFLER

Efficient networking and programming of large-scale computing systems

with R. Gerstenberger, M. Besta, R. Belli @ SPCL
presented at HP Labs, Palo Alto, CA, USA

PASC16
Platform for Advanced Scientific Computing Conference
Lausanne Switzerland | 08-10 June 2016

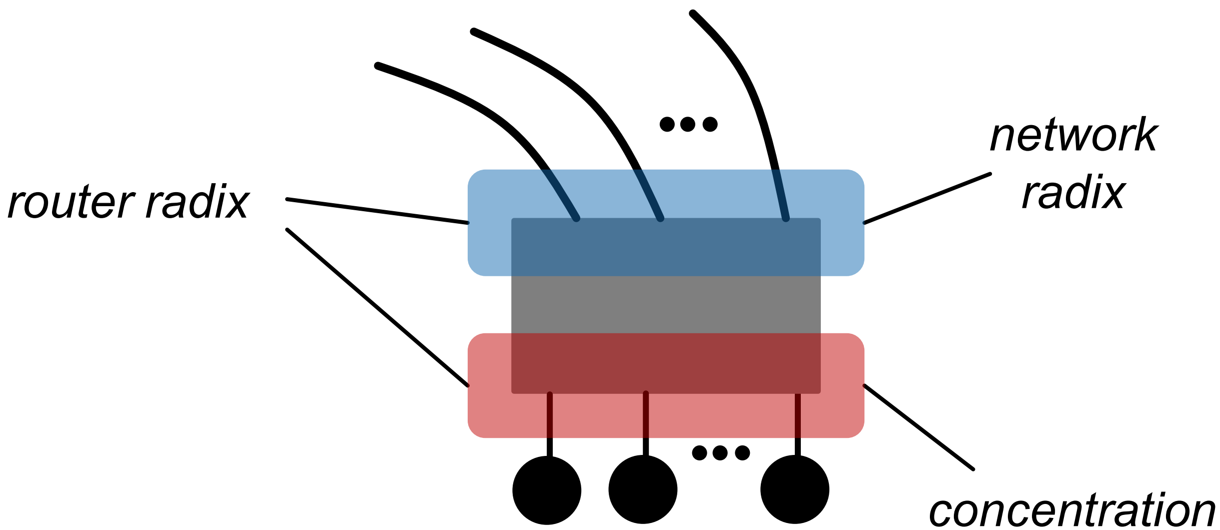
CLIMATE & WEATHER
SOLID EARTH
LIFE SCIENCE
CHEMISTRY & MATERIALS
PHYSICS
COMPUTER SCIENCE & MATHEMATICS
ENGINEERING
EMERGING DOMAINS

Networks, Limits, and Design Space

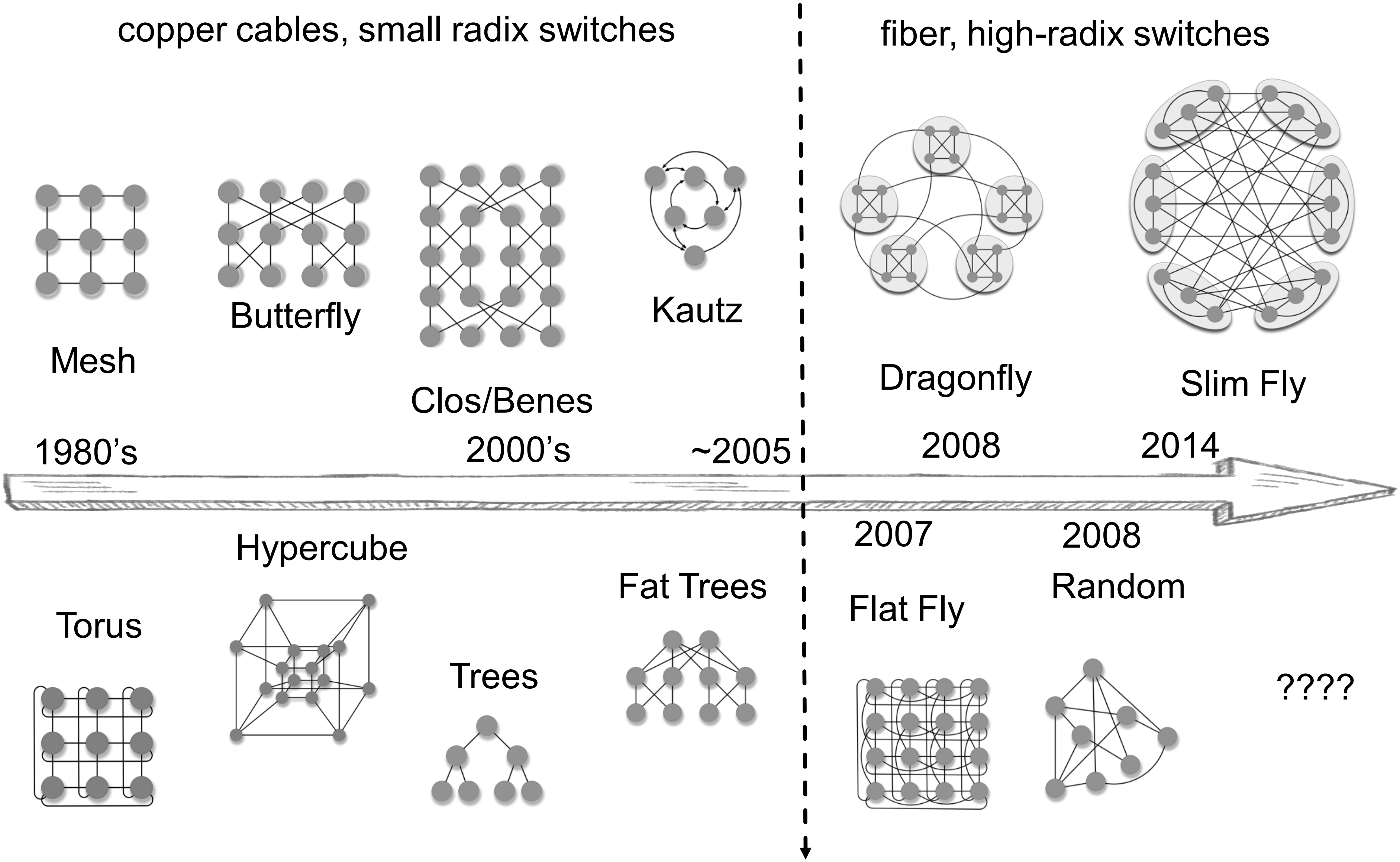
- **Networks cost 25-30% of a large supercomputer**

- **Hard limits:**
  - Router radix
  - Cable length

- **Soft limits:**
  - Cost
  - Performance



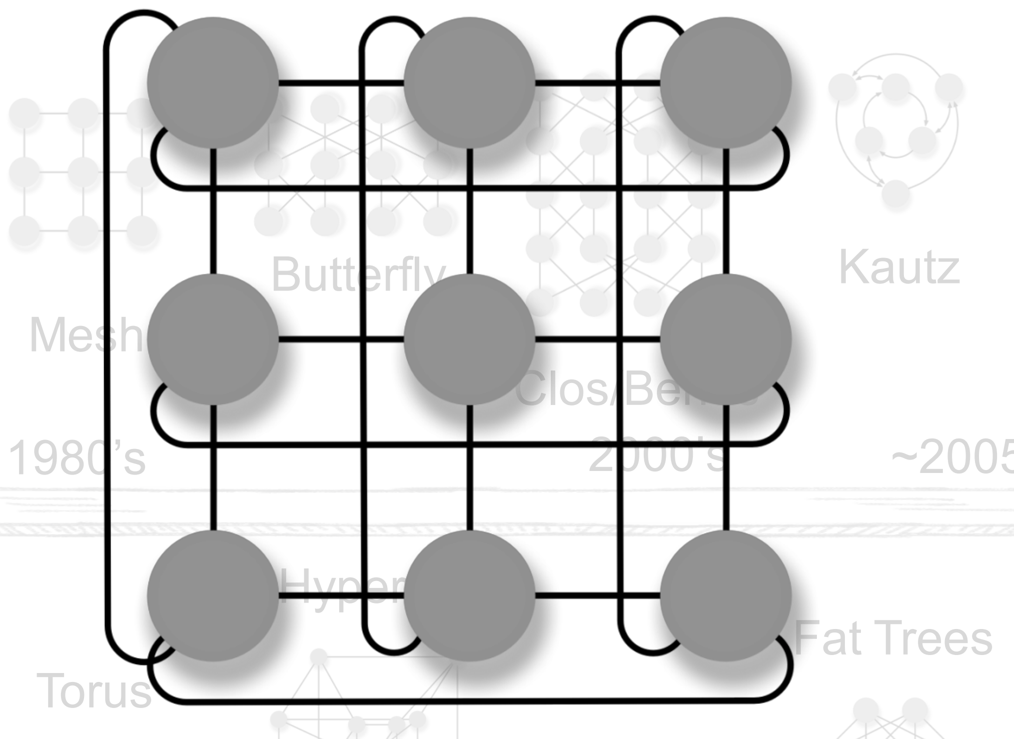
A BRIEF HISTORY OF NETWORK TOPOLOGIES

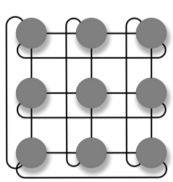


A BRIEF HISTORY OF NETWORK TOPOLOGIES

copper cables, small radix switches

fiber, high-radix switches





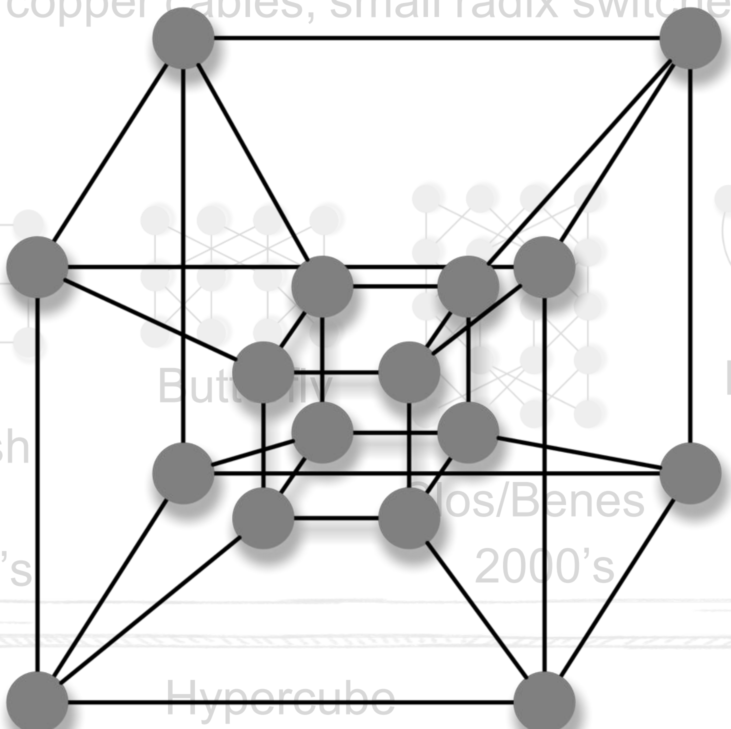
2008
2014

$$\text{Bandwidth} = 2\sqrt[d]{N^{d-1}}$$
$$\text{Latency} = \frac{d}{2}\sqrt[d]{N}$$
$$\text{Radix} = 2d$$

A BRIEF HISTORY OF NETWORK TOPOLOGIES

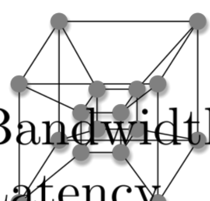
copper cables, small radix switches

fiber, high-radix switches



Mesh, Butterfly, Kautz, Clos/Benes, Hypercube, Torus, Trees, Fat Trees, Flat Fly, Random, ????

1980's, 2000's, ~2005, 2007, 2008, 2008, 2014

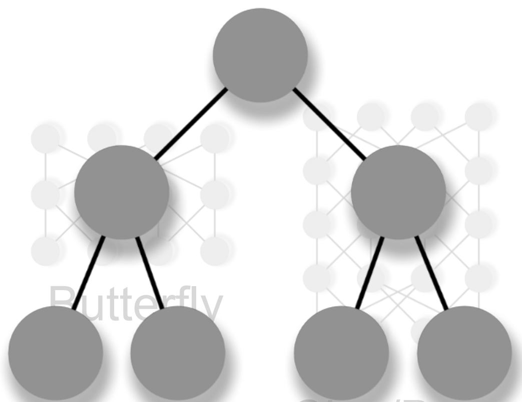


$$\text{Bandwidth} = \frac{N}{2}$$
$$\text{Latency} = \log_2 N$$
$$\text{Radix} = \log_2 N$$

A BRIEF HISTORY OF NETWORK TOPOLOGIES

copper cables, small radix switches

fiber, high-radix switches



Mesh

Butterfly

Kautz

Clos/Benes

1980's

2000's

~2005

2008

2014

2007

2008

Hypercube

Fat Trees

Random

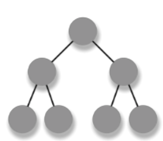
Flat Fly

Torus

Trees

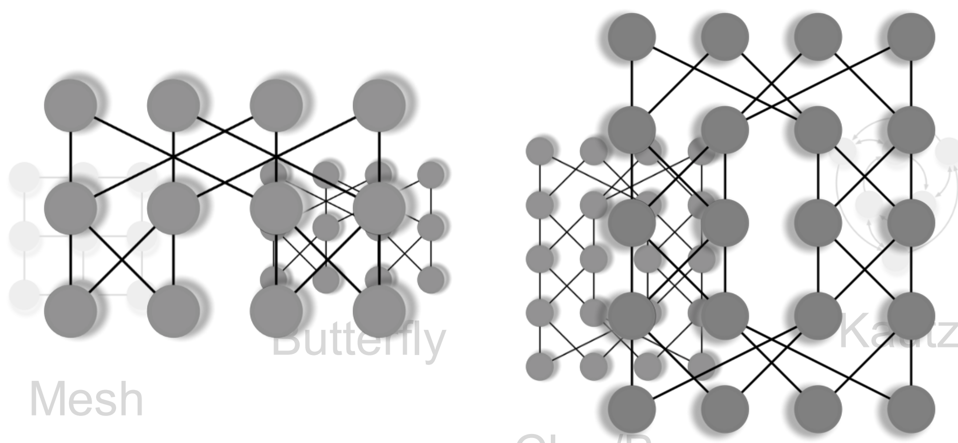
????

$$\begin{aligned} \text{Bandwidth} &= 1 \\ \text{Latency} &= 2\log_2 N \\ \text{Radix} &= 2 \end{aligned}$$



A BRIEF HISTORY OF NETWORK TOPOLOGIES

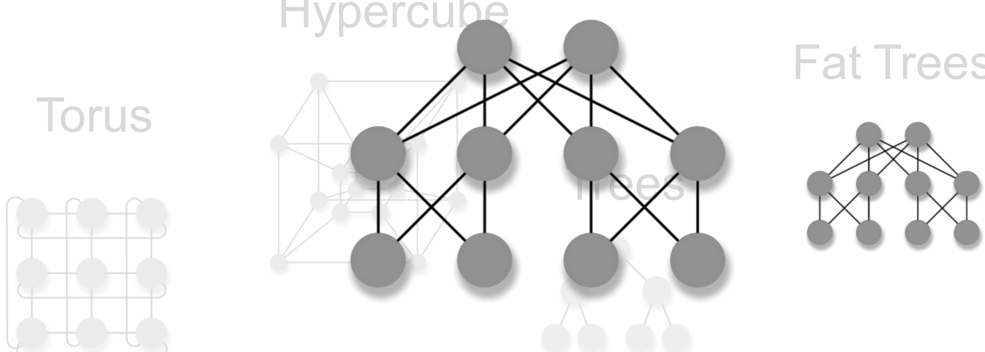
copper cables, small radix switches



$$\text{Bandwidth} = \frac{N}{2}$$
$$\text{Latency} = 2\log_2 N$$
$$\text{Radix} = 4$$

1980's 2000's ~2005 2008 2014

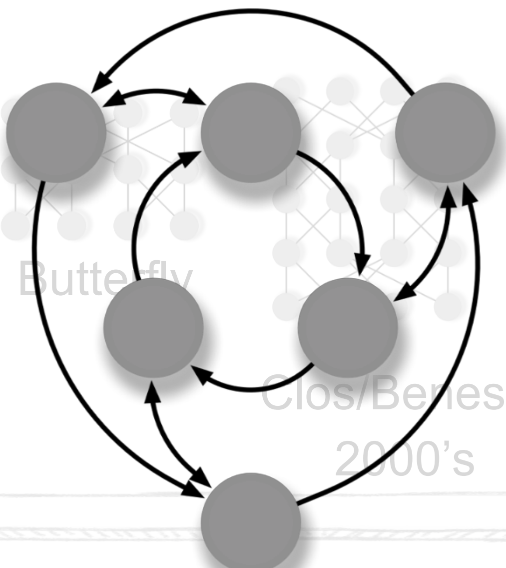
Hypercube

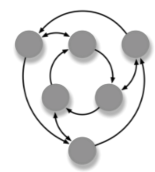


A BRIEF HISTORY OF NETWORK TOPOLOGIES

copper cables, small radix switches

fiber, high-radix switches





Kautz

Mesh

Butterfly

Clos/Benes

Dragonfly

Slim Fly

1980's

2000's

~2005

Hypercube

Fat Trees

Torus

Trees

????

$$\text{Bandwidth} = \rightarrow \frac{N}{4}$$
$$\text{Latency} = \log_k N$$
$$\text{Radix} = k$$

A BRIEF HISTORY OF NETWORK TOPOLOGIES

copper cables, small radix switches

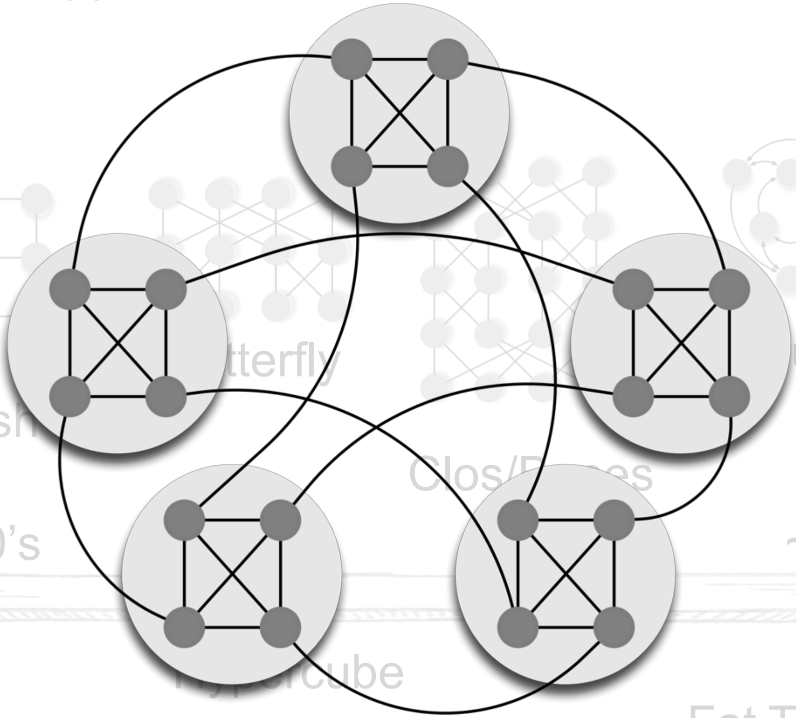
fiber, high-radix switches

1980's

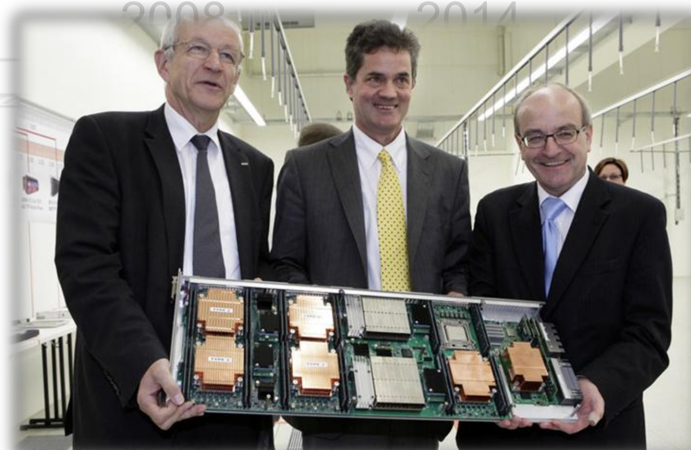
~2005

Torus

Fat Trees



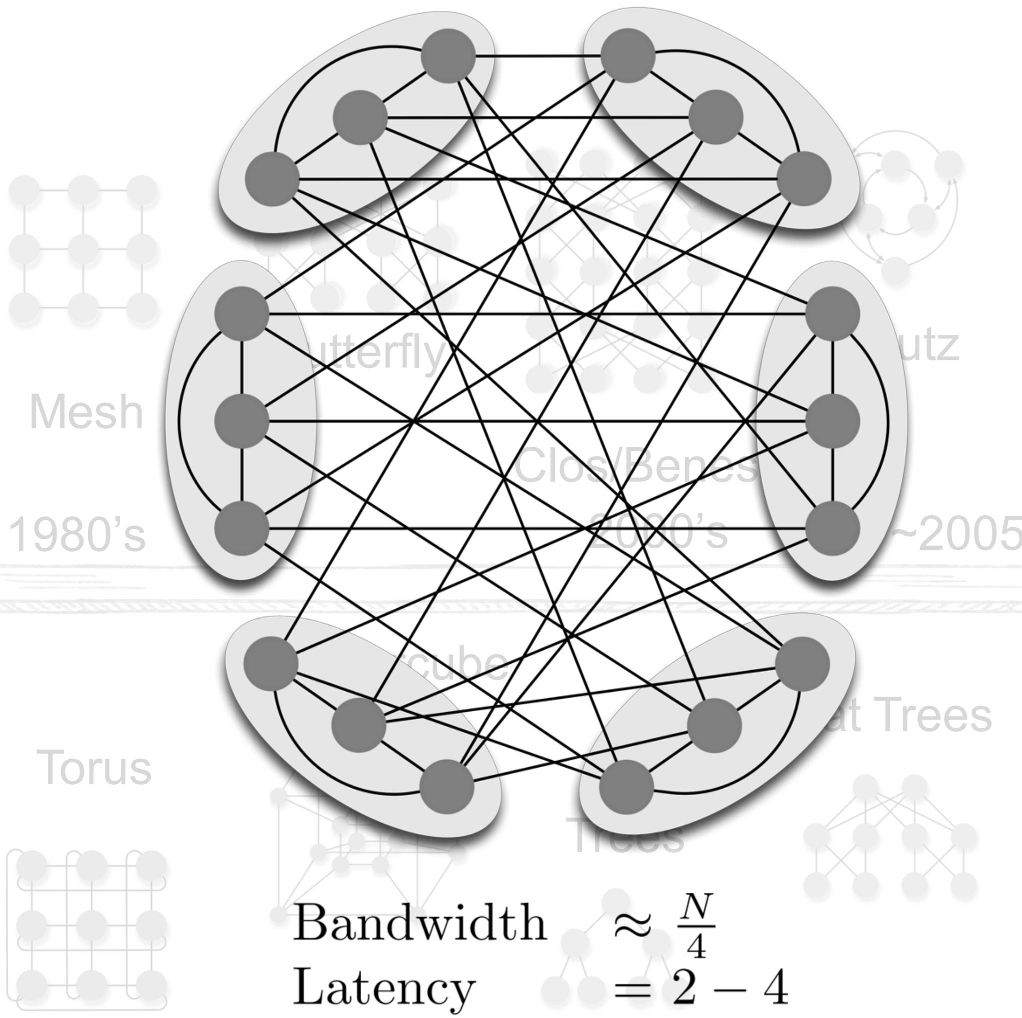
$$\begin{aligned} \text{Bandwidth} &\approx \frac{N}{4} \\ \text{Latency} &= 3 - 5 \\ \text{Radix} &= 48 - 64 \end{aligned}$$

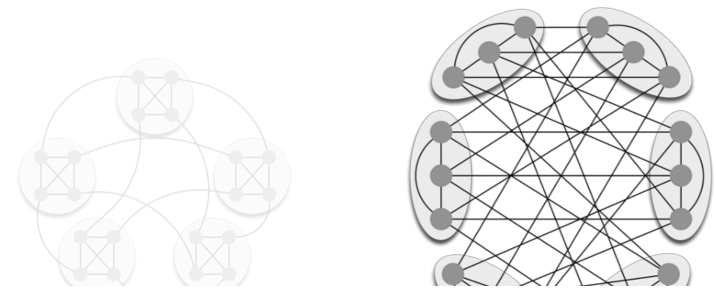


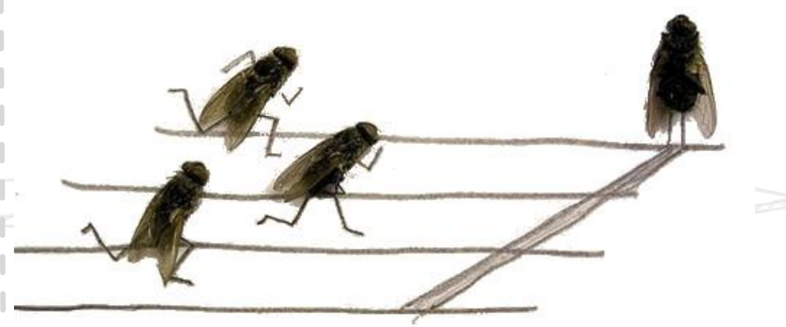
A BRIEF HISTORY OF NETWORK TOPOLOGIES

copper cables, small radix switches

fiber, high-radix switches







????

$$\text{Bandwidth} \approx \frac{N}{4}$$
$$\text{Latency} = 2 - 4$$
$$\text{Radix} = k$$

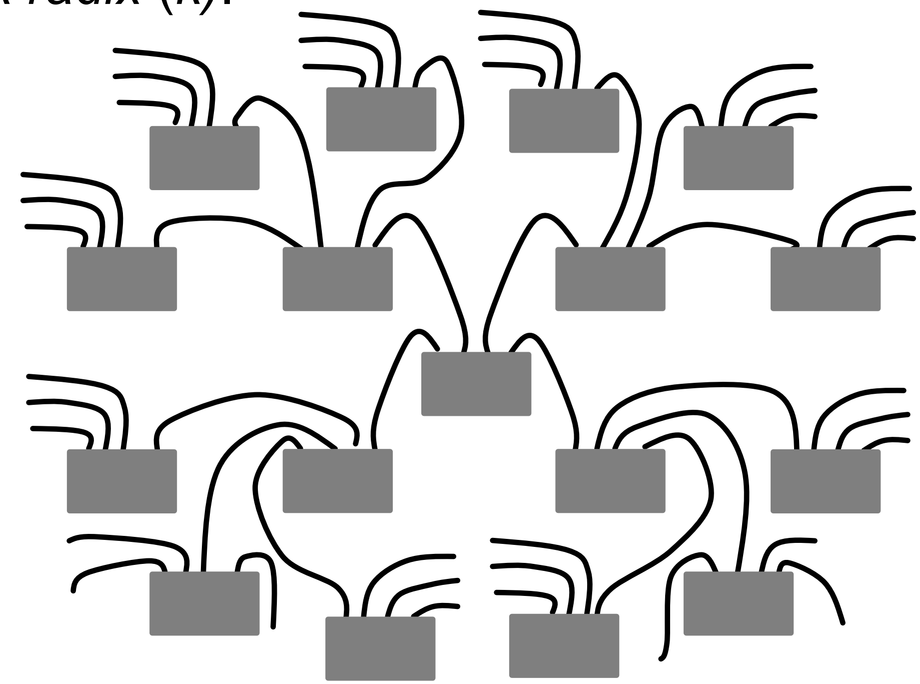
Designing An Efficient Network Topology

Connecting routers

- **Intuition: lower average distance → lower resource needs**
  - A new view as primary optimization target!
- Moore Bound [1]: upper bound on the *number of routers* in a graph with given *diameter* (*D*) and *network radix* (*k*).

$$MB(D,k) = 1 + k + k(k-1) + k(k-1)^2 + \cdots$$

$$MB(D,k) = 1 + k\sum_{i=0}^{D-1}(k-1)^i$$



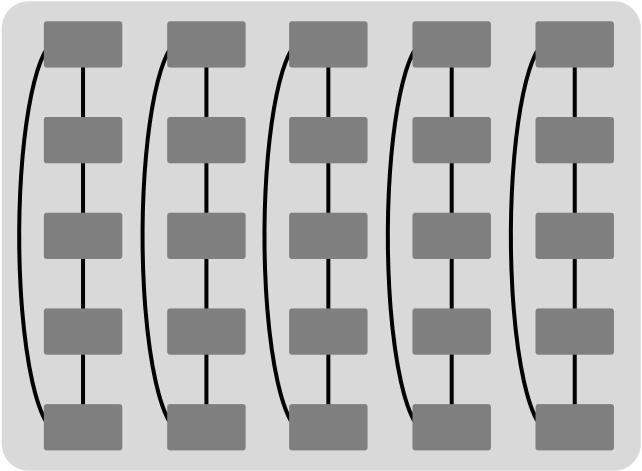
[1] M. Miller, J. Siráň. Moore graphs and beyond: A survey of the degree/diameter problem, Electronic Journal of Combinatorics, 2005.

Designing An Efficient Network Topology

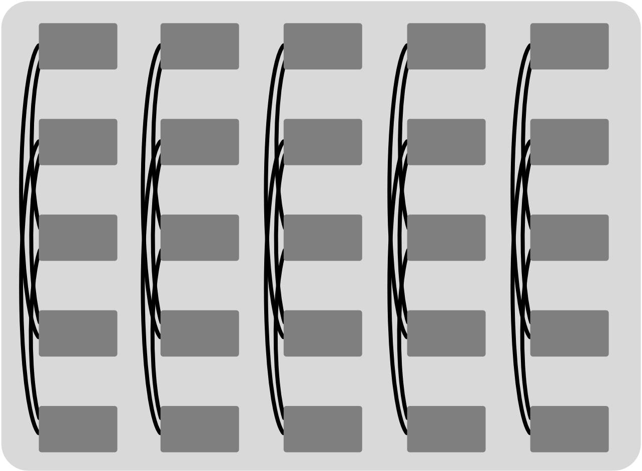
Connecting routers: Diameter 2

- Example Slim Fly design for *diameter* = 2: *MMS graphs* [1] (utilizing graph covering)

A subgraph with
identical groups of routers



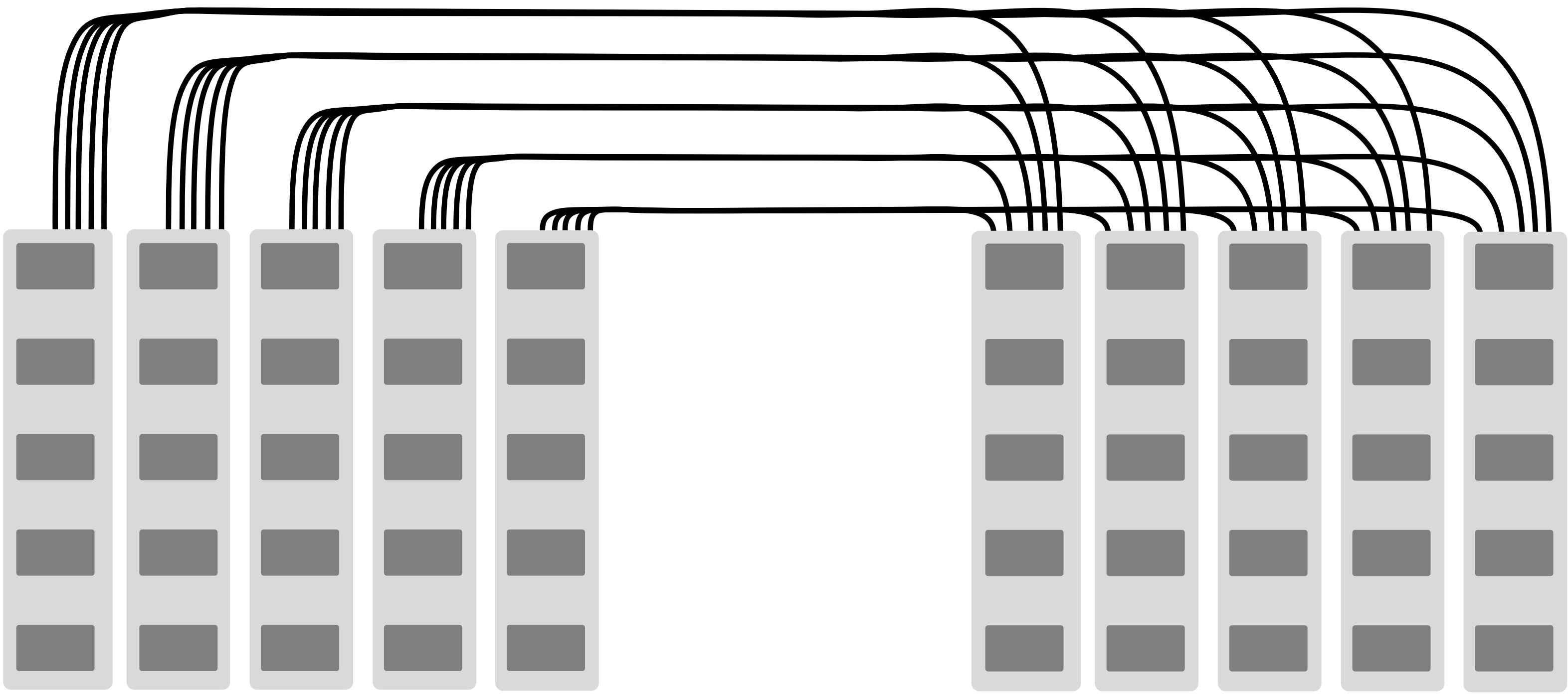
A subgraph with
identical groups of routers



[1] B. D. McKay, M. Miller, and J. Siráň. A note on large graphs of diameter two and given maximum degree. Journal of Combinatorial Theory, Series B, 74(1):110 – 118, 1998

Designing an Efficient Network Topology

Connecting routers: Diameter 2



Groups form a fully-connected bipartite graph

Designing an Efficient Network Topology

Connecting routers: Diameter 2

1 *Select a prime power q*

$$q = 4w + \delta;$$

$w \in \mathbb{N} \quad \delta \in \{-1,0,1\},$

A Slim Fly based on q :

Number of routers: $2q^2$

Network radix: $(3q - \delta)/2$

2 *Construct a finite field $\mathcal{F}_q$.*

Assuming *q* is prime:

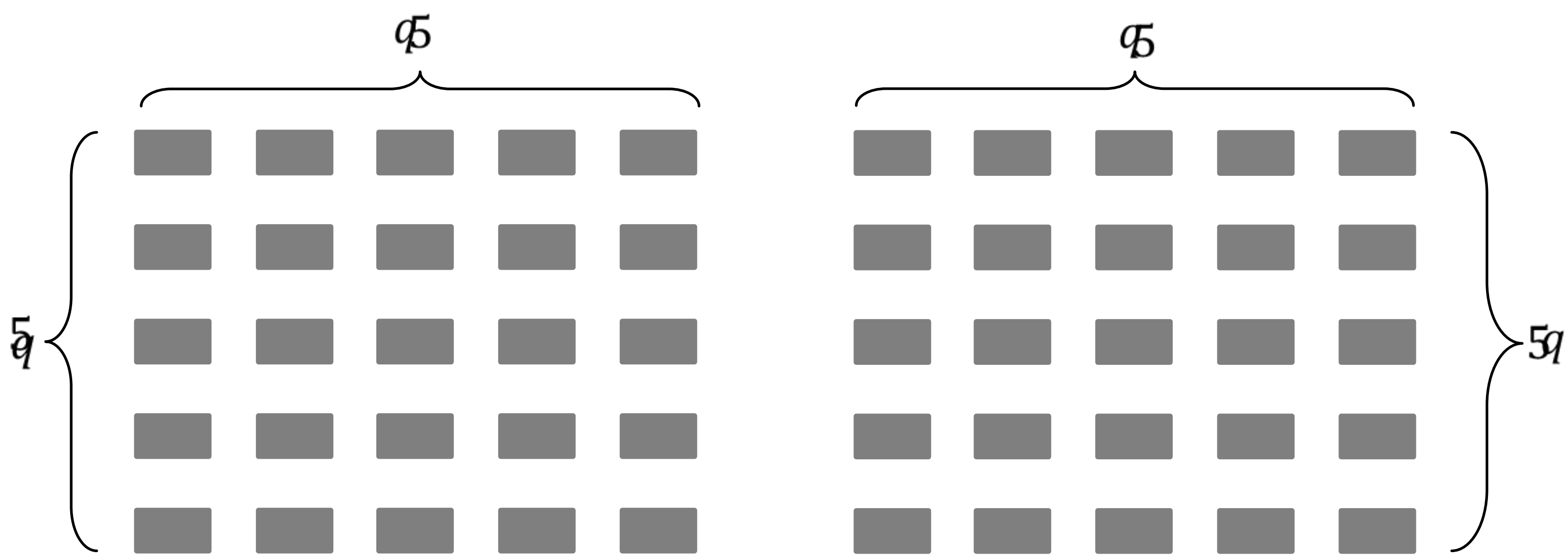
$$\mathcal{F}_q = \mathbb{Z}/q\mathbb{Z} = \{0,1,\dots,q-1\}$$

with modular arithmetic.

E *Example*: $q = 5$

50 routers
network radix: 7

$$\mathcal{F}_5 = \{0,1,2,3,4\}$$



Designing an Efficient Network Topology

Connecting routers: Diameter 2

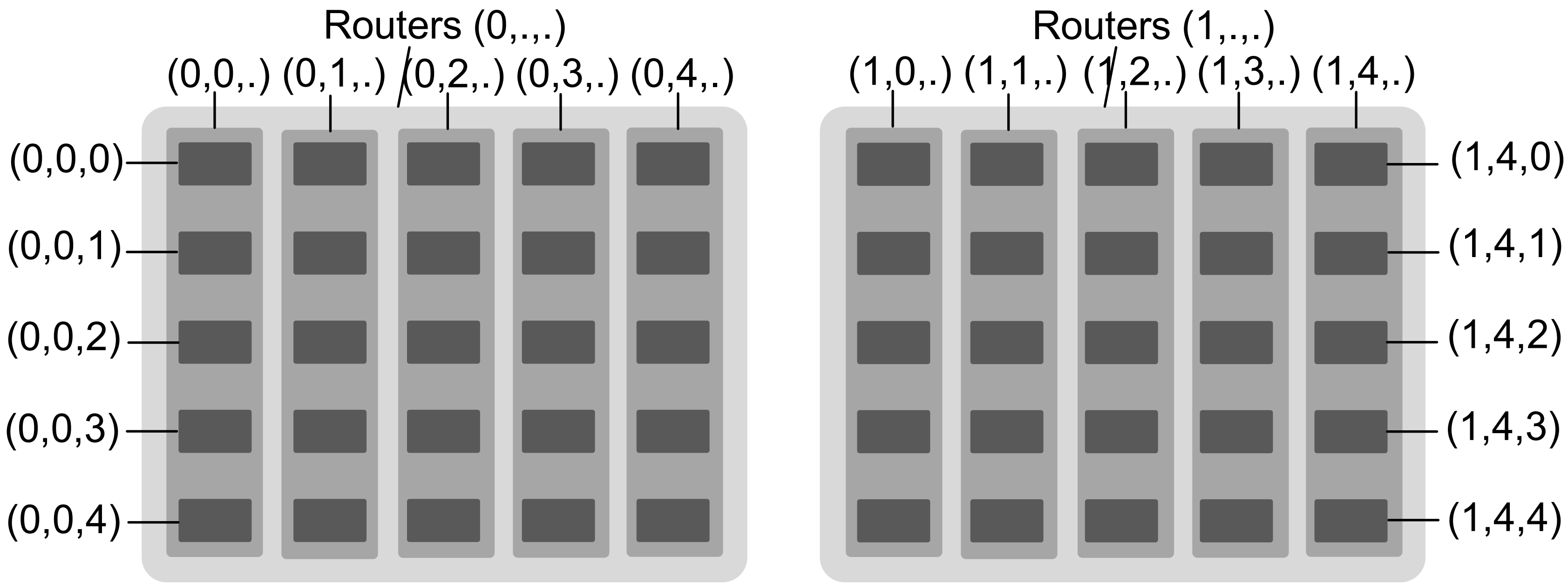
3 *Label the routers*

Set of routers:

$\{0,1\} \times \mathcal{F}_q \times \mathcal{F}_q$

E *Example:* $q = 5$

…



Designing An Efficient Network Topology

Connecting routers: Diameter 2

4 *Find primitive element* ξ

$\xi \in \mathcal{F}_q$ generates $\mathcal{F}_q$:

All non-zero elements of $\mathcal{F}_q$ can be written as $\xi^i;\ i \in \mathbb{N}$

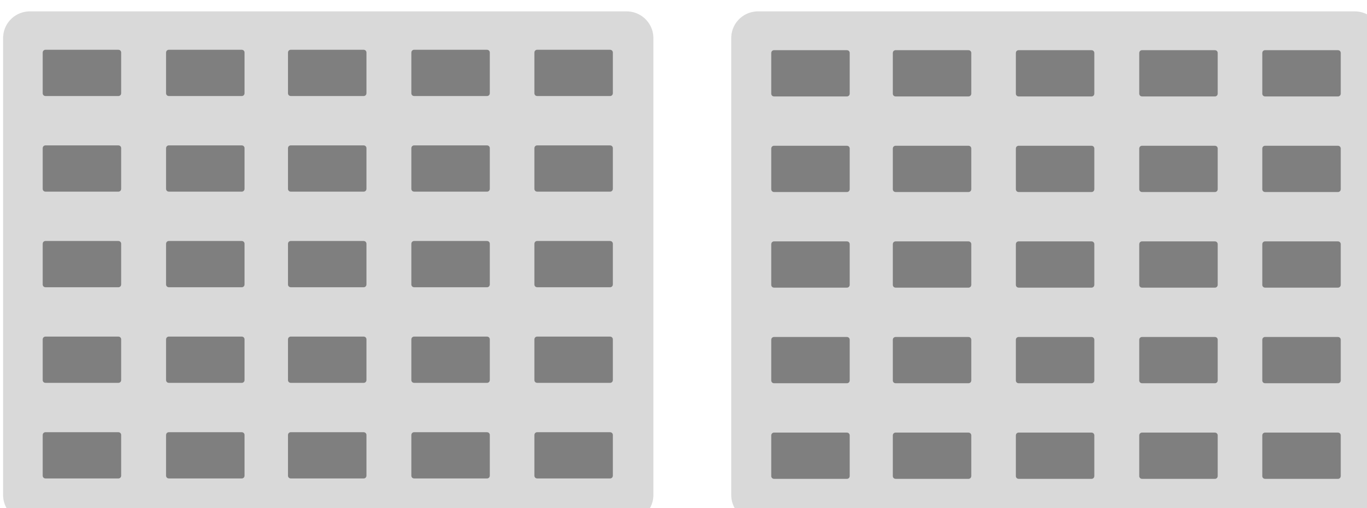
5 *Build Generator Sets*

$$X = \{1, \xi^2, \dots, \xi^{q-3}\}$$
$$X' = \{\xi, \xi^3, \dots, \xi^{q-2}\}$$

E *Example:* $q = 5$

$$\mathcal{F}_5 = \{0,1,2,3,4\}$$
$$\xi = 2$$
$$1 = \xi^4 \bmod 5 = 2^4 \bmod 5 = 16 \bmod 5$$

$$X = \{1,4\}$$
$$X' = \{2,3\}$$



Designing an Efficient Network Topology

Connecting routers: Diameter 2

❻ *Intra-group connections*

Two routers in one group are connected iff their “vertical Manhattan distance” is an element from:

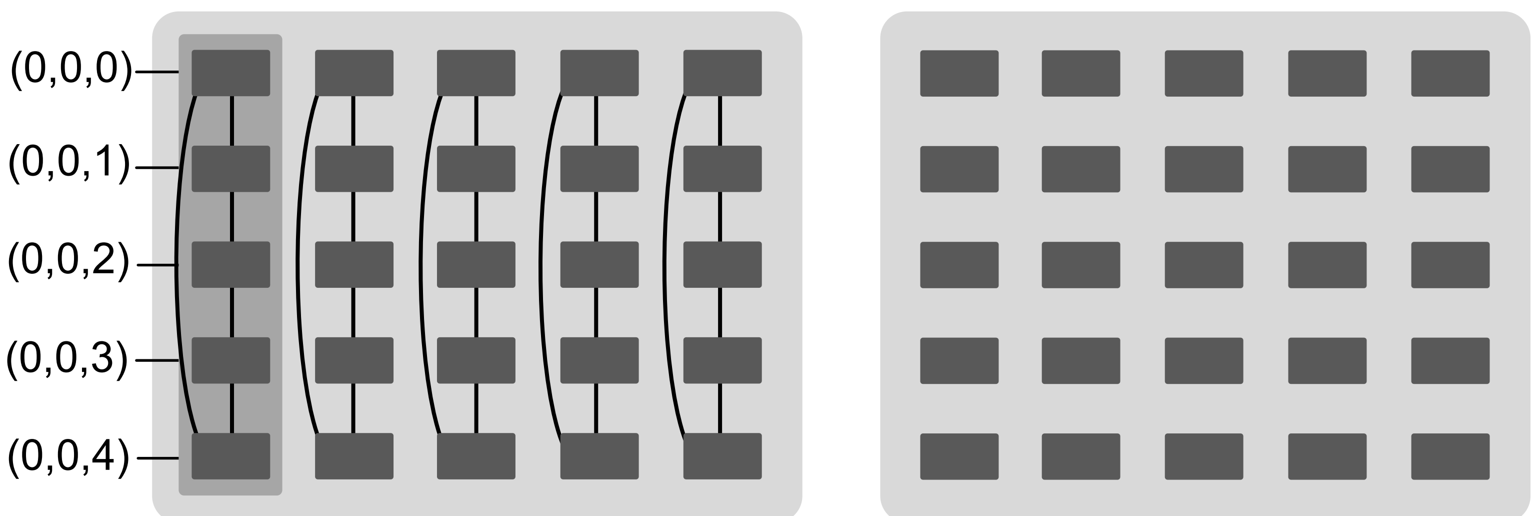
$$X = \{1, \xi^2, \ldots, \xi^{q-3}\} \text{ (for subgraph 0)}$$

$$X' = \{\xi, \xi^3, \ldots, \xi^{q-2}\} \text{ (for subgraph 1)}$$

Ⓔ *Example:* $q = 5$

Take Routers $(0,0,.)$

$$X = \{1, 4\}$$



Designing an Efficient Network Topology

Connecting routers: Diameter 2

6 *Intra-group connections*

Two routers in one group are connected iff their “vertical Manhattan distance” is an element from:

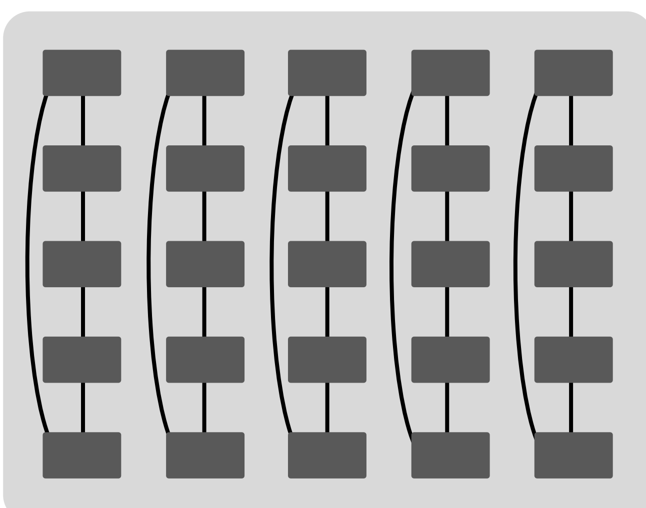
$X = \{1, \xi^2, \dots, \xi^{q-3}\}$ (for subgraph 0)

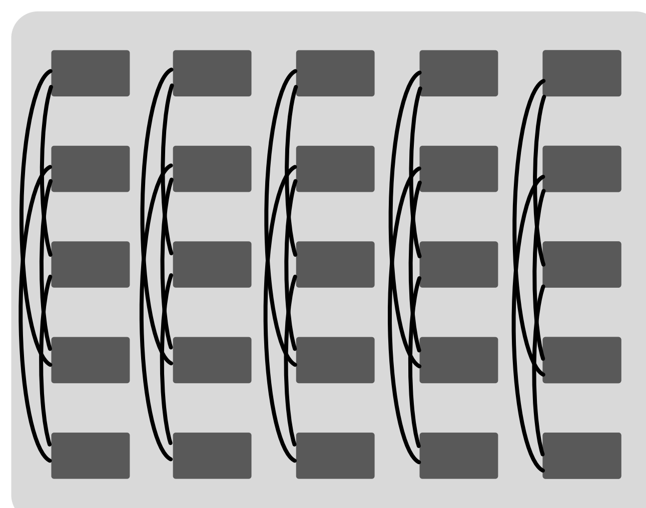
$X' = \{\xi, \xi^3, \dots, \xi^{q-2}\}$ (for subgraph 1)

E *Example:* $q = 5$

Take Routers $(1,4,.)$

$X' = \{2,3\}$





Designing An Efficient Network Topology

Connecting routers: Diameter 2

7 *Inter-group connections*

Router $(0, x, y) \leftrightarrow (1, m, c)$

iff $y = mx + c$

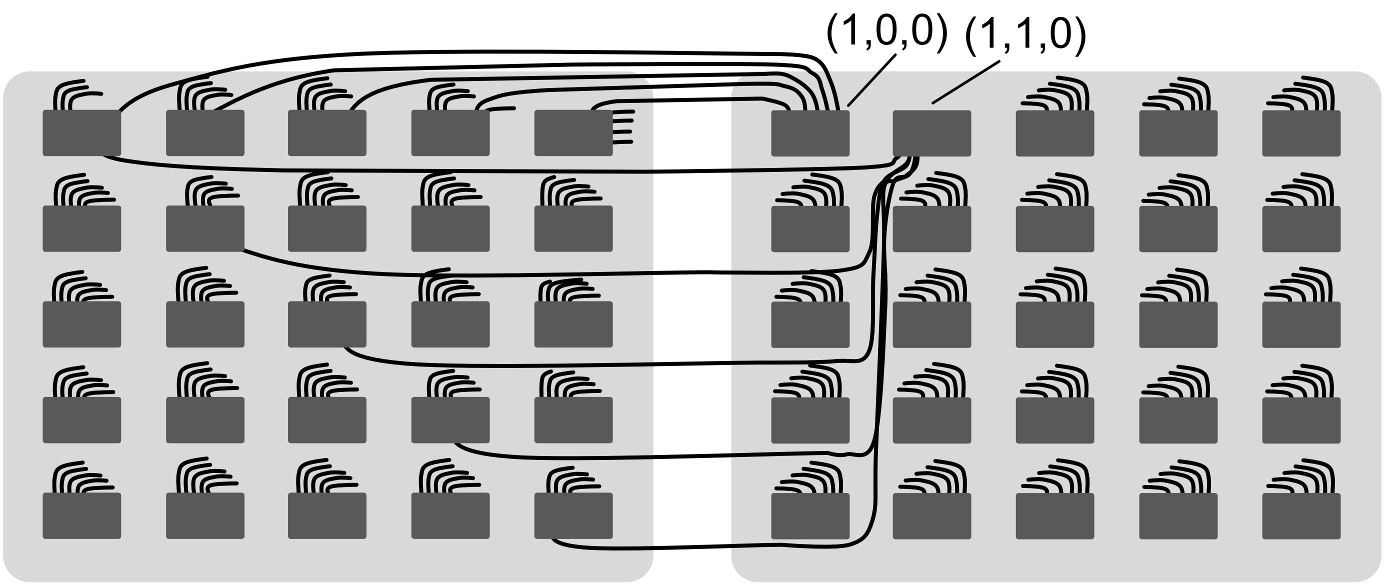
E *Example:* $q = 5$

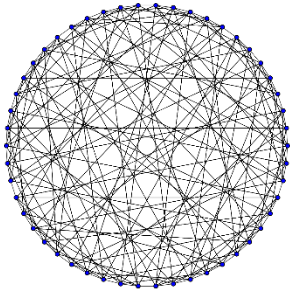
Take Router $(1{,}0{,}0)$ $m = 0, c = 0$

$(1{,}0{,}0) \leftrightarrow (0, x, 0)$

Take Router $(1{,}1{,}0)$ $m = 1, c = 0$

$(1{,}1{,}0) \leftrightarrow (0, x, x)$

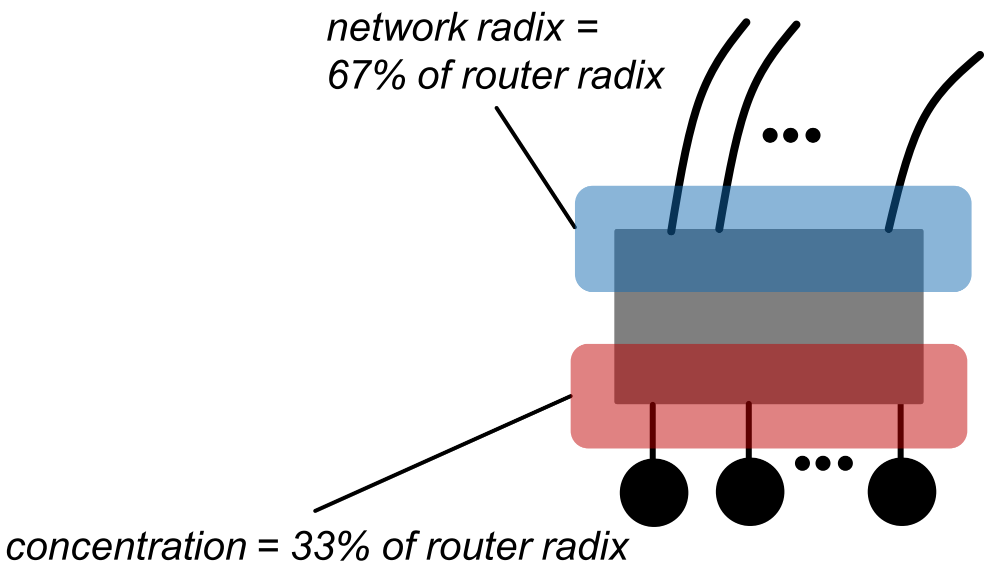




Designing An Efficient Network Topology

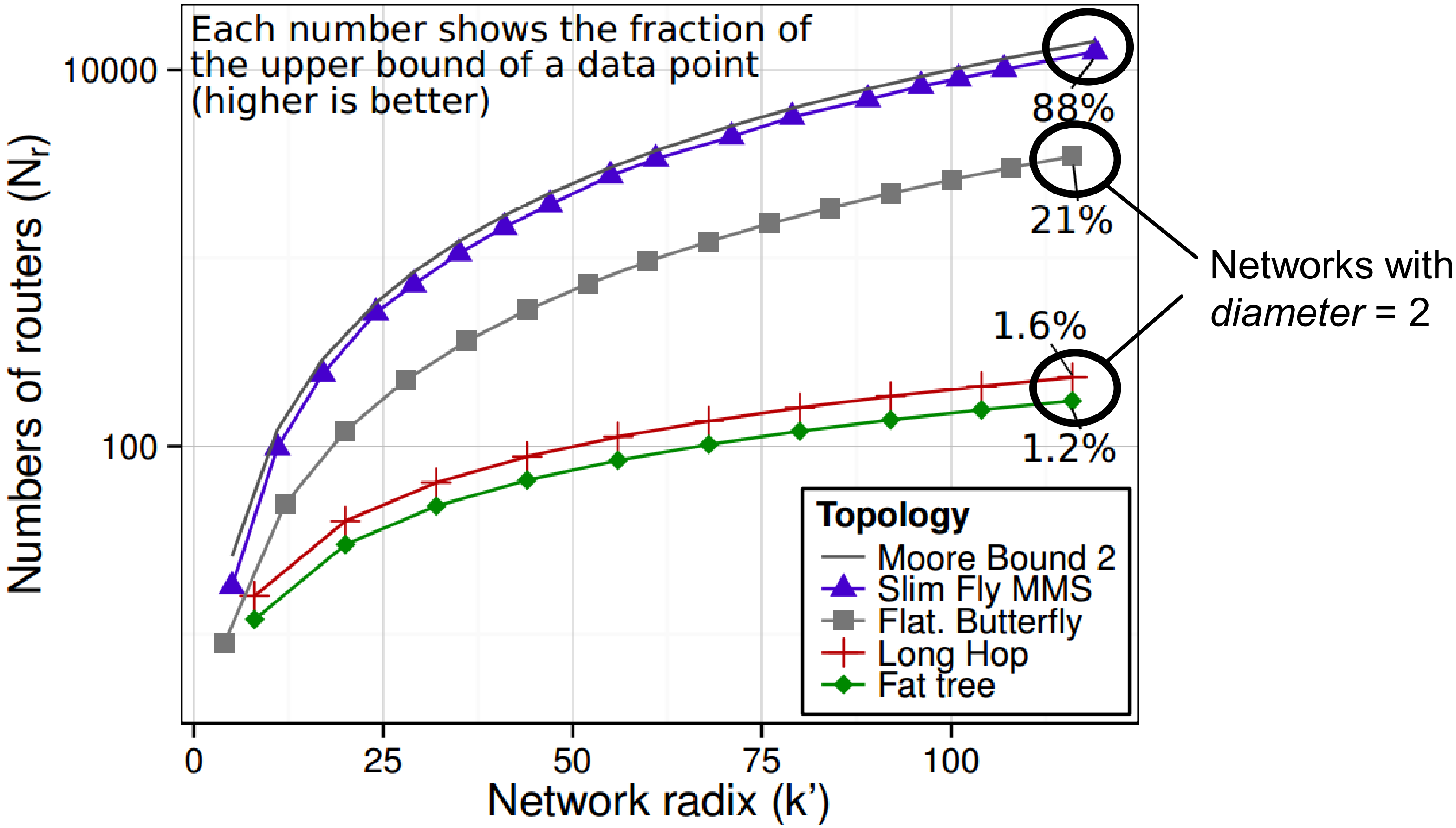
Attaching endpoints: Diameter 2

- How many endpoints do we attach to each router?
- As many to ensure *full global bandwidth:*
  - Global bandwidth: the theoretical cumulative throughput if all endpoints simultaneously communicate with all other endpoints in a steady state



Comparison to Optimality

- How close is the presented Slim Fly network to the Moore Bound?



Overview of our Research

Topology design

Optimizing towards Moore Bound

Attaching endpoints

Comparison of optimality

Cost, power, resilience analysis

Physical layout

Cost model

Comparison targets

Cost & power results

Detailed case-study

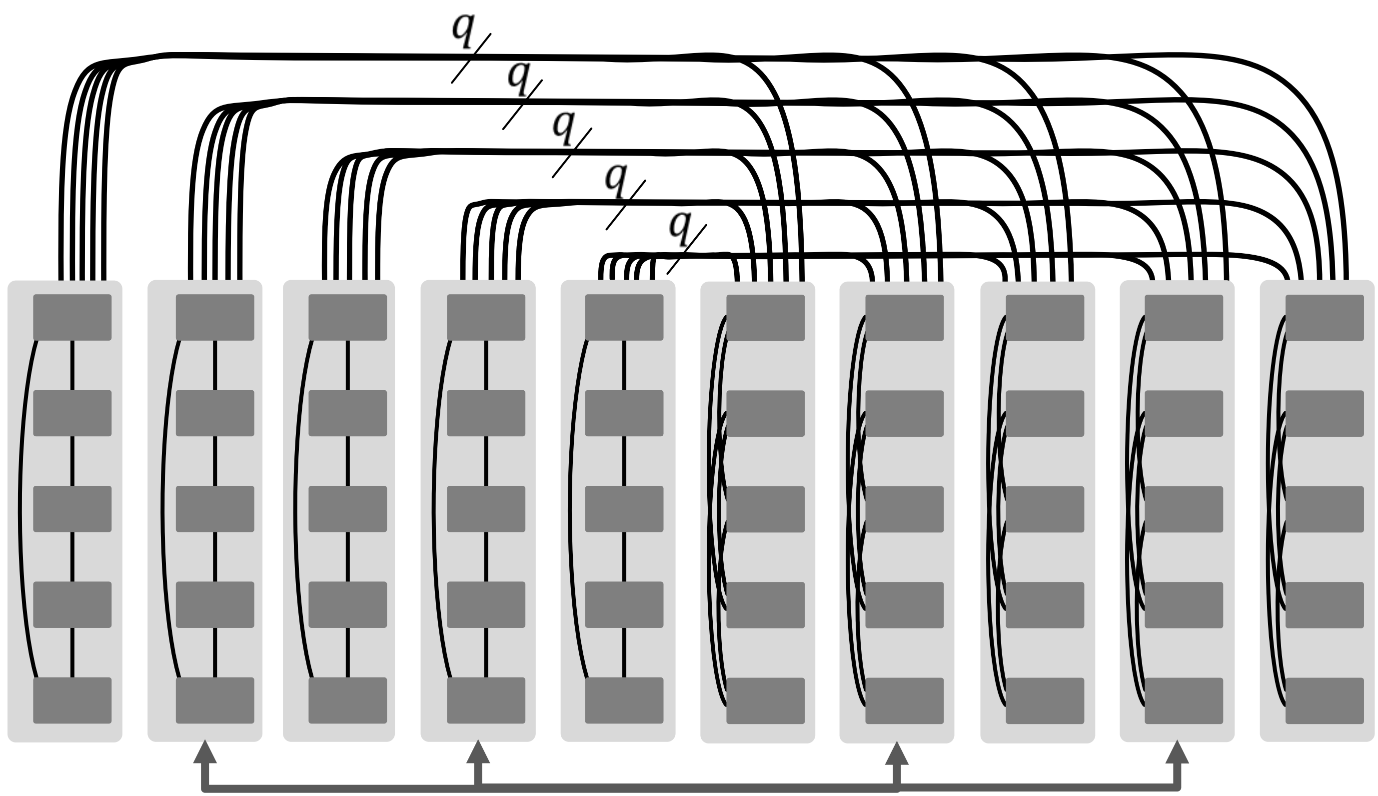
Resilience

Routing and performance

Routing

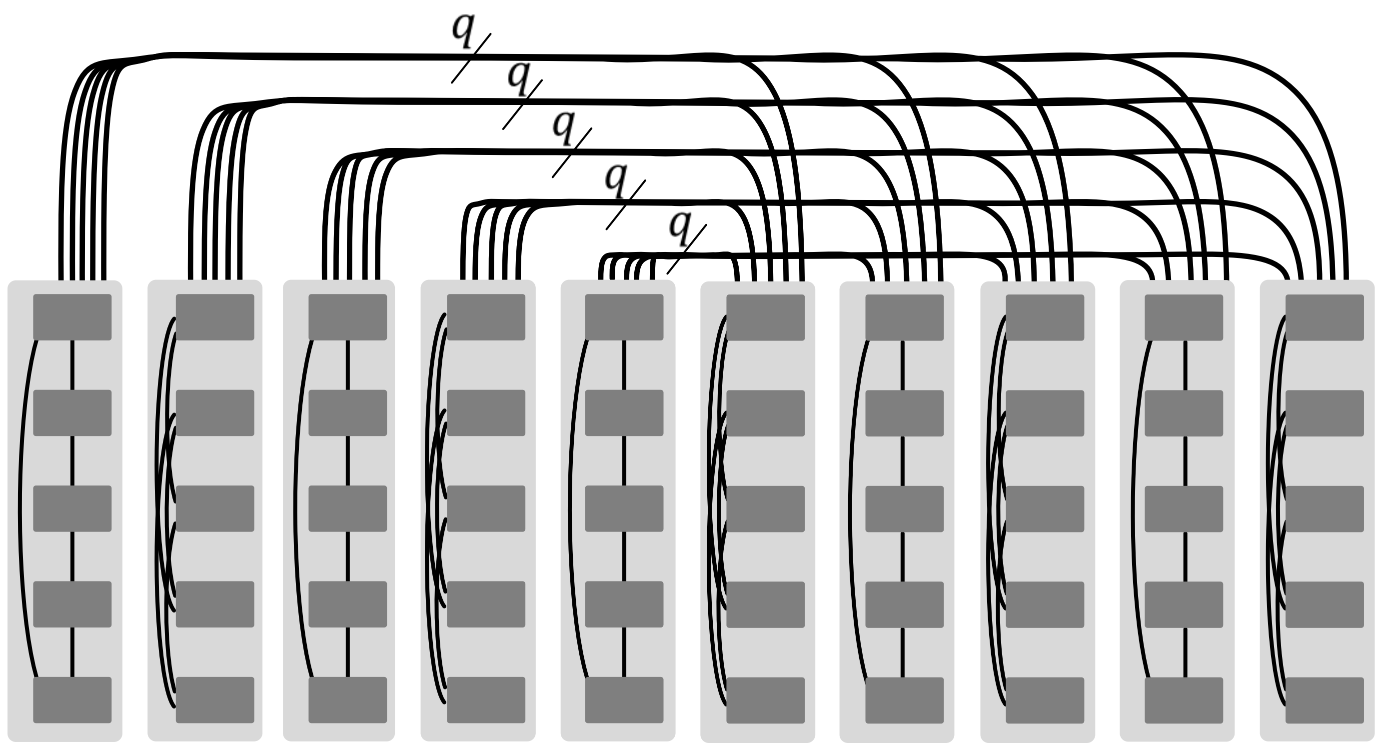
Performance, latency, bandwidth

Physical Layout

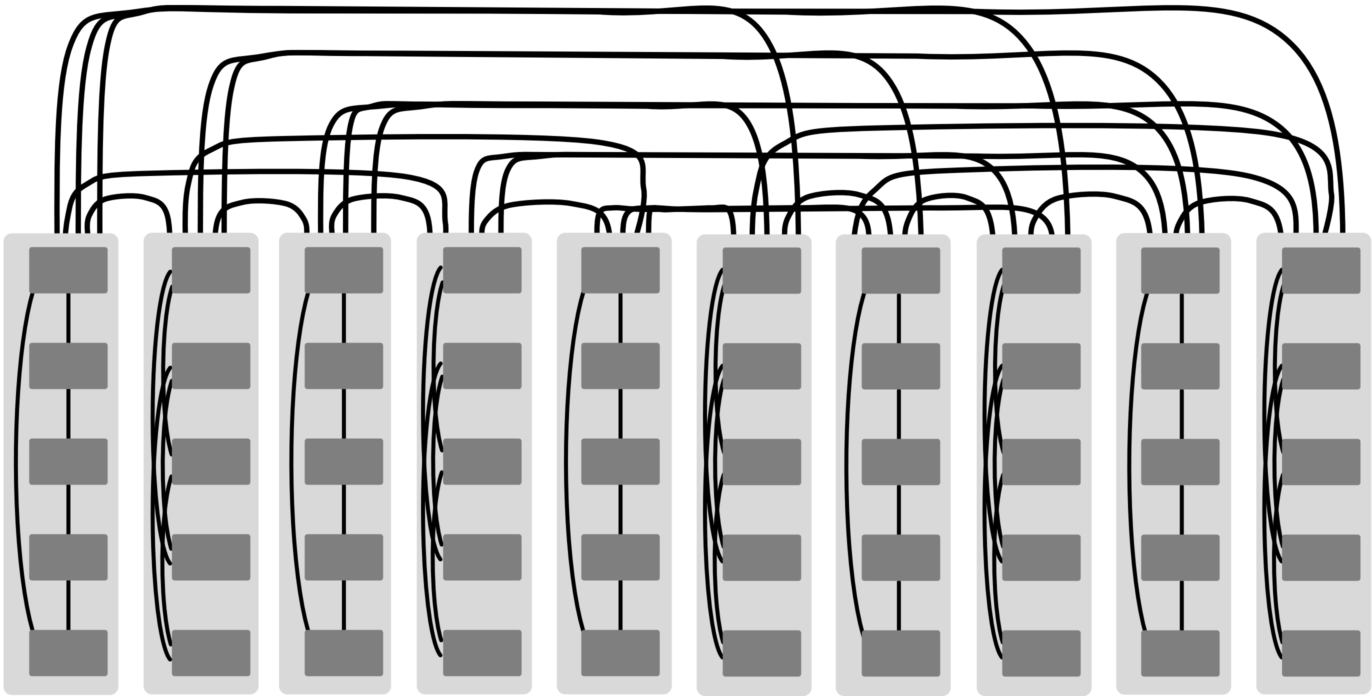


Mix (pairwise) groups
with different cabling patterns
to shorten inter-group cables

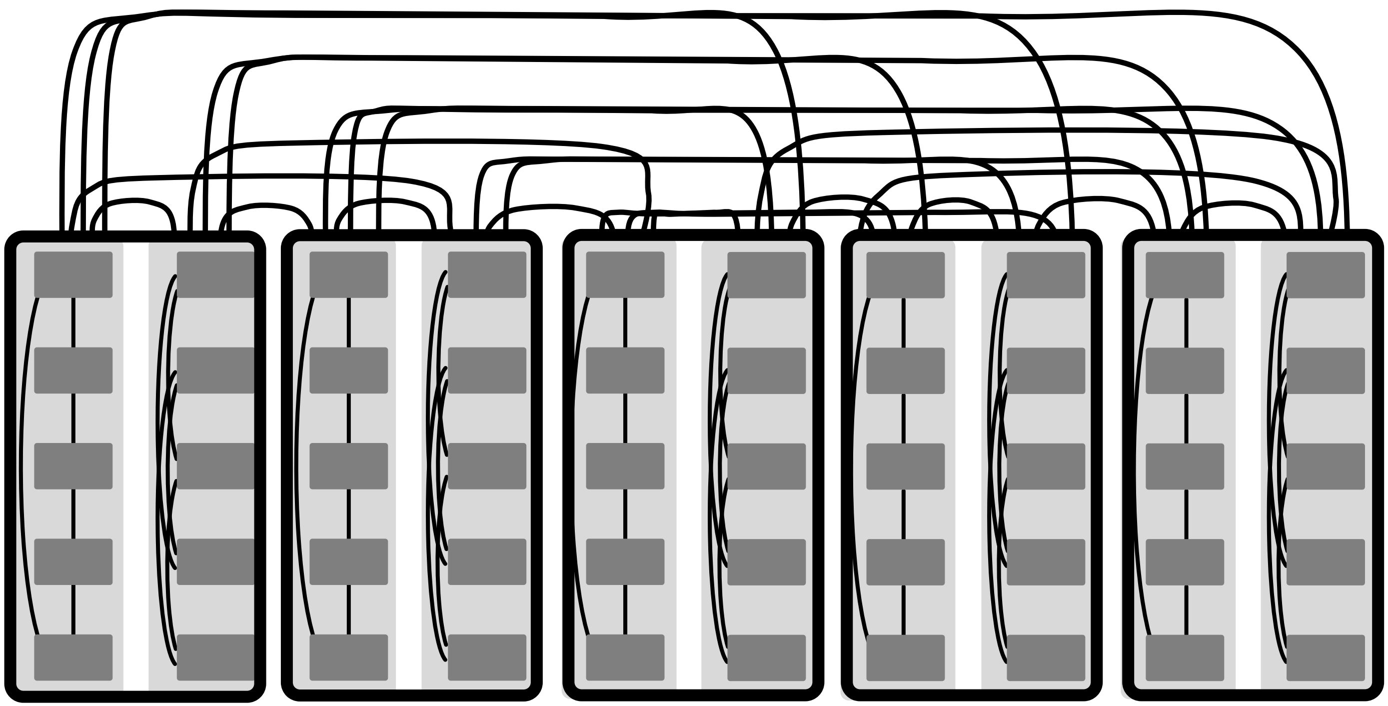
Physical Layout



Physical Layout

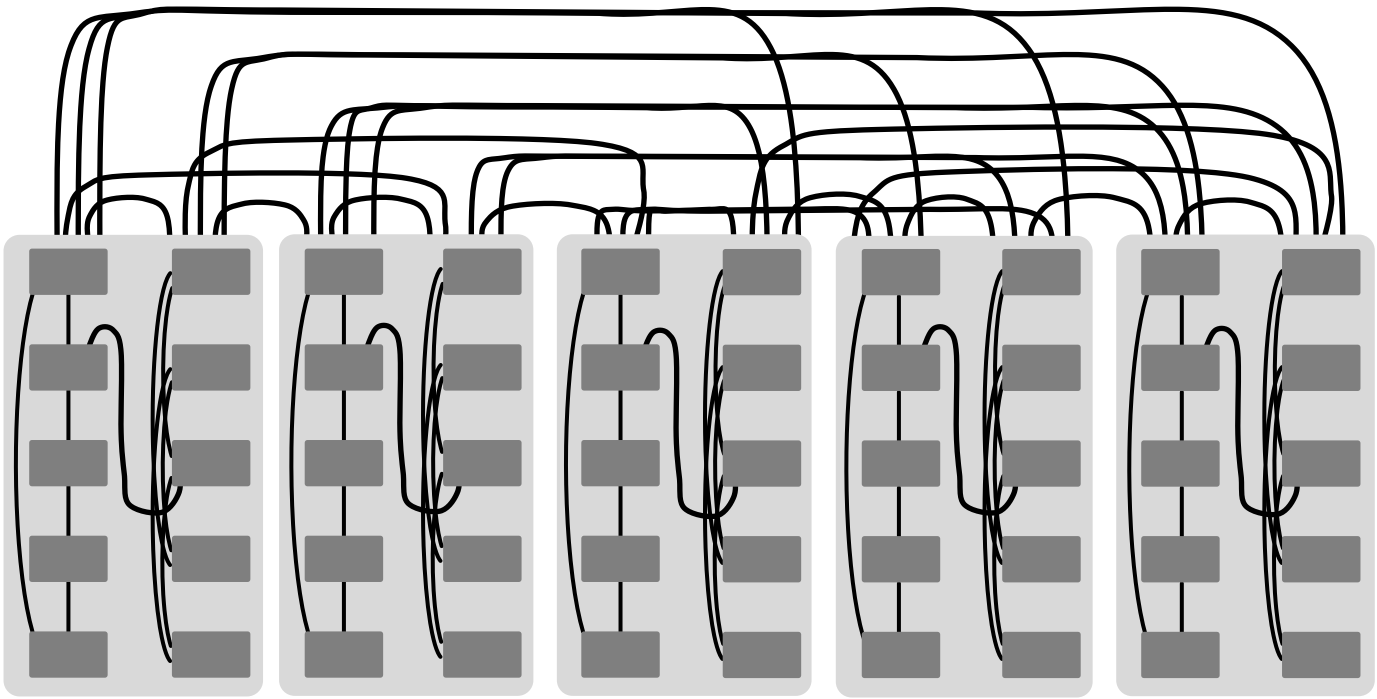


Physical Layout

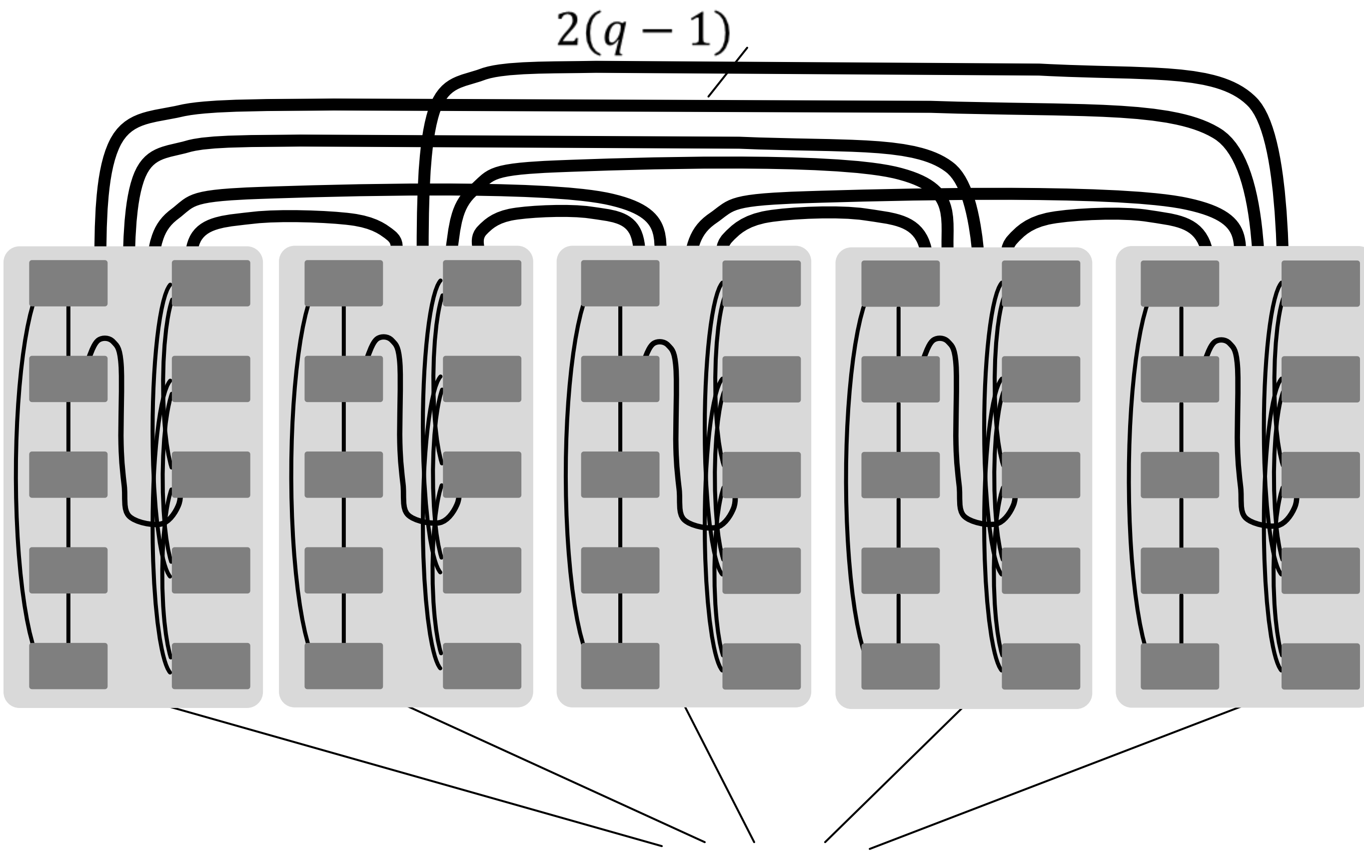


Merge groups pairwise
to create racks

Physical Layout

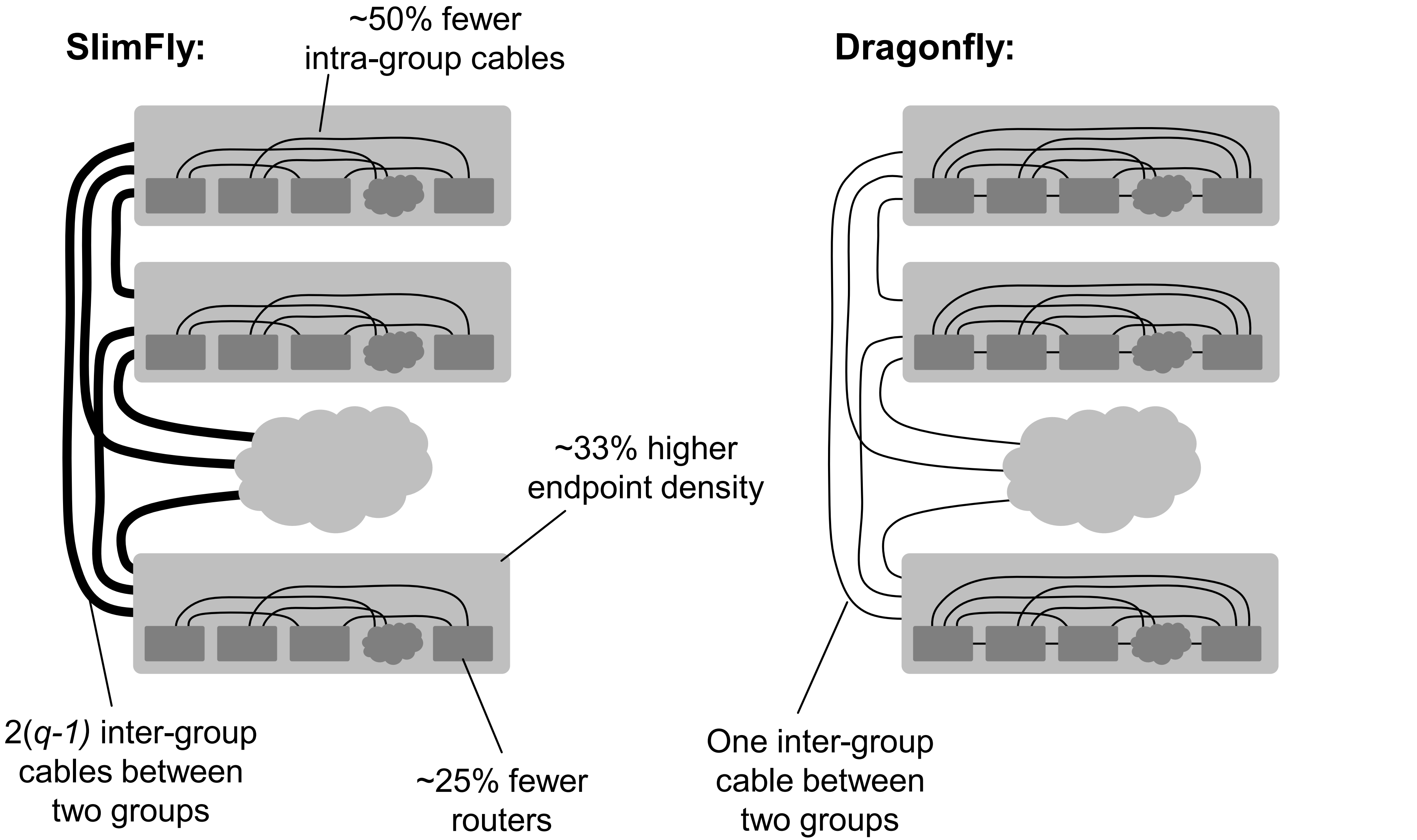


Physical Layout



Racks form
a fully-connected graph

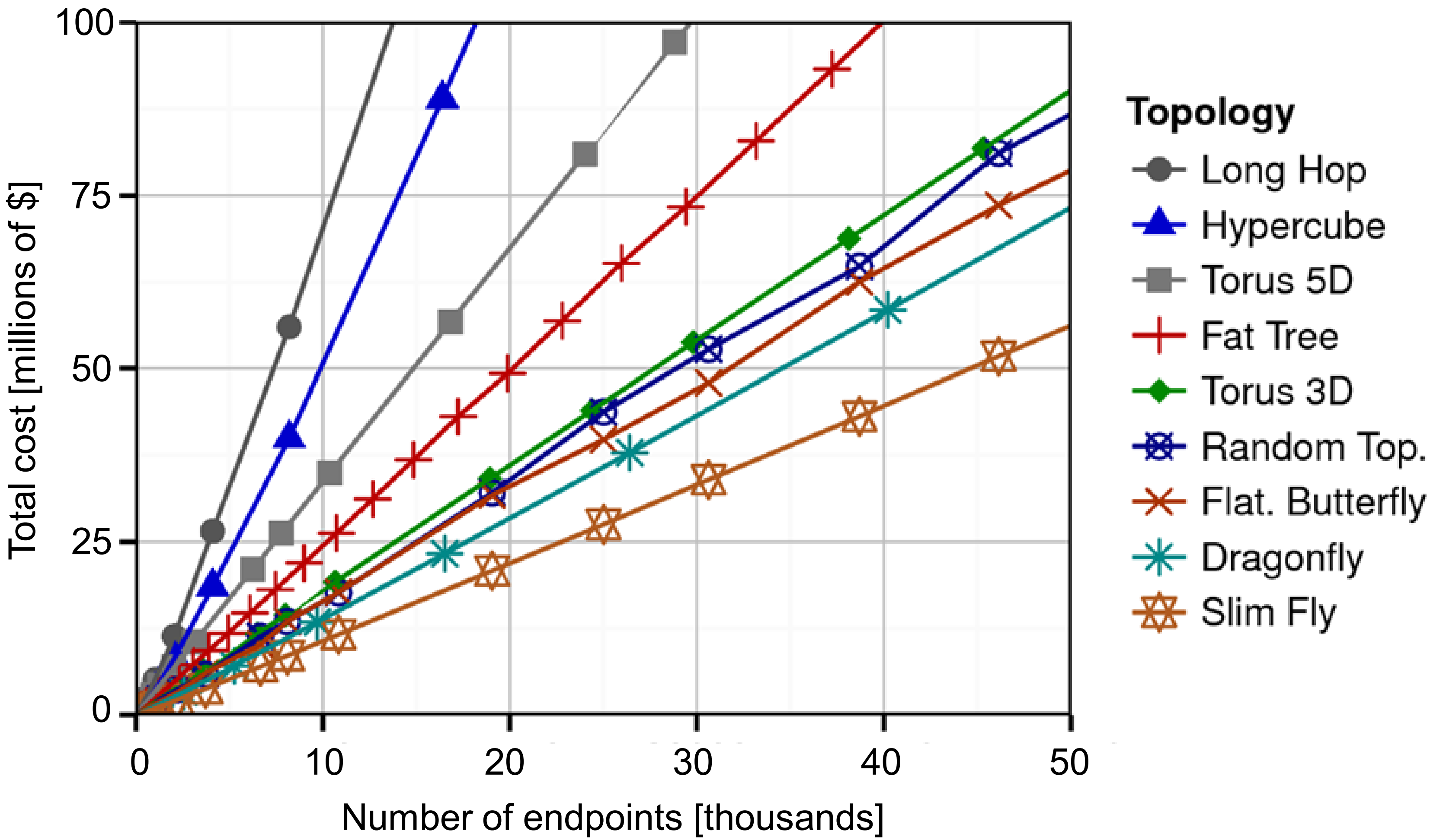
Physical Layout



Cost Comparison

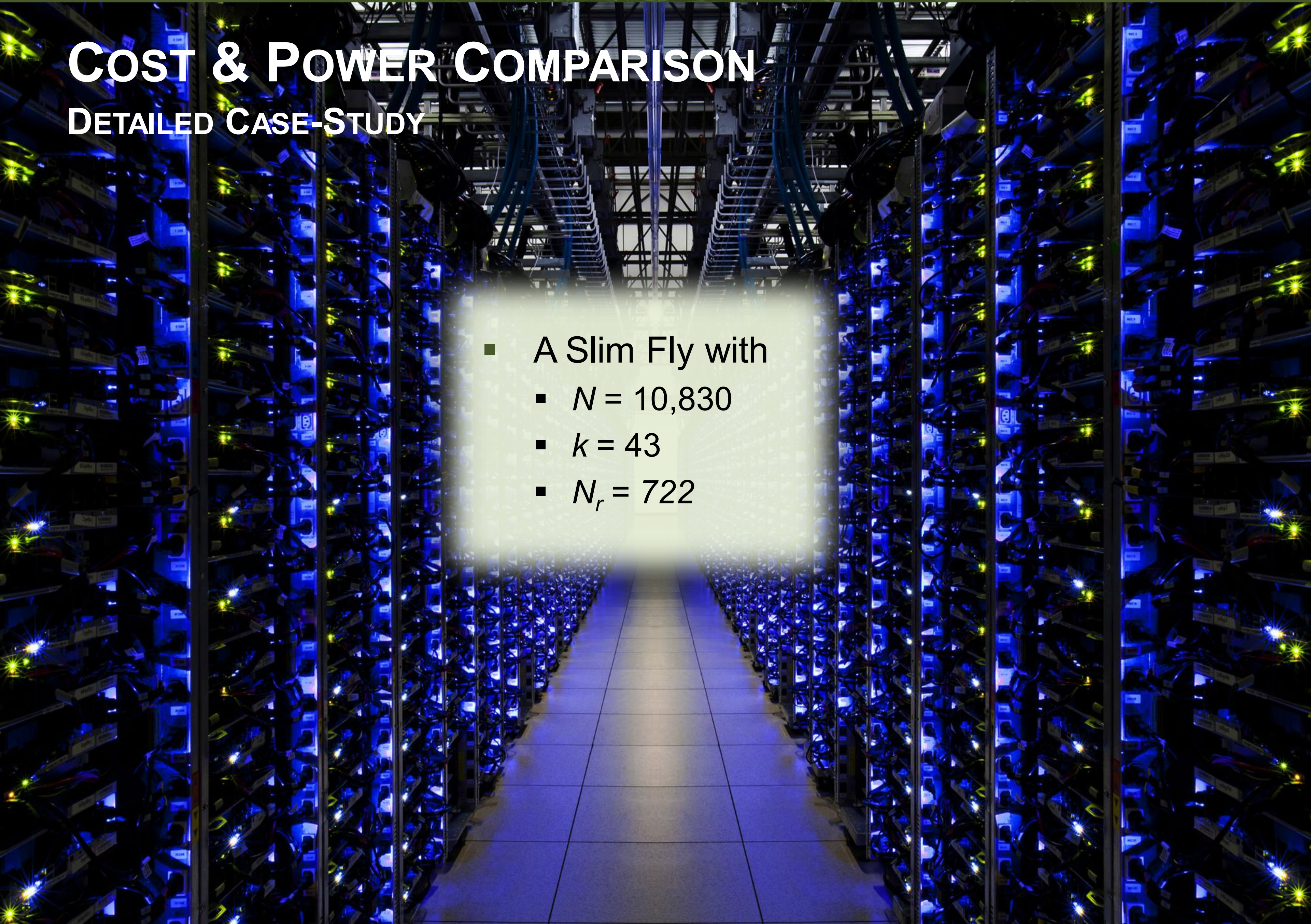
Results

Assuming COTS material costs and best known layout for each topology!



Cost & Power Comparison

Detailed Case-Study



- A Slim Fly with
  - N = 10,830
  - k = 43
  - N_r = 722

Cost & Power Comparison

Detailed Case-Study: High-Radix Topologies

Topology	Fat tree	Random	Flat. Butterfly	Dragonfly	**Slim Fly**
Endpoints (N)	19,876	40,200	20,736	58,806	**10,830**
Routers (N_r)	2,311	4,020	1,728	5,346	**722**
Radix (k)	**43**	**43**	**43**	**43**	**43**
Electric cables	19,414	32,488	9,504	56,133	**6,669**
Fiber cables	40,215	33,842	20,736	29,524	**6,869**
Cost per node [$]	2,346	1,743	1,570	1,438	**1,033**
Power per node [W]	14.0	12.04	10.8	10.9	**8.02**

Topology	Fat tree	Random	Flat. Butterfly	Dragonfly	**Slim Fly**
Endpoints (N)	**10,718**	**9,702**	**10,000**	**9,702**	**10,830**
Routers (N_r)	1,531	1,386	1,000	1,386	**722**
Radix (k)	35	28	33	27	**43**
Electric cables	7,350	6,837	4,500	9,009	**6,669**
Fiber cables	24,806	7,716	10,000	4,900	**6,869**
Cost per node [$]	2,315	1,566	1,535	1,342	**1,033**
Power per node [W]	14.0	11.2	10.8	10.8	**8.02**

OVERVIEW OF OUR RESEARCH

Topology design

Optimizing towards Moore Bound

Attaching endpoints

Comparison of optimality

Cost, power, resilience analysis

Physical layout

Cost model

Comparison targets

Cost & power results

Detailed case-study

Resilience

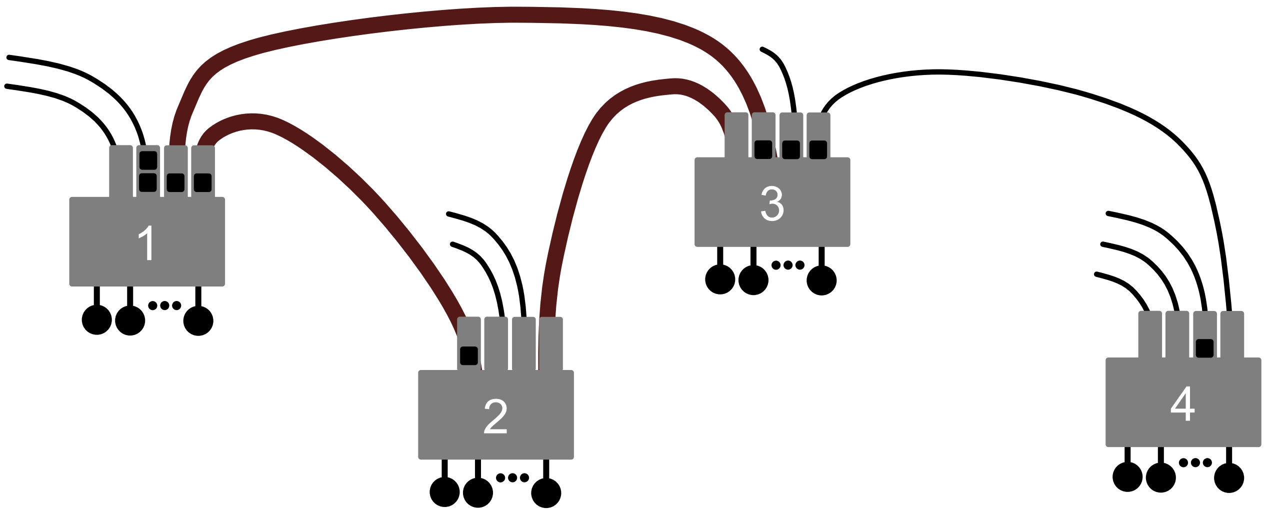
Routing and performance

Routing

Performance, latency, bandwidth

PERFORMANCE & ROUTING

- Cycle-accurate simulations [1]
- Routing protocols:
  - Minimum static routing
  - Valiant routing [2]
  - Universal Globally-Adaptive Load-Balancing routing [3]
    UGAL-L: each router has access to its local output queues
    UGAL-G: each router has access to the sizes of all router queues in the network

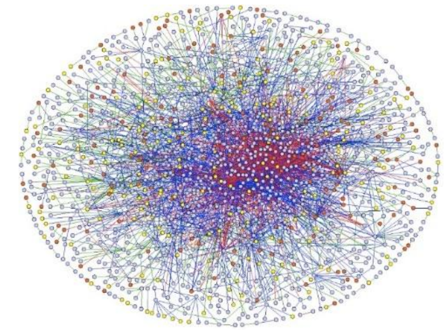


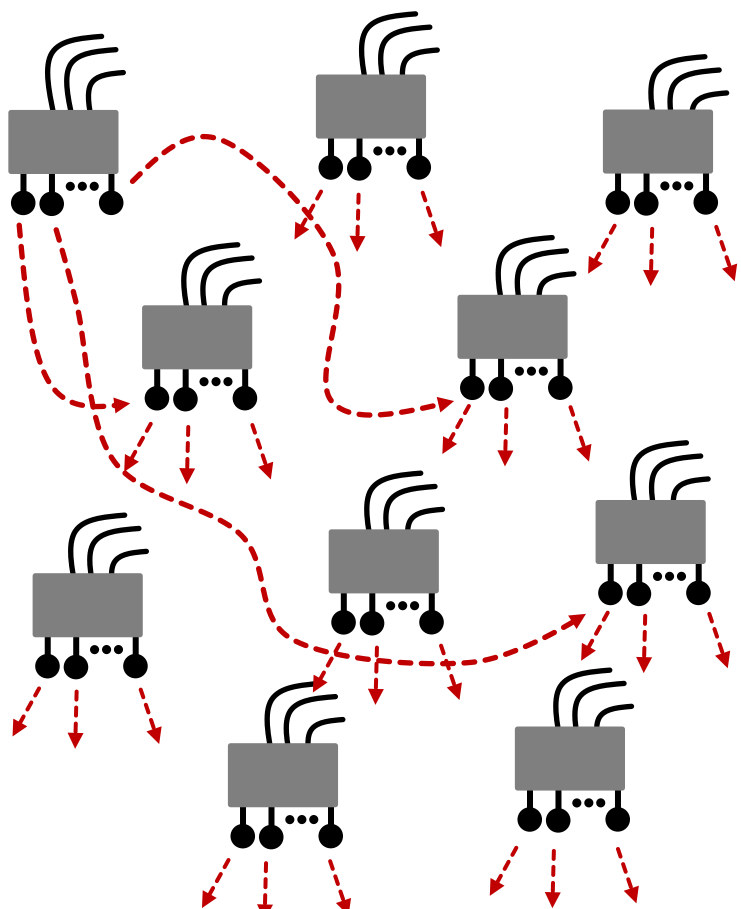
[1] N. Jiang et al. A detailed and flexible cycle-accurate Network-on-Chip simulator. ISPASS'13
[2] L. Valiant. A scheme for fast parallel communication. SIAM journal on computing, 1982
[3] A. Singh. Load-Balanced Routing in Interconnection Networks. PhD thesis, Stanford University, 2005

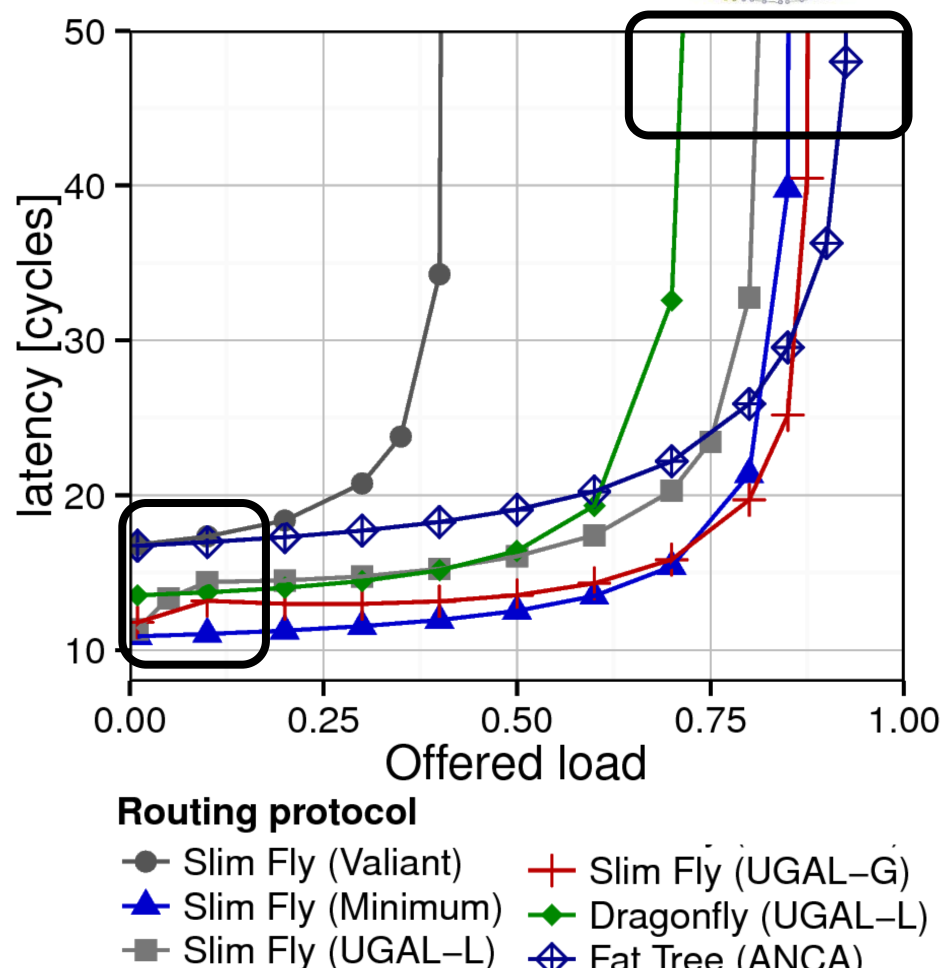
Performance & routing

Random uniform traffic

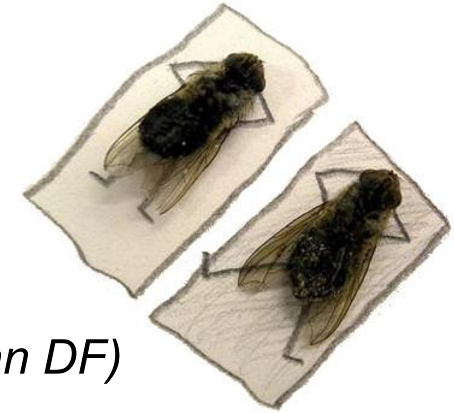
GRAPH 500



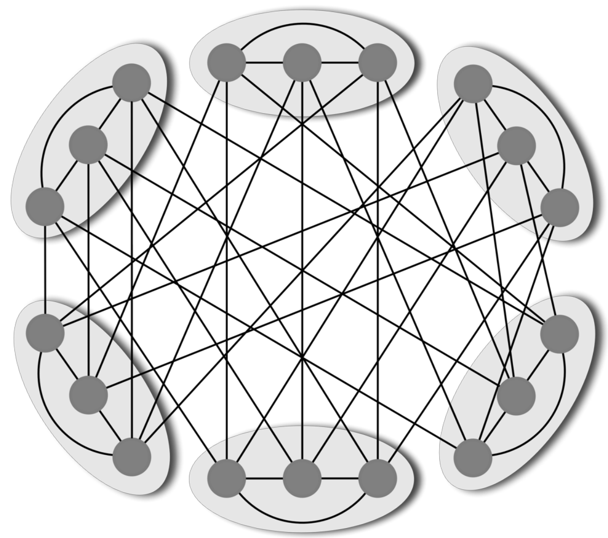




Intermediate conclusions



- **We have:**
  - The cheapest full-bandwidth topology (25% less than DF)
    Basing on group theory, large number of options (more than DF)
  - Requires advanced routing techniques (adaptive)
    Works somewhat with next-gen IB, we work on Ethernet solutions

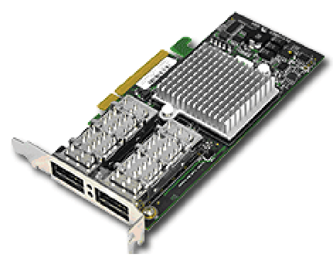


- **Is that all?**
  - No – the endpoint is actually most (more?) important for performance!
  - So let’s see ….

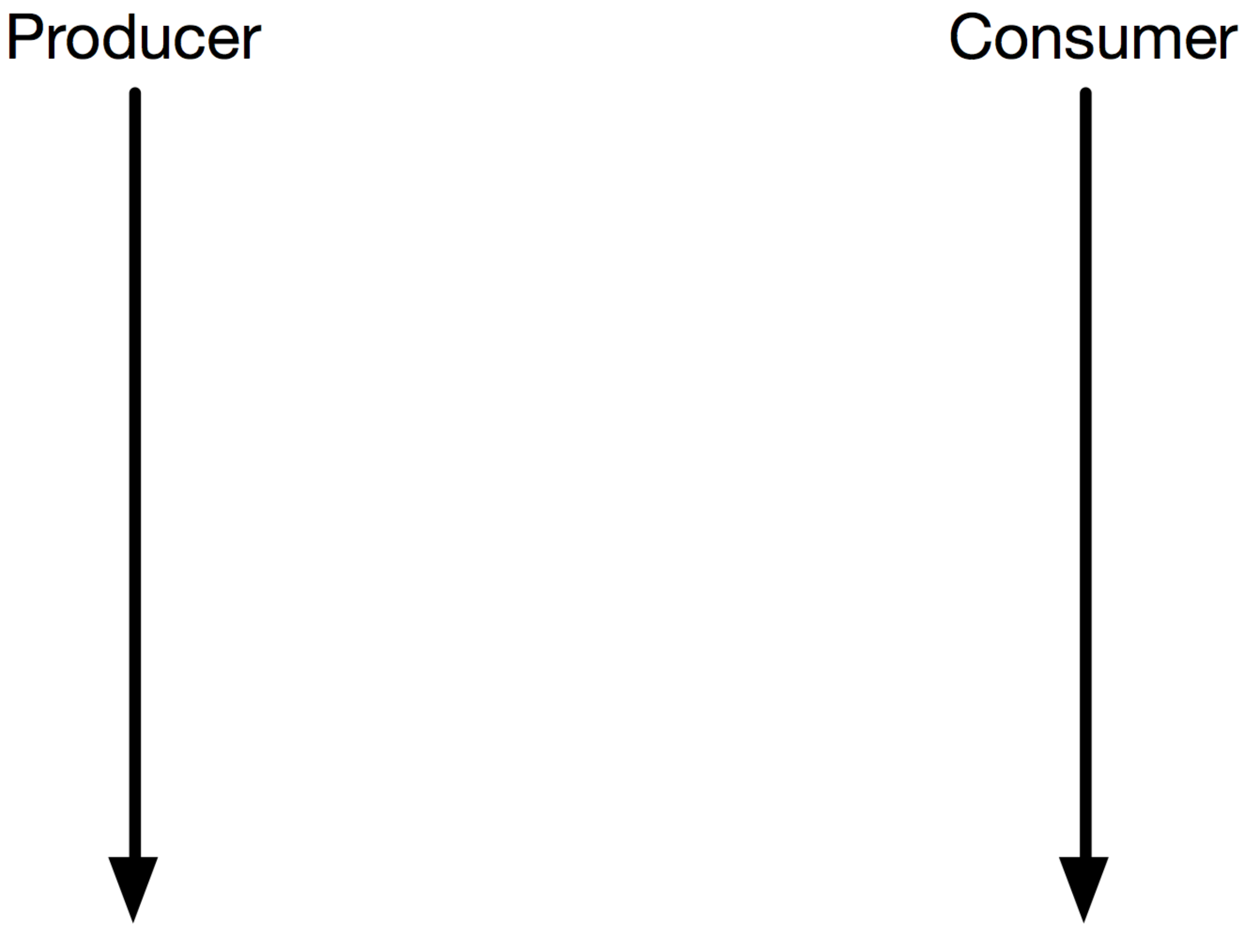
Communication in Today's HPC Systems

- **The de-facto programming model: MPI-1**
  - Using send/recv messages and collectives

- **The de-facto network standard: RDMA, SHM**
  - Zero-copy, user-level, os-bypass, fuzz-bang

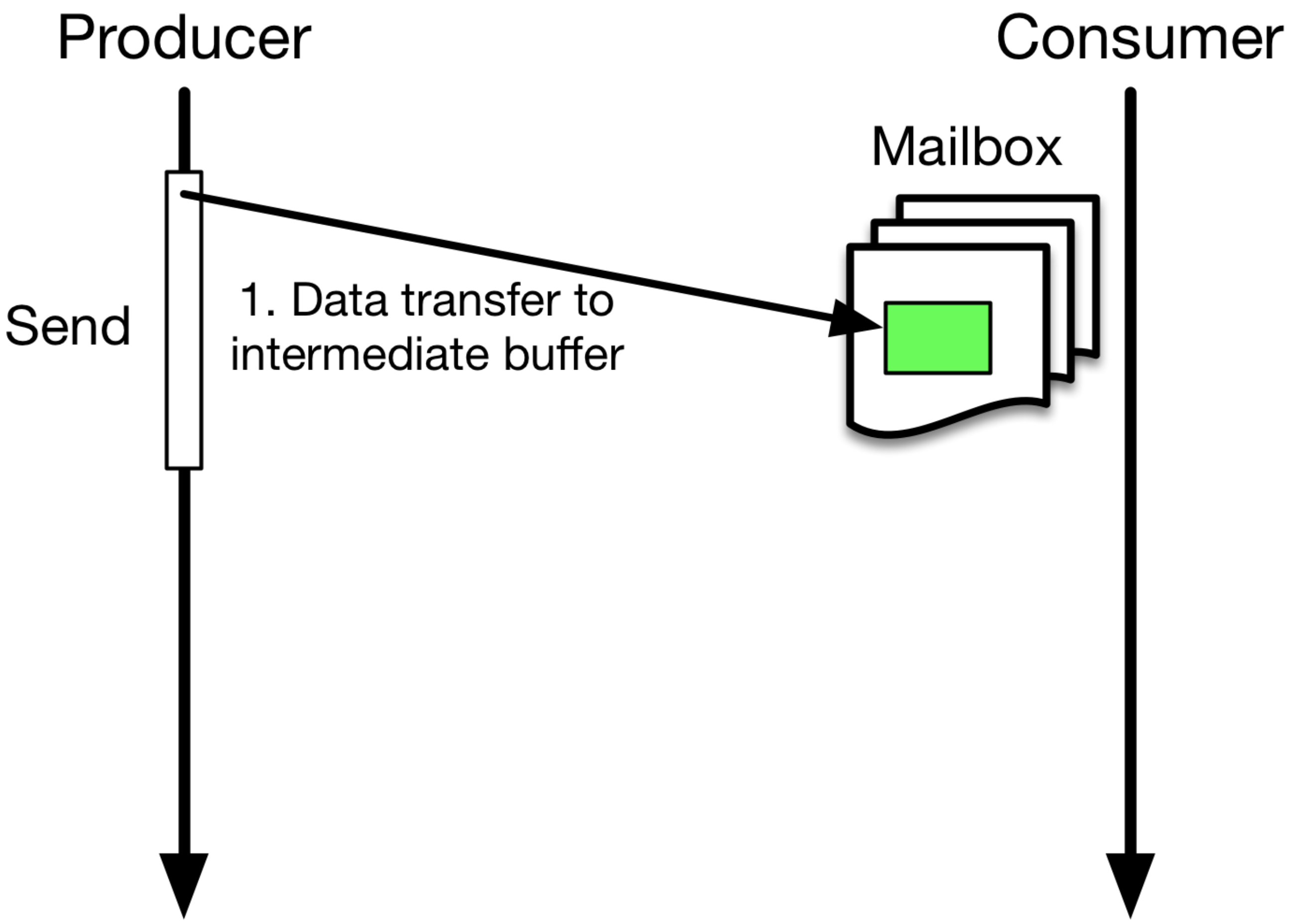


MPI-1 Message Passing – Simple Eager



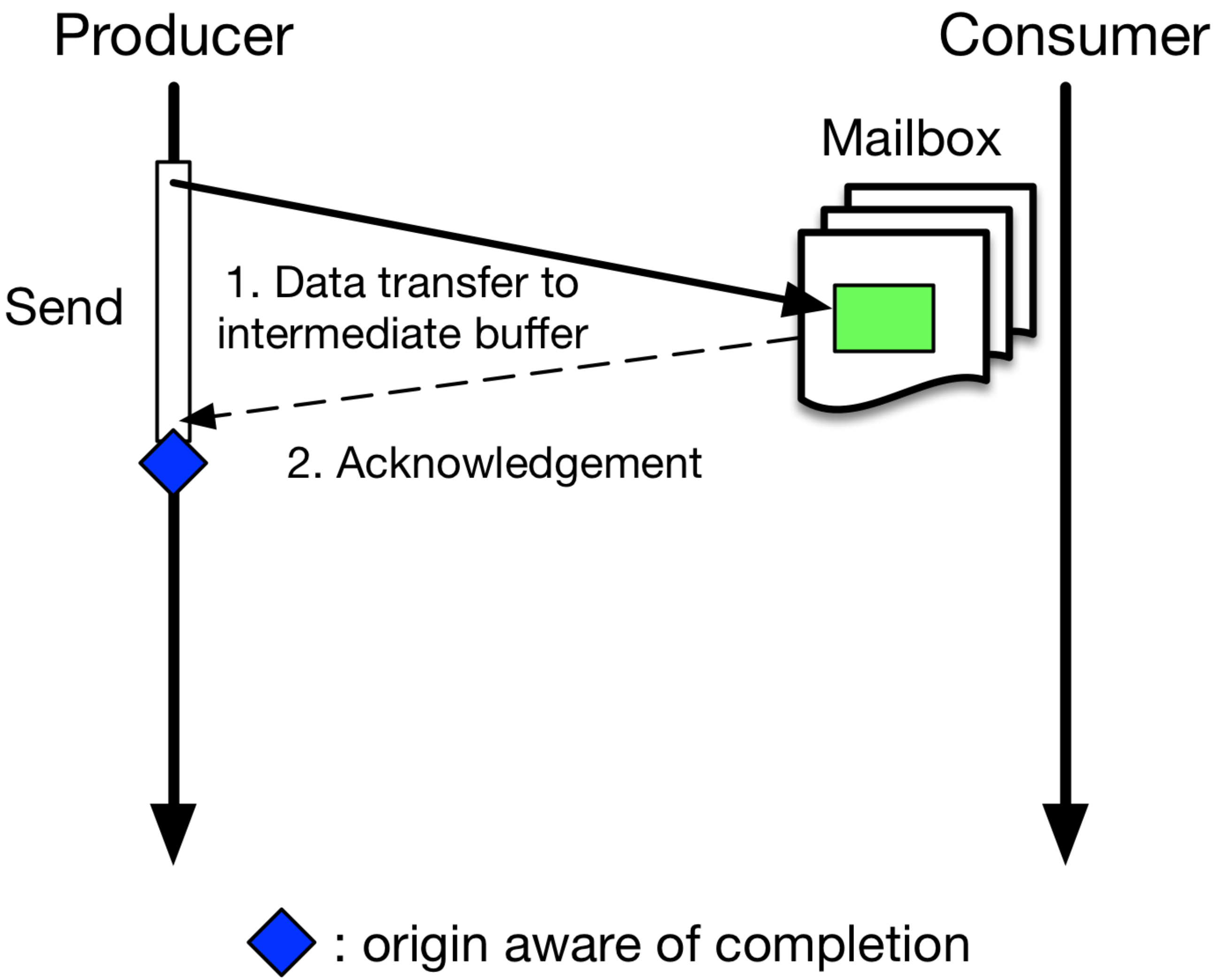
[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, “High performance RDMA protocols in HPC.”, EuroMPI’06

MPI-1 Message Passing – Simple Eager



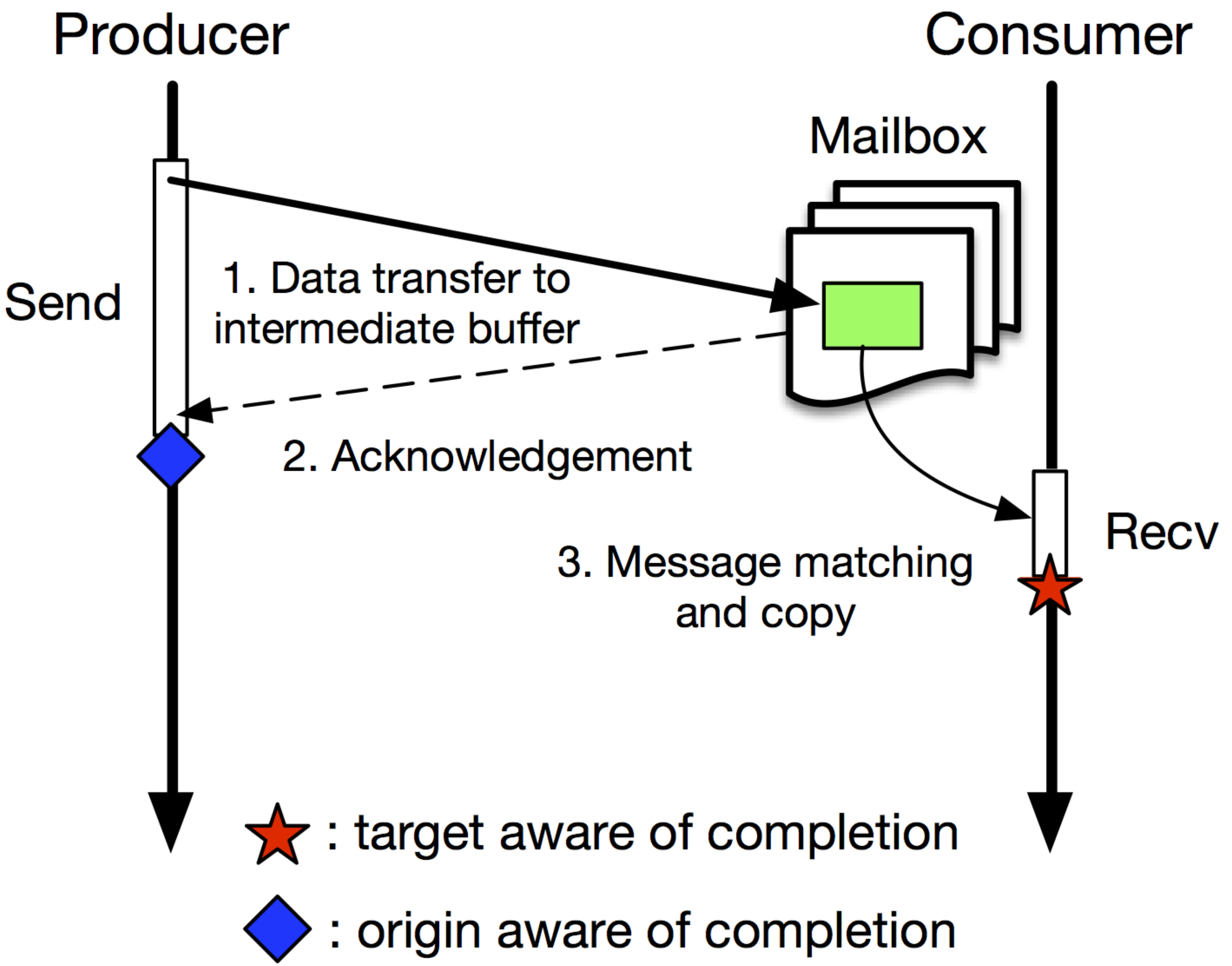
[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, "High performance RDMA protocols in HPC.", EuroMPI'06

MPI-1 Message Passing – Simple Eager



[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, “High performance RDMA protocols in HPC.”, EuroMPI’06

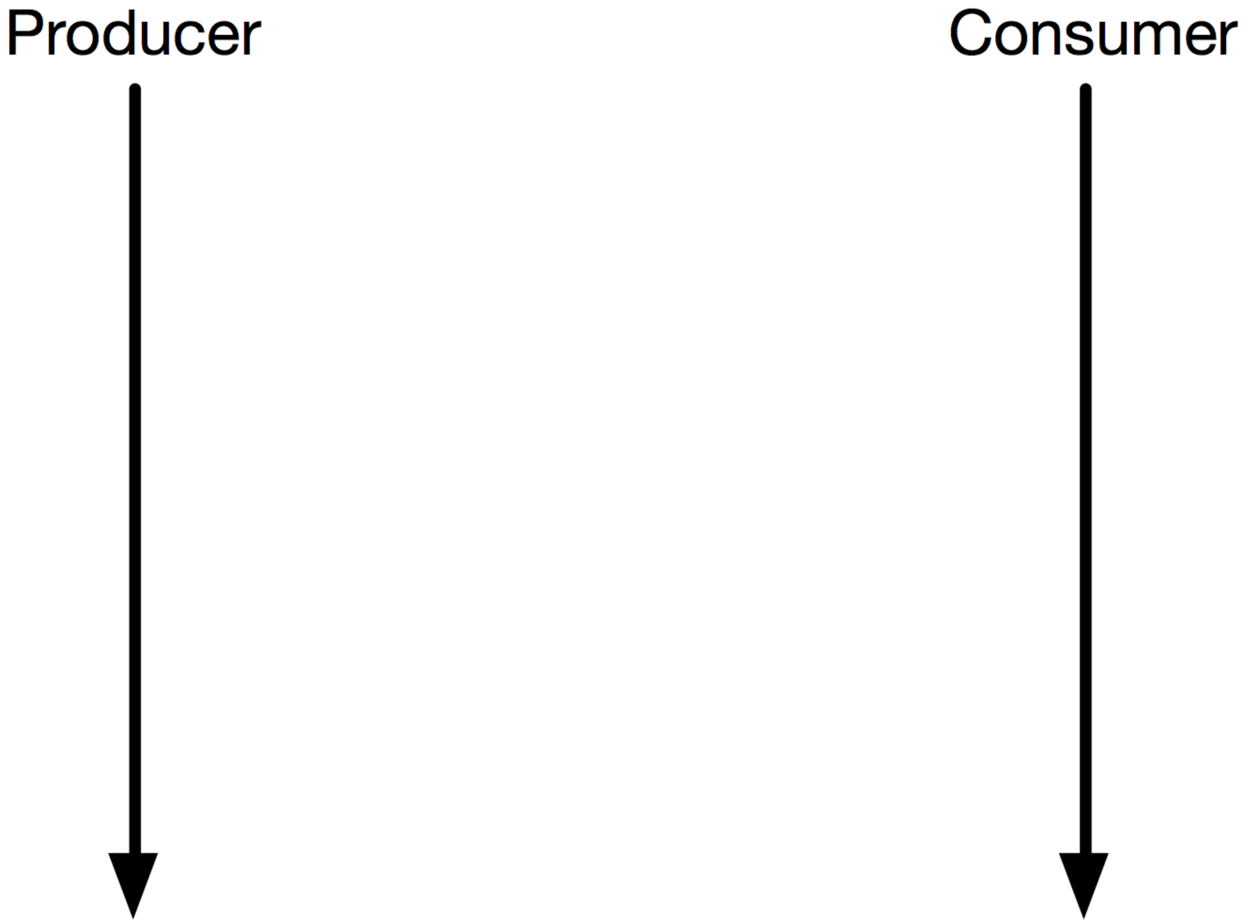
MPI-1 Message Passing – Simple Eager



Critical path: 1 latency + 1 copy

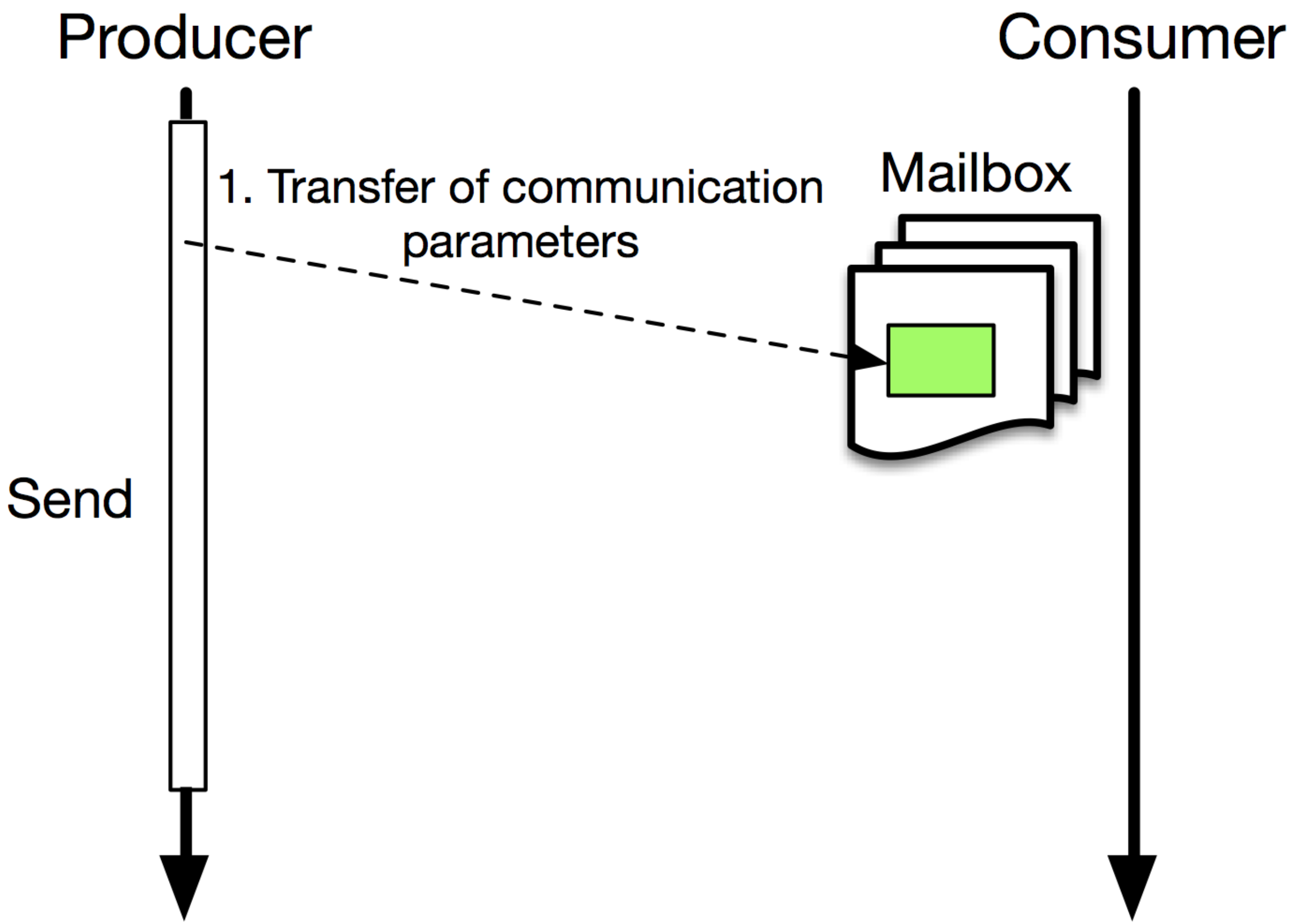
[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, "High performance RDMA protocols in HPC.", EuroMPI'06

MPI-1 Message Passing – Simple Rendezvous



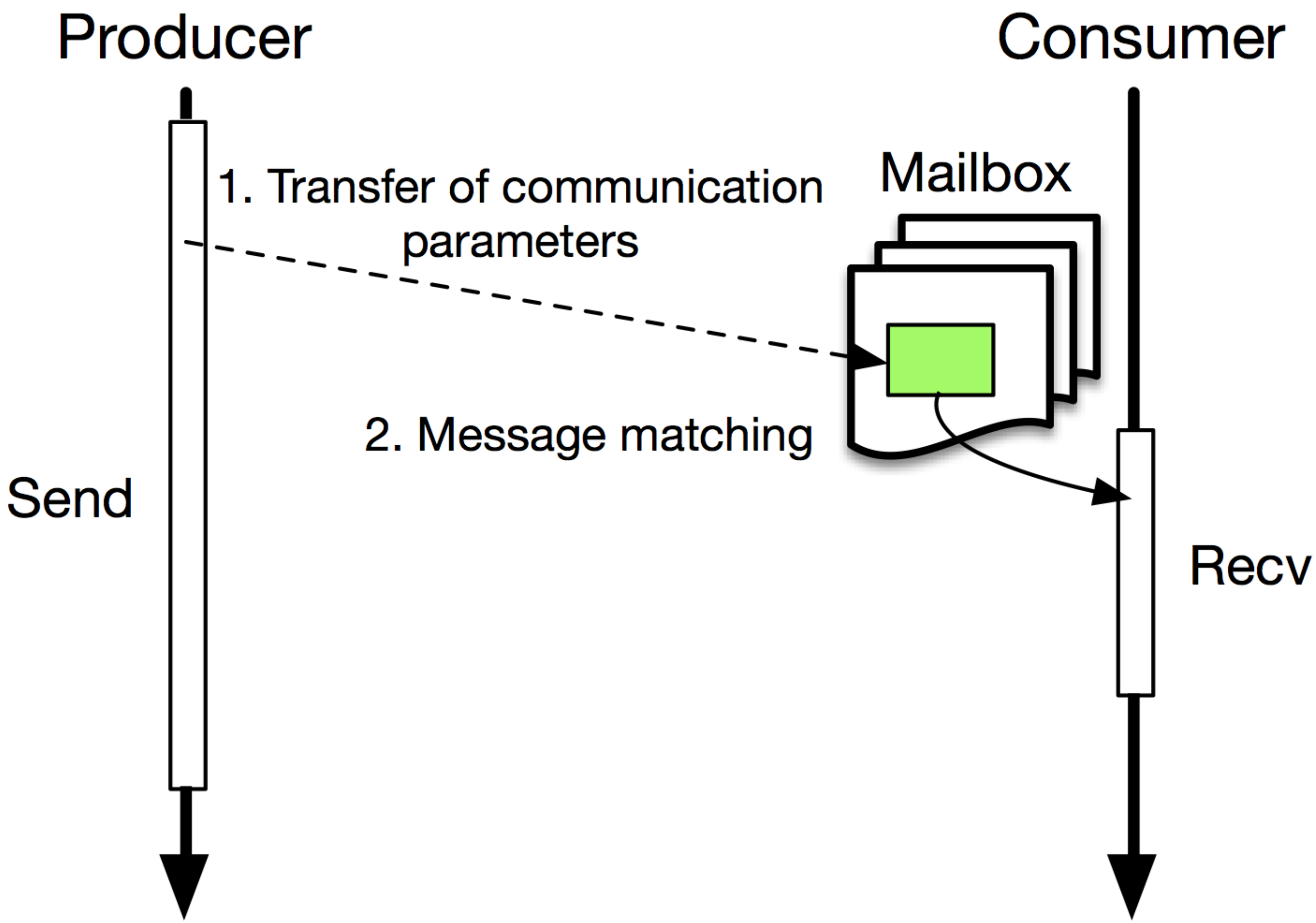
[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, “High performance RDMA protocols in HPC.”, EuroMPI’06

MPI-1 MESSAGE PASSING – SIMPLE RENDEZVOUS



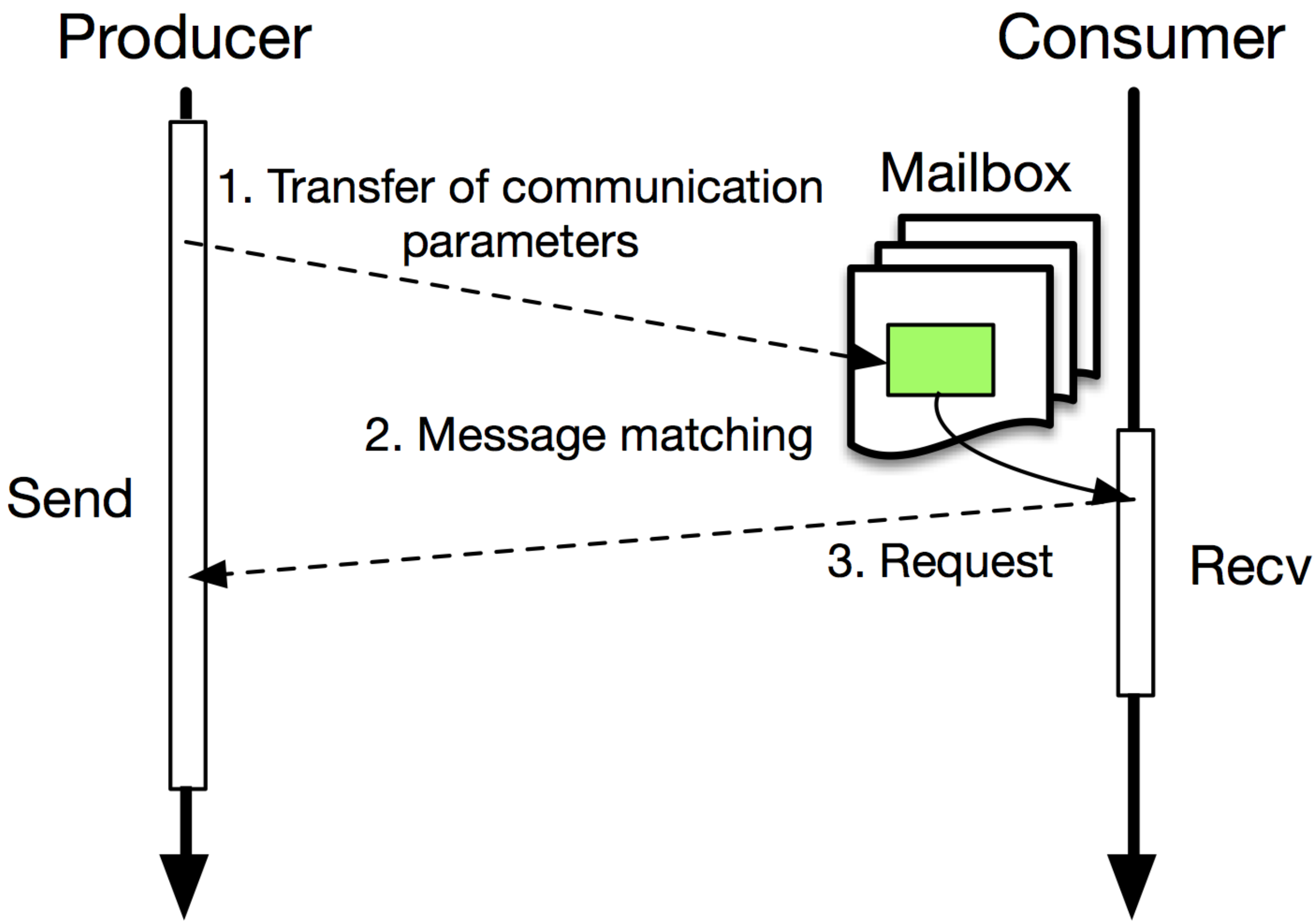
[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, "High performance RDMA protocols in HPC.", EuroMPI'06

MPI-1 Message Passing – Simple Rendezvous



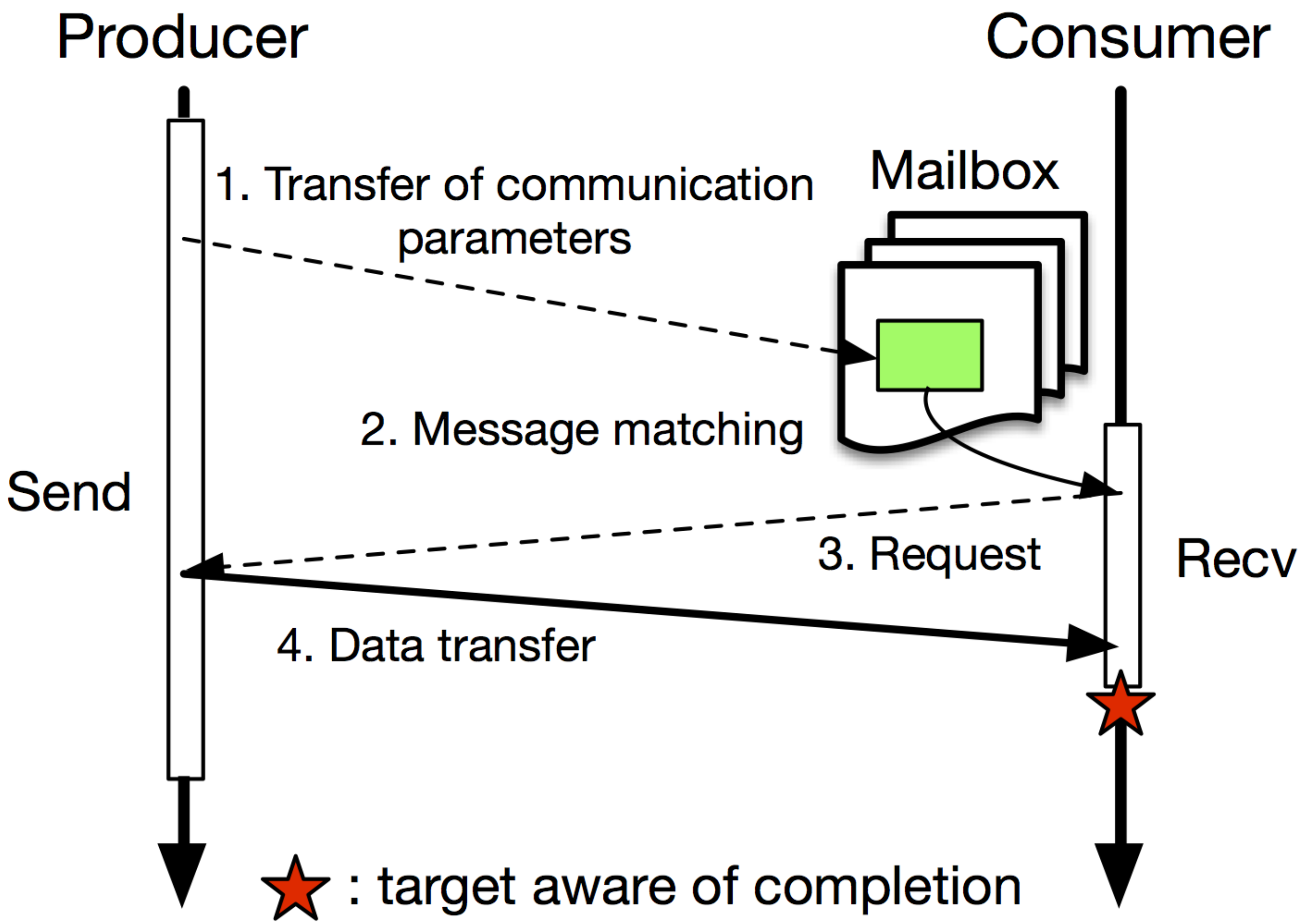
[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, "High performance RDMA protocols in HPC.", EuroMPI'06

MPI-1 Message Passing – Simple Rendezvous



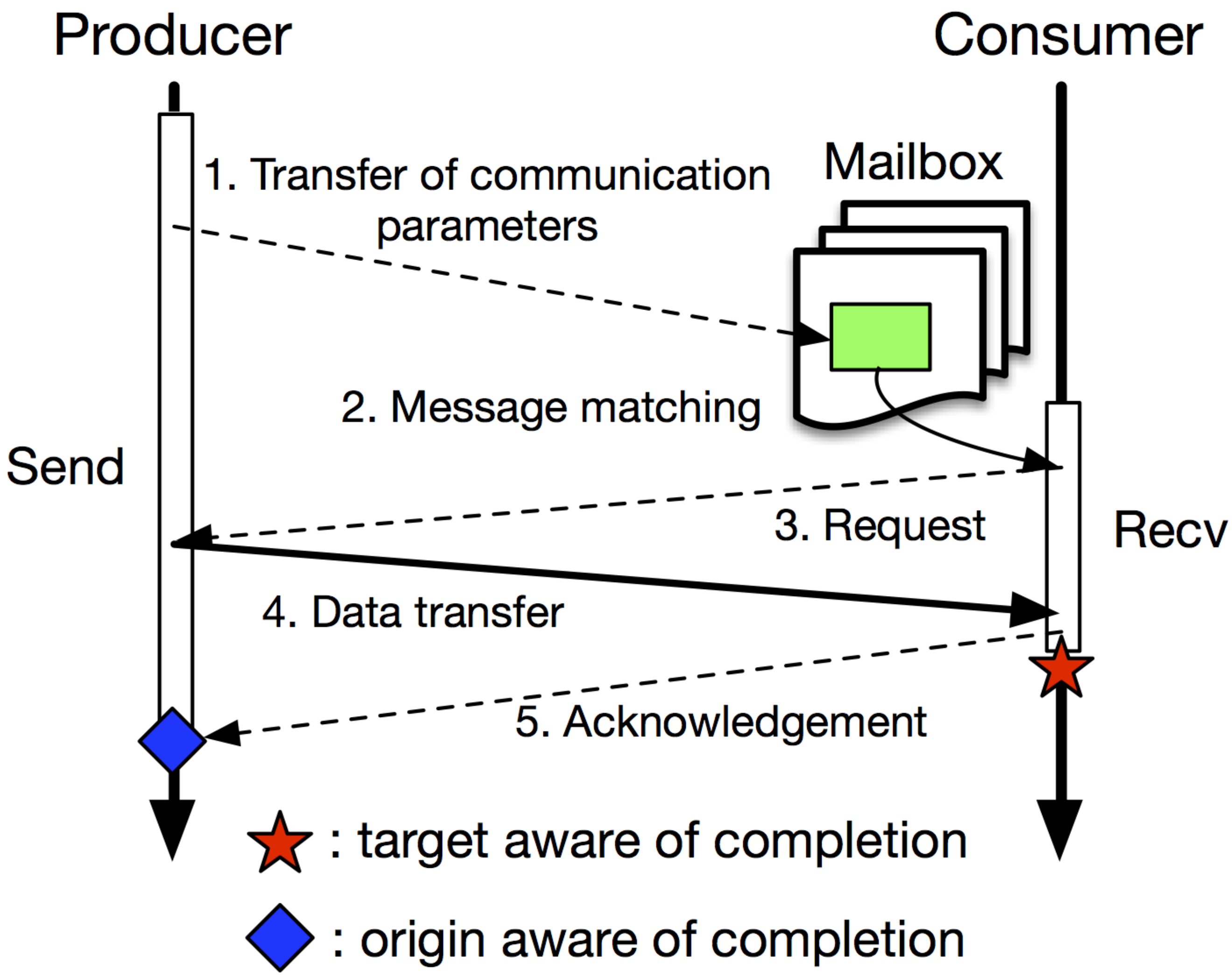
[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, “High performance RDMA protocols in HPC.”, EuroMPI’06

MPI-1 Message Passing – Simple Rendezvous



[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, "High performance RDMA protocols in HPC.", EuroMPI'06

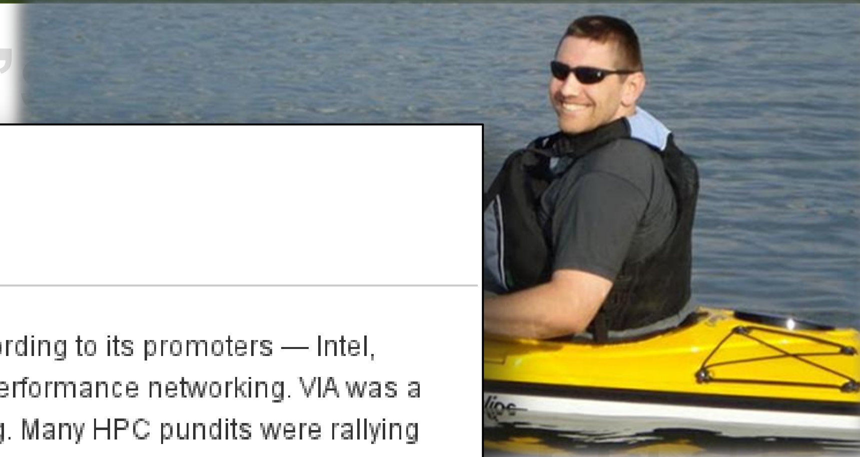
MPI-1 Message Passing – Simple Rendezvous



Critical path: 3 latencies

[1]: T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, "High performance RDMA protocols in HPC.", EuroMPI'06

Communication in Today'



August 18, 2006

A Critique of RDMA

by Patrick Geoffray, Ph.D.

Do you remember VIA, the Virtual Interface Architecture? I do. In 1998, according to its promoters — Intel, Compaq, and Microsoft — VIA was supposed to change the face of high-performance networking. VIA was a buzzword at the time; Venture Capital was flowing, and startups multiplying. Many HPC pundits were rallying behind this low-level programming interface, which promised scalable, low-overhead, high-throughput communication, initially for HPC and eventually for the data center. The hype was on and doom was spelled for the non-believers.

It turned out that VIA, based on RDMA (Remote Direct Memory Access, or Remote DMA), was not an improvement on existing APIs to support widely used application-software interfaces such as MPI and Sockets. After a while, VIA faded away, overtaken by other developments.

VIA was eventually reborn into the RDMA programming model that is the basis of various InfiniBand Verbs implementations, as well as DAPL (Direct Access Provider Library) and iWARP (Internet Wide Area RDMA Protocol). The pundits have returned, VCs are spending their money, and RDMA is touted as an ideal solution for the efficiency of high-performance networks.

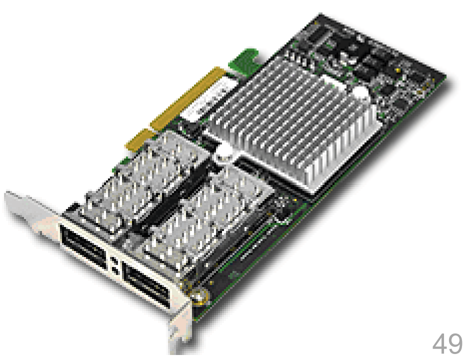
However, the evidence I'll present here shows that the revamped RDMA model is more a problem than a solution. What's more, the objective that RDMA pretends to address of efficient user-level communication between computing nodes is already solved by the two-sided Send/Recv model in products such as Quadrics QsNet, Cray SeaStar (implementing Sandia Portals), Qlogic InfiniPath, and Myricom's Myrinet Express (MX).

Send/Recv versus RDMA

The difference between these two paradigms, Send/Receive (Send/Recv) and RDMA, resides essentially in the

Remote Memory Access Programming

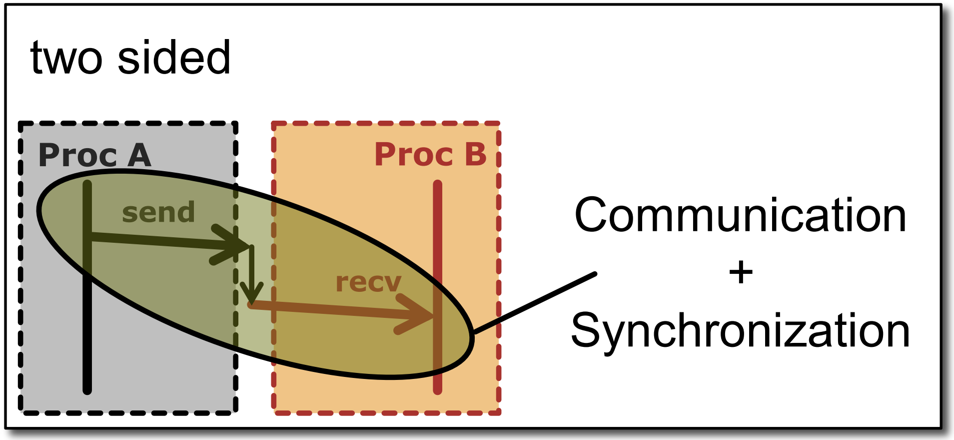
- **Why not use these RDMA features more directly?**
  - A global address space may simplify programming
  - … and accelerate communication
  - … and there could be a widely accepted standard
- **MPI-3 RMA ("MPI One Sided") was born ('13)**
  - Just one among many others (UPC, CAF, …)
  - Designed to react to hardware trends, learn from others
  - Direct (hardware-supported) remote access
  - New way of thinking for programmers

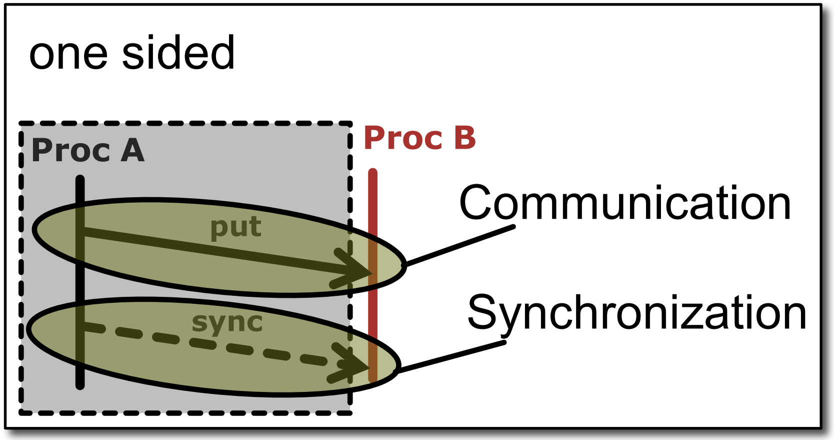


[1] http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf

MPI-3 RMA SUMMARY

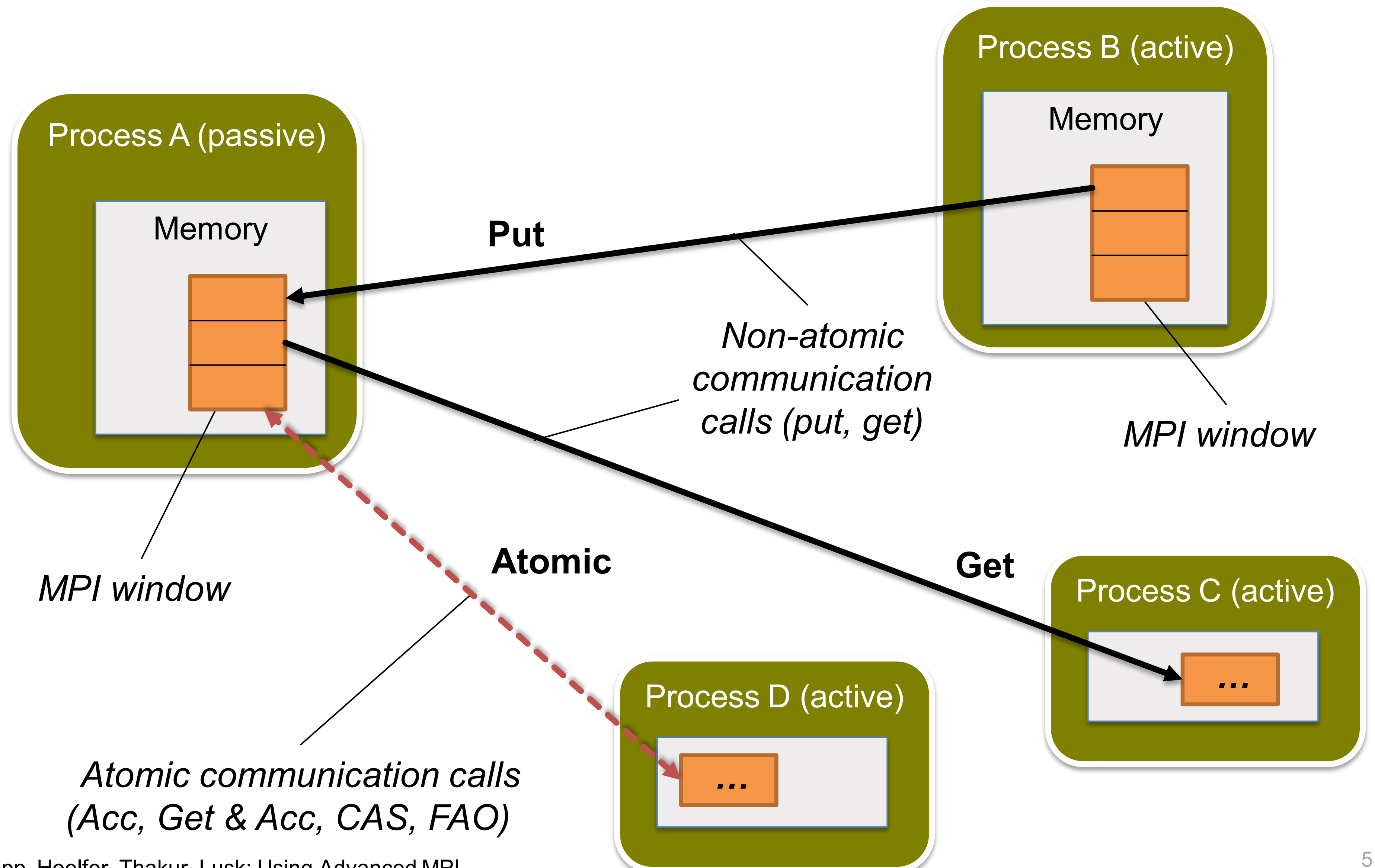
- **MPI-3 updates RMA (“MPI One Sided”)**
  - Significant change from MPI-2
- **Communication is „one sided” (no involvement of destination)**
  - Utilize direct memory access
- **RMA decouples communication & synchronization**
  - Fundamentally different from message passing





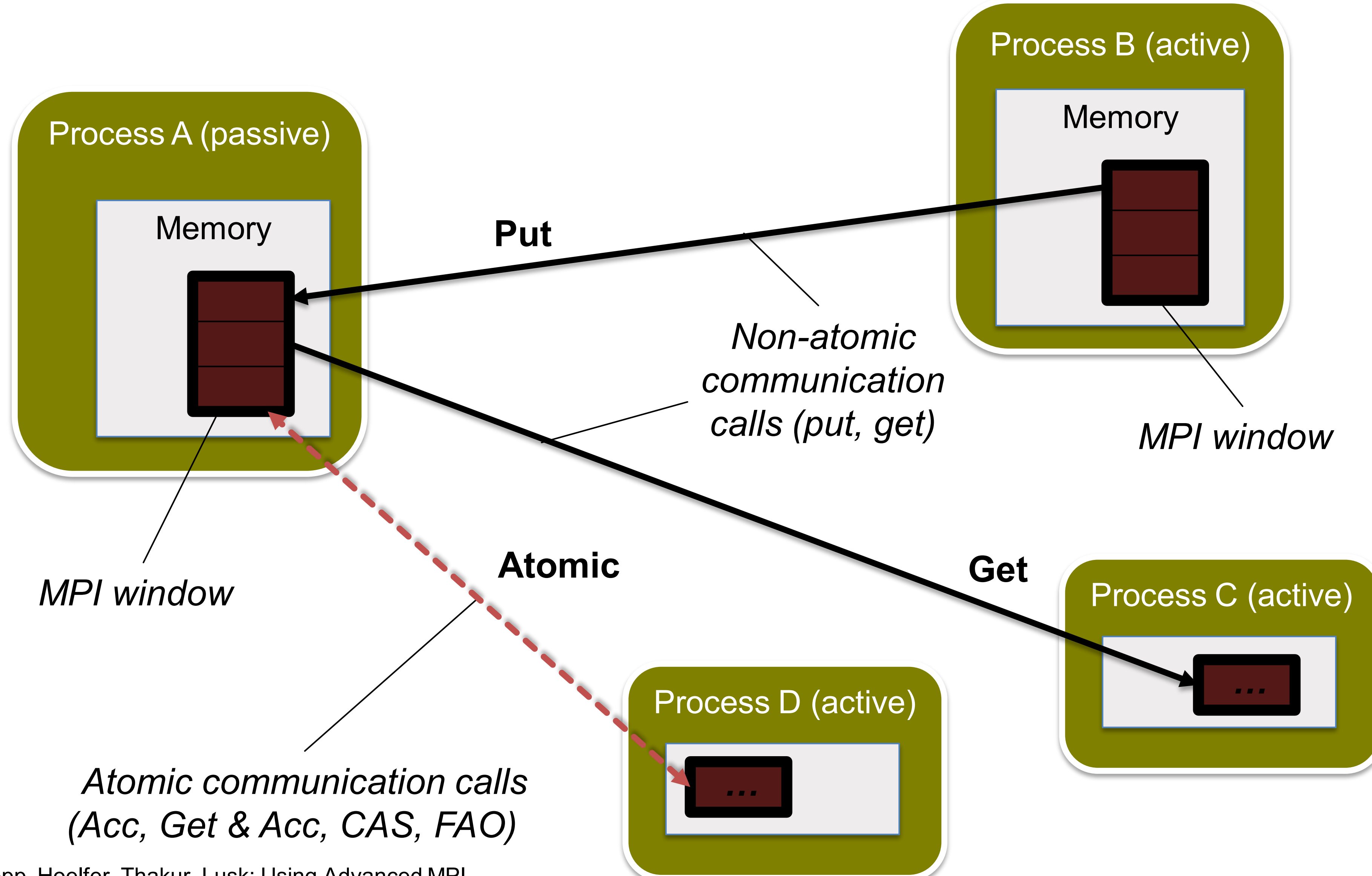
[1] http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf

MPI-3 RMA Communication Overview



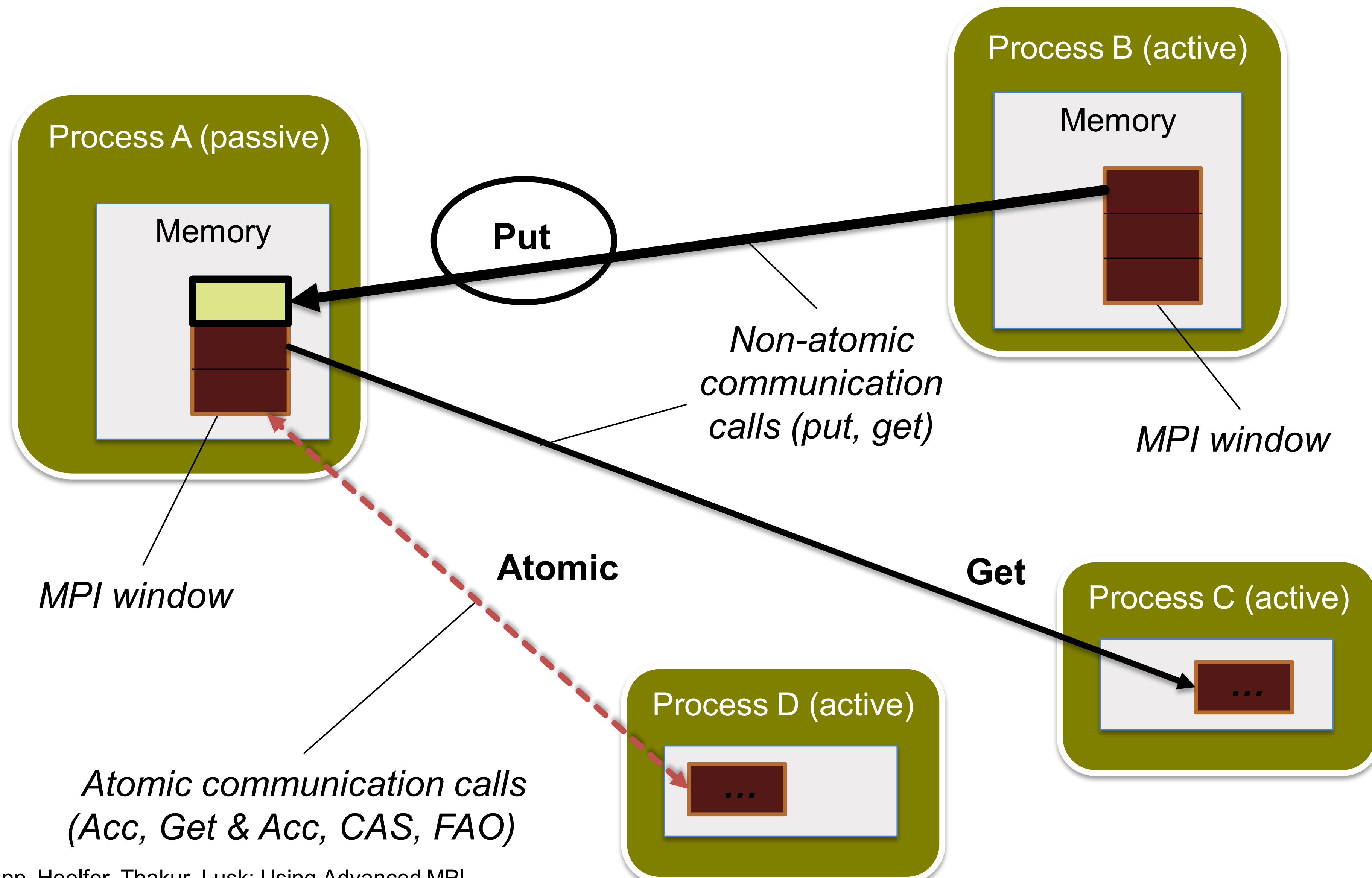
Gropp, Hoelfer, Thakur, Lusk: Using Advanced MPI

MPI-3 RMA Communication Overview



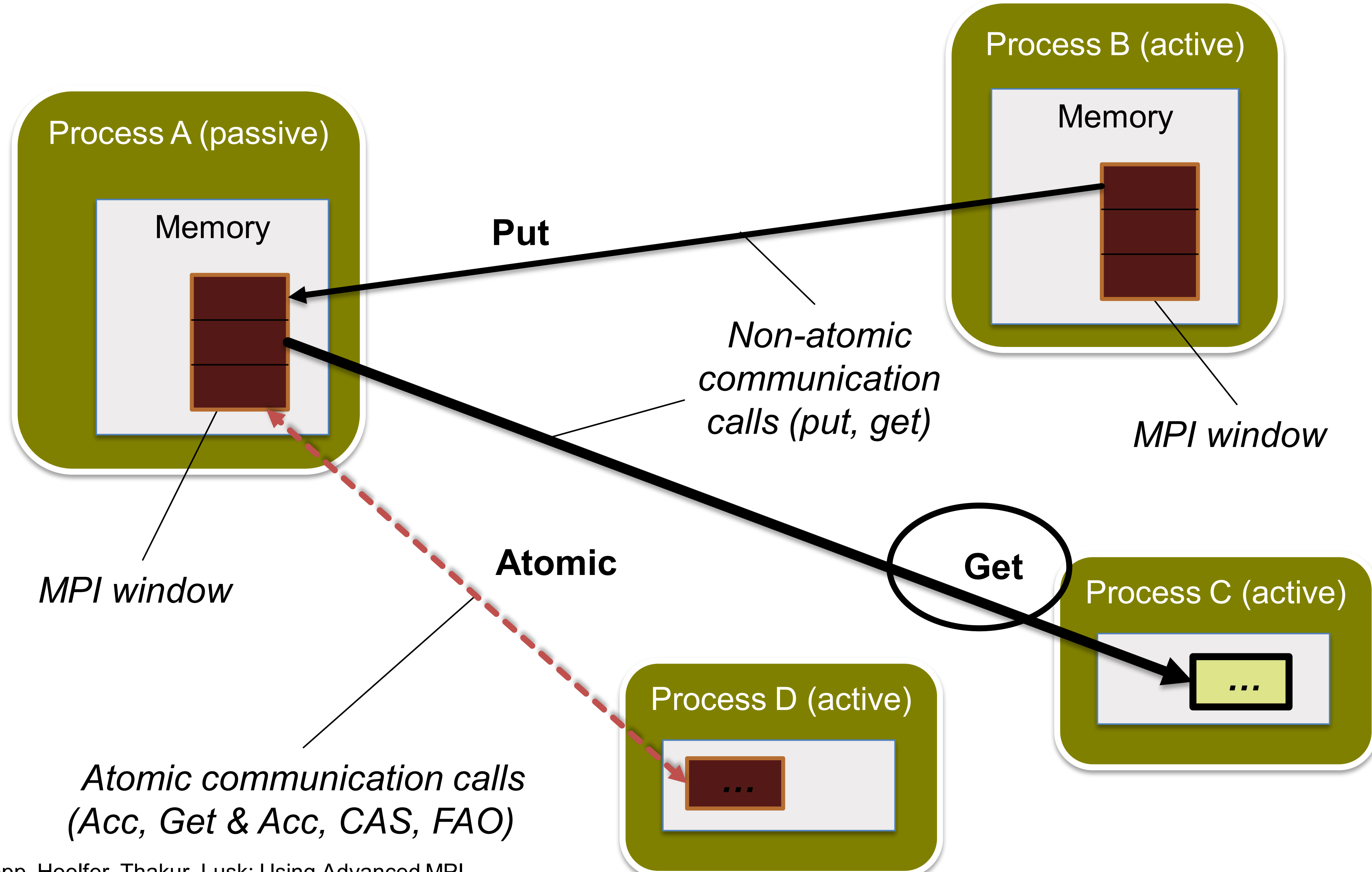
Gropp, Hoelfer, Thakur, Lusk: Using Advanced MPI

MPI-3 RMA Communication Overview



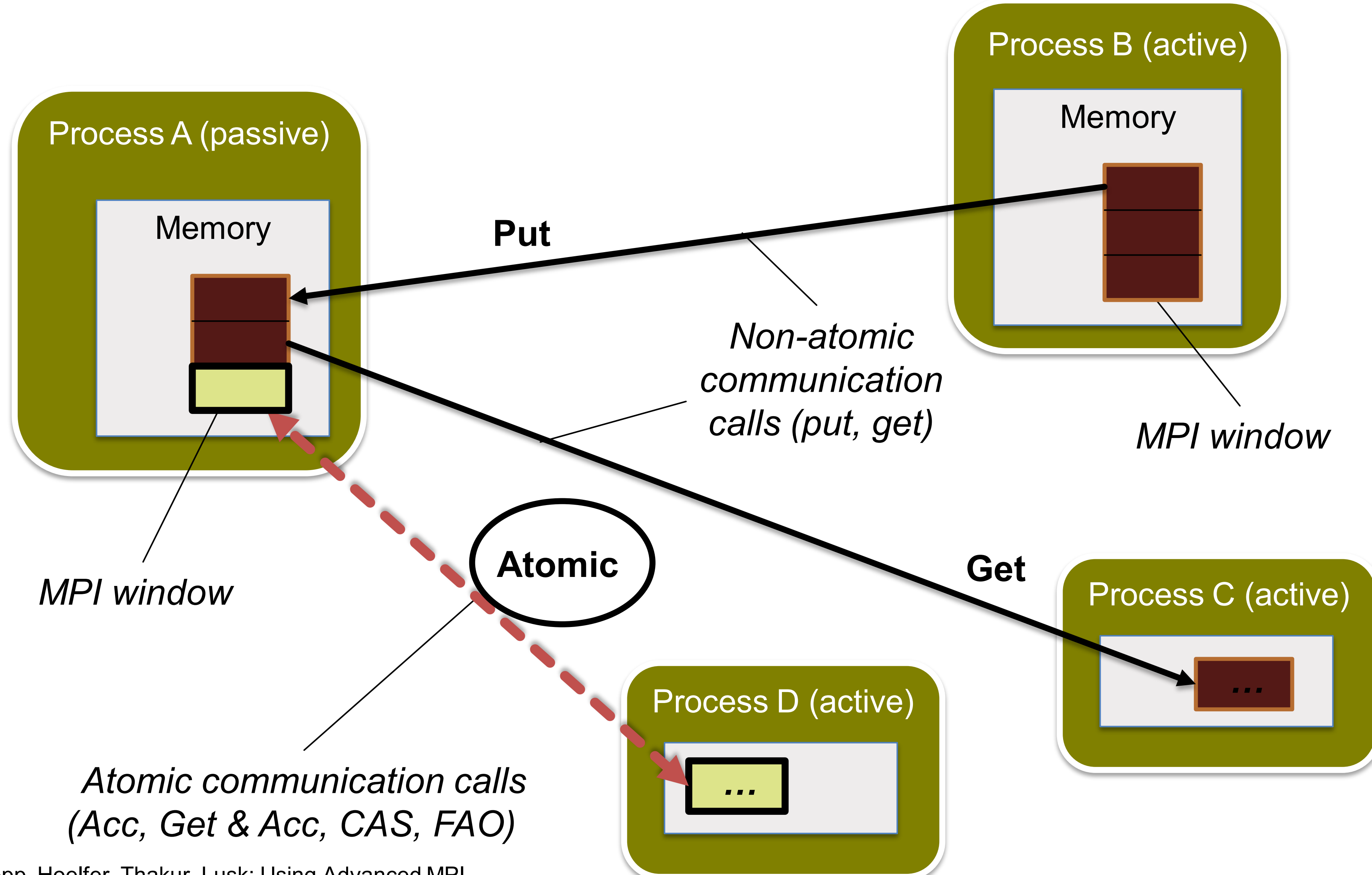
Gropp, Hoelfer, Thakur, Lusk: Using Advanced MPI

MPI-3 RMA Communication Overview



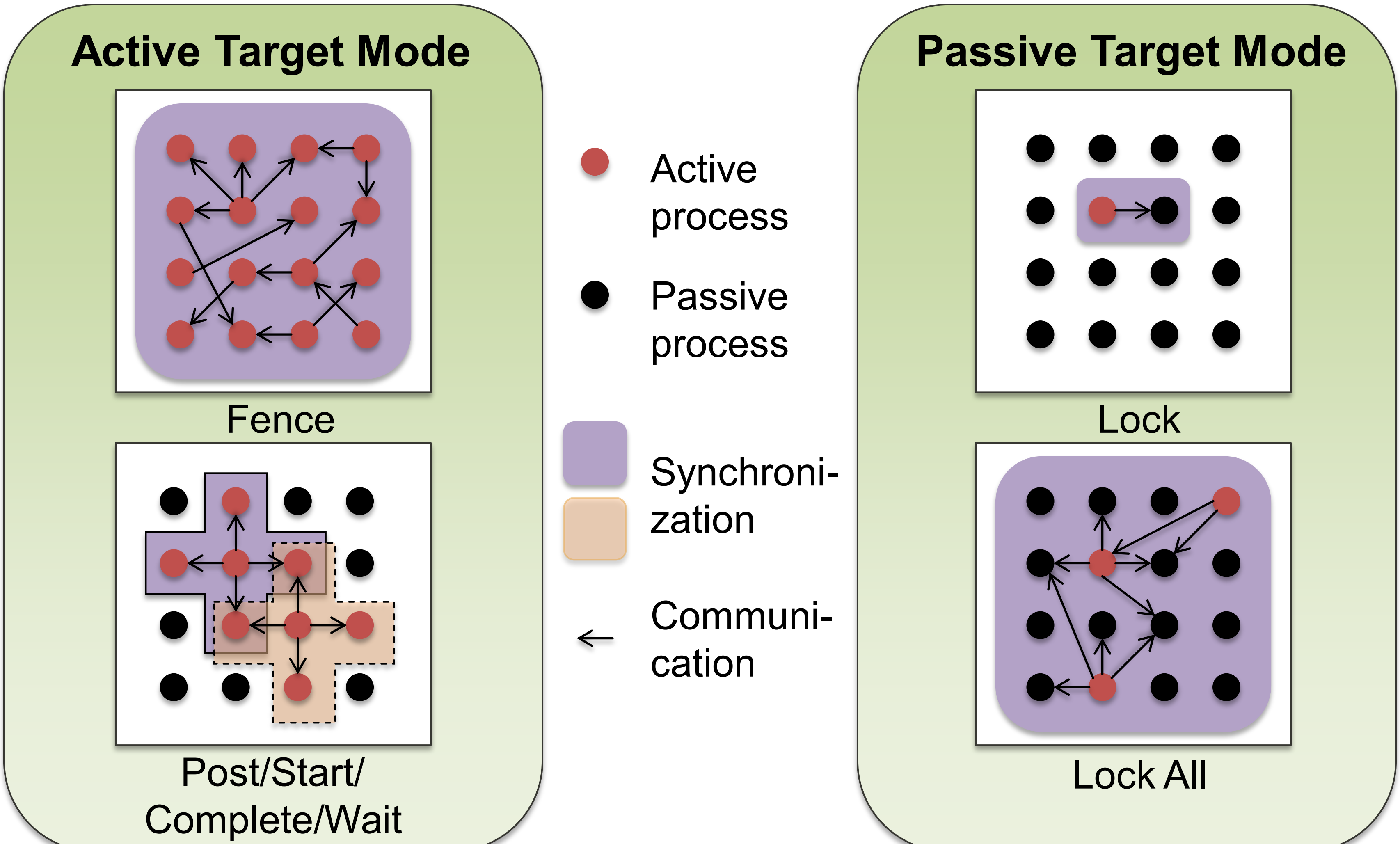
Gropp, Hoelfer, Thakur, Lusk: Using Advanced MPI

MPI-3 RMA Communication Overview



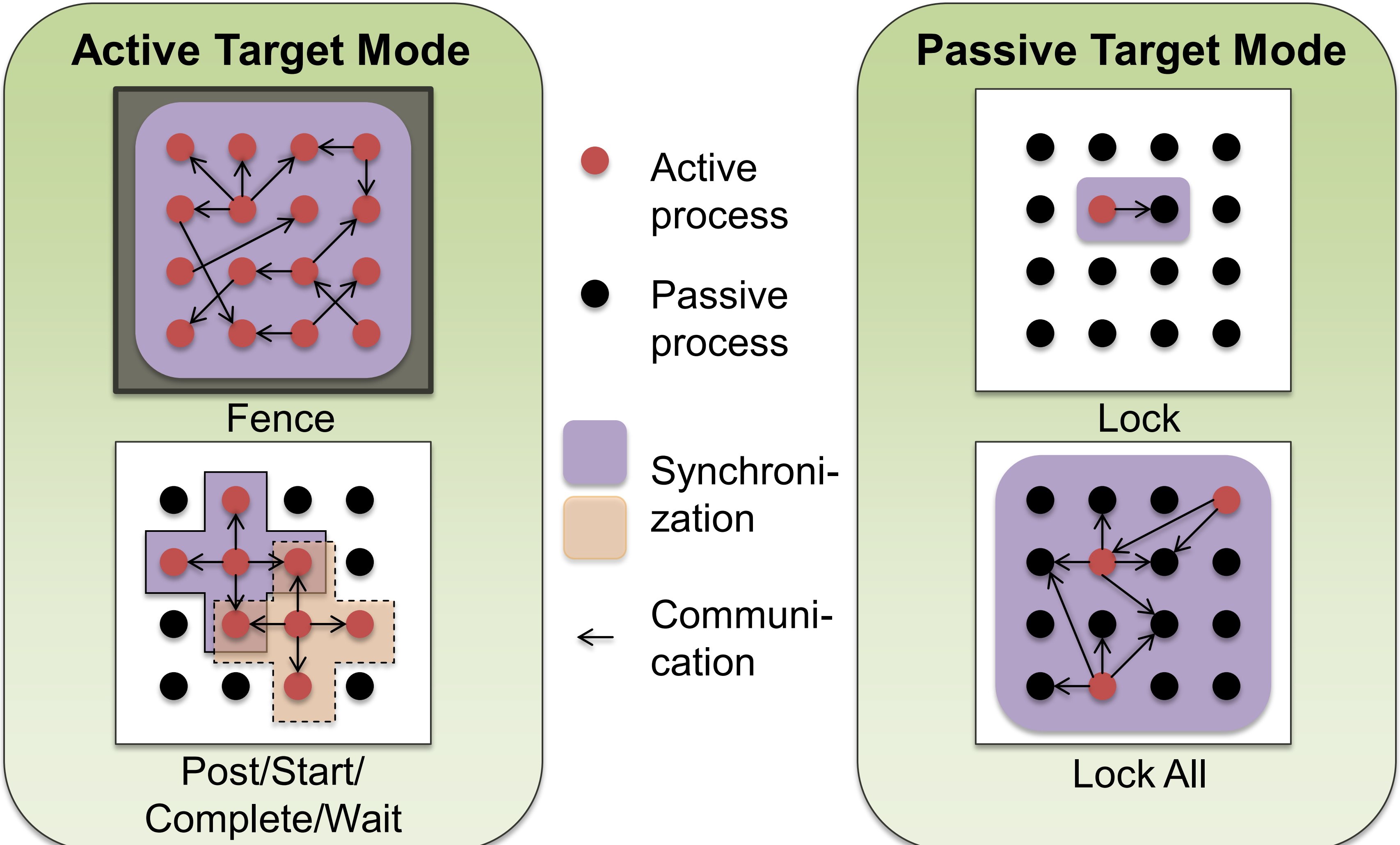
Gropp, Hoelfer, Thakur, Lusk: Using Advanced MPI

MPI-3 RMA Synchronization Overview



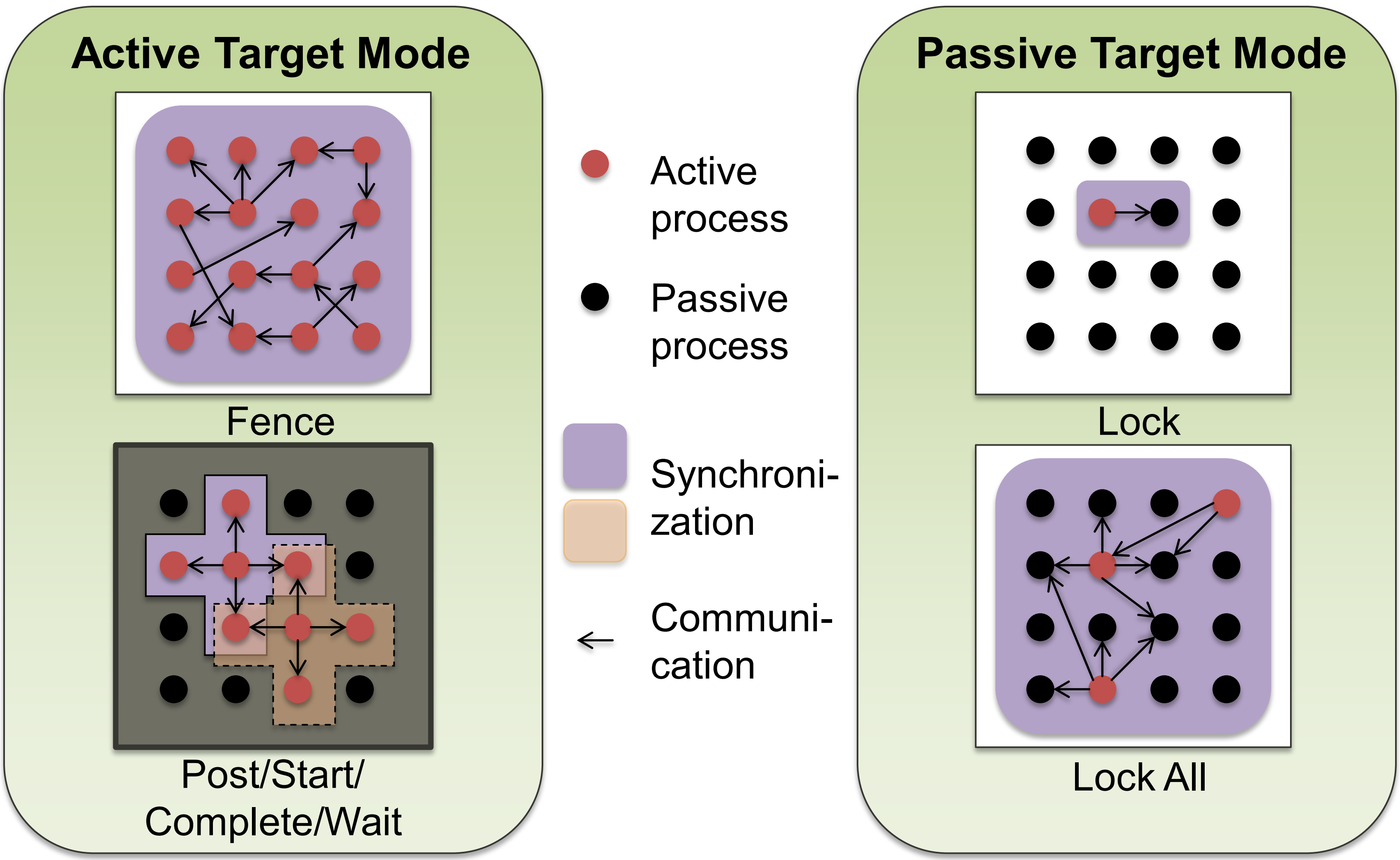
Gropp, Hoelfer, Thakur, Lusk: Using Advanced MPI

MPI-3 RMA Synchronization Overview



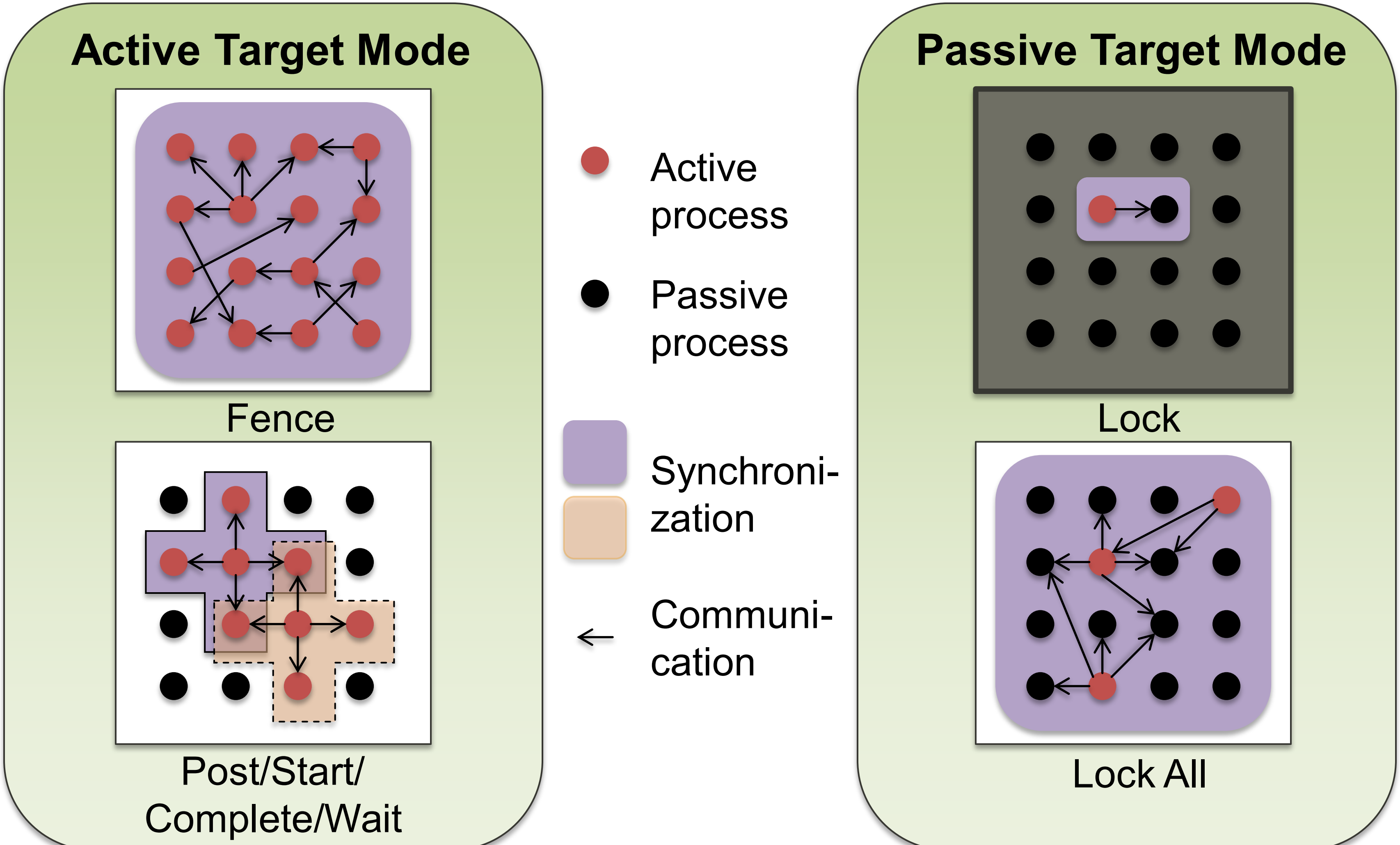
Gropp, Hoelfer, Thakur, Lusk: Using Advanced MPI

MPI-3 RMA Synchronization Overview

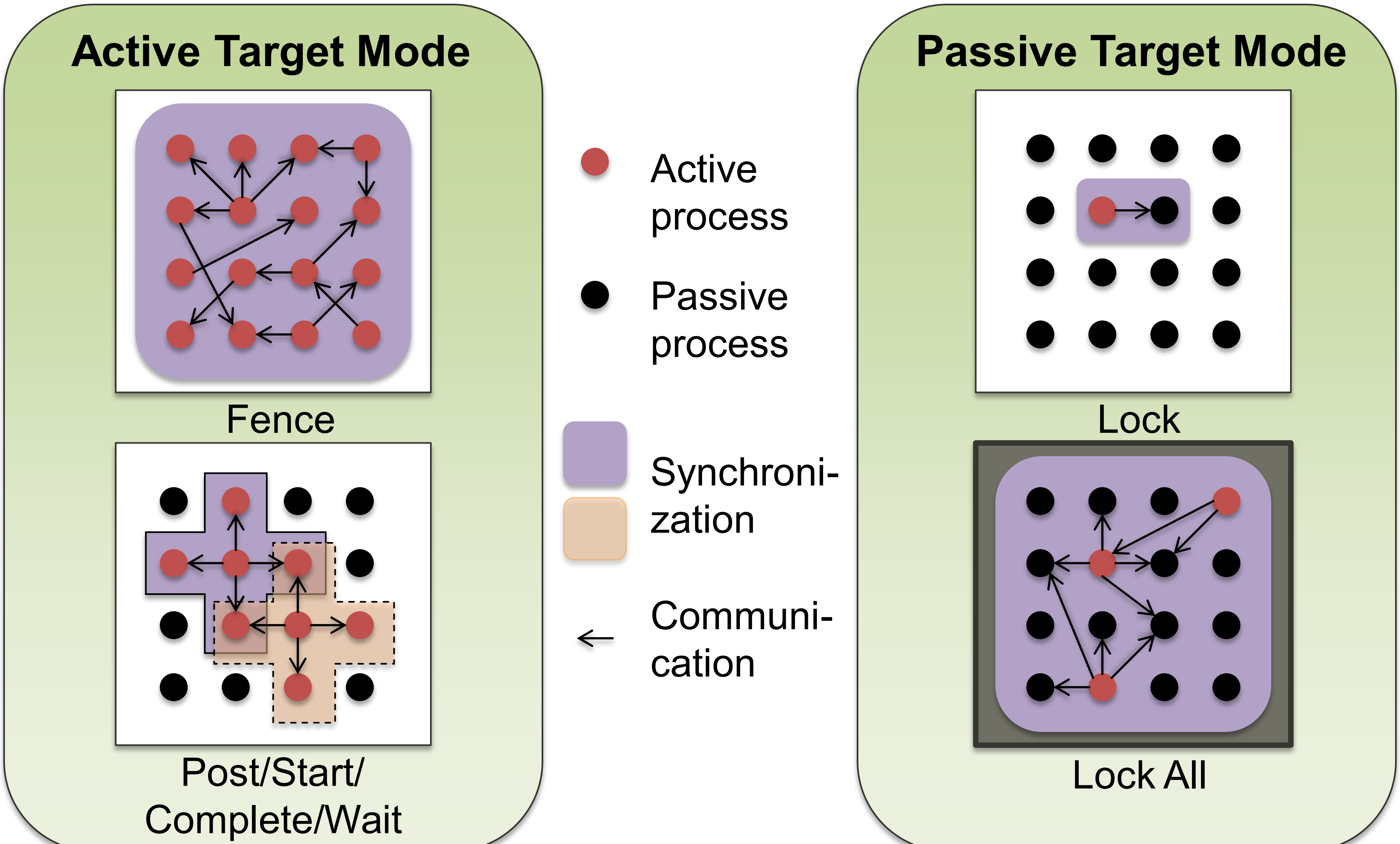


Gropp, Hoelfer, Thakur, Lusk: Using Advanced MPI

MPI-3 RMA Synchronization Overview

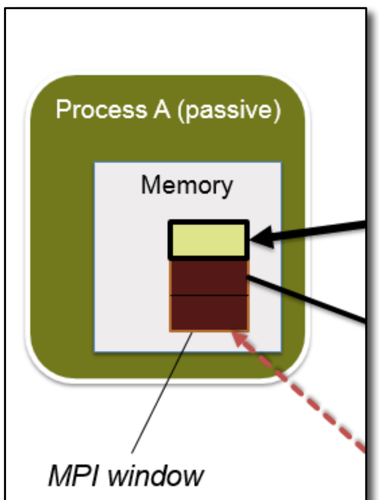


MPI-3 RMA Synchronization Overview

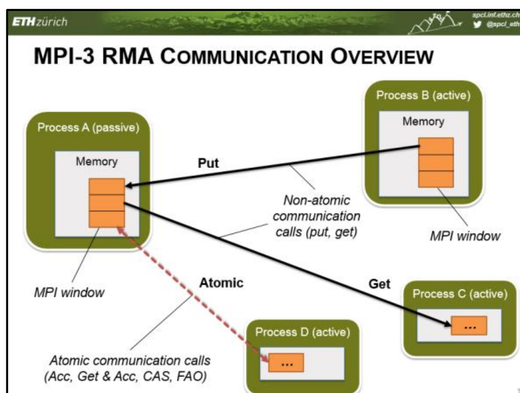


Scalable Protocols & Reference Implementation

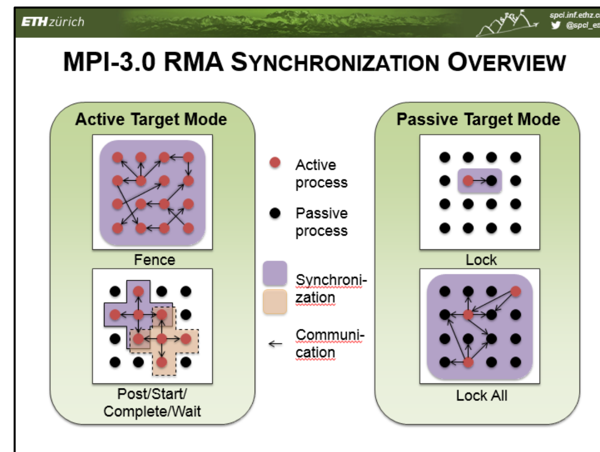
- Scalable & generic protocols
  - Can be used on any RDMA network (e.g., OFED/IB)
  - Window creation, communication and synchronization



Window creation



Communication

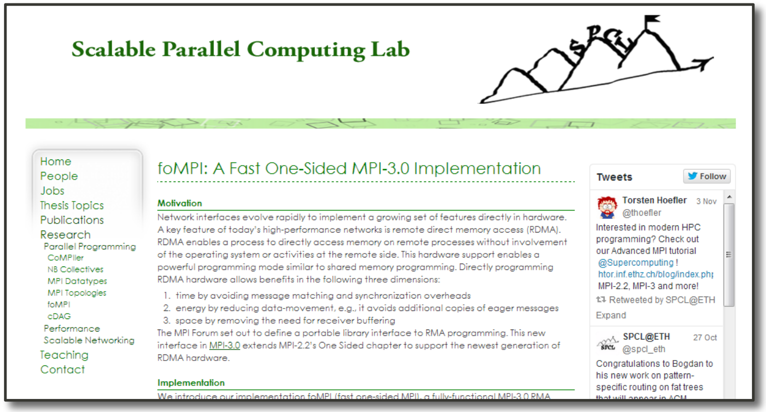


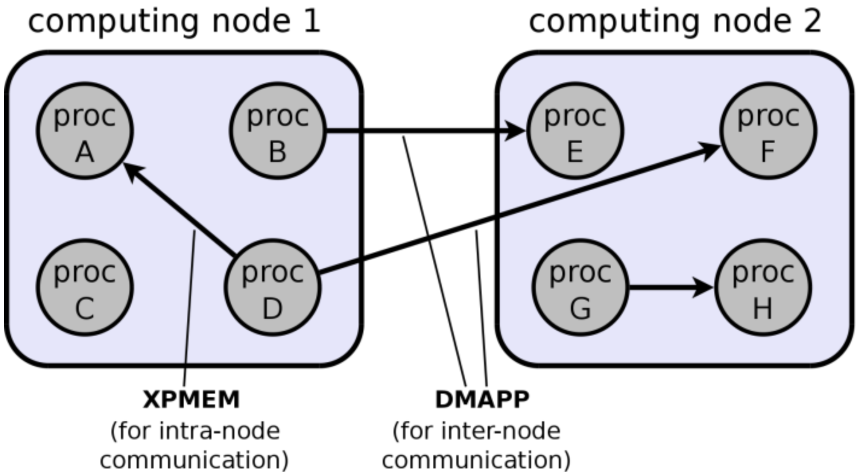
Synchronization

Scalable Protocols & Reference Implementation

- Scalable & generic protocols
  - Can be used on any RDMA network (e.g., OFED/IB)
  - Window creation, communication and synchronization

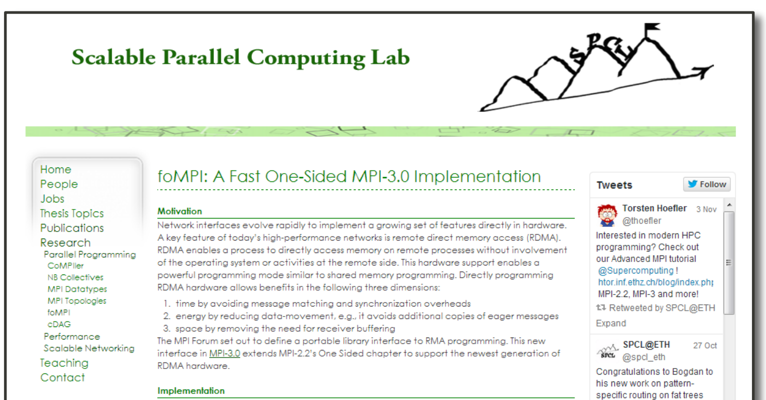
- foMPI, a fully functional MPI-3 RMA implementation
  - DMAPP: lowest-level networking API for Cray Gemini/Aries systems
  - XPMEM, a portable Linux kernel module

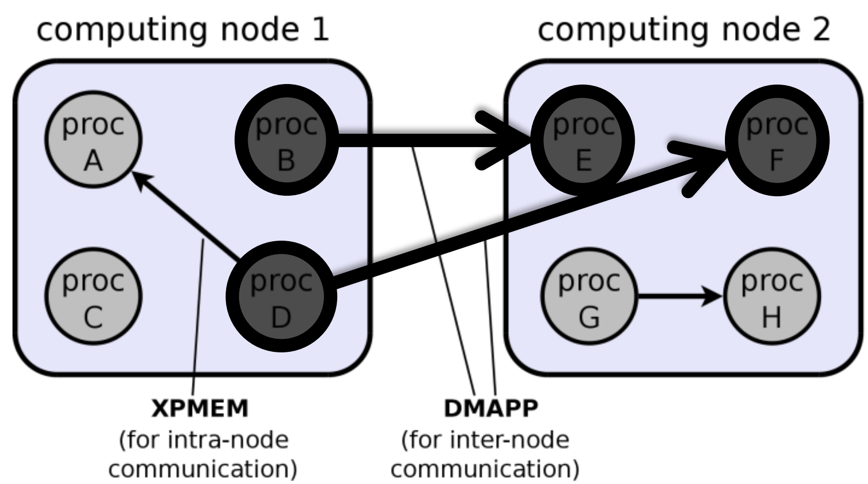




Scalable Protocols & Reference Implementation

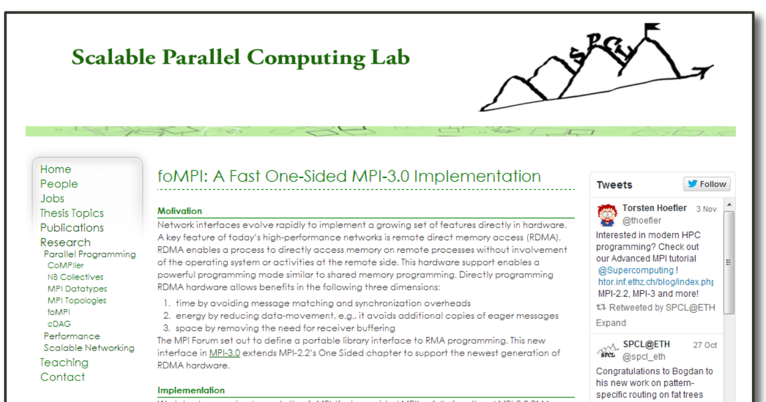
- Scalable & generic protocols
  - Can be used on any RDMA network (e.g., OFED/IB)
  - Window creation, communication and synchronization
- foMPI, a fully functional MPI-3 RMA implementation
  - DMAPP: lowest-level networking API for Cray Gemini/Aries systems
  - XPMEM, a portable Linux kernel module

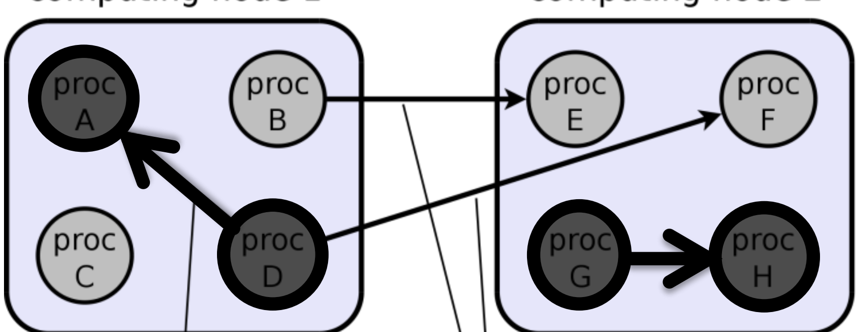




Scalable Protocols & Reference Implementation

- Scalable & generic protocols
  - Can be used on any RDMA network (e.g., OFED/IB)
  - Window creation, communication and synchronization
- foMPI, a fully functional MPI-3 RMA implementation
  - DMAPP: lowest-level networking API for Cray Gemini/Aries systems
  - XPMEM: a portable Linux kernel module

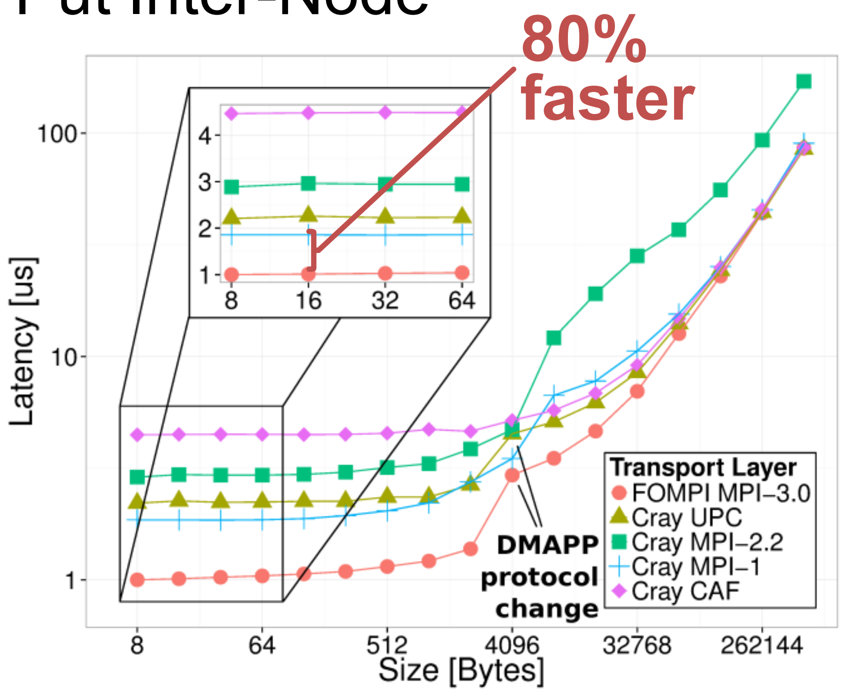




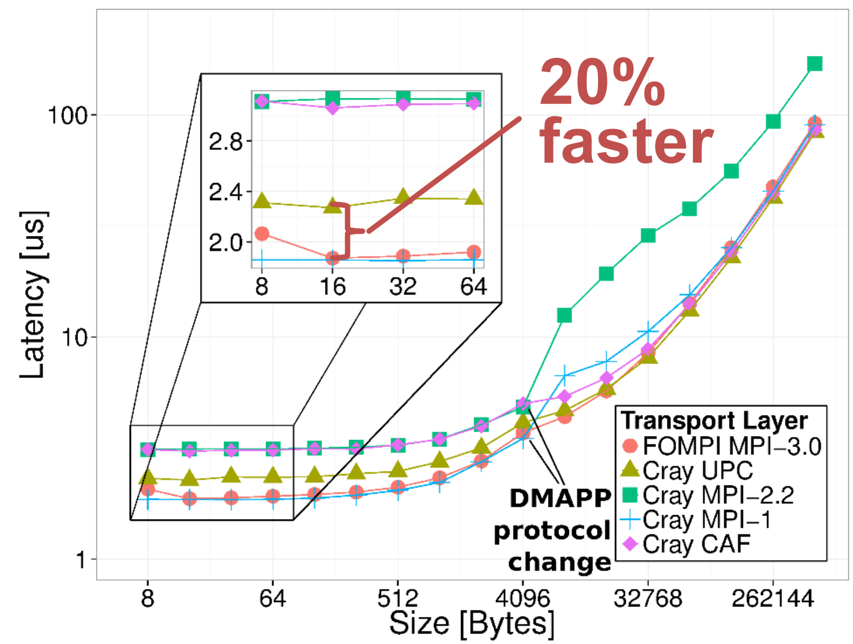
http://spcl.inf.ethz.ch/Research/Parallel_Programming/foMPI

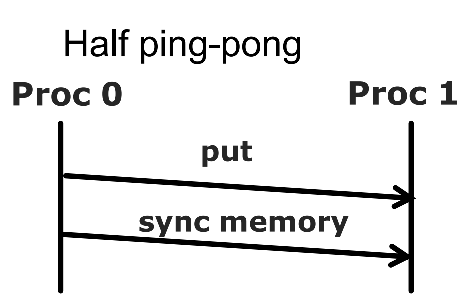
Performance Inter-node: Latency

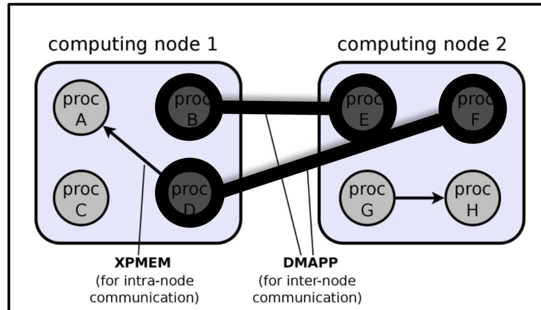
Put Inter-Node



Get Inter-Node

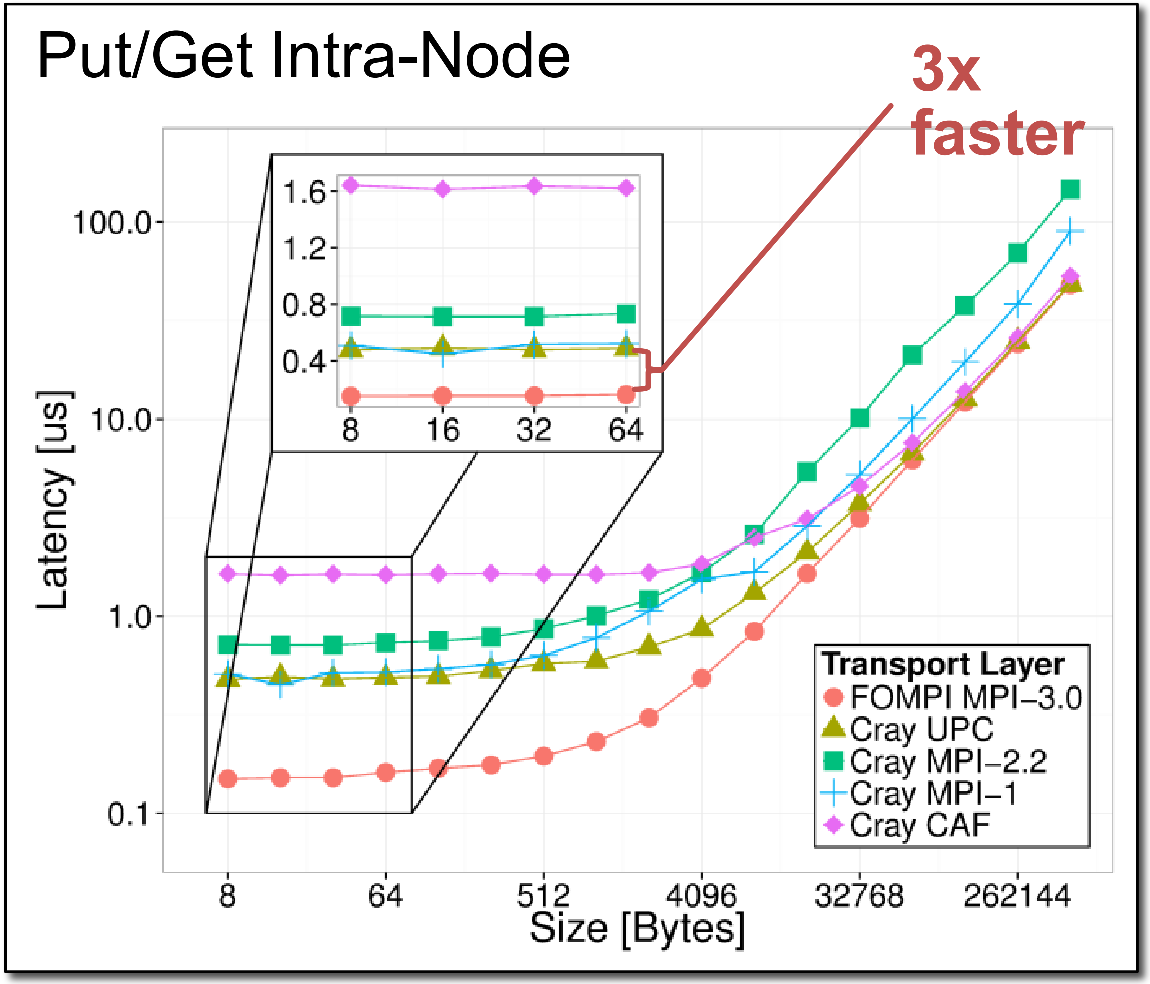


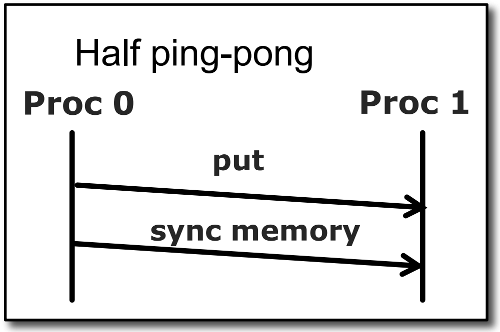


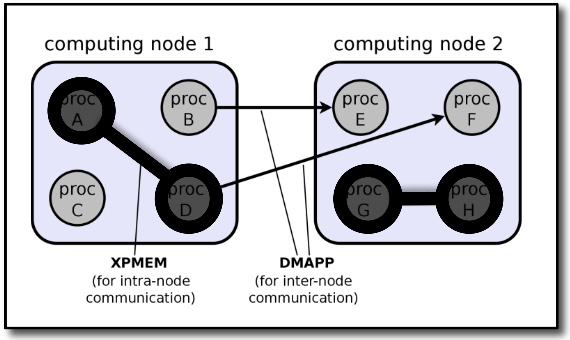


Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13

PERFORMANCE INTRA-NODE: LATENCY

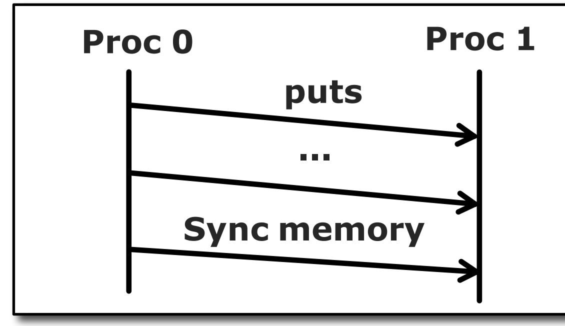


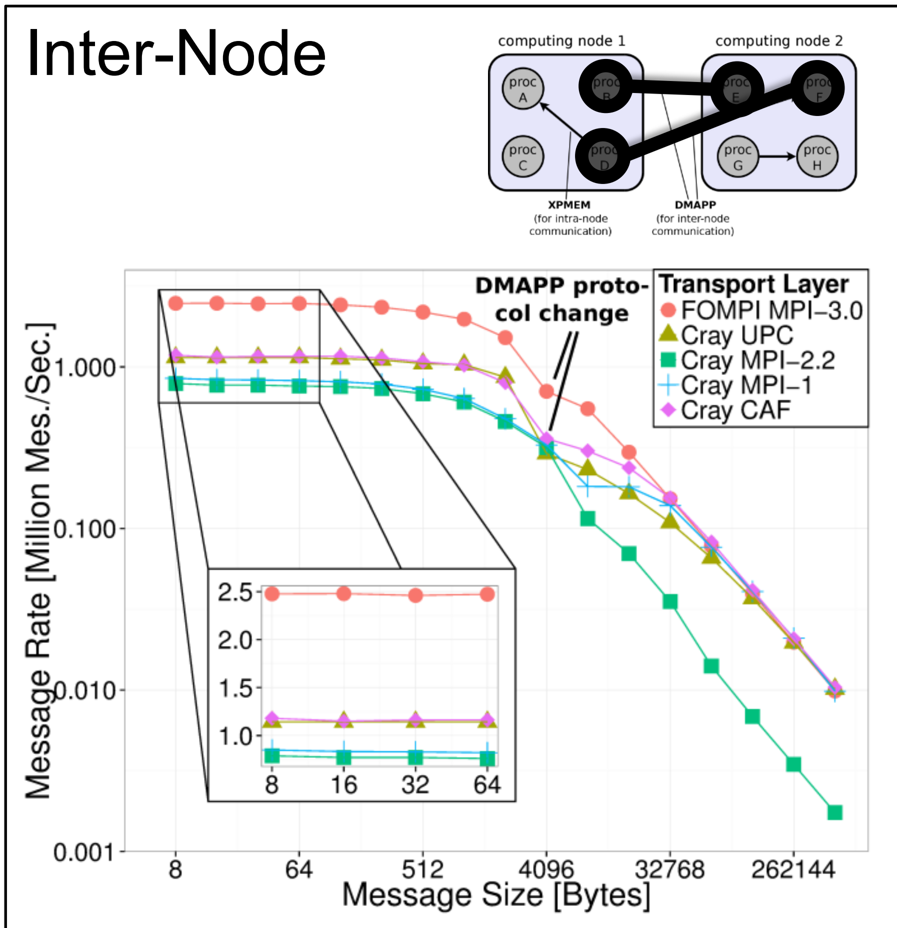


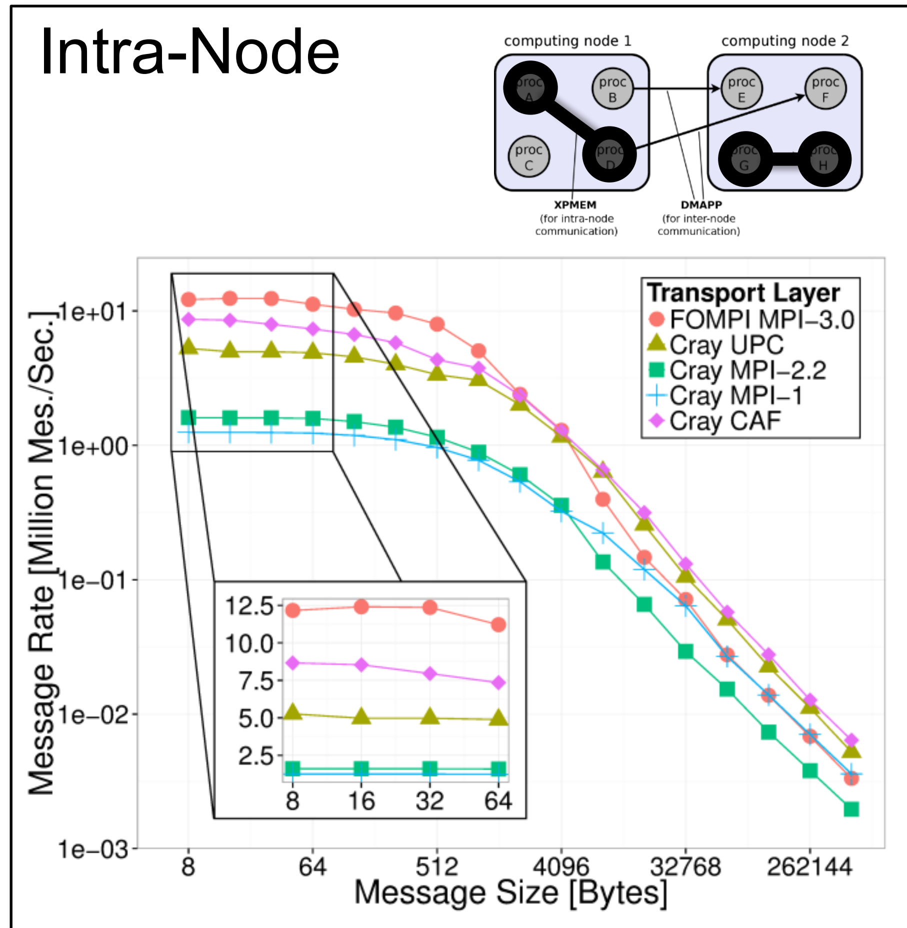


Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13

PERFORMANCE: MESSAGE RATE

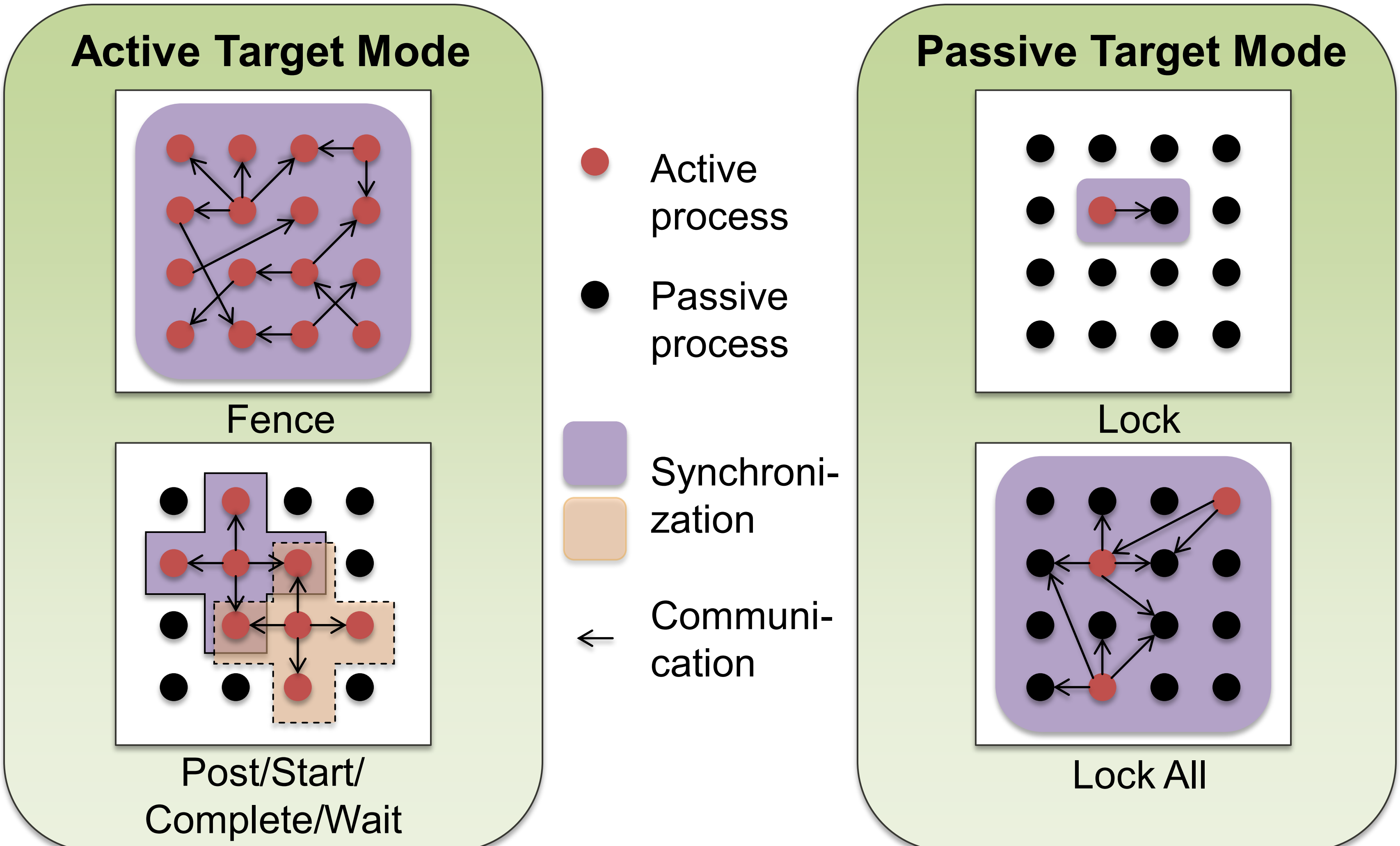




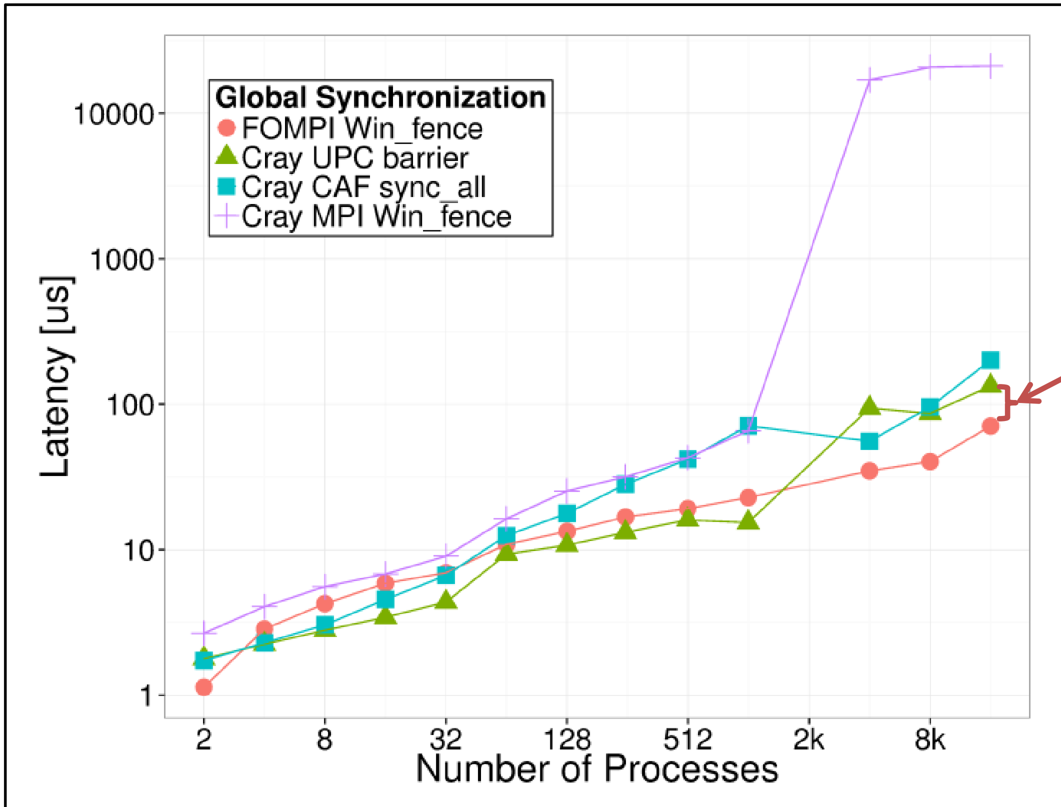


Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13

Part 3: Synchronization



Scalable Fence Performance



90% faster

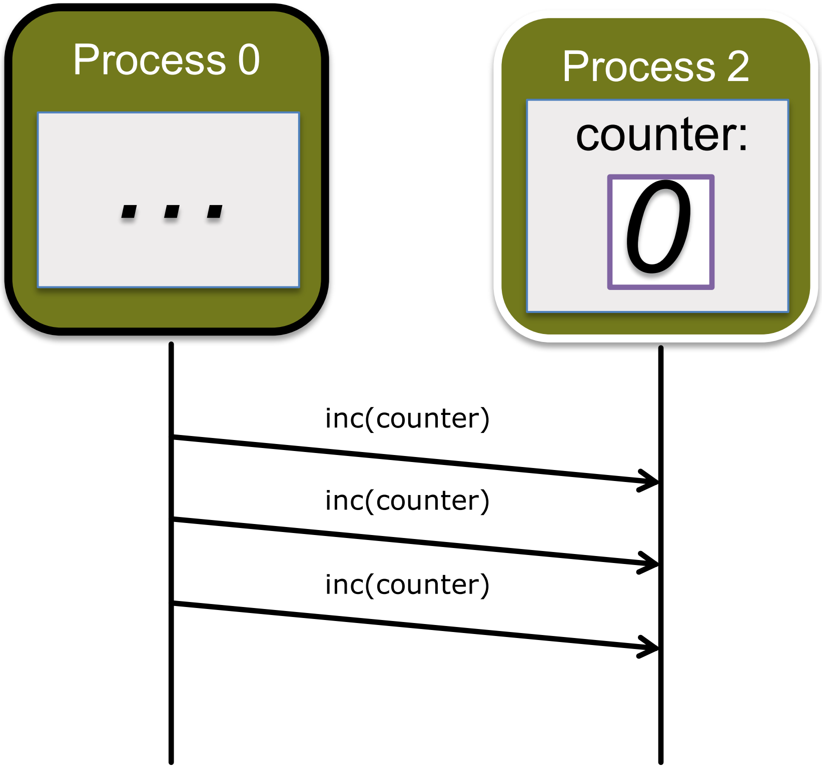
Time bound	$\mathcal{O}(\log p)$
Memory bound	$\mathcal{O}(1)$

Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13

Flush Synchronization

Time bound	$\mathcal{O}(1)$
Memory bound	$\mathcal{O}(1)$

- Guarantees remote completion
- Performs a remote bulk synchronization and an x86 `mfence`
- One of the most performance critical functions, we add only 78 x86 CPU instructions to the critical path

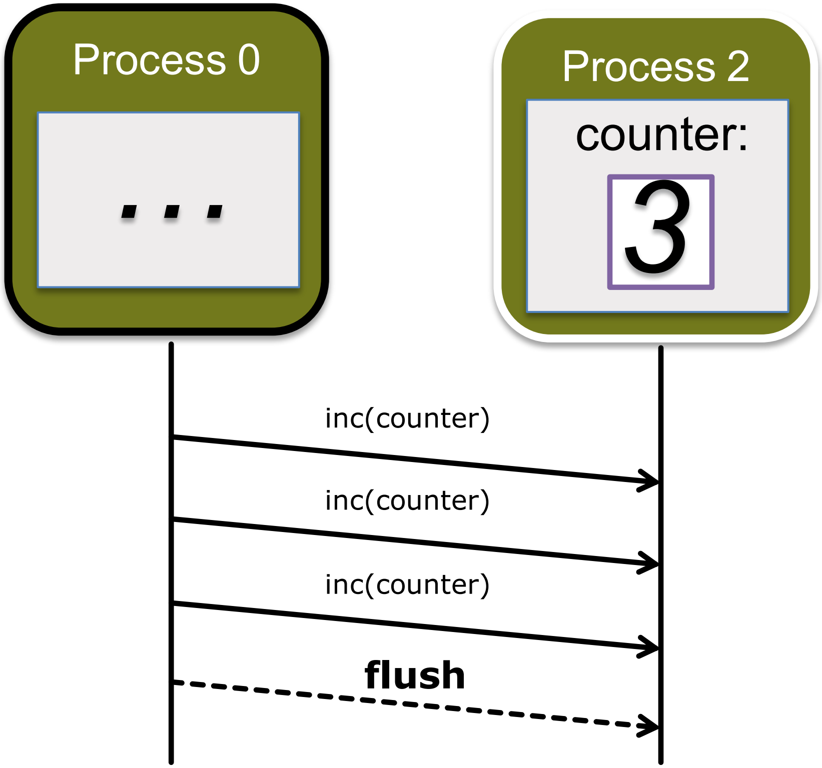


Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13

Flush Synchronization

Time bound	$\mathcal{O}(1)$
Memory bound	$\mathcal{O}(1)$

- Guarantees remote completion
- Performs a remote bulk synchronization and an x86 `mfence`
- One of the most performance critical functions, we add only 78 x86 CPU instructions to the critical path



Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13

Performance Modeling

Performance functions for synchronization protocols

Fence	$\mathcal{P}_{fence} = 2.9\mu s \cdot \log_2(p)$
PSCW	$\mathcal{P}_{start} = 0.7\mu s, \mathcal{P}_{wait} = 1.8\mu s$ $\mathcal{P}_{post} = \mathcal{P}_{complete} = 350ns \cdot k$
Locks	$\mathcal{P}_{lock,excl} = 5.4\mu s$ $\mathcal{P}_{lock,shrd} = \mathcal{P}_{lock\_all} = 2.7\mu s$ $\mathcal{P}_{unlock} = \mathcal{P}_{unlock\_all} = 0.4\mu s$ $\mathcal{P}_{flush} = 76ns$ $\mathcal{P}_{sync} = 17ns$

Performance functions for communication protocols

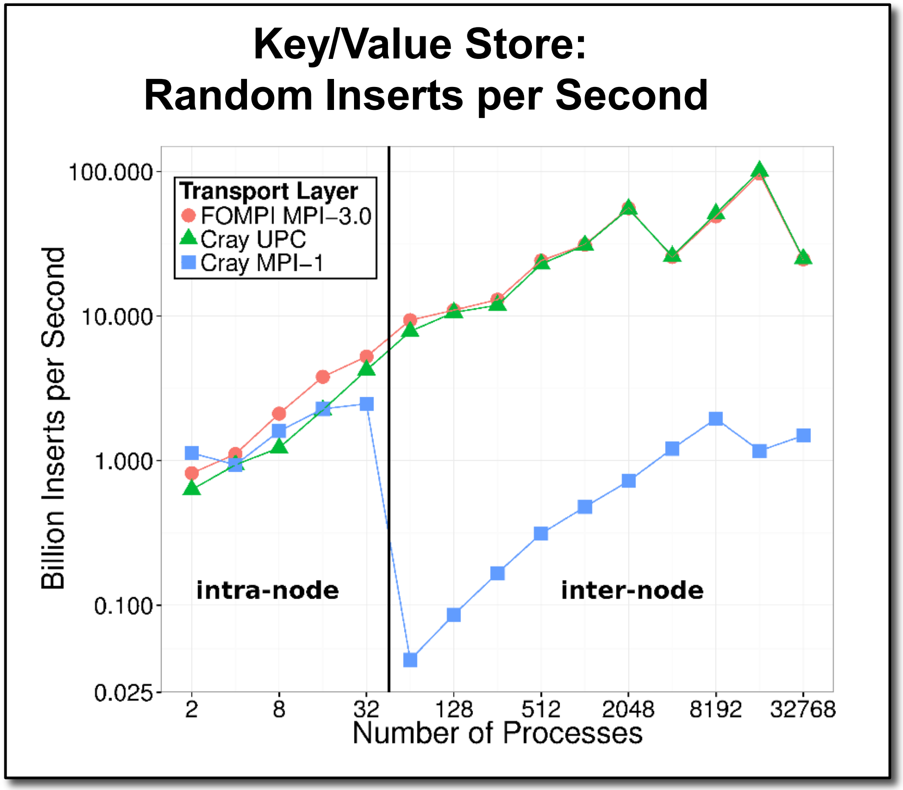
Put/get	$\mathcal{P}_{put} = 0.16ns \cdot s + 1\mu s$ $\mathcal{P}_{get} = 0.17ns \cdot s + 1.9\mu s$
Atomics	$\mathcal{P}_{acc,sum} = 28ns \cdot s + 2.4\mu s$ $\mathcal{P}_{acc,min} = 0.8ns \cdot s + 7.3\mu s$

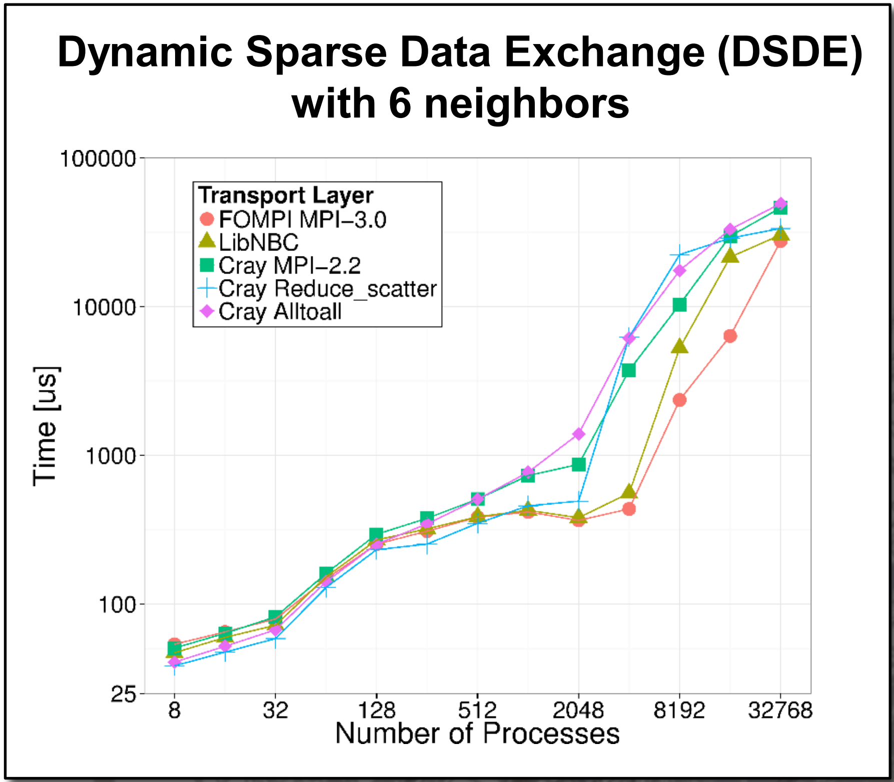
Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13

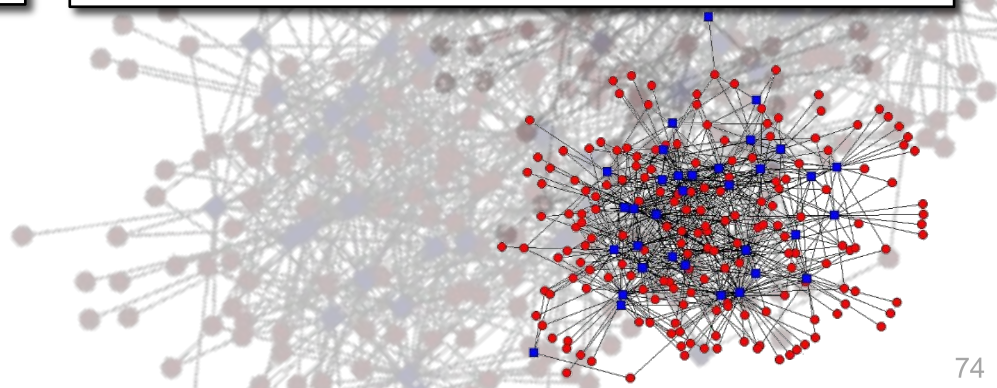
Application Performance

- Evaluation on Blue Waters System
  - 22,640 computing Cray XE6 nodes
  - 724,480 schedulable cores
- All microbenchmarks
- 4 applications
- One nearly full-scale run ☺

Performance: Motif Applications



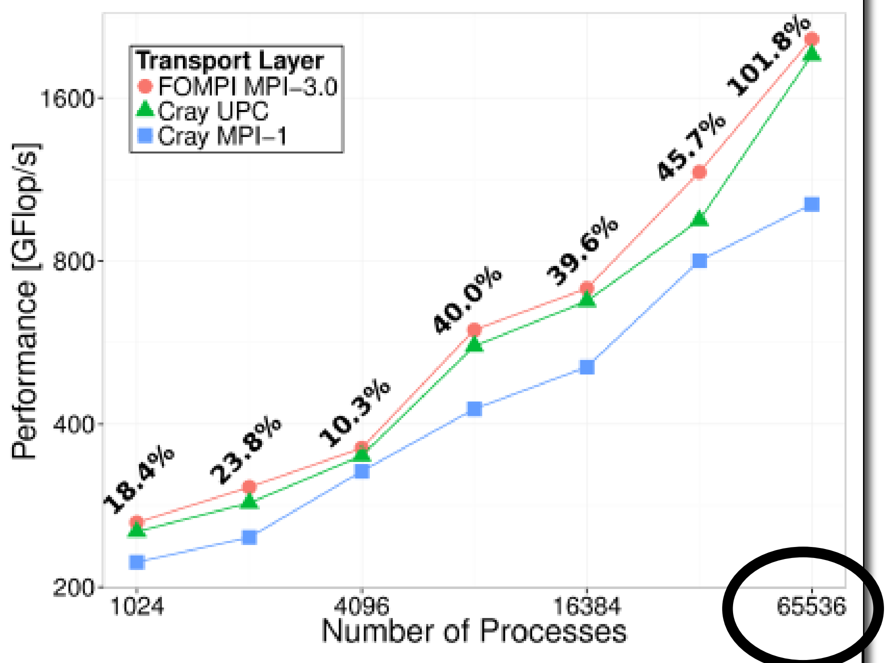




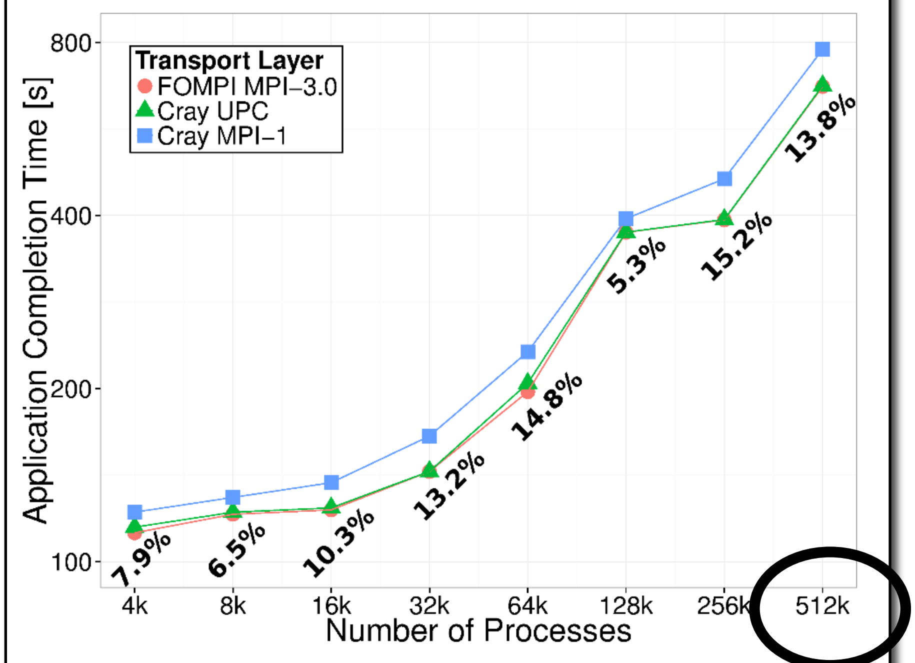
PERFORMANCE: APPLICATIONS

Annotations represent performance gain of foMPI [3] over Cray MPI-1.

NAS 3D FFT [1] Performance



MILC [2] Application Execution Time



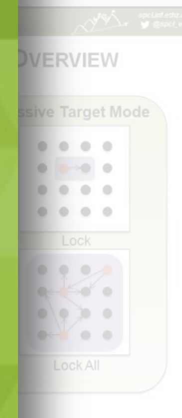
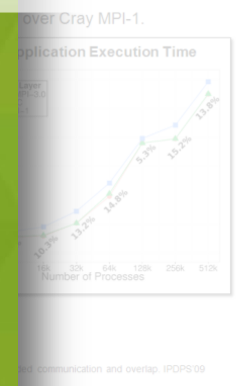
scale to 65k procs

scale to 512k procs

[1] Nishtala et al.: Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap. IPDPS'09
[2] Shan et al.: Accelerating applications at scale using one-sided communication. PGAS'12
[3] Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13

In case you want to learn more

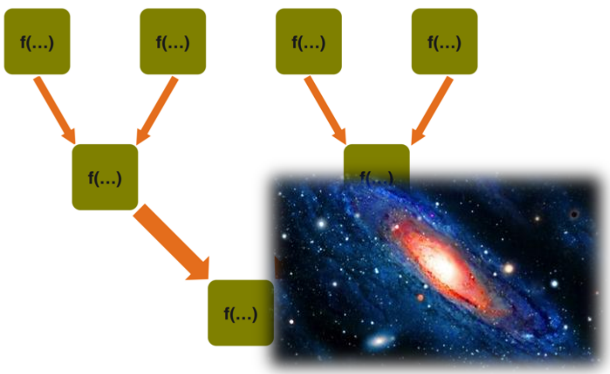
- Available
  - Some ar

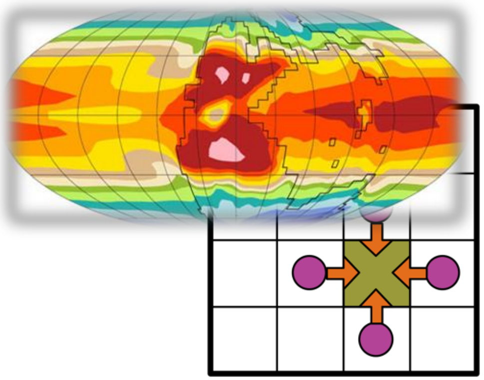



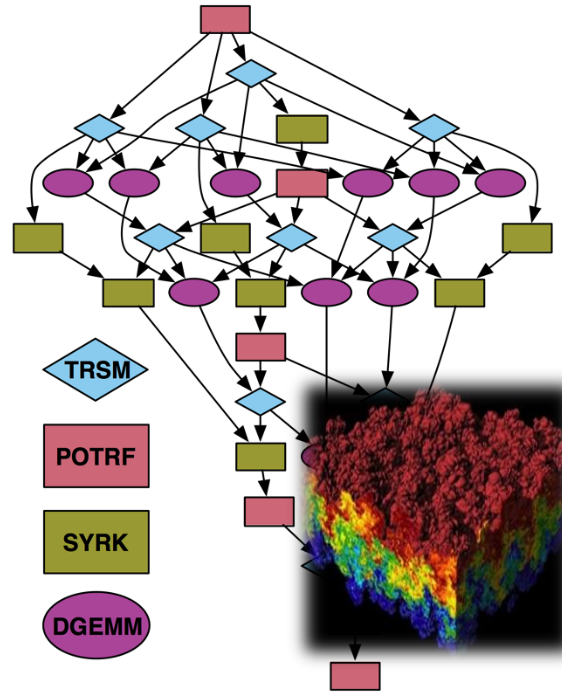
How to implement producer/consumer in passive mode?

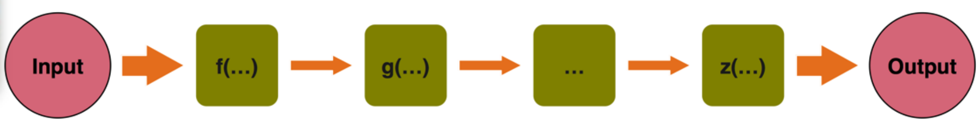
PRODUCER-CONSUMER RELATIONS

- **Most important communication idiom**
  - Some examples:

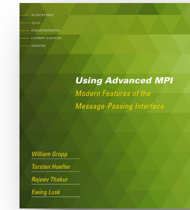




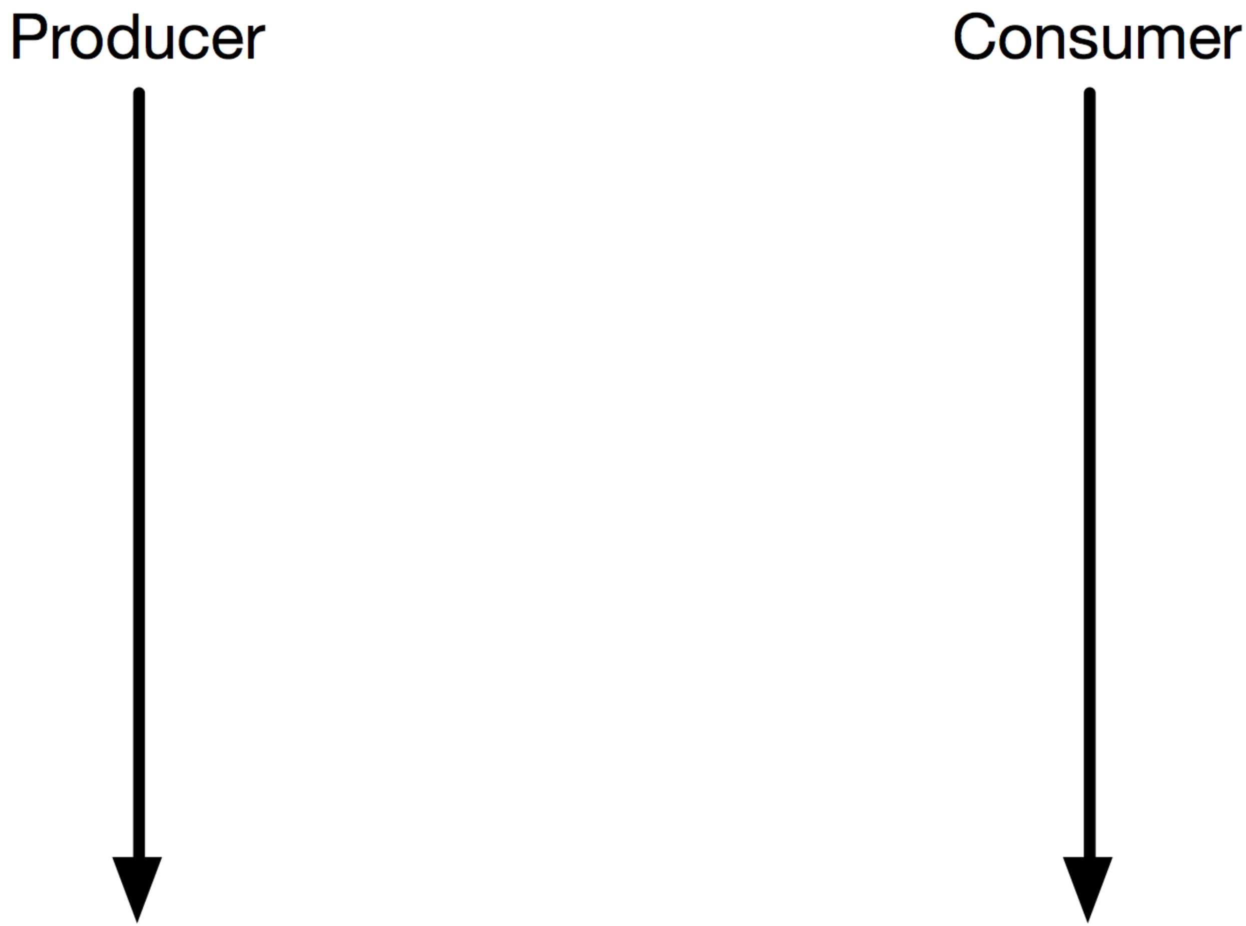




- **Perfectly supported by MPI-1 Message Passing**
  - But how does this actually work over RDMA?

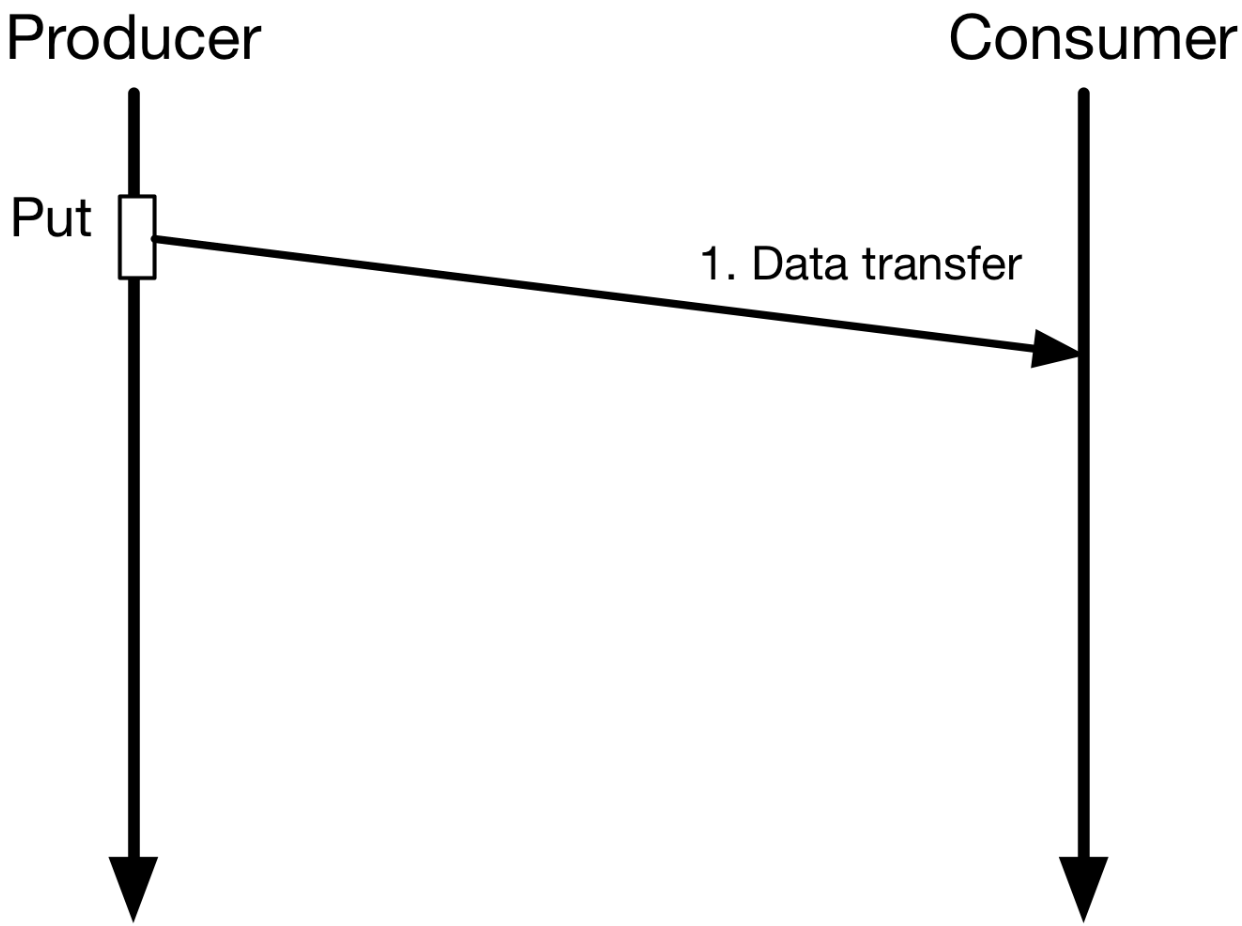


ONE SIDED – PUT + SYNCHRONIZATION



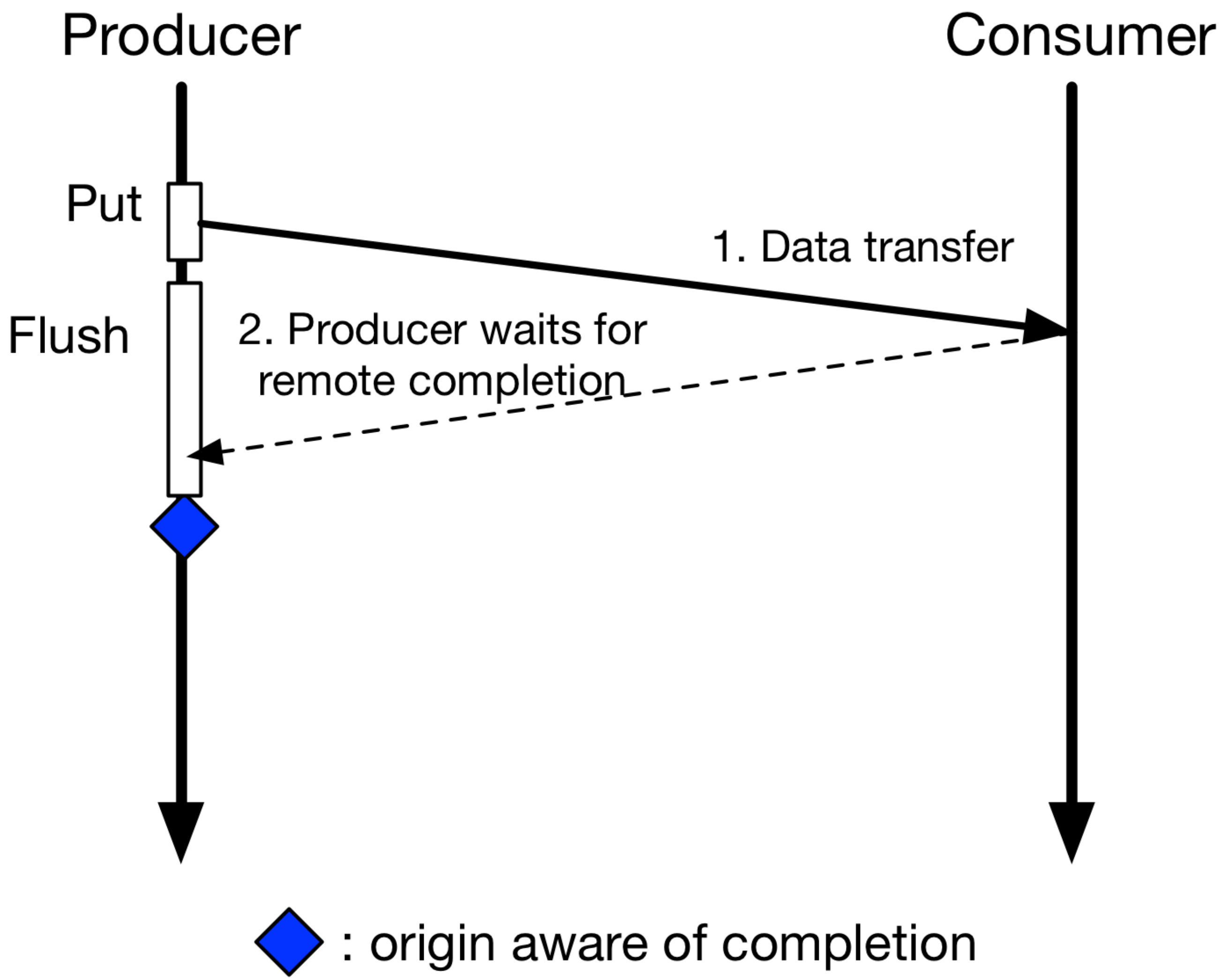
Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

ONE SIDED – PUT + SYNCHRONIZATION



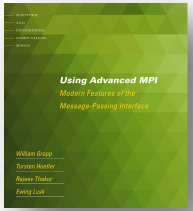
Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

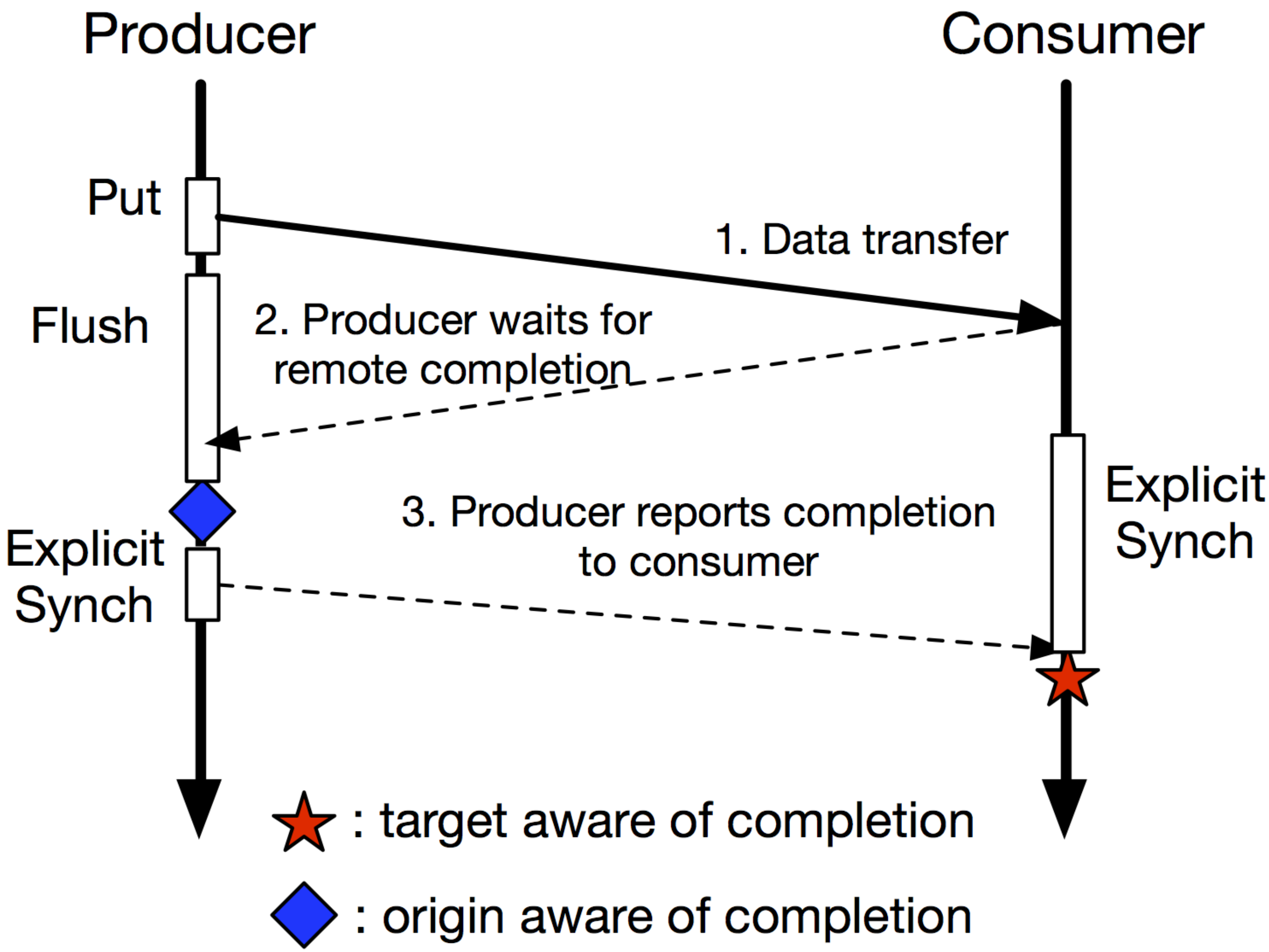
ONE SIDED – PUT + SYNCHRONIZATION



Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

ONE SIDED – PUT + SYNCHRONIZATION

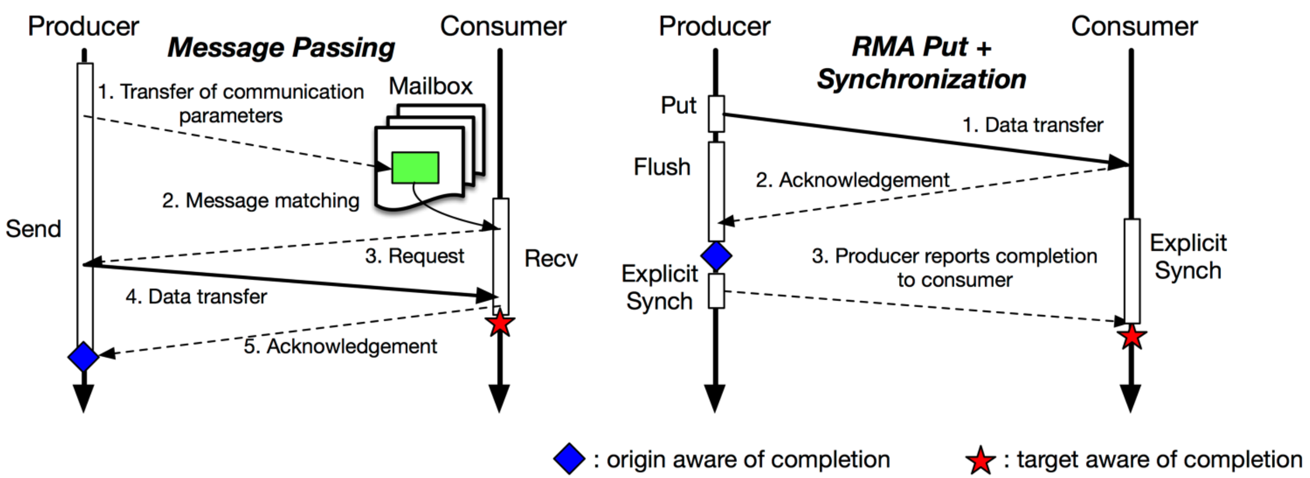




Critical path: 3 latencies

Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

Comparing Approaches



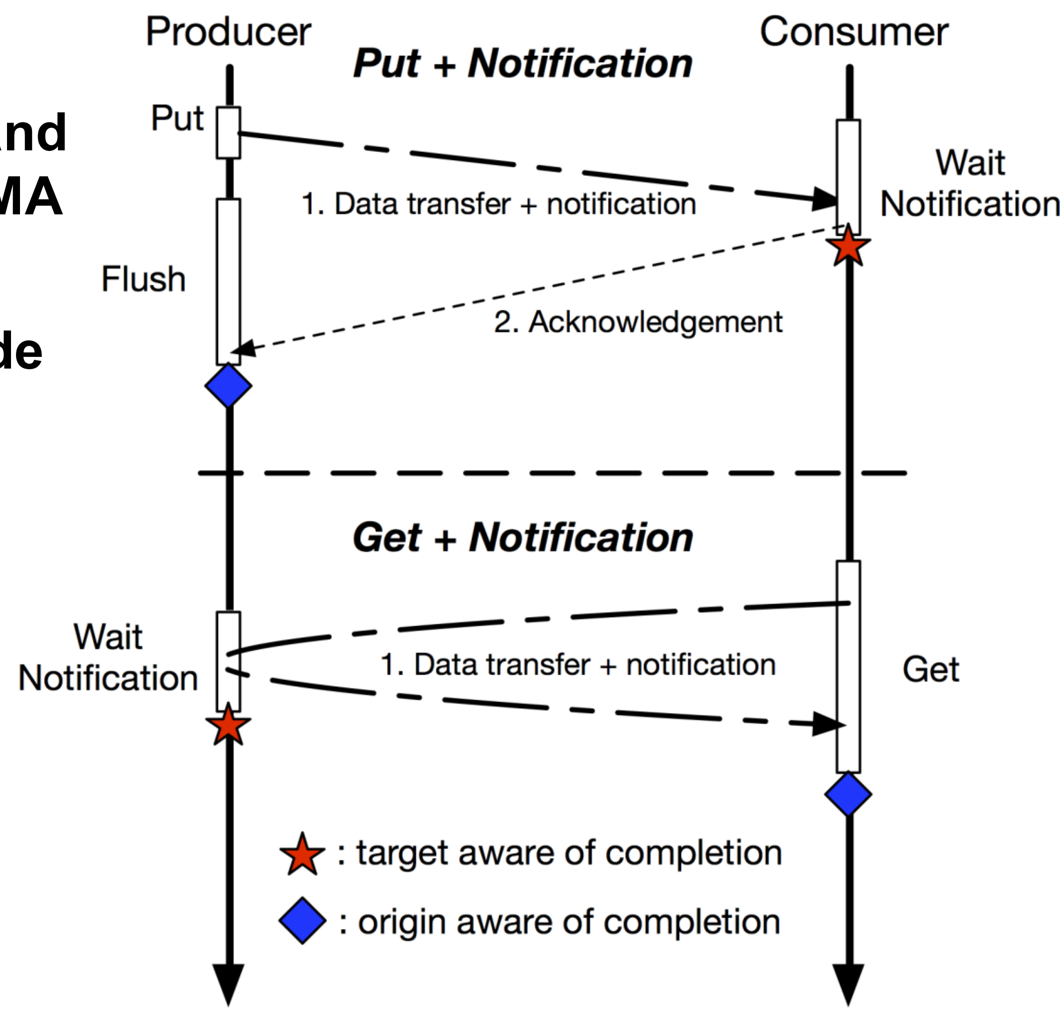
Message Passing
1 latency + copy / 3 latencies

One Sided
3 latencies

Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

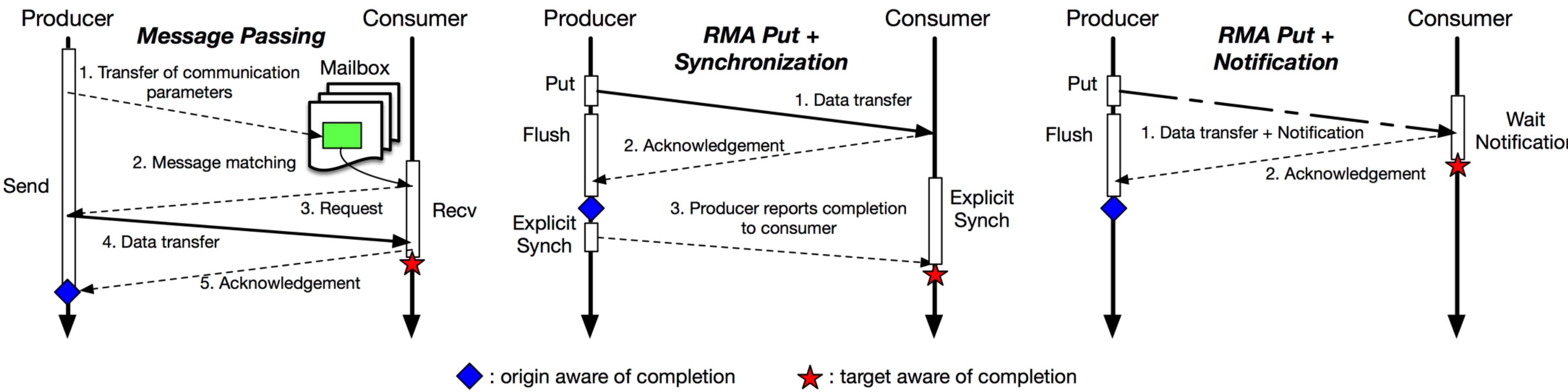
Idea: RMA Notifications

- **First seen in Split-C (1992)**
- **Combine communication and synchronization using RDMA**
- **RDMA networks can provide various notifications**
  - Flags
  - Counters
  - Event Queues



Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

COMPARING APPROACHES



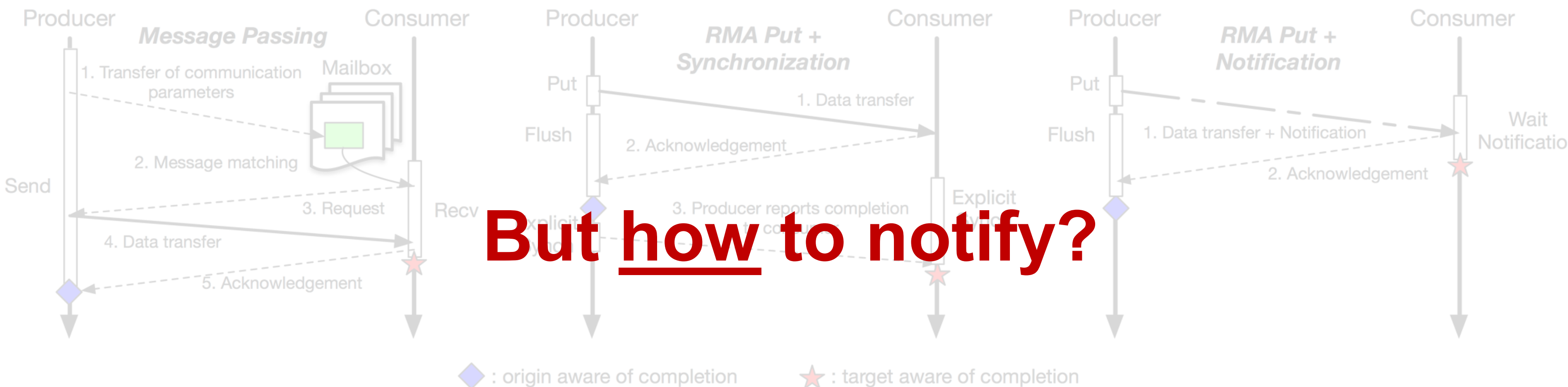
Message Passing
1 latency + copy / 3 latencies

One Sided
3 latencies

Notified Access
1 latency

Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

COMPARING APPROACHES



But <u>how</u> to notify?

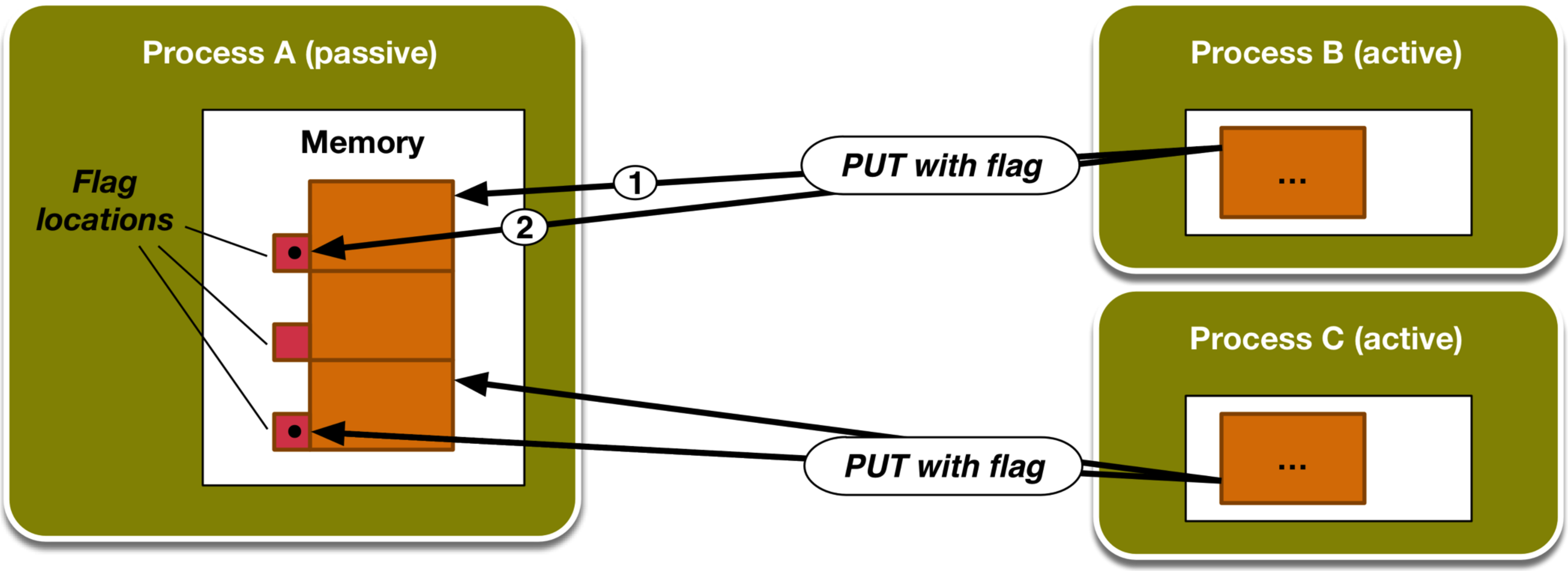
Message Passing
1 latency + copy / 3 latencies

One Sided
3 latencies

Notified Access
1 latency

PREVIOUS WORK: OVERWRITING INTERFACE

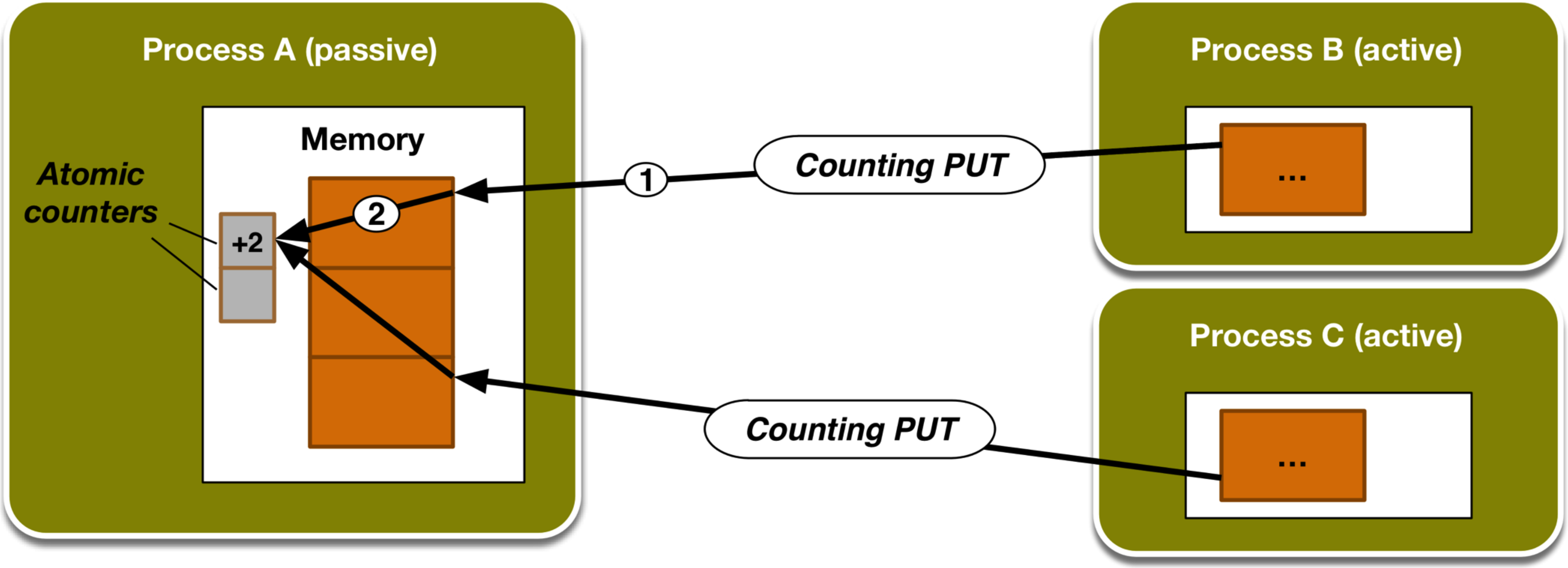
- **Flags (polling at the remote side)**
  - Used in *GASPI, DMAPP, NEON*



- **Disadvantages**
  - Location of the flag chosen at the sender side
  - Consumer needs at least one flag for every process
  - Polling a high number of flags is inefficient

PREVIOUS WORK: COUNTING INTERFACE

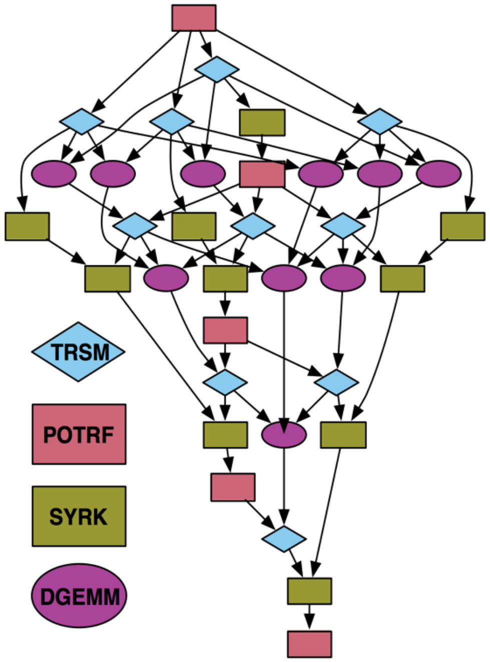
- **Atomic counters (accumulate notifications → scalable)**
  - Used in *Split-C, LAPI, SHMEM - Counting Puts, …*



- **Disadvantages**
  - Dataflow applications may require many counters
  - High polling overhead to identify accesses
  - Does not preserve order (may not be linearizable)

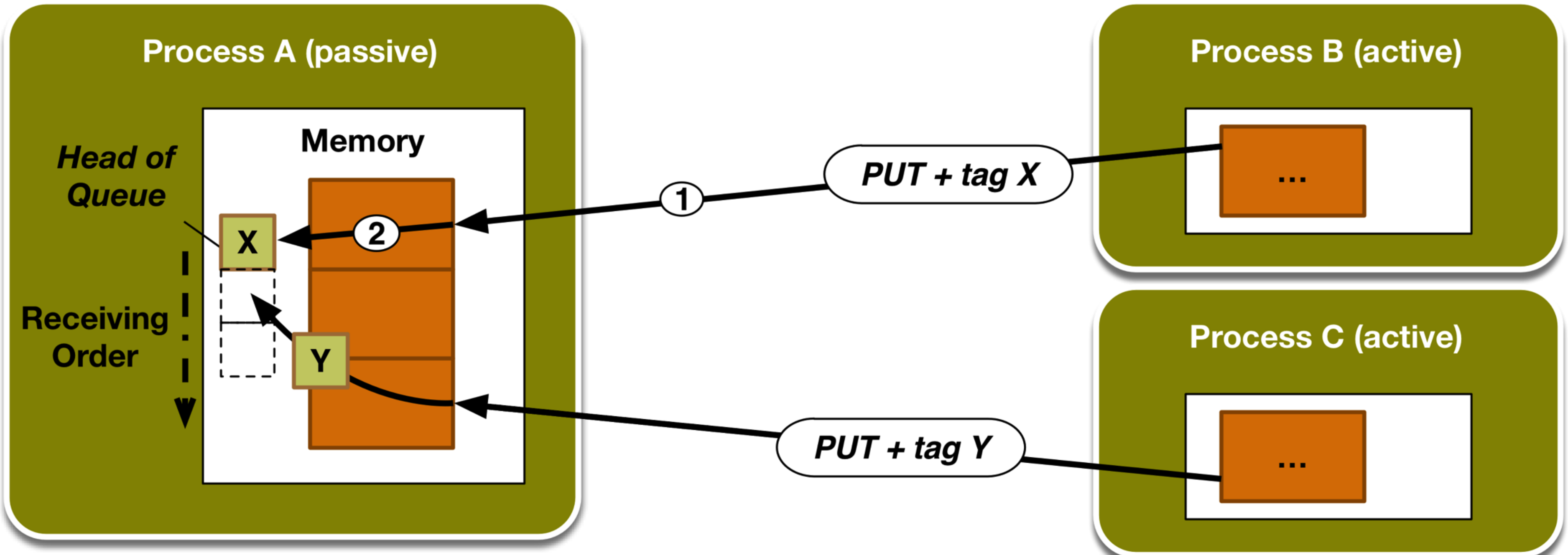
What is a Good Notification Interface?

- **Scalable to yotta-scale**
  - Does memory or polling overhead grow with # of processes?
- **Computation/communication overlap**
  - Do we support maximum asynchrony? (better than MPI-1)
- **Complex data flow graphs**
  - Can we distinguish between different accesses locally?
  - Can we avoid starvation?
  - What about load balancing?
- **Ease-of-use**
  - Does it use standard mechanisms?



Our Approach: Notified Access

- **Notifications with MPI-1 (queue-based) matching**
  - Retains benefits of previous notification schemes
  - Poll only head of queue
  - Provides linearizable semantics



Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

Notified Access – An MPI Interface

- **Minor interface evolution**
  - Leverages MPI two sided <source, tag> matching
  - Wildcards matching with FIFO semantics

Example Communication Primitives

```
int MPI_Put        (void *origin_addr, int origin_count, MPI_Datatype origin_type, int target_rank,
                    MPI_Aint target_disp, int target_count, MPI_Datatype target_type, MPI_Win win);

int MPI_Get        (void *origin_addr, int origin_count, MPI_Datatype origin_type, int target_rank,
                    MPI_Aint target_disp, int target_count, MPI_Datatype target_type, MPI_Win win);
```

Example Synchronization Primitives

```
/*Functions already available in MPI*/
int MPI_Start(MPI_Request *request);
int MPI_Test(MPI_Request *request, int *flag, MPI_Status *status);
int MPI_Wait(MPI_Request *request, MPI_Status *status);
```

Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

Notified Access – An MPI Interface

- **Minor interface evolution**
  - Leverages MPI two sided <source, tag> matching
  - Wildcards matching with FIFO semantics

Example Communication Primitives

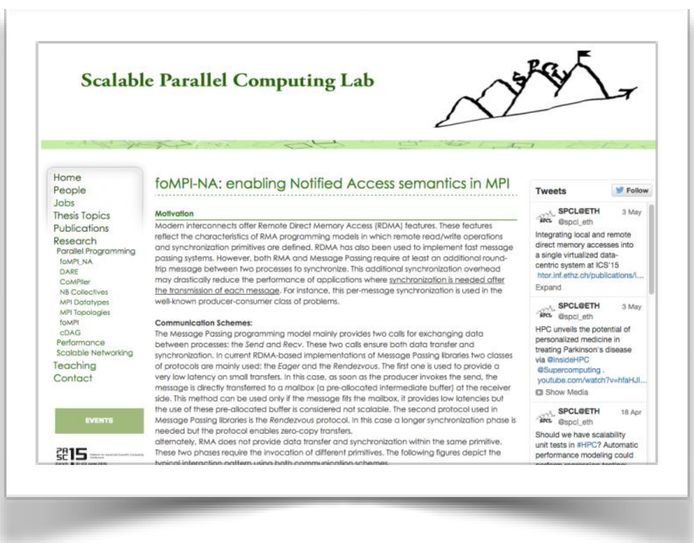
```
int MPI_Put_notify(void *origin_addr, int origin_count, MPI_Datatype origin_type, int target_rank,
                   MPI_Aint target_disp, int target_count, MPI_Datatype target_type, MPI_Win win,
                   int tag);
int MPI_Get_notify(void *origin_addr, int origin_count, MPI_Datatype origin_type, int target_rank,
                   MPI_Aint target_disp, int target_count, MPI_Datatype target_type, MPI_Win win,
                   int tag);
```

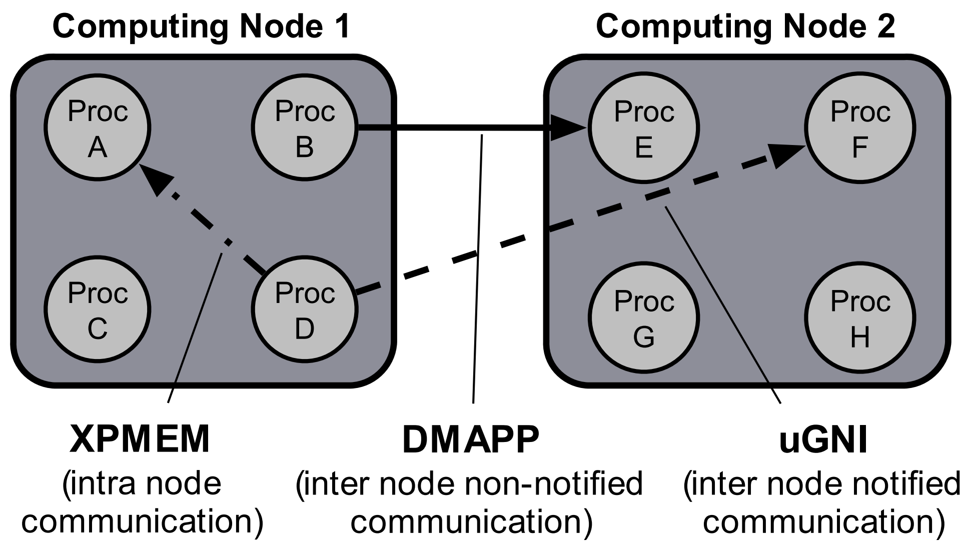
Example Synchronization Primitives

```
int MPI_Notify_init(MPI_Win win, int src_rank, int tag, int expected_count, MPI_Request *request);
/*Functions already available in MPI*/
int MPI_Start(MPI_Request *request);
int MPI_Test(MPI_Request *request, int *flag, MPI_Status *status);
int MPI_Wait(MPI_Request *request, MPI_Status *status);
```

Notified Access - Implementation

- **foMPI – a fully functional MPI-3 RMA implementation**
  - Runs on newer Cray machines (Aries, Gemini)
  - DMAPP: low-level networking API for Cray systems
  - XPMEM: a portable Linux kernel module
- **Implementation of Notified Access via uGNI [1]**
  - Leverages uGNI queue semantics
  - Adds unexpected queue
  - Uses 32-bit immediate value to encode source and tag





[1] http://spcl.inf.ethz.ch/Research/Parallel_Programming/foMPI_NA/

EXPERIMENTAL SETTING

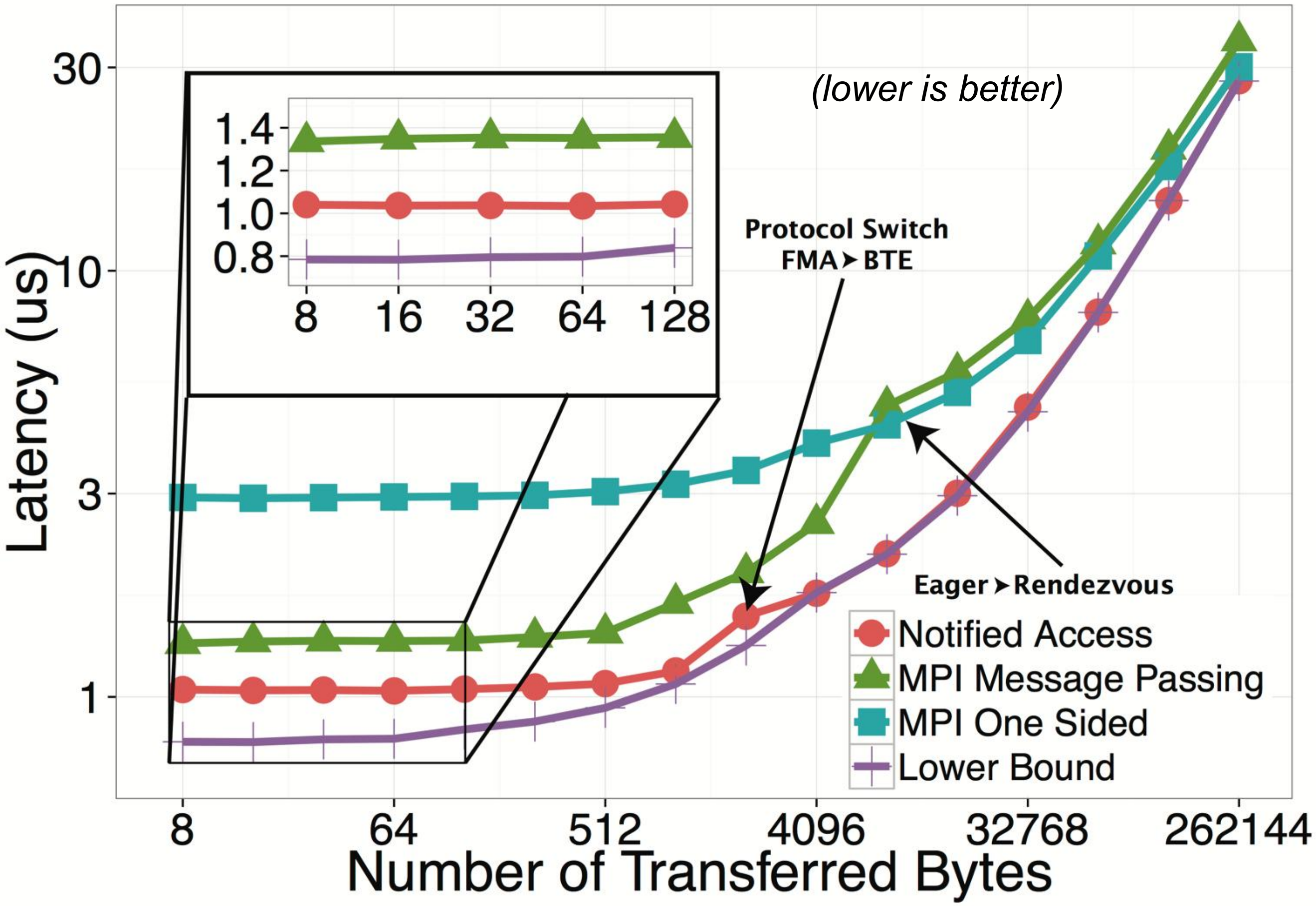
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

- **Piz Daint**
  - Cray XC30, Aries interconnect
  - 5'272 computing nodes (Intel Xeon E5-2670 + NVIDIA Tesla K20X)
  - Theoretical Peak Performance 7.787 Petaflops
  - Peak Network Bisection Bandwidth 33 TB/s

[1] http://www.cscs.ch

PING PONG PERFORMANCE (INTER-NODE)

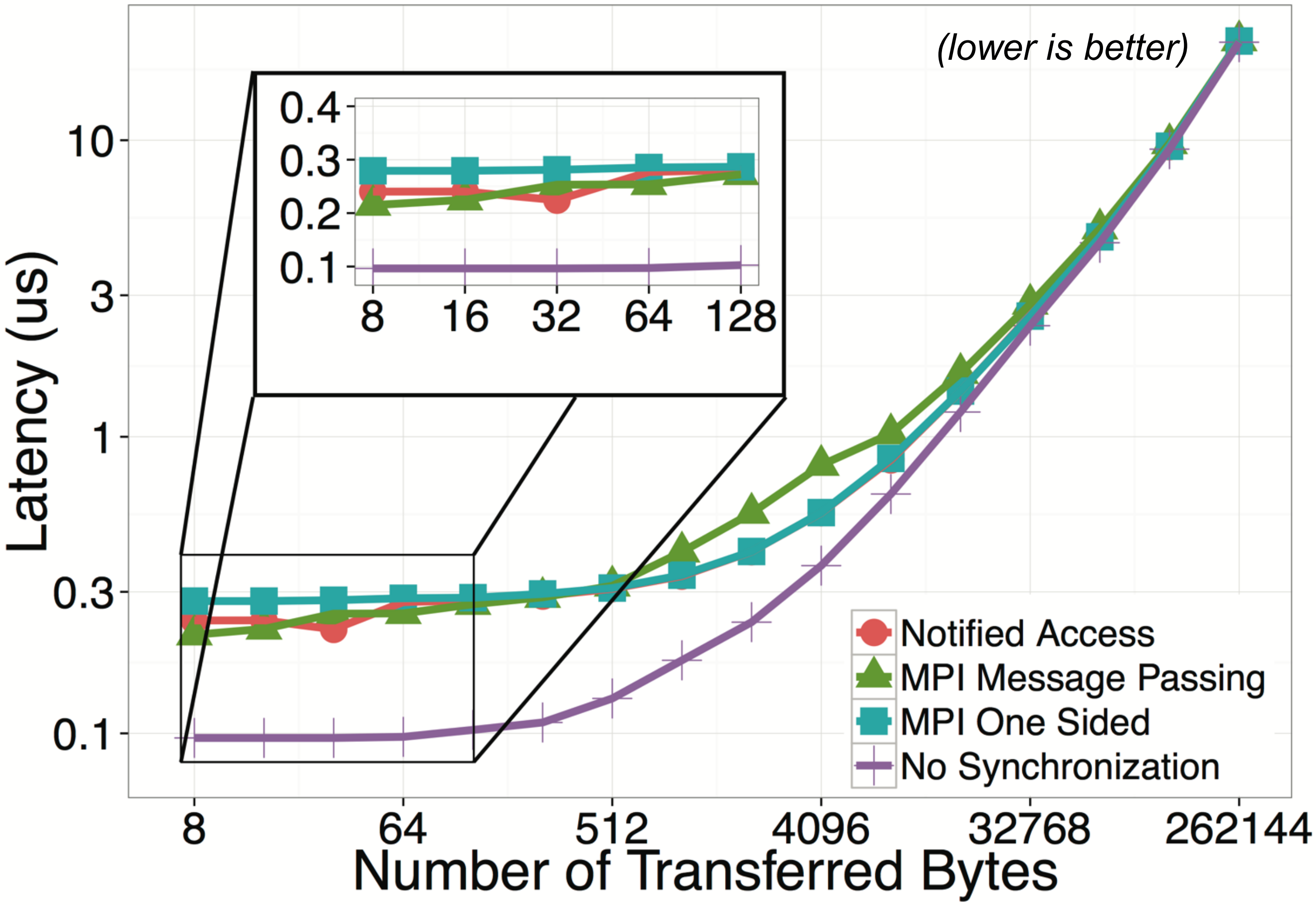
- **1000 repetitions, each timed separately, RDTSC timer**
- **95% confidence interval always within 1% of median**



Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

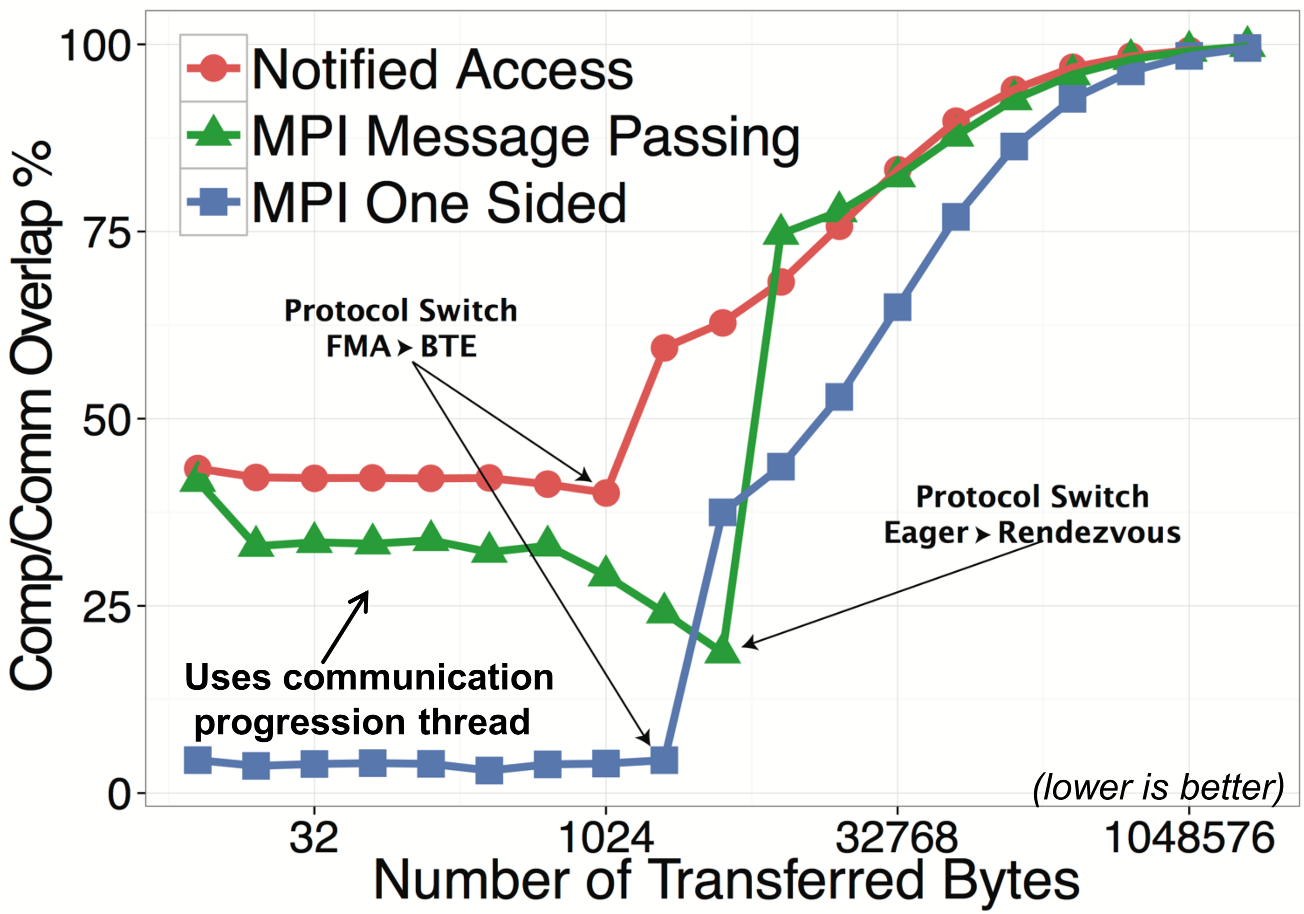
Ping Pong Performance (Intra-Node)

- 1000 repetitions, each timed separately, RDTSC timer
- 95% confidence interval always within 1% of median



COMPUTATION/COMMUNICATION OVERLAP

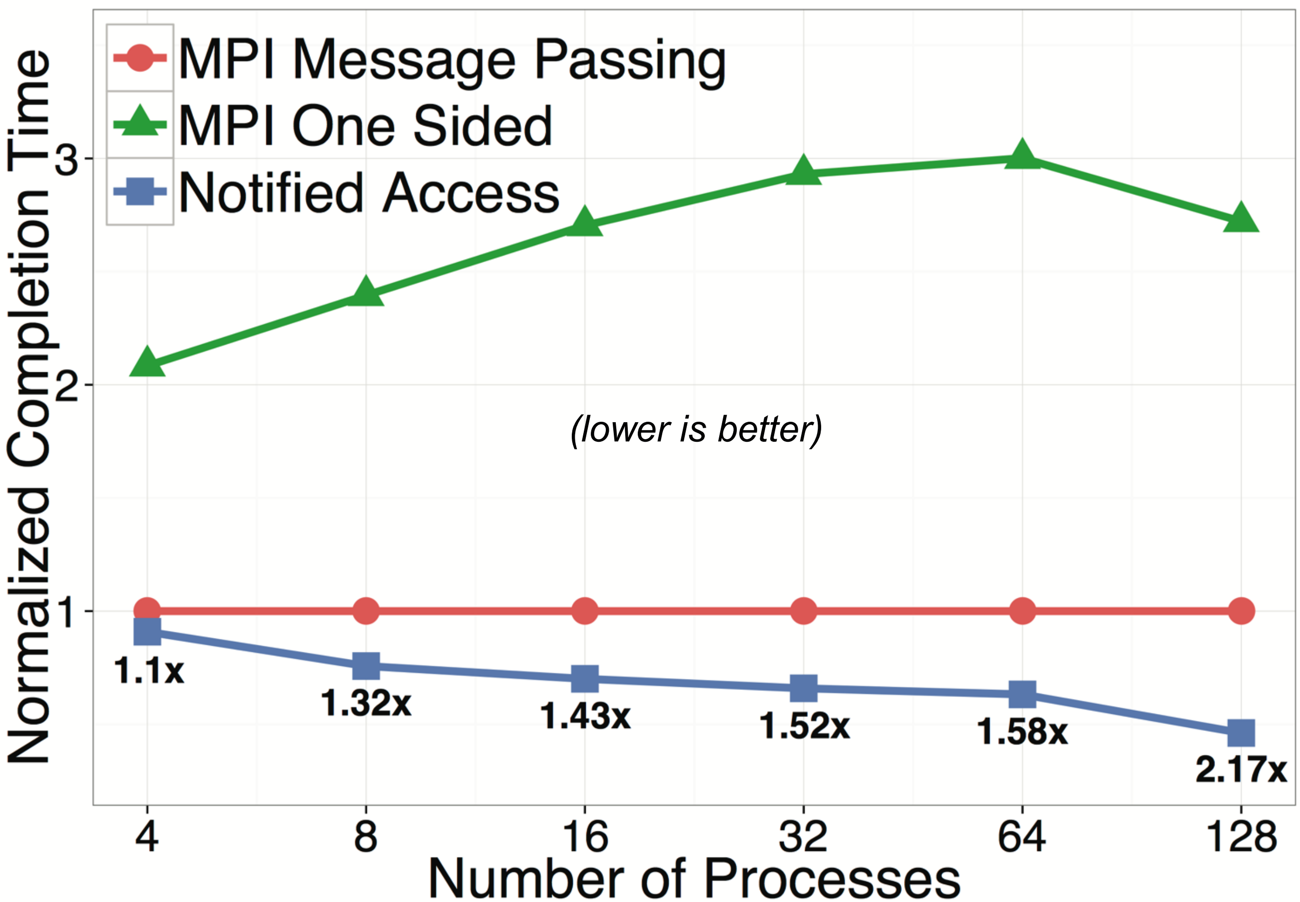
- 1000 repetitions, each timed separately, RDTSC timer
- 95% confidence interval always within 1% of median

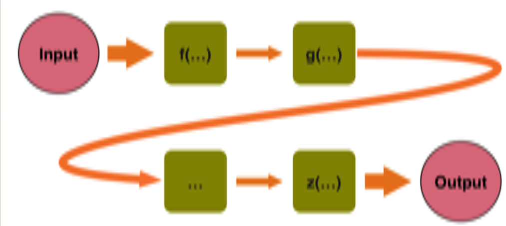


Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

Pipeline – One-to-One Synchronization

- **1000 repetitions, each timed separately, RDTSC timer**
- **95% confidence interval always within 1% of median**

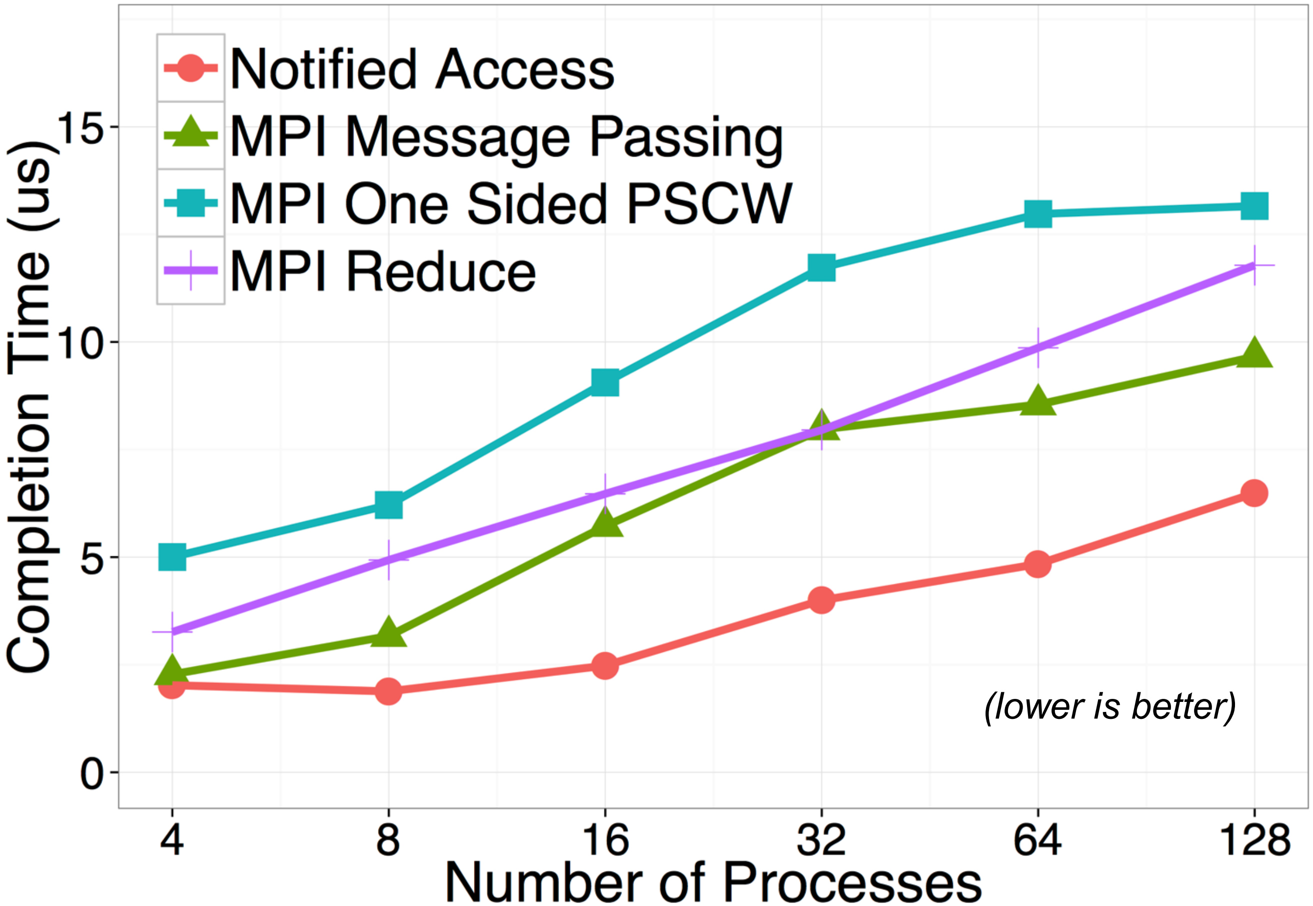


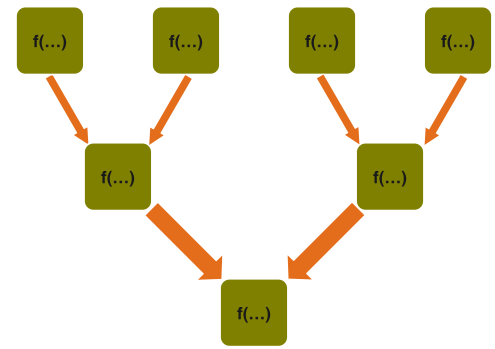


[1] https://github.com/intelesg/PRK2

Reduce – One-to-Many Synchronization

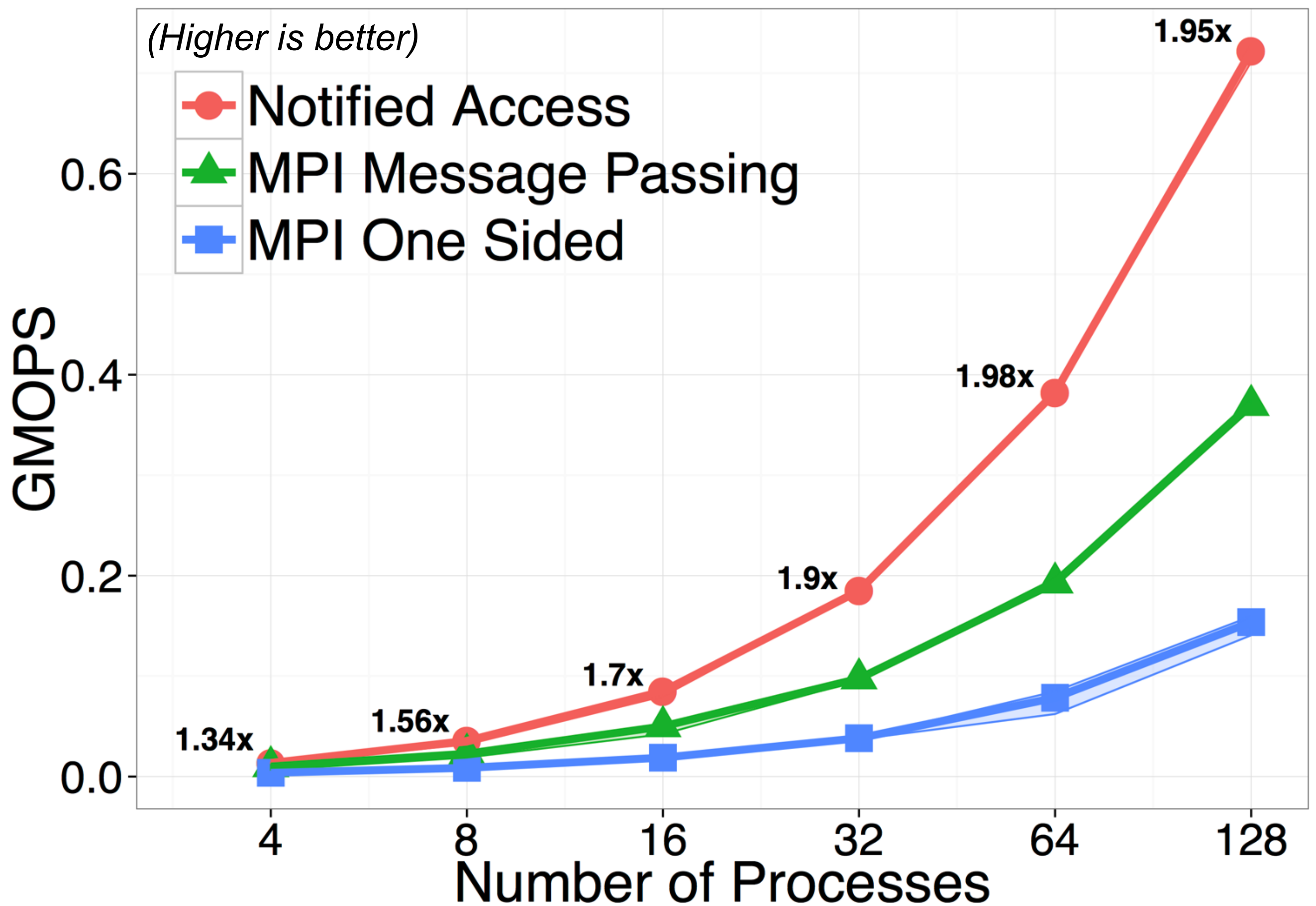
- **Reduce as an example (same for FMM, BH, etc.)**
  - Small data (8 Bytes), 16-ary tree
  - 1000 repetitions, each timed separately with RDTSC

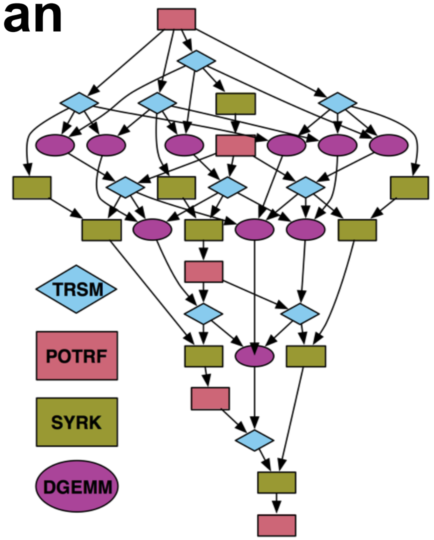




Cholesky – Many-to-Many Synchronization

- 1000 repetitions, each timed separately, RDTSC timer
- 95% confidence interval always within 10% of median





[1]: J. Kurzak, H. Ltaief, J. Dongarra, R. Badia: "Scheduling dense linear algebra operations on multicore processors", CCPE 2010

Discussion and Conclusions

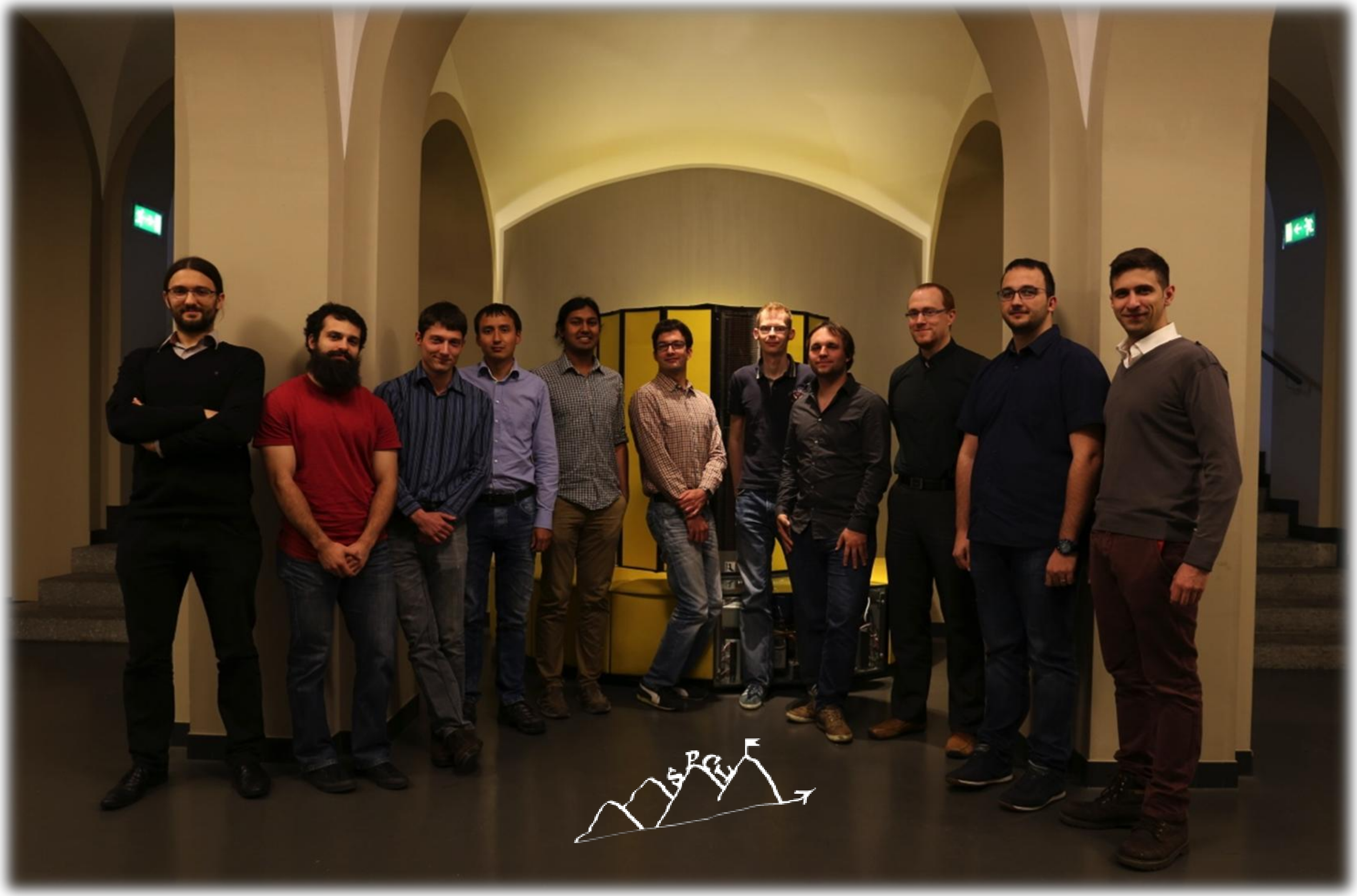
- **We develop a close-to-optimal network topology**
  - Spawns new research on adaptive routing
- **RDMA+SHM are de-facto hardware mechanisms**
  - Gives rise to RMA programming
- **MPI-3 RMA standardizes clear semantics**
  - Builds on existing practice (UPC, CAF, ARMCI etc.)
  - Rich set of synchronization mechanisms
- **Notified Access can support producer/consumer**
  - Maintains benefits of RDMA
- **Fully parameterized LogGP-like performance model**
  - Aids algorithm development and reasoning

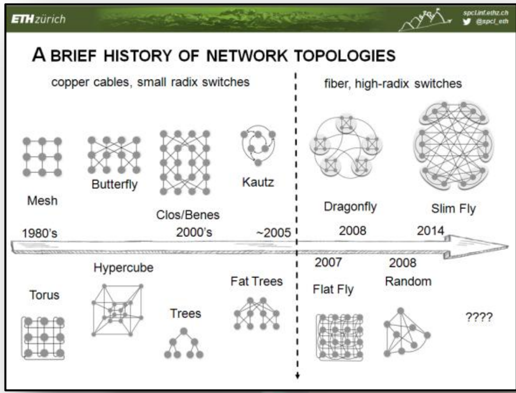
applicable at least to: OpenFabrics Alliance, portals, Cray Supercomputer

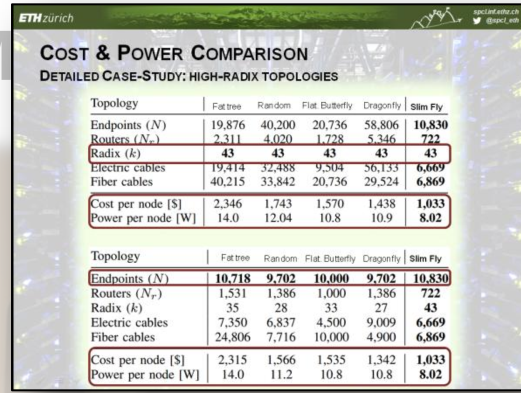
	Shared Memory	uGNI FMA	uGNI BTE
L	$0.25\mu s$	$1.02\mu s$	$1.32\mu s$
G	$0.08ns$	$0.105ns$	$0.101ns$

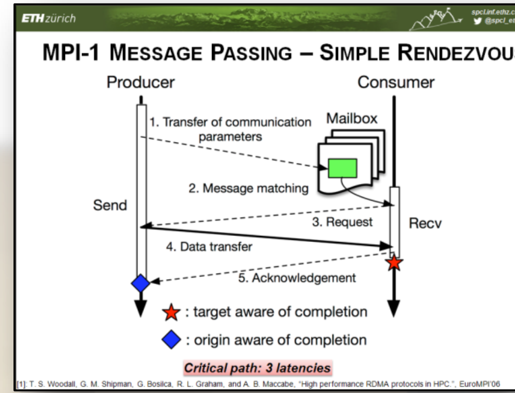
Function	Time
MPI_Notify_init	$t_{init} = 0.07\mu s$
MPI_Request_free	$t_{free} = 0.04\mu s$
MPI_Start	$t_{start} = 0.008\mu s$
MPI_{Put|Get}_notify	$t_{na} = 0.29\mu s$

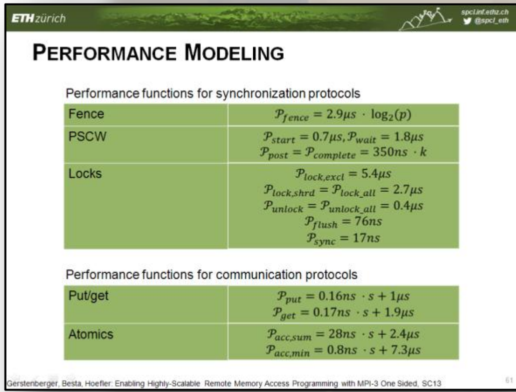
ACKNOWLEDGMENTS

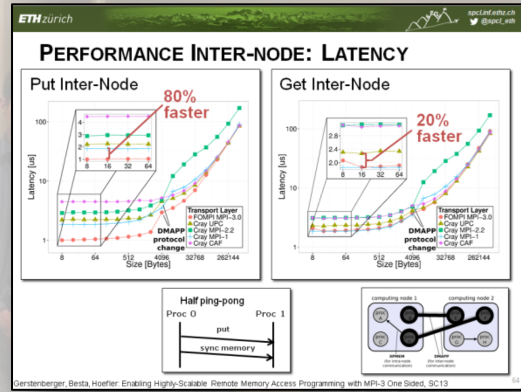


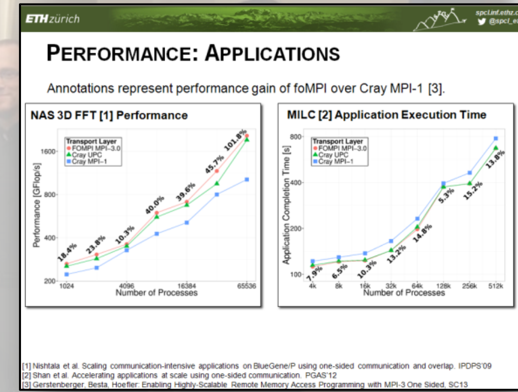

A BRIEF HISTORY OF NETWORK TOPOLOGIES
copper cables, small radix switches
fiber, high-radix switches
Mesh
Butterfly
Clos/Benes
Kautz
Dragonfly
Slim Fly
1980's
2000's
~2005
2008
2014
Hypercube
Torus
Trees
Fat Trees
2007
Flat Fly
2008
Random
????

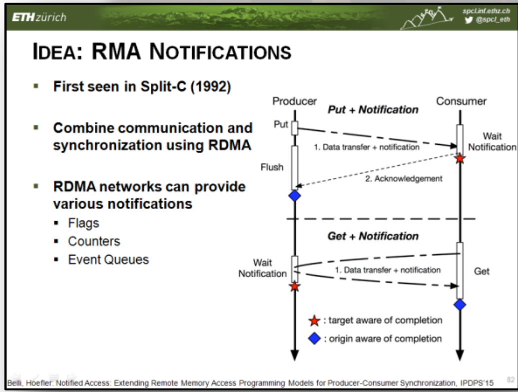

COST & POWER COMPARISON
DETAILED CASE-STUDY: HIGH-RADIX TOPOLOGIES
Topology | Fat tree | Random | Flat Butterfly | Dragonfly | Slim Fly
Endpoints (N) | 19,876 | 40,200 | 20,736 | 58,806 | 10,830
Routers (Nr) | 2,311 | 4,020 | 1,728 | 5,346 | 722
Radix (k) | 43 | 43 | 43 | 43 | 43
Electric cables | 19,414 | 32,488 | 9,504 | 56,133 | 6,669
Fiber cables | 40,215 | 33,842 | 20,736 | 29,524 | 6,869
Cost per node [$] | 2,346 | 1,743 | 1,570 | 1,438 | 1,033
Power per node [W] | 14.0 | 12.04 | 10.8 | 10.9 | 8.02
Topology | Fat tree | Random | Flat Butterfly | Dragonfly | Slim Fly
Endpoints (N) | 10,718 | 9,702 | 10,000 | 9,702 | 10,830
Routers (Nr) | 1,531 | 1,386 | 1,000 | 1,386 | 722
Radix (k) | 35 | 28 | 33 | 27 | 43
Electric cables | 7,350 | 6,837 | 4,500 | 9,009 | 6,669
Fiber cables | 24,806 | 7,716 | 10,000 | 4,900 | 6,869
Cost per node [$] | 2,315 | 1,566 | 1,535 | 1,342 | 1,033
Power per node [W] | 14.0 | 11.2 | 10.8 | 10.8 | 8.02

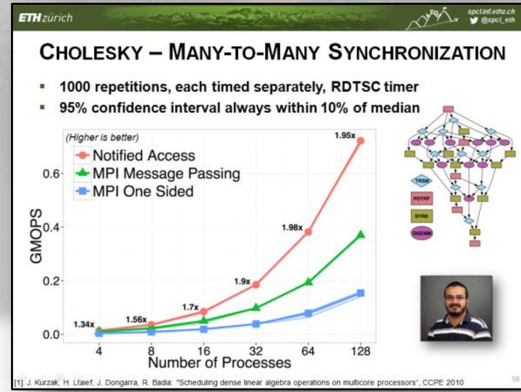

MPI-1 MESSAGE PASSING – SIMPLE RENDEZVOUS
Producer
Consumer
1. Transfer of communication parameters
Mailbox
2. Message matching
Send
Recv
3. Request
4. Data transfer
5. Acknowledgement
: target aware of completion
: origin aware of completion
Critical path: 3 latencies
[1] T. S. Woodall, G. M. Shipman, G. Bosilca, R. L. Graham, and A. B. Maccabe, "High performance RDMA protocols in HPC.", EuroMPI'06

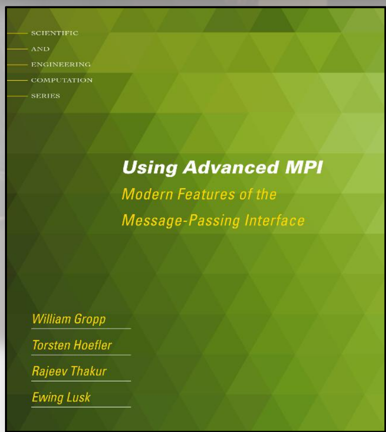

PERFORMANCE MODELING
Performance functions for synchronization protocols
Fence $\mathcal{P}_{fence} = 2.9\mu s \cdot \log_2(p)$
PSCW $\mathcal{P}_{start} = 0.7\mu s, \mathcal{P}_{wait} = 1.8\mu s$
$\mathcal{P}_{post} = \mathcal{P}_{complete} = 350ns \cdot k$
Locks $\mathcal{P}_{lock,excl} = 5.4\mu s$
$\mathcal{P}_{lock,shrd} = \mathcal{P}_{lock\_all} = 2.7\mu s$
$\mathcal{P}_{unlock} = \mathcal{P}_{unlock\_all} = 0.4\mu s$
$\mathcal{P}_{flush} = 76ns$
$\mathcal{P}_{sync} = 17ns$
Performance functions for communication protocols
Put/get $\mathcal{P}_{put} = 0.16ns \cdot s + 1\mu s$
$\mathcal{P}_{get} = 0.17ns \cdot s + 1.9\mu s$
Atomics $\mathcal{P}_{acc,sum} = 28ns \cdot s + 2.4\mu s$
$\mathcal{P}_{acc,min} = 0.8ns \cdot s + 7.3\mu s$
Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13


PERFORMANCE INTER-NODE: LATENCY
Put Inter-Node
80% faster
Get Inter-Node
20% faster
Latency [us]
Size [Bytes]
Transport Layer
FOMPI MPI-3.0
Cray UPC
Cray MPI-2.2
Cray MPI-1
Cray CAF
DMAPP protocol change
Half ping-pong
Proc 0
Proc 1
put
sync memory
Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13


PERFORMANCE: APPLICATIONS
Annotations represent performance gain of foMPI over Cray MPI-1 [3].
NAS 3D FFT [1] Performance
MILC [2] Application Execution Time
Performance [GFlop/s]
Number of Processes
Application Completion Time [s]
[1] Nishtala et al. Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap. IPDPS'09
[2] Shan et al. Accelerating applications at scale using one-sided communication. PGAS'12
[3] Gerstenberger, Besta, Hoefler: Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided, SC13


IDEA: RMA NOTIFICATIONS
First seen in Split-C (1992)
Combine communication and synchronization using RDMA
RDMA networks can provide various notifications
Flags
Counters
Event Queues
Producer
Consumer
Put + Notification
Put
Wait Notification
1. Data transfer + notification
Flush
2. Acknowledgement
Get + Notification
Wait Notification
Get
: target aware of completion
: origin aware of completion
Belli, Hoefler: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15

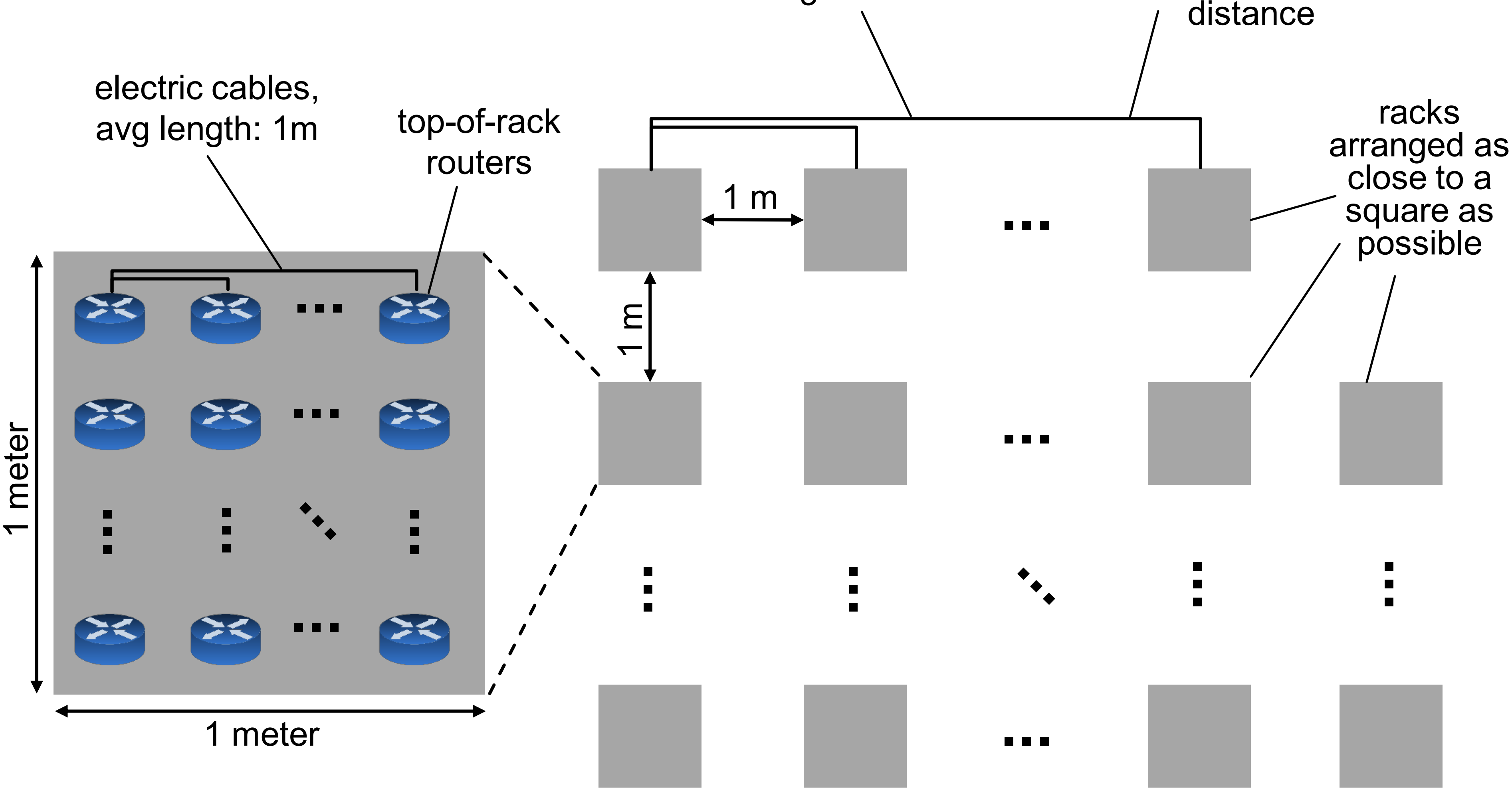

CHOLESKY – MANY-TO-MANY SYNCHRONIZATION
1000 repetitions, each timed separately, RDTSC timer
95% confidence interval always within 10% of median
(Higher is better)
Notified Access
MPI Message Passing
MPI One Sided
GMOPS
Number of Processes
[1] J. Kurzak, H. Ltaief, J. Dongarra, R. Badia: "Scheduling dense linear algebra operations on multicore processors", CCPE 2010


SCIENTIFIC
AND
ENGINEERING
COMPUTATION
SERIES
Using Advanced MPI
Modern Features of the Message-Passing Interface
William Gropp
Torsten Hoefler
Rajeev Thakur
Ewing Lusk

Cost Comparison

Cost Model

*Most cables skipped for clarity



Cost Comparison

Cable Cost Model

*Prices based on: COLFAX DIRECT HPC and Data Center Gear

- Cable cost as a function of distance
  - The functions obtained using linear regression*
  - Cables used:
    Mellanox IB FDR10 40Gb/s QSFP

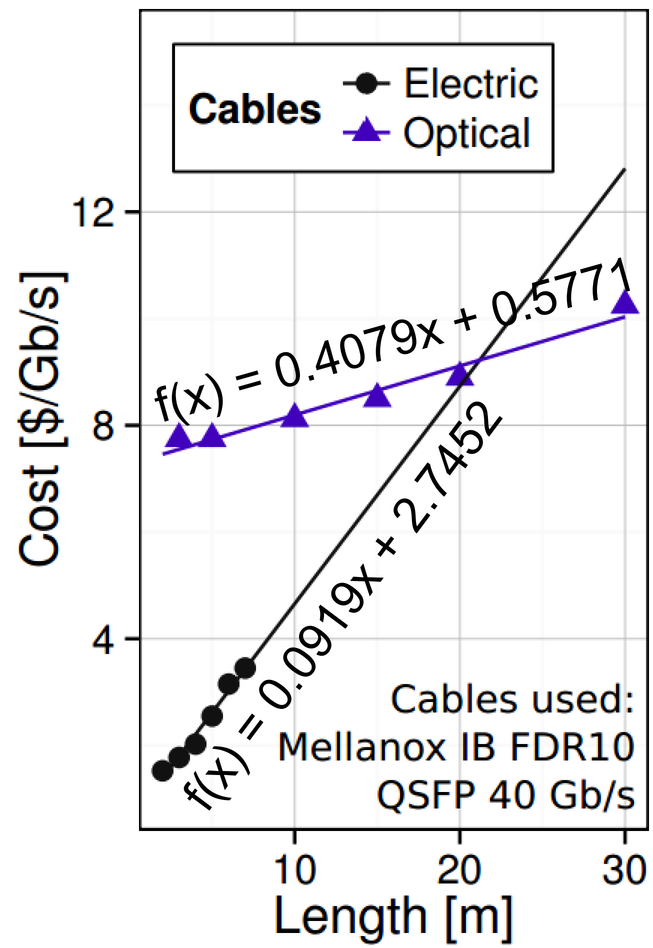
- Other used cables:

Mellanox IB QDR 56Gb/s QSFP

Mellanox Ethernet 40Gb/s QSFP

Mellanox Ethernet 10Gb/s SFP+

Elpeus Ethernet 10Gb/s SFP+



Cost Comparison

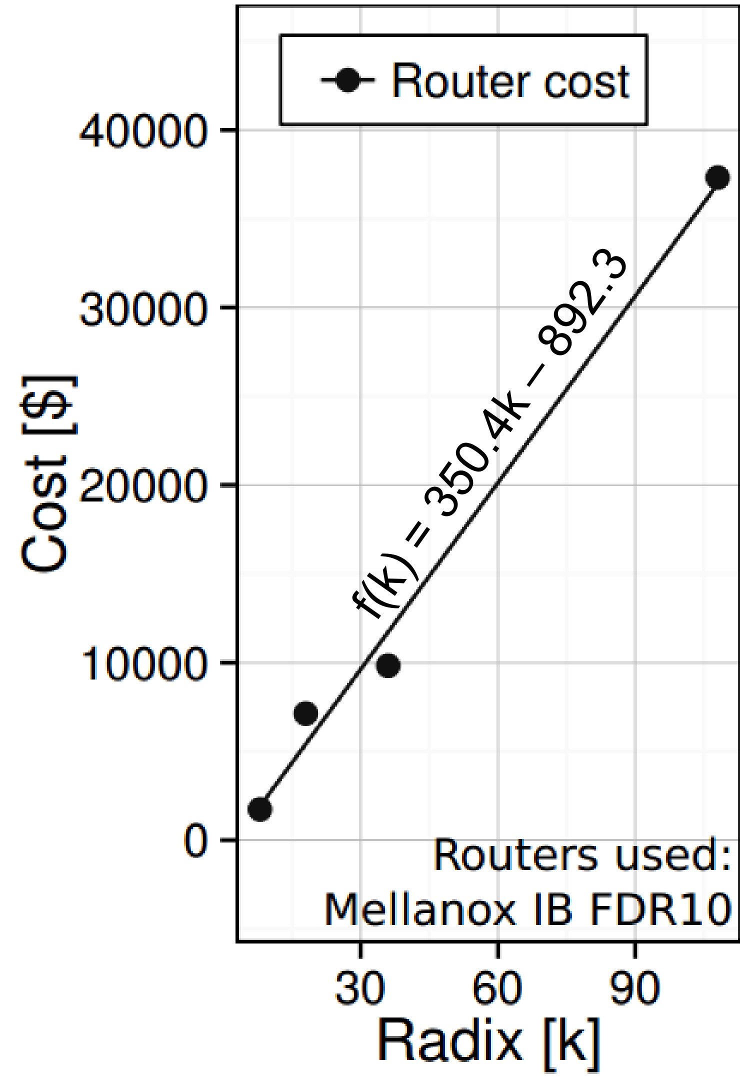
Router Cost Model

*Prices based on: COLFAX DIRECT HPC and Data Center Gear

- Router cost as a function of radix
  - The function obtained using linear regression*
  - Routers used:

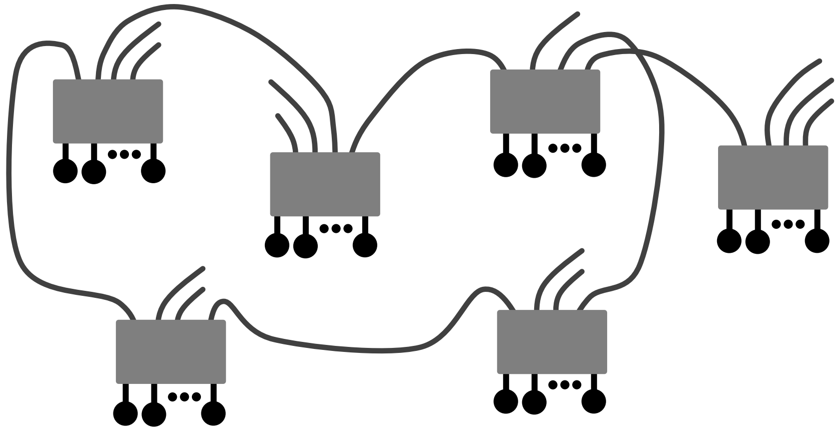
Mellanox IB FDR10

Mellanox Ethernet 10/40 Gb



STRUCTURE ANALYSIS

RESILIENCY



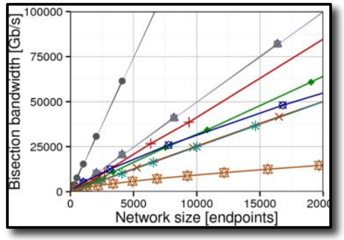
- Disconnection metrics*
- Other studied metrics:
  - Average path length (increase by 2); SF is 10% more resilient than DF

$\approx N$	Torus3D	Torus5D	Hypercube	Long Hop	Fat tree	Dragonfly	Flat. Butterfly	Random	**Slim Fly**
512	30%	-	40%	55%	35%	-	55%	60%	**60%**
1024	25%	40%	40%	55%	40%	50%	60%	-	**-**
2048	20%	-	40%	55%	40%	55%	65%	65%	**65%**
4096	15%	-	45%	55%	55%	60%	70%	70%	**70%**
8192	10%	35%	45%	55%	60%	65%	-	75%	**75%**

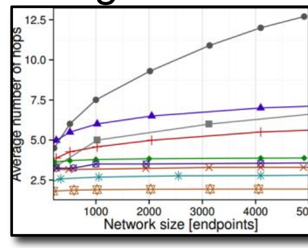
*Missing values indicate the inadequacy of a balanced topology variant for a given N

OTHER RESULTS

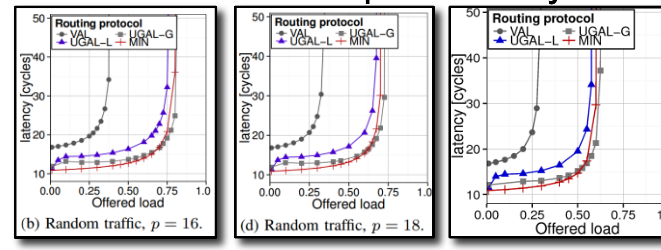
Bisection b.



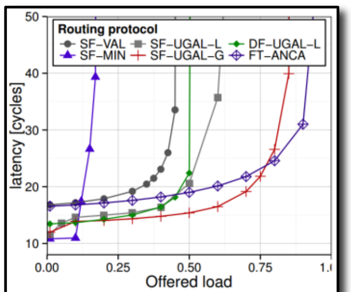
Avg. distance



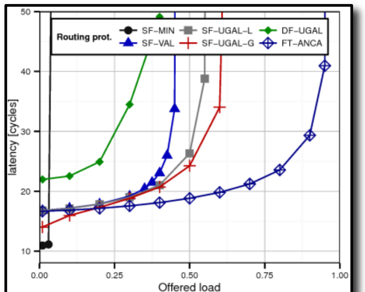
Oversubscription analysis



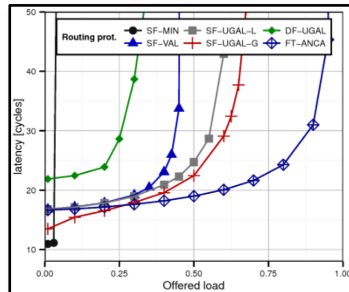
Bit reverse



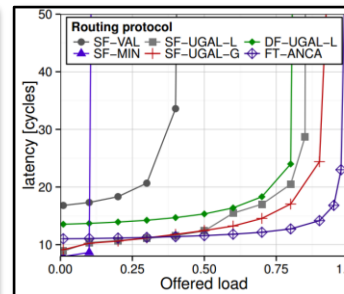
Bit complement



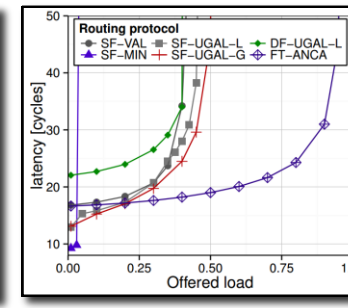
Shuffle



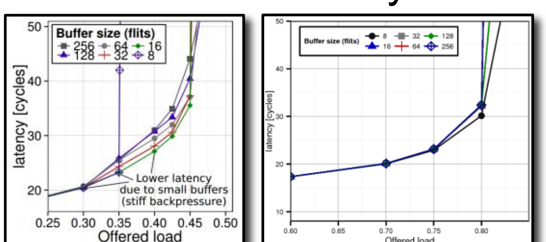
Shift



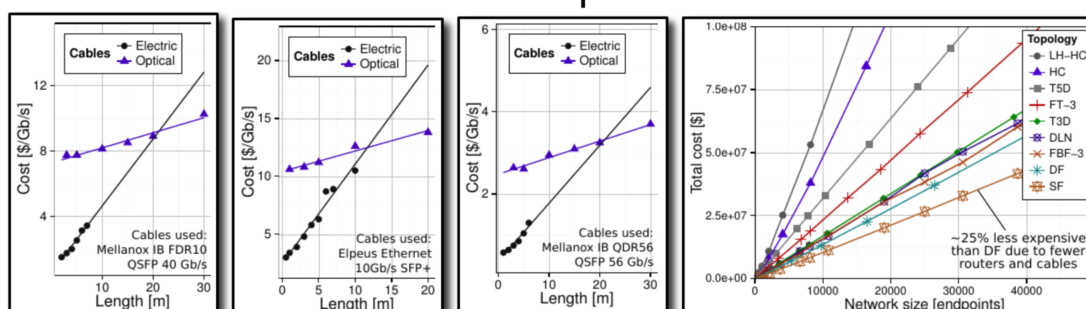
Adversarial

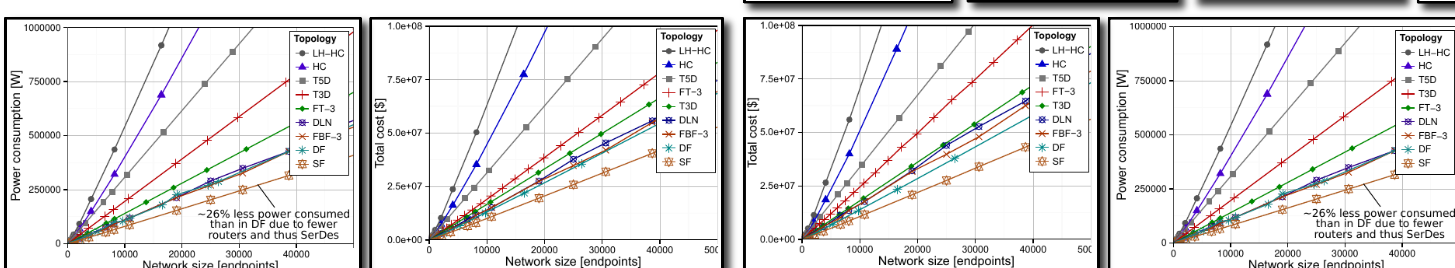


Buffer size analysis



Other cost & power results



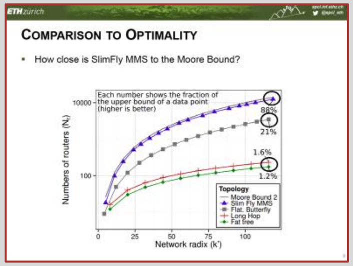


Topology	Dragonfly	Slim Fly
Endpoints (N)	10,890	10,830
Routers (N_r)	990	722
Radix (k)	43	43
Electric cables	6,885	6,669
Fiber cables	1,012	6,869
Cost per node [$]	1,365	1,033
Power per node [W]	10.9	8.02

SUMMARY

Topology design

Optimizing towards the Moore Bound reduces expensive network resources

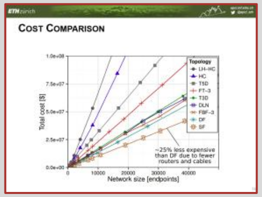


Credits

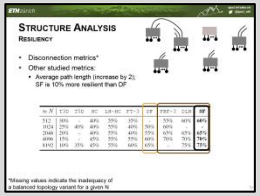
Maciej Besta
(PhD Student @SPCL)

Advantages of SlimFly

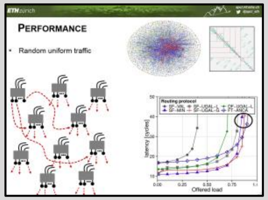
Cost & power



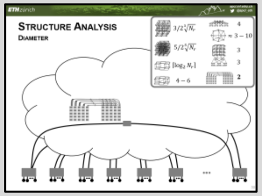
Resilience



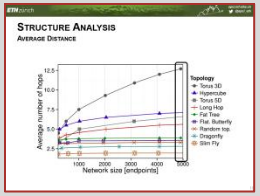
Performance



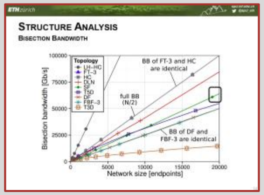
Diameter



Avg. distance

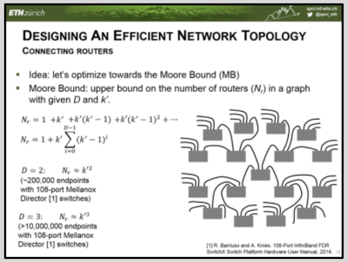


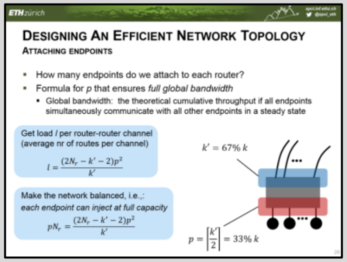
Bandwidth



Optimization approach

Combining mathematical optimization and current technology trends effectively tackles challenges in networking

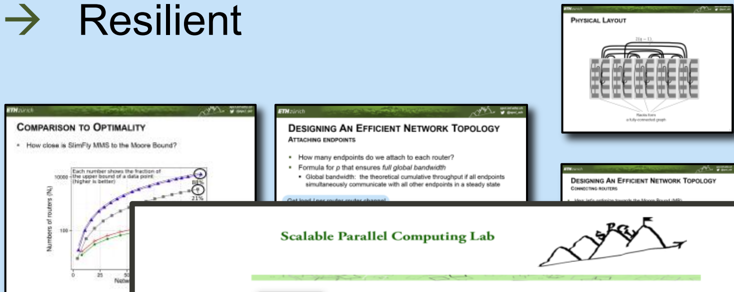


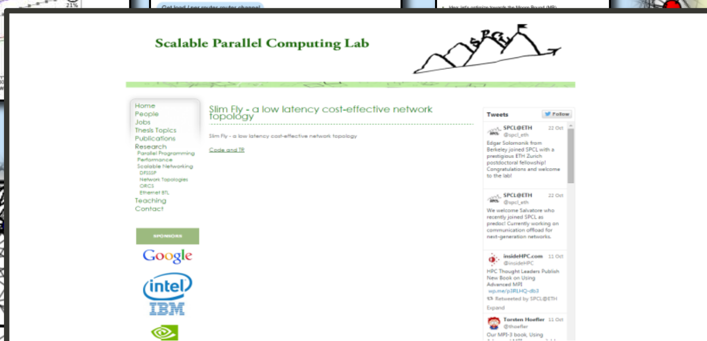


A lowest-diameter topology

→ Viable set of configurations

→ Resilient



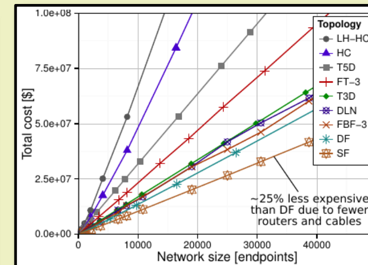


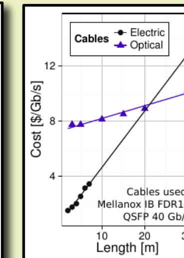
http://spcl.inf.ethz.ch/Research/Scalable_Networking/SlimFly

A cost & power effective topology

→ 25% less expensive than Dragonfly,

→ 26% less power-hungry than Dragonfly





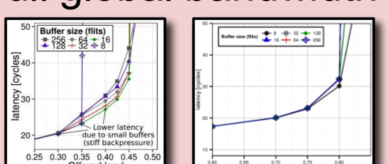
AWAY M

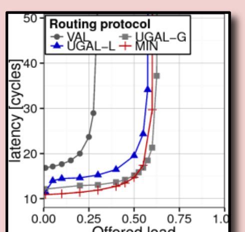
Thank you for your attention

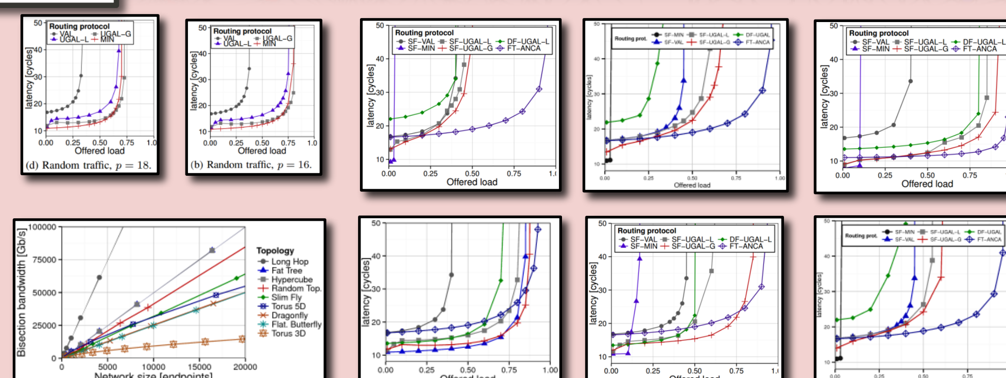
A high-performance topology

→ Lowest latency

→ Full global bandwidth



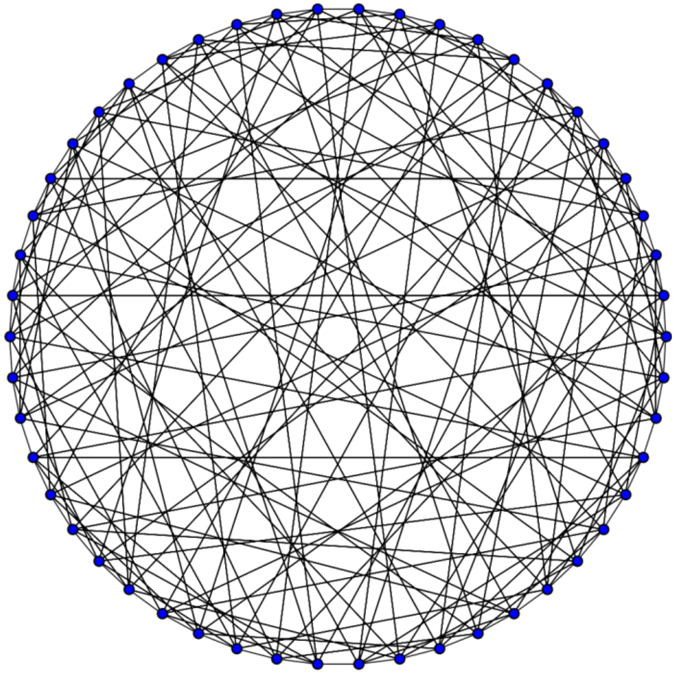




Designing an Efficient Network Topology

Connecting routers: Diameter 2

- Viable set of configurations
  - 10 SF networks with *the number of endpoints* < 11,000 (compared to 6 balanced Dragonflies [1])
- Let's pick *network radix* = 7...
  - ... We get the Hoffman-Singleton graph (attains the Moore Bound)



Designing an Efficient Network Topology

Attaching endpoints: Diameter 2

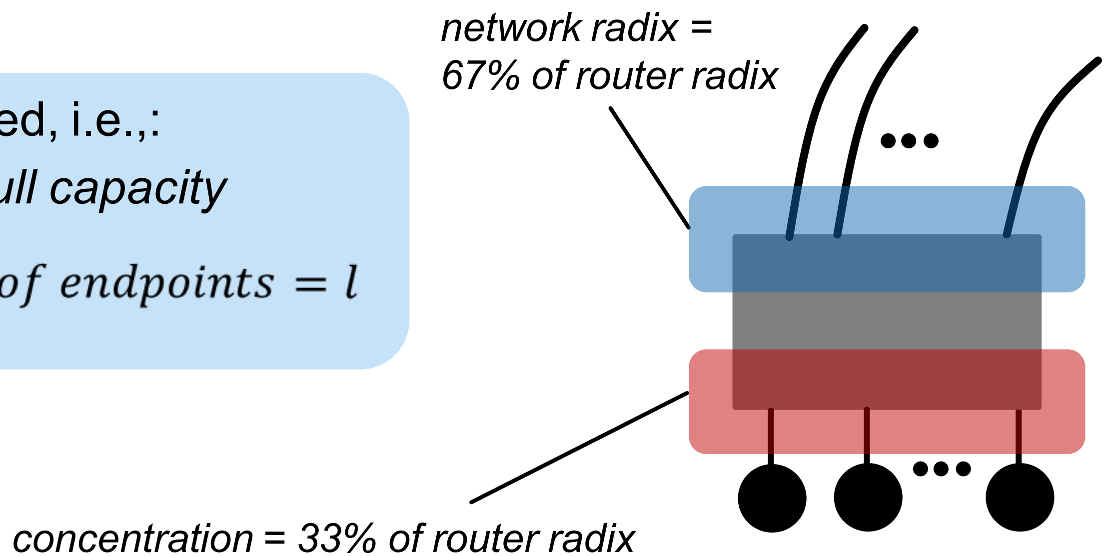
1 Get load *l* per router-router channel (average number of routes per channel)

$$l = \frac{total\ number\ of\ routes}{total\ number\ of\ channels}$$

2 Make the network balanced, i.e.,:

each endpoint can inject at full capacity

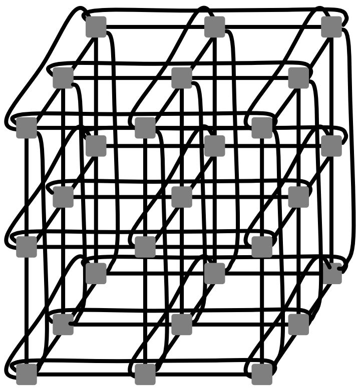
$$local\ uplink\ load = number\ of\ endpoints = l$$



Comparison Targets

Low-radix topologies

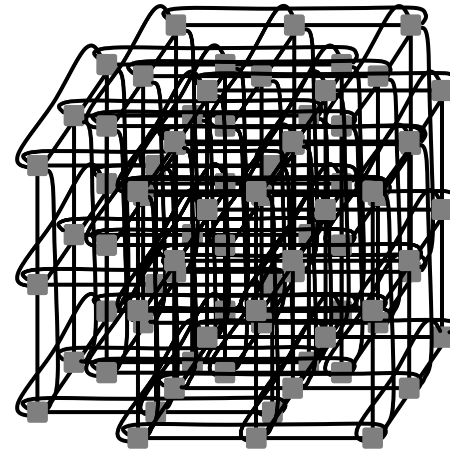
Torus 3D



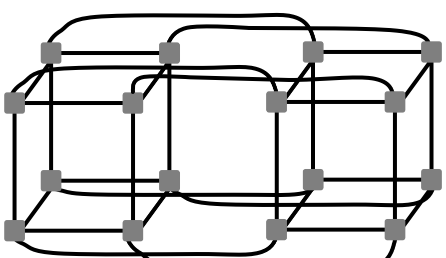
Cray XE6

IBM BG/Q

Torus 5D



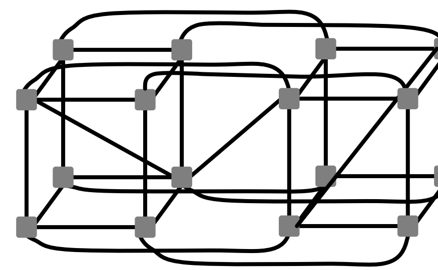
Hypercube



NASA Pleiades

Infinetics

Long Hop [1]

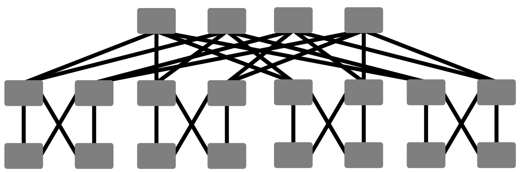


[1] Tomic, Ratko V. Optimal networks from error correcting codes. 2013 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)

Comparison Targets

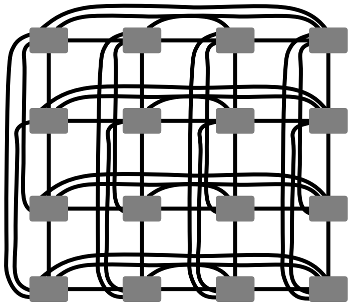
High-radix topologies

Fat tree [1]

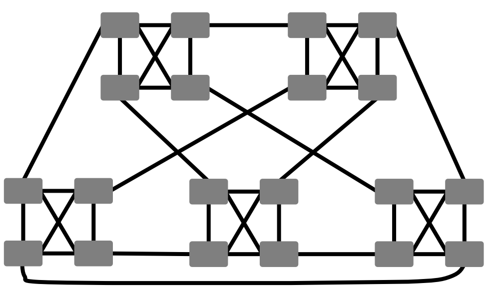


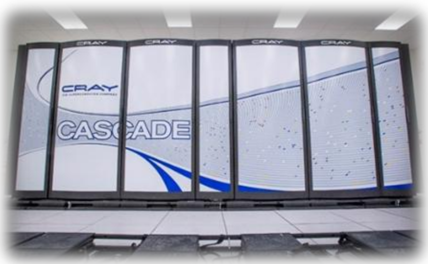
TSUBAME2.0

Flattened Butterfly [2]



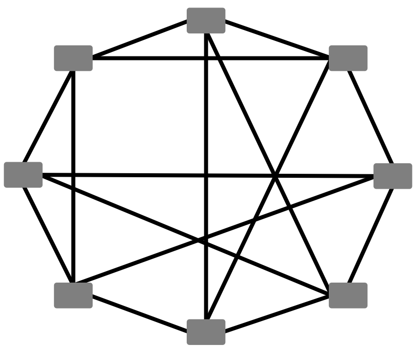
Dragonfly [3]





Cray Cascade

Random Topologies [4,5]



[1] C. E. Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. IEEE Transactions on Computers. 1985
[2] J. Kim, W. J. Dally, D. Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. ISCA'07
[3] J. Kim, W. J. Dally, S. Scott, D. Abts. Technology-Driven, Highly-Scalable Dragonfly Topology. ISCA'08
[4] A. Singla, C. Hong, L. Popa, P. B. Godfrey. Jellyfish: Networking Data Centers Randomly. NSDI'12
[5] M. Koibuchi, H. Matsutani, H. Amano, D. F. Hsu, H. Casanova. A case for random shortcut topologies for HPC interconnects. ISCA'12

PERFORMANCE & ROUTING

MINIMUM ROUTING

1 *Intra-group connections*

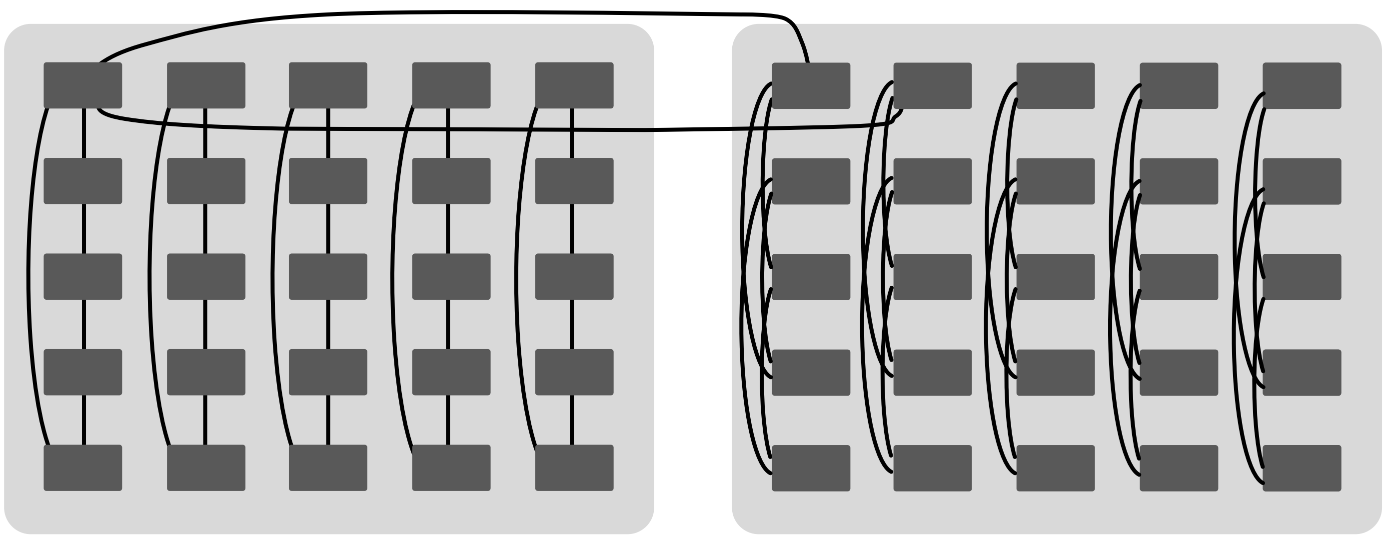
∃ Path of length 1 or 2 between two routers

2 *Inter-group connections (different types of groups)*

∃ Path of length 1 or 2 between two routers

3 *Inter-group connections (identical types of groups)*

∃ Path of length 2 between two routers



Designing An Efficient Network Topology

General construction scheme

- We split the problem into two pieces

Connect routers:

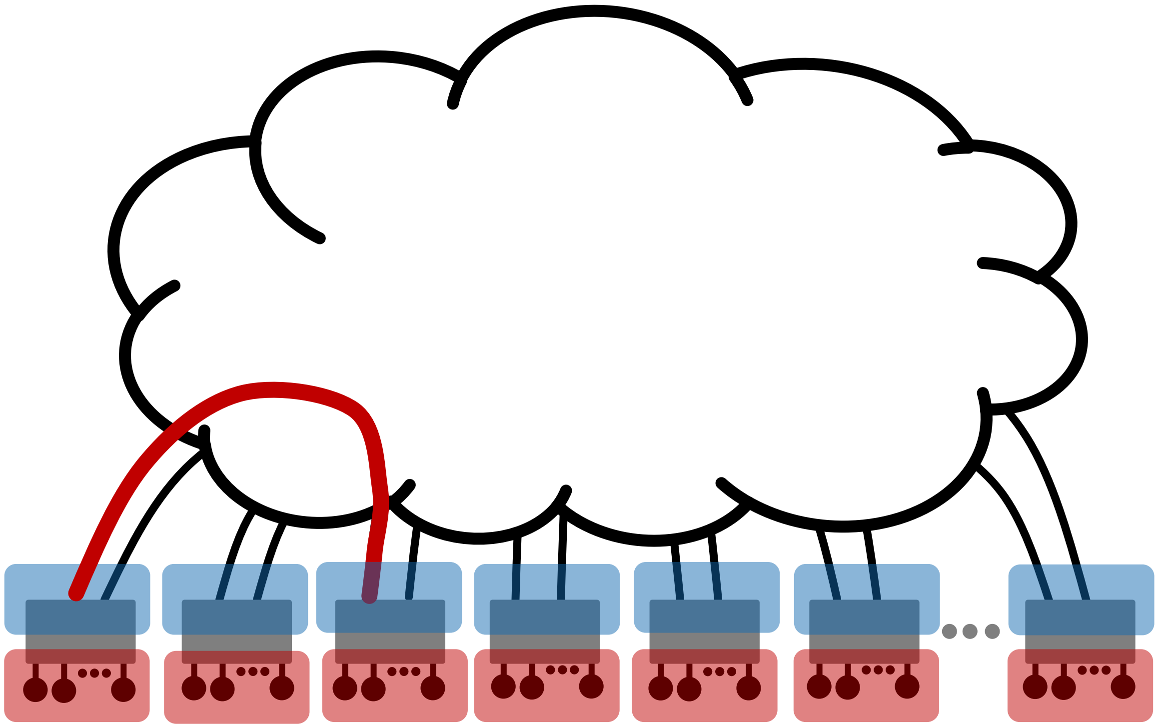
select *diameter*

select *network radix*

maximize *number of routers*

Attach endpoints

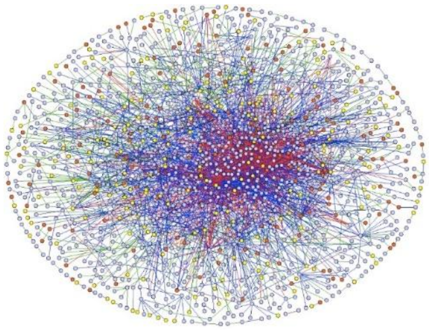
Derive *concentration* that provides full global bandwidth



Optimizing Network Topologies

GRAPH 500

- **Optimize for (bad-case) random uniform traffic**
  - Can often be generated by randomization of allocations
  - Important for permutations, transpose, graph computations …



- **Discrete optimization problem**
  - min(# of routers)
  - constraints:
    Router radix k
    Full "global" bandwidth ("guarantee cable capacity")
  - Implemented as SAT problem: $\binom{N-1}{k}$ options for neighbors alone!
    Maximum size solved was N=8 ☹
- **Intuition: lower average distance → lower resource needs**
  - A new view as primary optimization target!

Cost & Power Comparison

Detailed Case-Study: High-Radix Topologies

Topology	Fat tree	Random	Flat. Butterfly	Dragonfly	**Slim Fly**
Endpoints (N)	**10,718**	**9,702**	**10,000**	**9,702**	**10,830**
Routers (N_r)	1,531	1,386	1,000	1,386	**722**
Radix (k)	35	28	33	27	**43**
Electric cables	7,350	6,837	4,500	9,009	**6,669**
Fiber cables	24,806	7,716	10,000	4,900	**6,869**
Cost per node [$]	2,315	1,566	1,535	1,342	**1,033**
Power per node [W]	14.0	11.2	10.8	10.8	**8.02**