Parallel Computing

Interconnection Networks

Readings: Hager’s book (4.5)

Pacheco’s book (chapter 2.3.3)

http://pages.cs.wisc.edu/~tvrdik/5/html/Section5.html#AAAAATre
e-based topologies

Slides credit: Richard Vuduc, GaTech. Kayvon Fathalin, CMU

Interconnection Networks

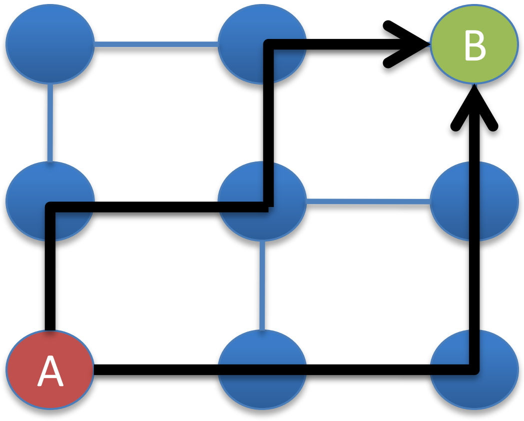
- Introduction and Terminology
- Topology
- Routing and Flow control

Interconnection Network Basics

- **Topology**
  - Specifies the way switches are wired
  - Affects routing, reliability, throughput, latency, building ease
- **Routing**
  - How does a message get from source to destination
  - Static or adaptive
- **Buffering and Flow Control**
  - What do we store within the network?
  - Entire packets, parts of packets, etc?
  - How do we manage and negotiate buffer space?
  - How do we throttle during oversubscription?
  - Tightly coupled with routing strategy

Topology & Routing

- **Topology:** Determines arrangement of nodes and links in network
- Significant impact on network performance
  - Determines number of hops
- **Routing:** Routing algorithm determines path(s) from source to destination

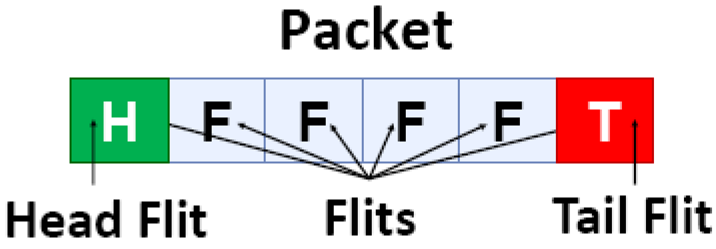


Terminology

- **Network interface**
  - Connects endpoints (e.g. cores) to network.
  - Decouples computation/communication
- **Links**
  - Bundle of wires that carries a signal
- **Switch/router**
  - Connects fixed number of input channels to fixed number of output channels
- **Channel**
  - A single logical connection between routers/switches

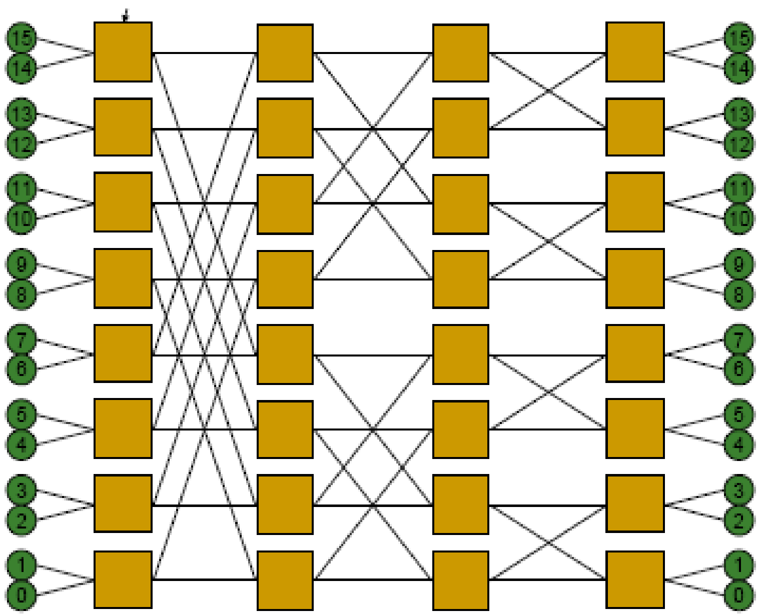
More Terminology

- **Node**
  - A network endpoint connected to a router/switch
- **Message**
  - Unit of transfer for network clients (e.g. cores, memory)
- **Packet**
  - Unit of transfer for network
- **Flit**
  - Flow control digit
  - Unit of flow control within network

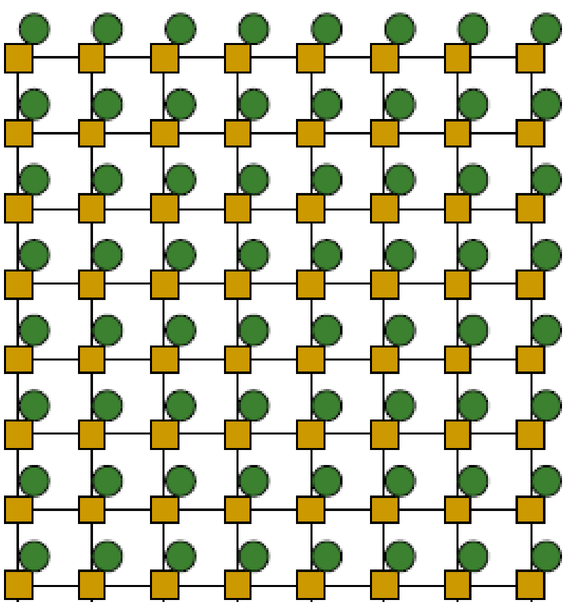


Some More Terminology

- **Direct or Indirect Networks**
  - Endpoints sit "inside" (direct) or "outside" (indirect) the network
  - E.g. mesh is direct; every node is both endpoint and switch
    - Router (switch), Radix of 2 (2 inputs, 2 outputs) Abbreviation: Radix-ary These routers are 2-ary



Indirect



Direct

Properties of a Topology/Network

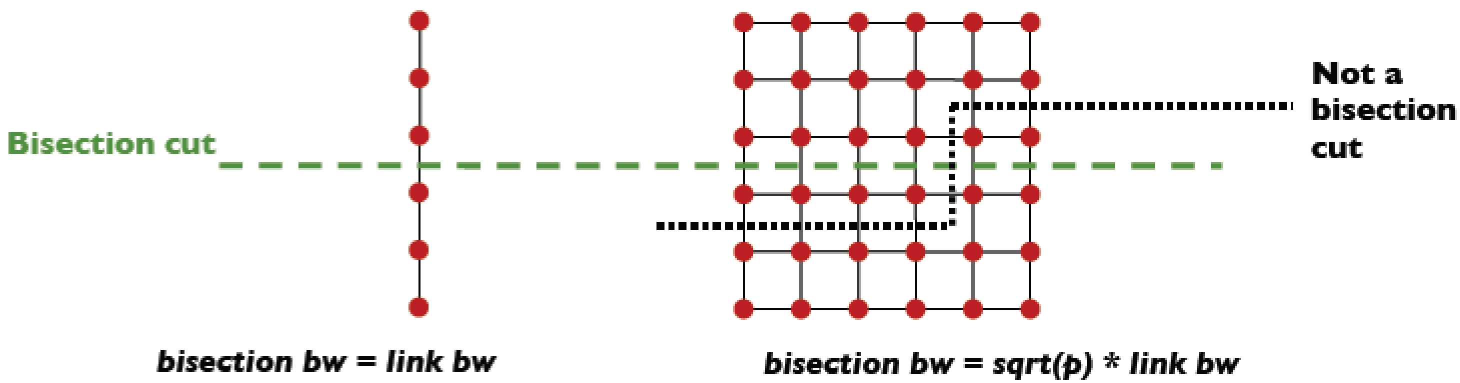
- **Regular or Irregular**
  - regular if topology is regular graph (e.g. ring, mesh)
- **Routing Distance**
  - number of links/hops along route
- **Diameter**
  - maximum routing distance
- **Average Distance**
  - average number of hops across all valid routes

Properties of a Topology/Network

- **Blocking vs. Non-Blocking**
  - If connecting any permutation of sources & destinations is possible, network is non-blocking; otherwise network is blocking.
- **Bisection Bandwidth**
  - Often used to describe network performance
  - Cut network in half and sum bandwidth of links severed
  - (Min # channels spanning two halves) * (BW of each channel)
  - Meaningful only for recursive topologies
  - Can be misleading, because does not account for switch and routing efficiency

Bisection Bandwidth

- Definition: # links across smallest cut that divides nodes in two (nearly) equal parts
- Important for **all-to-all communication**
- Variation: Bisection *bandwidth = bandwidth across smallest cut*



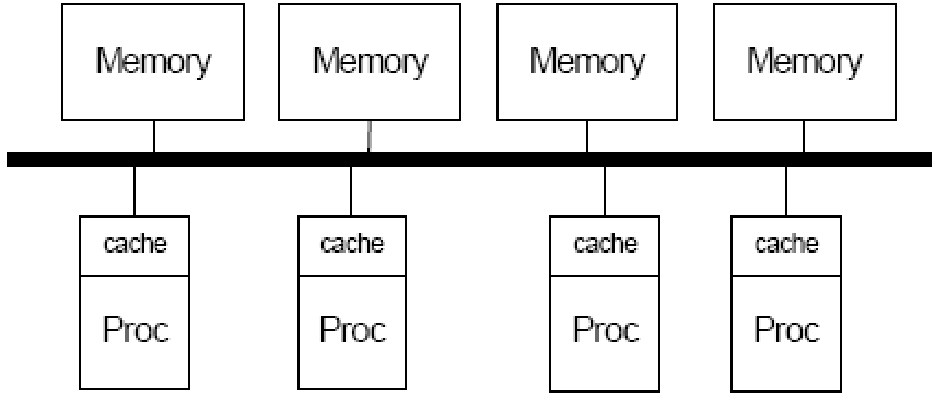
http://www.unixer.de/research/orcs/

Many Topology Examples

- **Bus**
- **Crossbar**
- **Ring**
- **Tree**
- **Omega**
- **Hypercube**
- **Mesh**
- **Torus**
- **Butterfly**
- …

Bus

+ Simple
+ Cost effective for a small number of nodes
+ Easy to implement coherence (snooping)
- Not scalable to large number of nodes (limited bandwidth, electrical loading -> reduced frequency)
- High contention



Crossbar

- Every node connected to all others (non-blocking)
- Good for small number of nodes

  + Low latency and high throughput

  - Expensive

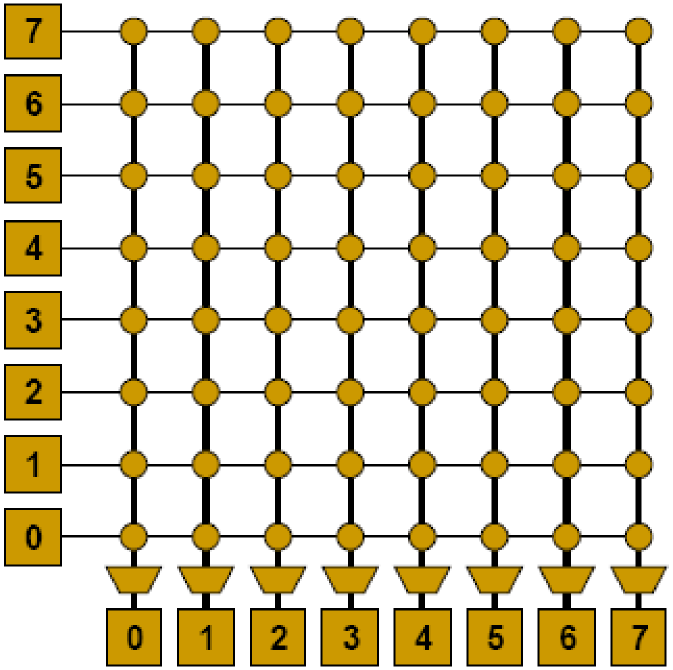
  - Not scalable -> $O(N^2)$ cost

  - Difficult to arbitrate

  Core-to-cache-bank networks:

  - IBM POWER5

  - Sun Niagara I/II

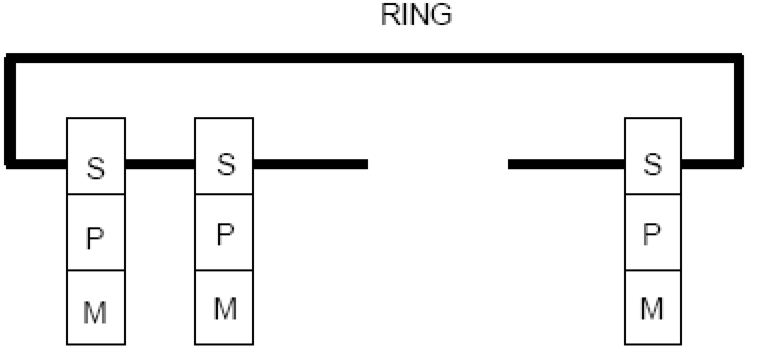


Ring

+ Cheap: O(N) cost
- High latency: O(N)
- Not easy to scale
- Bisection bandwidth remains constant

Used in:

- Intel Larrabee/Core i7
- IBM Cell



Linear and ring networks

- **Diameter : Length of shortest path between farthest pair**
- **Bisection bandwidth : bandwidth across smallest cut that bisects network**
- **Average distance**

Linear: P-1 links

Diameter = ?

Avg. dist. = ?

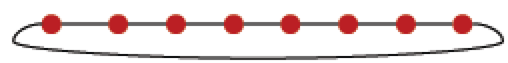
Bisection = ?

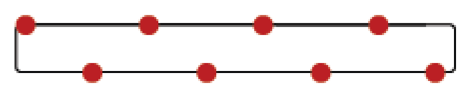
Ring/Torus: P links

Diameter = ?

Avg. dist. = ?

Bisection = ?





Linear and ring networks

Linear: P-1 links

Diameter = *P-1*

Avg. dist. ~ *P/3*

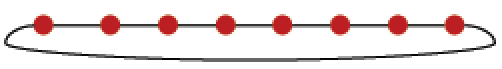
Bisection = 1

Ring/Torus: P links

Diameter ~ *P/2*

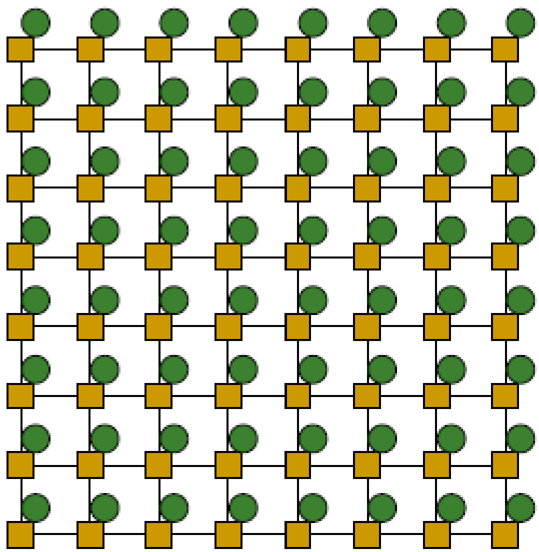
Avg. dist. ~ *P/4*

Bisection = 2



Mesh

- O(N) cost
- Average latency: O(sqrt(N))
- Easy to layout on-chip: regular & equal-length links
- Path diversity: many ways to get from one node to another
- Used in:
  - Tilera 100-core CMP
  - On-chip network prototypes



Torus

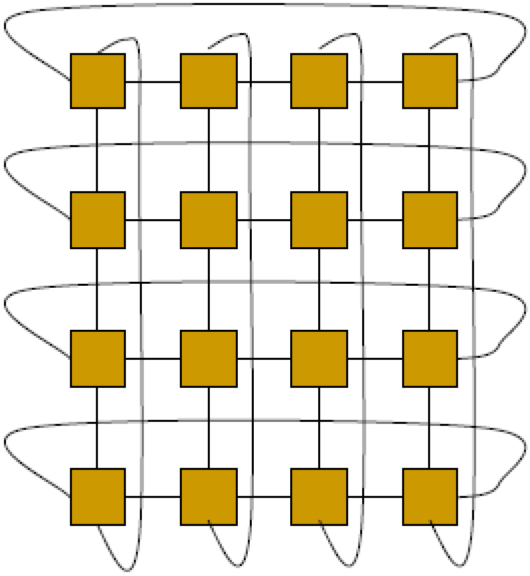
- **Mesh is not symmetric on edges: performance very sensitive to placement of task on edge vs. middle**
- **Torus avoids this problem**

  + Higher path diversity (& bisection bandwidth) than mesh

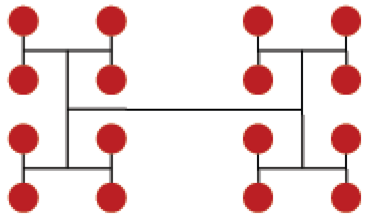
  - Higher cost

  - Harder to lay out on-chip

  - Unequal link lengths



Trees



Planar, hierarchical topology

Latency: O(logN)

Good for local traffic

+ Cheap: O(N) cost

+ Easy to Layout

- Root can become a bottleneck

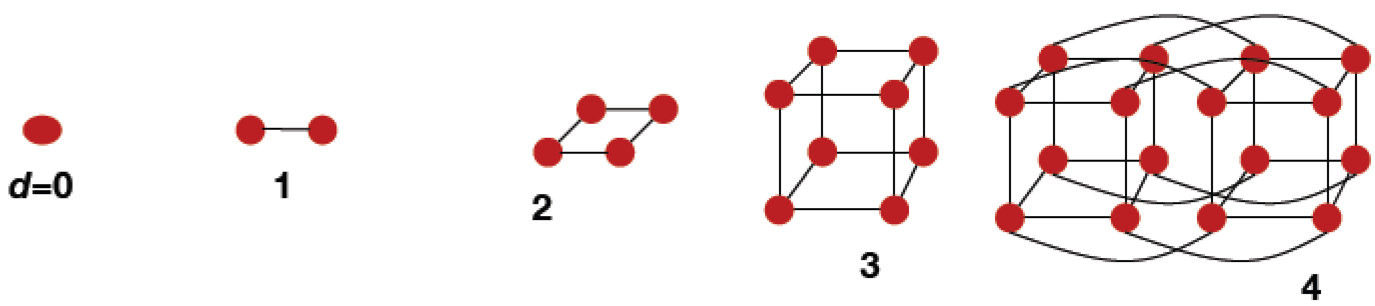
Fat trees avoid this problem (CM-5)

Hypercube

- **Latency: O(logN)**
- **Radix: O(logN)**
- **#links: O(NlogN)**
- **+ Low latency**
- **- Hard to lay out in 2D/3D**
- **Used in some early message passing machines, e.g.:**
  - **- Intel iPSC**
  - **- nCube**

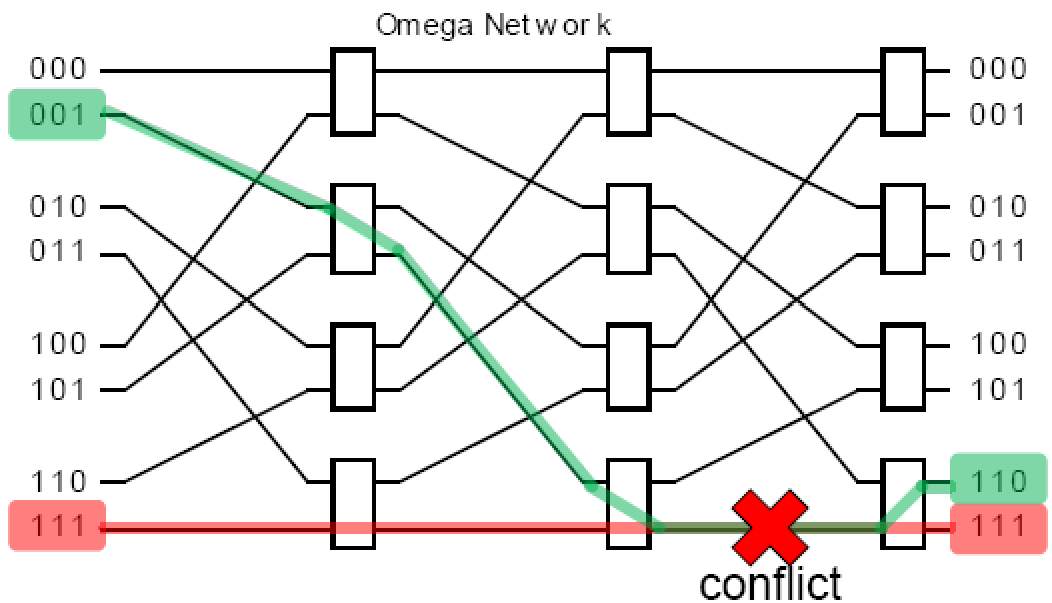
No. of nodes = 2*d for dimension d*

- Diameter = *d*
- Bisection = *p/2*



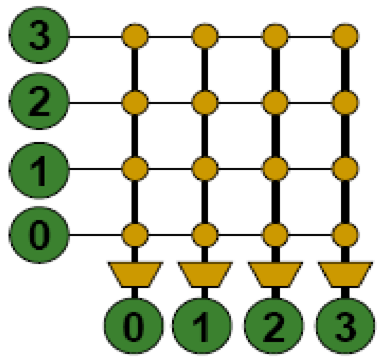
Multistage Logarithmic Networks

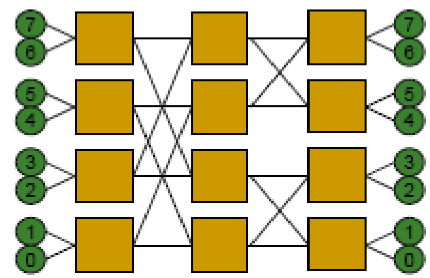
- Idea: Indirect networks with multiple layers of switches between terminals
- Cost: O(NlogN), Latency: O(logN)
- Many variations (Omega, Butterfly, Benes, Banyan, …)
- E.g. Omega Network:

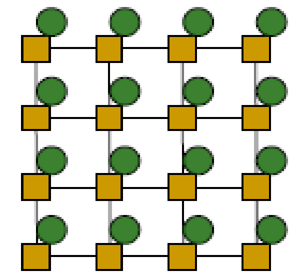


Q: Blocking or non-blocking?

Review: Topologies



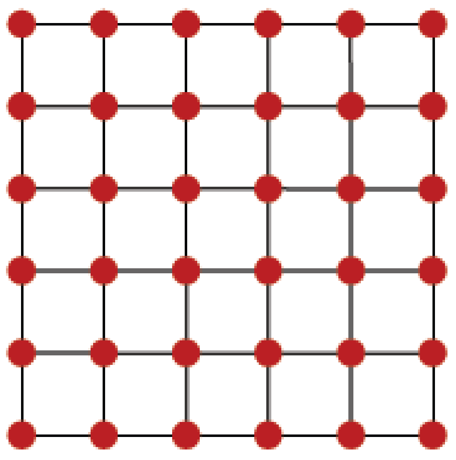




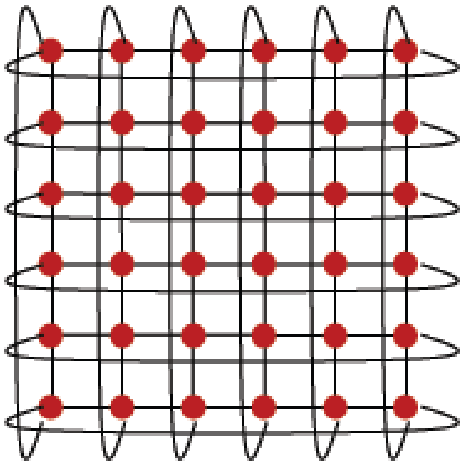
Topology	**Crossbar**	**Multistage Logarith.**	**Mesh**
Direct/Indirect	**Indirect**	**Indirect**	**Direct**
Blocking/ Non-blocking	**Non-blocking**	**Blocking**	**Blocking**
Cost	**$O(N^2)$**	**O(NlogN)**	**O(N)**
Latency	**O(1)**	**O(logN)**	**O(sqrt(N))**

Multidimensional meshes and tori

2-D mesh: ~ 2*P links
Diameter = ?
Bisection = ?

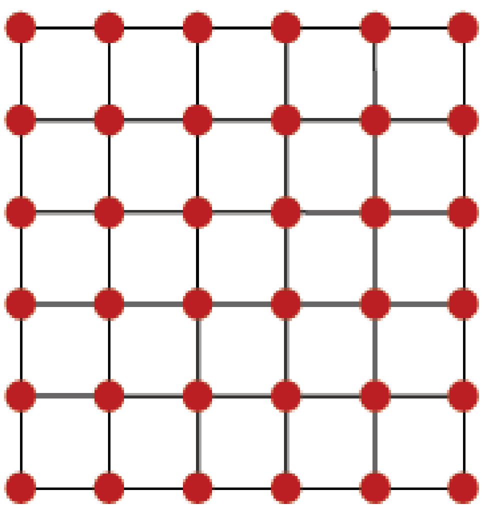


2-D torus: 2*P links
Diameter = ?
Bisection = ?

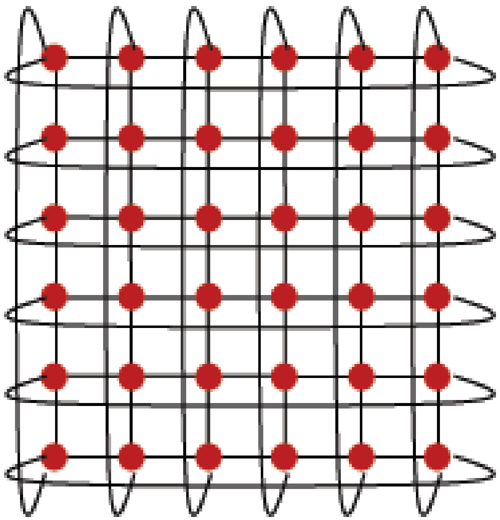


Multidimensional meshes and tori

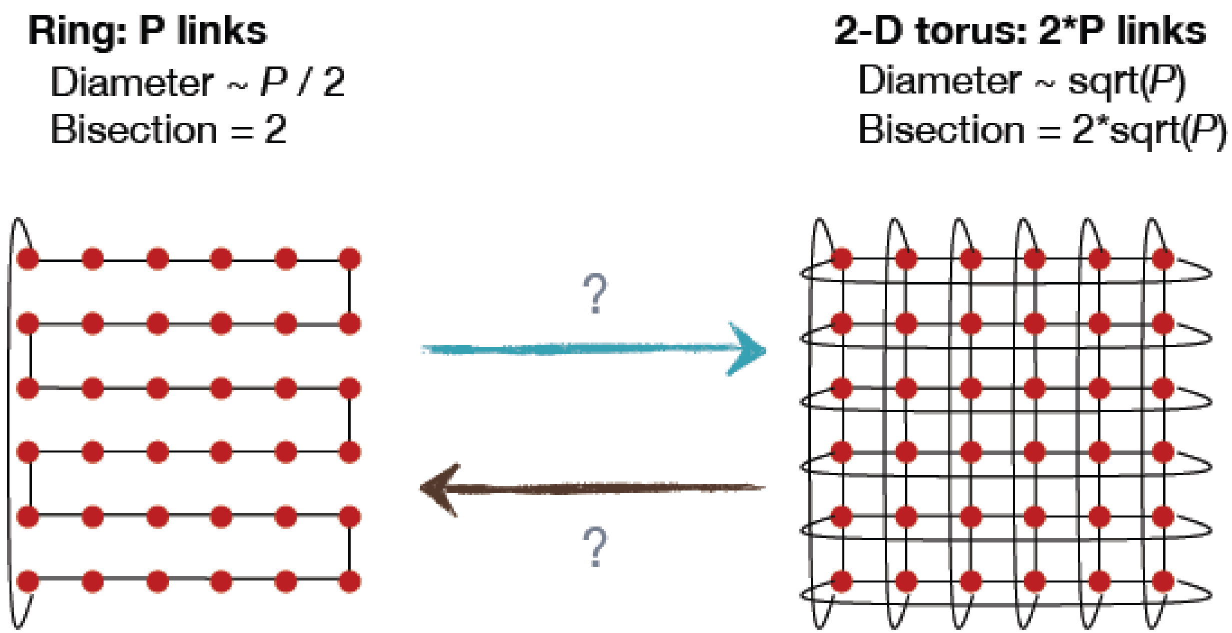
2-D mesh: ~ 2*P links
Diameter ~ 2*sqrt(*P*)
Bisection = sqrt(*P*)



2-D torus: 2*P links
Diameter ~ sqrt(*P*)
Bisection = 2*sqrt(*P*)

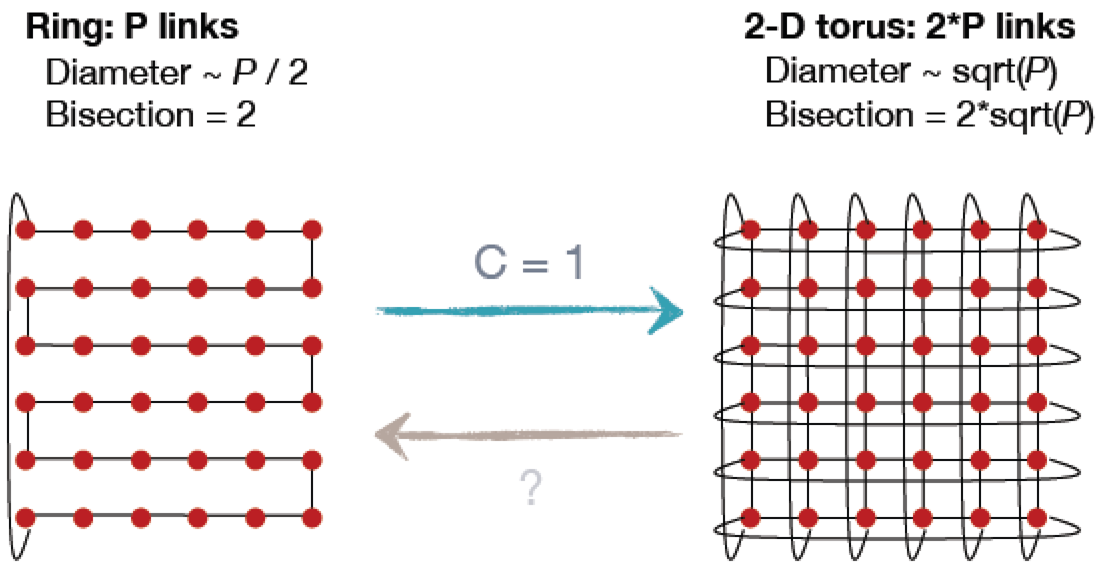


Mappings and congestion



Node mapping implies an edge mapping.
Congestion = maximum number of source edges that map to a target edge.

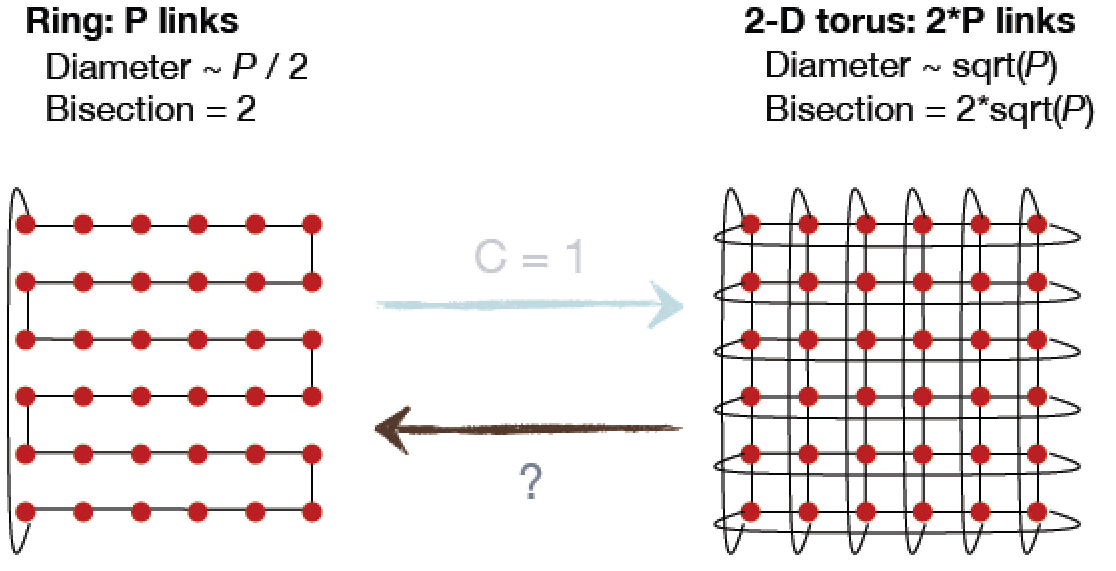
Mappings and congestion



Node mapping implies an edge mapping.

Congestion = maximum number of source edges that map to a target edge.

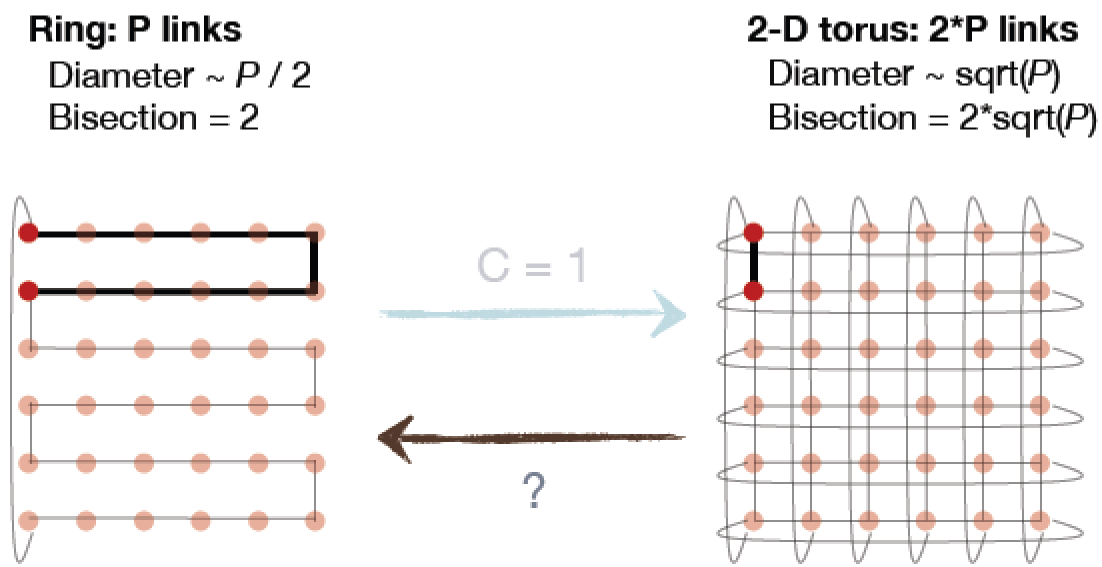
Mappings and congestion



Node mapping implies an edge mapping.

Congestion = maximum number of source edges that map to a target edge.

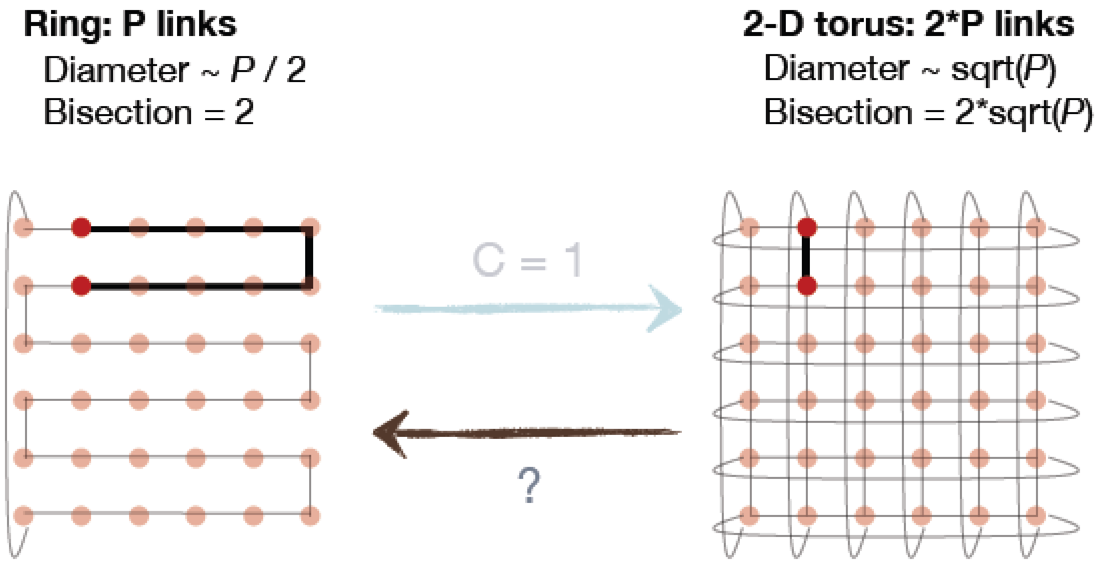
Mappings and congestion



Node mapping implies an edge mapping.

Congestion = maximum number of source edges that map to a target edge.

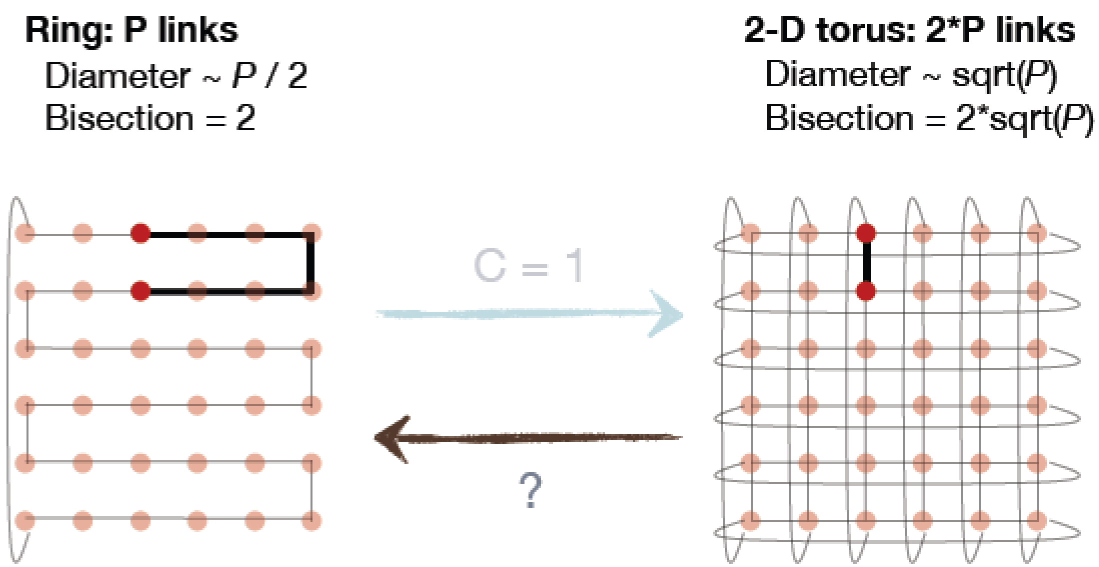
Mappings and congestion



Node mapping implies an edge mapping.

Congestion = maximum number of source edges that map to a target edge.

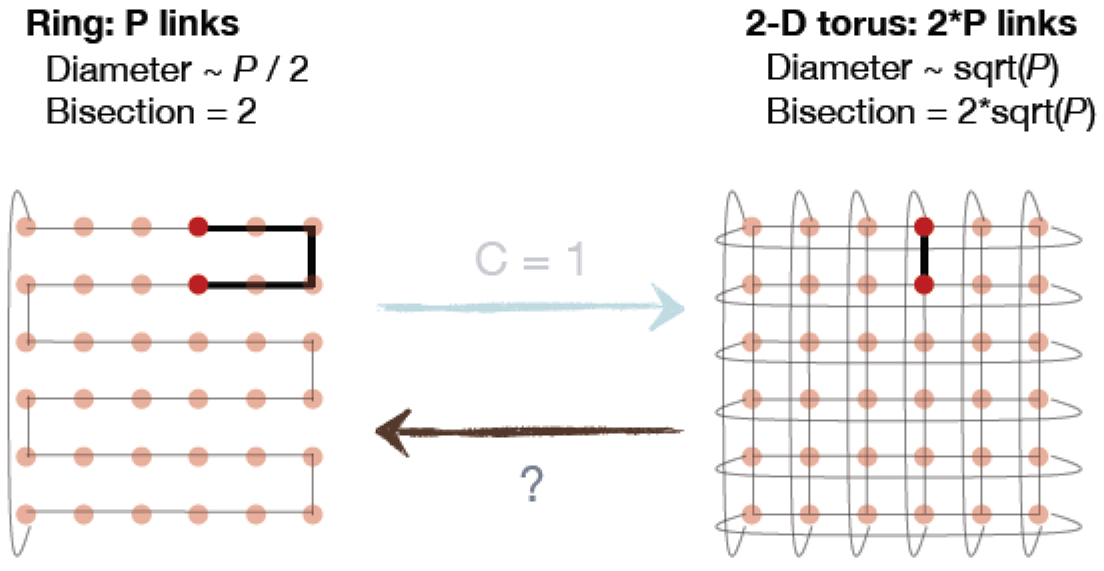
Mappings and congestion



Node mapping implies an edge mapping.

Congestion = maximum number of source edges that map to a target edge.

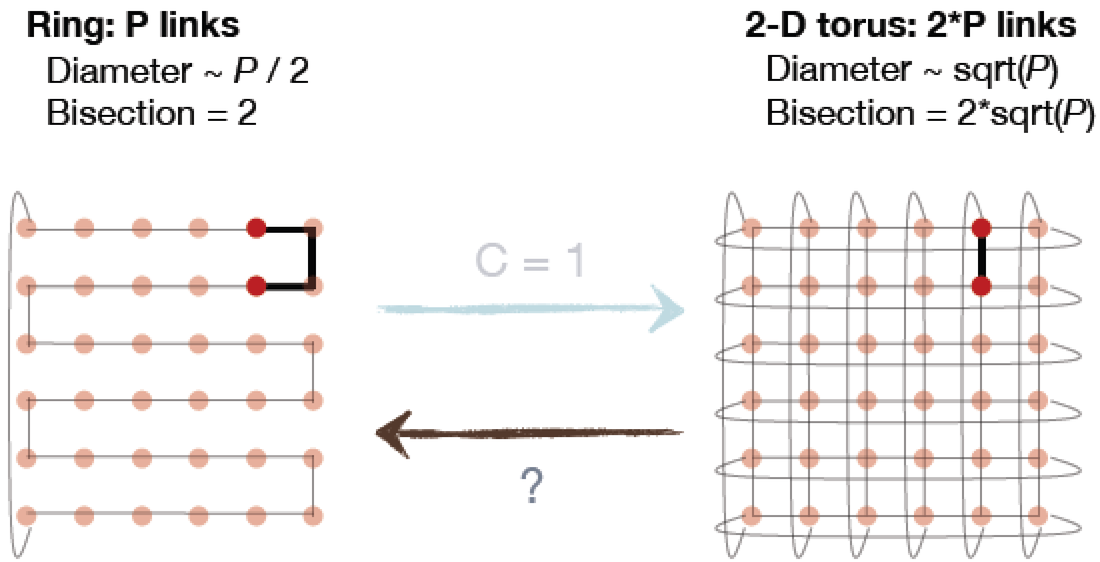
Mappings and congestion



Node mapping implies an edge mapping.

Congestion = maximum number of source edges that map to a target edge.

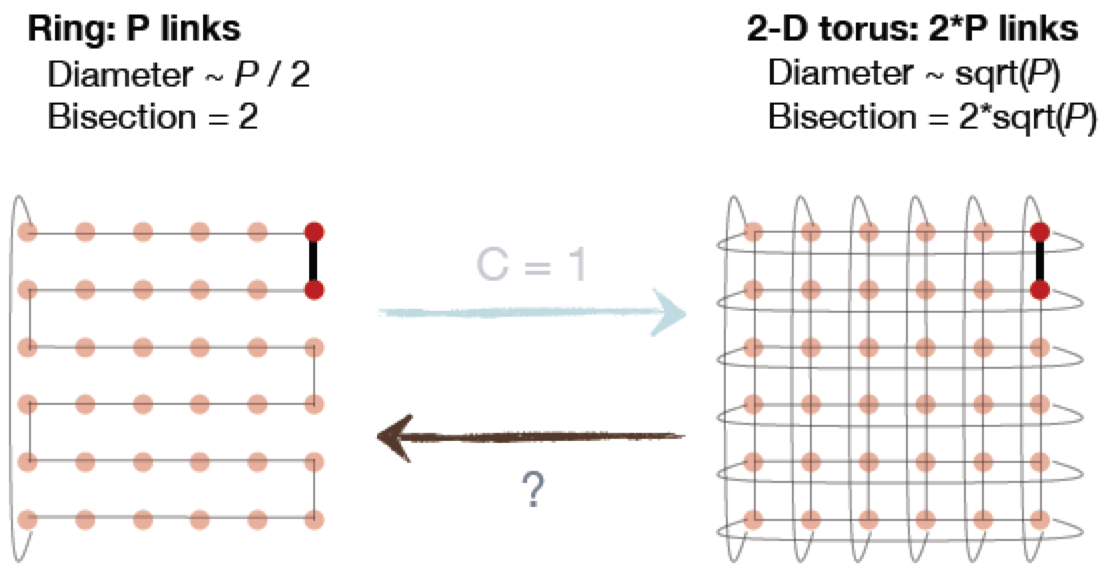
Mappings and congestion



Node mapping implies an edge mapping.

Congestion = maximum number of source edges that map to a target edge.

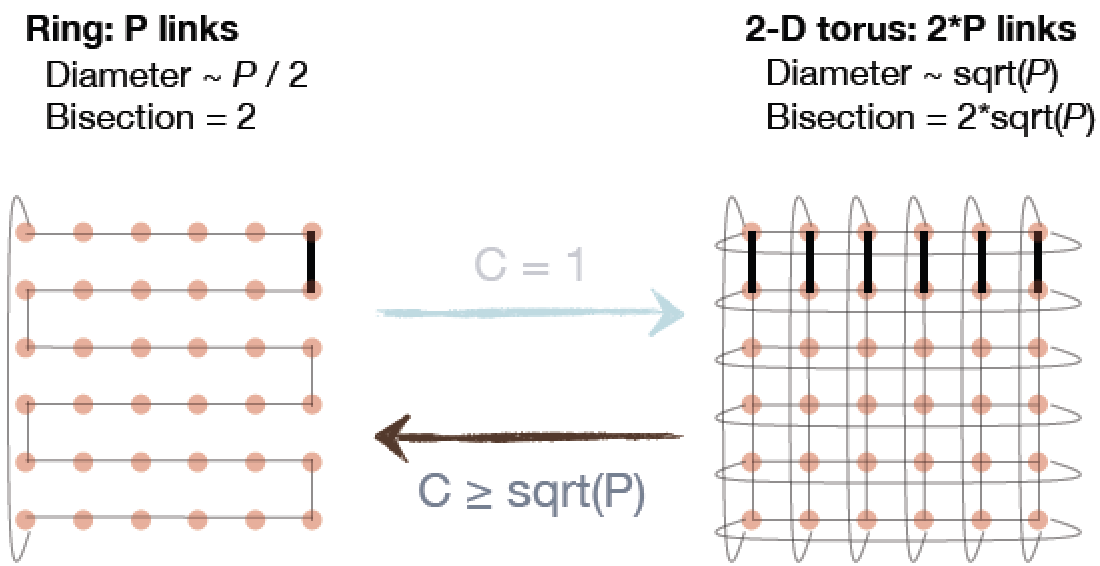
Mappings and congestion



Node mapping implies an edge mapping.

Congestion = maximum number of source edges that map to a target edge.

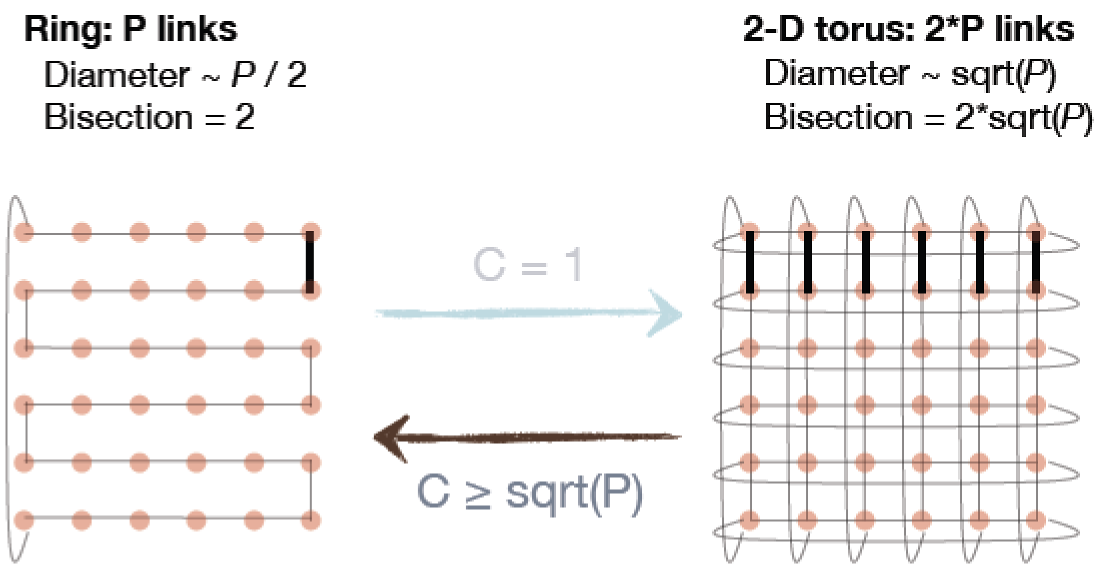
Mappings and congestion



Node mapping implies an edge mapping.

Congestion = maximum number of source edges that map to a target edge.

Mappings and congestion



General principle: Ratio of bisection widths is a **lower bound on** congestion

Node mapping implies an edge mapping.
Congestion = maximum number of source edges that map to a target edge.

Topology properties (*n nodes total)*

Topology	Diameter	Bisection	Arc connectivity	# links
Linear	$n-1$	1	1	$n-1$
Ring	$\approx n/2$	2	2	n
2-D mesh	$\approx 2\sqrt{n}$	$\sqrt{n}$	2	$n-1$
2-D torus	$\approx \sqrt{n}$	$2\sqrt{n}$	4	$2n$
Hypercube	$\log n$	$n/2$	$\log n$	$1/2 \cdot n \log n$
k-ary tree	$2 \log_k n$	1	1	$n-1$
Butterfly	$\log n$	n	n	$\approx n \log n$
d-D torus	$\approx \sqrt[d]{n} \cdot d/2$	$2n^{(d-1)/d)}$	1	$n-1$
Completely connected	1	$n^2/4$	$n-1$	$n(n-1)/2$

Source: Grama, et al. (2003), Intro. to Parallel Computing.

Topologies in practice

Machine	Network
ORNL Titan (Cray XK7)	3D torus
IBM Blue Gene/Q	5D torus
K computer	6D torus
Tianhe-1A (GPU)	Fat tree (?)
Tsubame (GPU)	Fat tree
Cray XE6	3D torus
Cray XT3, XT4, XT5	3D torus
BG/L, BG/P	3D torus (+ others)
SGI Altix	Fat tree
Cray X1	4D hypercube*
Millennium (UCB, Myricom)	Arbitrary*
HP Alphaserver (Quadrics)	Fat tree
IBM SP	~ Fat tree
SGI Origin	Hypercube
Intel Paragon	2D mesh
BBN Butterfly	Butterfly

“α-β” (latency-bandwidth) cost model

- Model time to send a message in terms of latency and bandwidth

$$t(n) \quad = \quad \alpha + \frac{n}{\beta}$$

α (latency) , β (bandwidth)

- Node may send to any other
- May send and receive Simultaneously

- Usually, cost(flop) << $1/\beta$ << α
  - One long message cheaper than many short ones
  - Can do ~ thousands of flops for each message
  - Want large computation-to communication ratio

Does network topology matter?

- Mapping algorithms to networks used to be a "hot topic"
  - Key metric: Minimize hops
  - Modern networks hide hop cost (e.g., wormhole routing) and software overheads dominate wire latencies, so topology seemed less important over time
- Gap in **hardware/software latency: On IBM SP, *cf. 1.5 usec to 36 usec***
- Topology affects **bisection bandwidth, so still relevant**

Wormhole flow control

Switching/Flow Control Overview

- Topology: determines connectivity of network
- Routing: determines paths through network
- Flow Control: determine allocation of resources to messages as they traverse network
  - Buffers and links
  - Significant impact on throughput and latency of network

Packets

- Messages: composed of one or more packets
  - If message size is <= maximum packet size only one packet created
- Packets: composed of one or more flits
- Flit: flow control digit
- Phit: physical digit
  - Subdivides flit into chunks = to link width
  - In on-chip networks, flit size == phit size.
    - Due to very wide on-chip channels

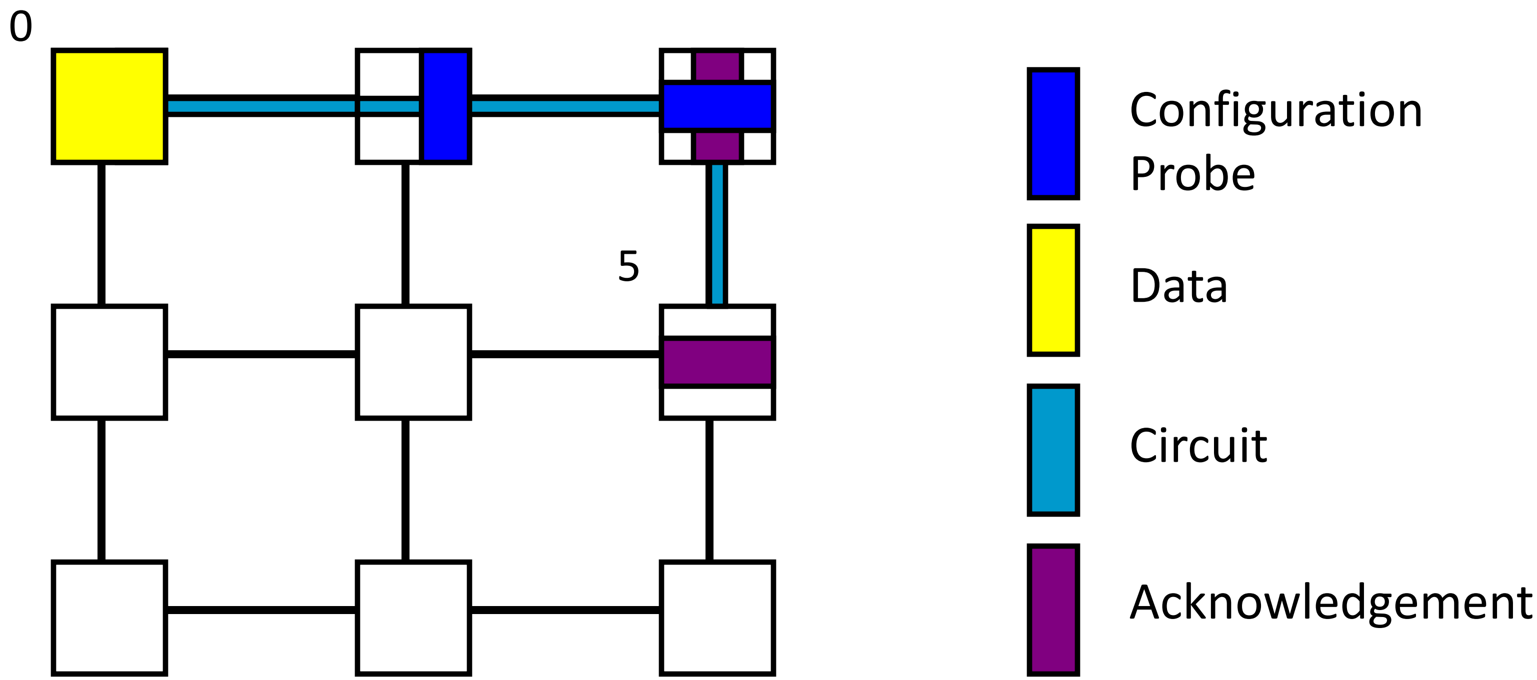
Switching

- Different flow control techniques based on granularity
- Circuit-switching: operates at the granularity of messages
- Packet-based: allocation made to whole packets
- Flit-based: allocation made on a flit-by-flit basis

Circuit Switching

- All resources (from source to destination) are allocated to the message prior to transport
  - Probe sent into network to reserve resources
- Once probe sets up circuit
  - Message does not need to perform any routing or allocation at each network hop
  - Good for transferring large amounts of data
    - Can amortize circuit setup cost by sending data with very low per-hop overheads
- No other message can use those resources until transfer is complete
  - Throughput can suffer due setup and hold time for circuits

Circuit Switching Example

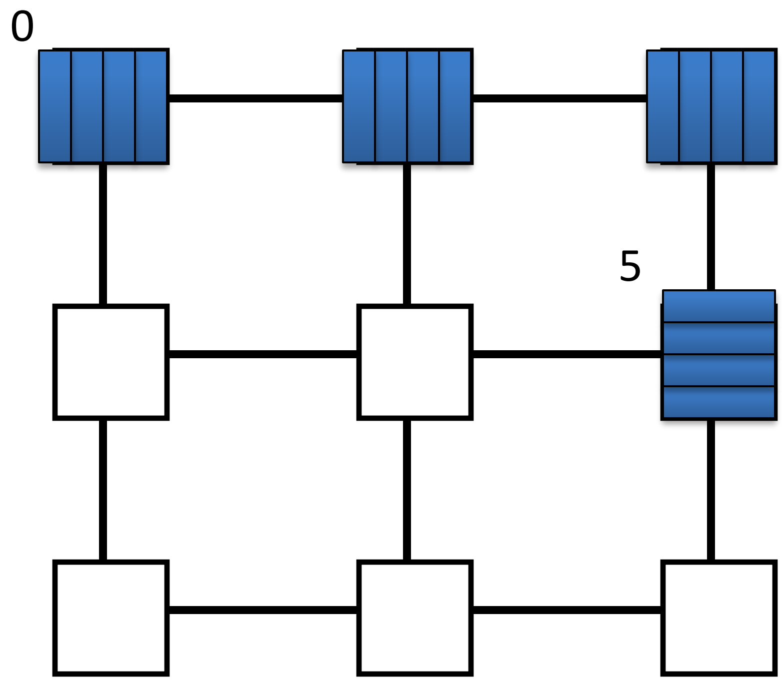


- Significant latency overhead prior to data transfer
- Other requests forced to wait for resources

Packet-based Flow Control

- Store and forward
- Links and buffers are allocated to entire packet
- Head flit waits at router until entire packet is buffered before being forwarded to the next hop
- Not suitable for on-chip
  - Requires buffering at each router to hold entire packet
  - Incurs high latencies (pays serialization latency at each hop)

Store and Forward Example

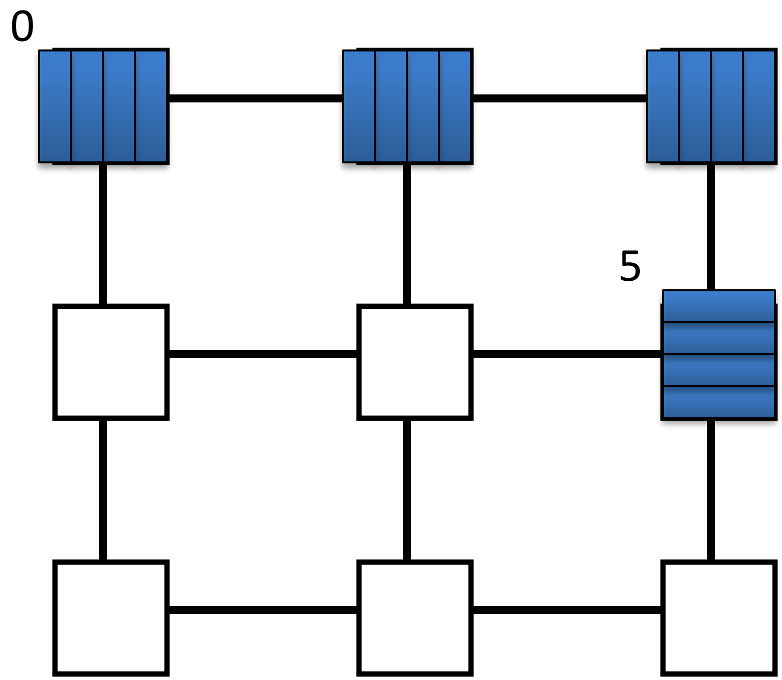


- High per-hop latency
- Larger buffering required

Virtual Cut Through

- Packet-based: similar to Store and Forward
- Links and Buffers allocated to entire packets
- Flits can proceed to next hop before tail flit has been received by current router
  - But only if next router has enough buffer space for entire packet
- Reduces the latency significantly compared to SAF
- But still requires large buffers
  - Unsuitable for on-chip

Virtual Cut Through Example

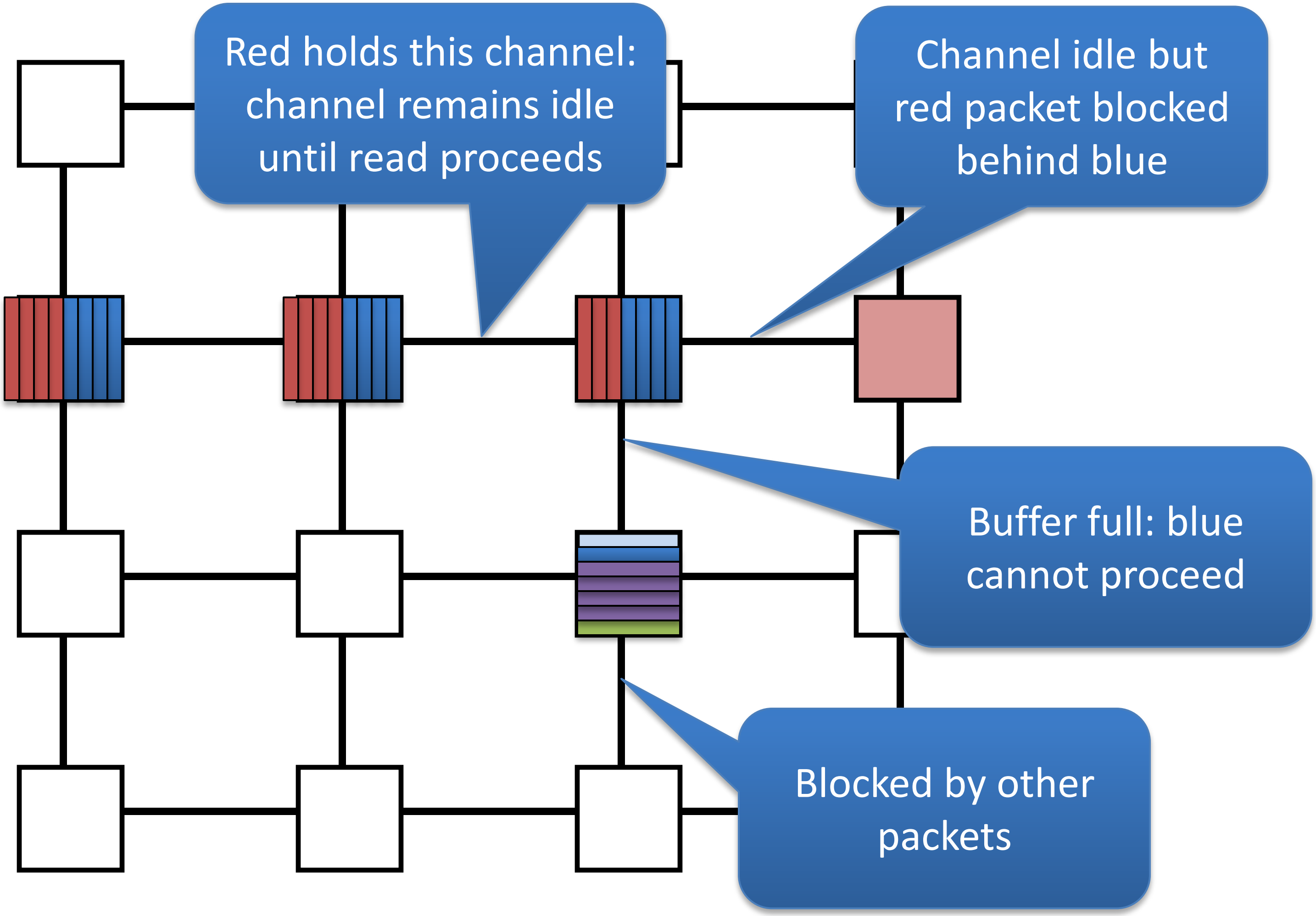


- Lower per-hop latency
- Larger buffering required

Flit Level Flow Control

- Wormhole flow control
- Flit can proceed to next router when there is buffer space available for that flit
  - Improved over SAF and VCT by allocating buffers on a flit-basis
- Pros
  - More efficient buffer utilization (good for on-chip)
  - Low latency
- Cons
  - Poor link utilization: if head flit becomes blocked, all links spanning length of packet are idle
    - Cannot be re-allocated to different packet
    - Suffers from head of line (HOL) blocking

Wormhole Example

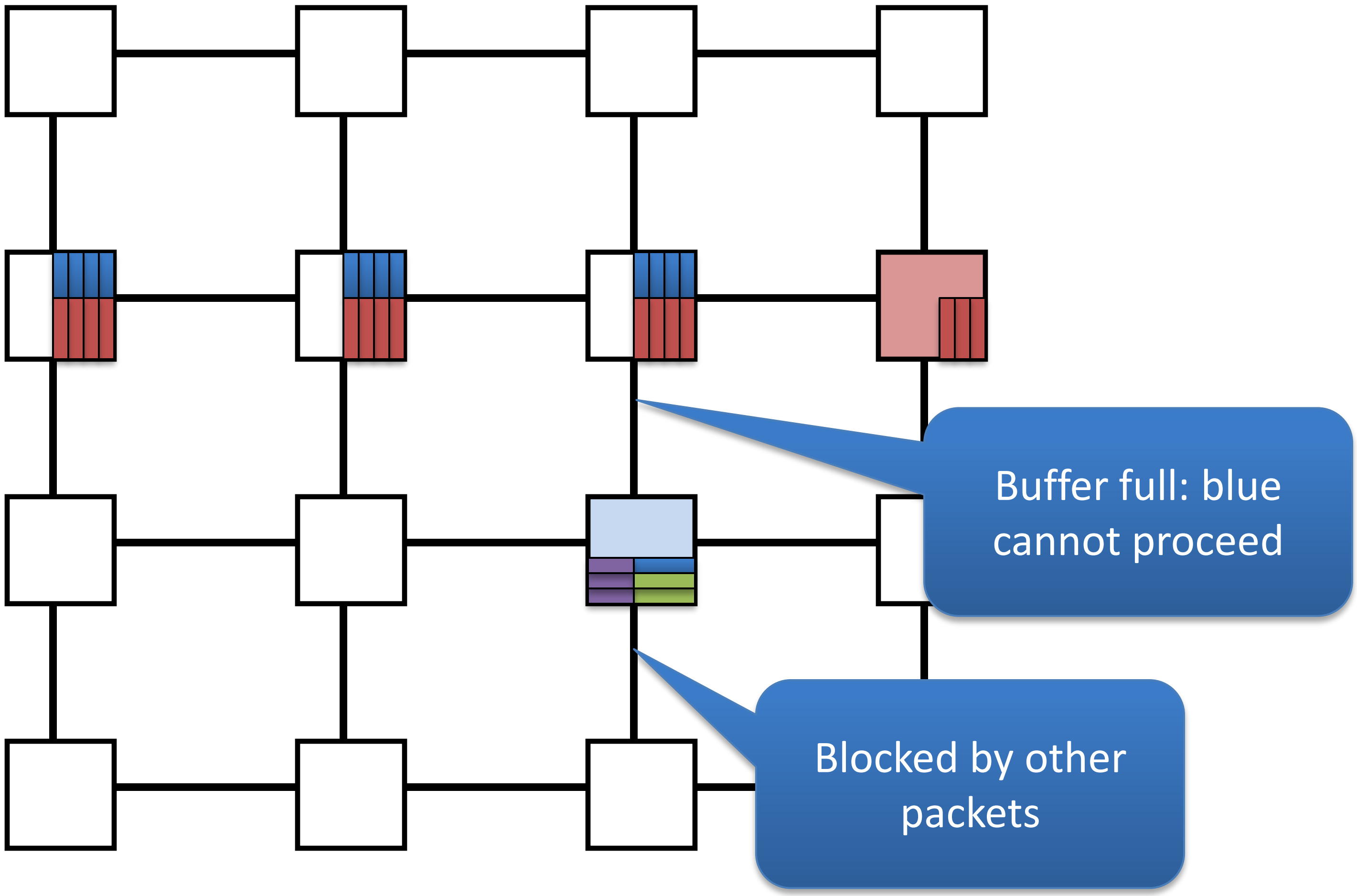


- 6 flit buffers/input port

Virtual Channel Flow Control

- Virtual channels used to combat HOL block in wormhole
- Virtual channels: multiple flit queues per input port
  - Share same physical link (channel)
- Link utilization improved
  - Flits on different VC can pass blocked packet

Virtual Channel Example



- 6 flit buffers/input port
- 3 flit buffers/VC

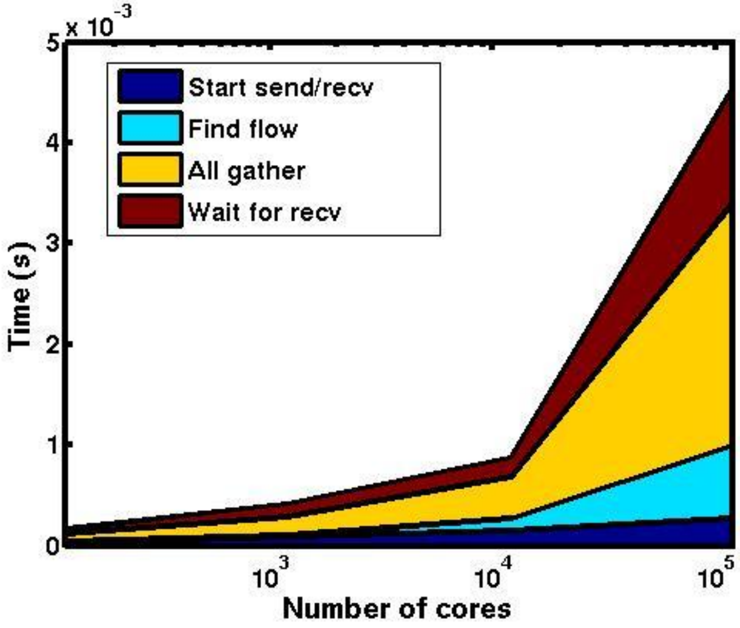
Why the need for Topology Mapping now ?

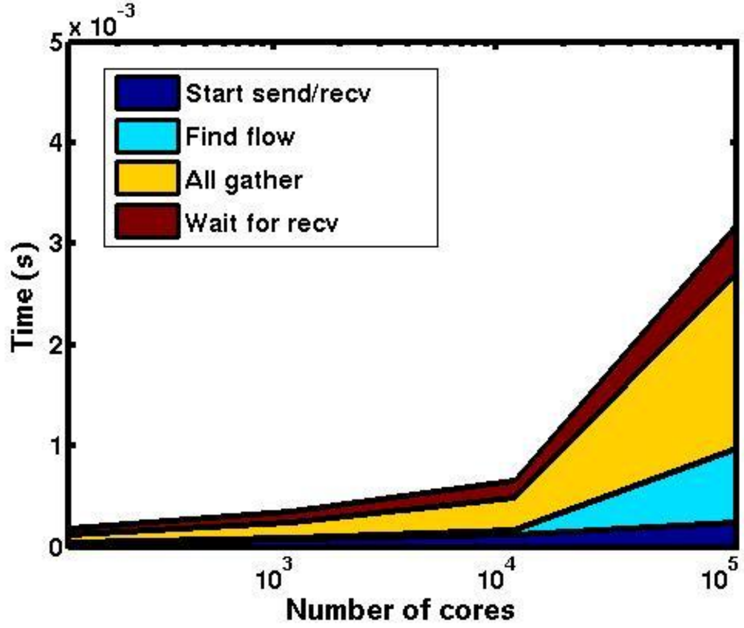
- Large-scale systems are built with low-dimensional network topologies
- E.g., 3D-Torus Jaguar (18k nodes), BG/P (64k nodes)
- Number of nodes grows (~100k-1M for Exascale)
- At this large scale, high chance of network congestion, hence advantages of hop count independence of wormhole routing are not applicable.
- Problem has been analysed for mapping Cartesian topologies [Yu’06, Bhatele’09, Krishna’11], arbitrary topologies [Hoefler’11]

The Mapping Problem

- **Definition:** Given a set of communicating parallel “entities”, map them on to physical processors to optimize communication
- **Goals:**
  - ✓ Minimize communication traffic and hence contention
  - ✓ Balance computational load (when n > p)
- **Case Study: Petascale Quantum Monte Carlo Application**

Task Assignment in Load Balancing



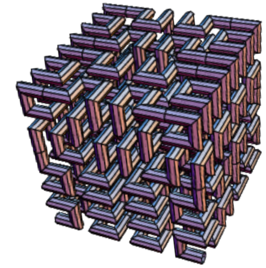


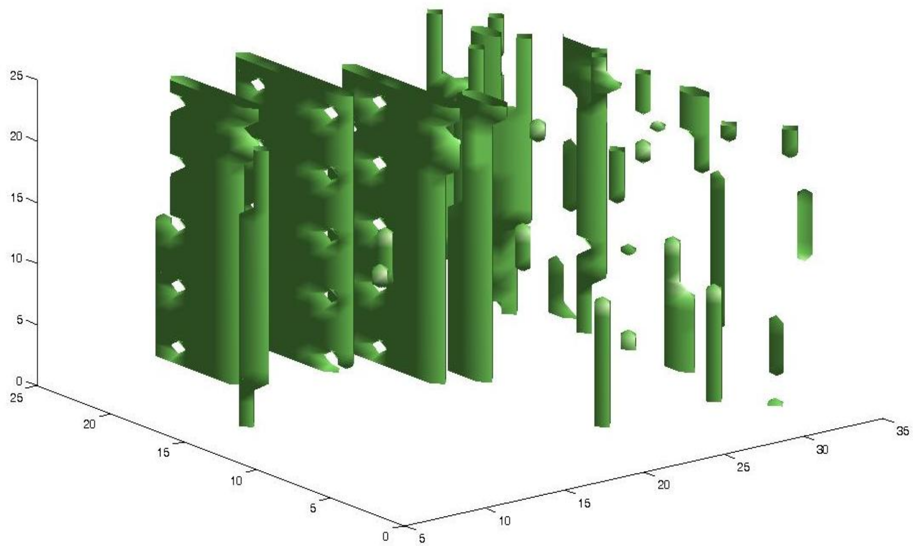
Time taken for different components of the new load balancing scheme with the default process ranks from MPI (left) and with our assignment (right)

- Our assignment reduces send/recv wait time by up to 60%
  - ✓ It reduces MPI_Allgather too by up to 30%

Paper published in Elsevier Computer Physics Communications Journal (Impact Factor: 3.268)

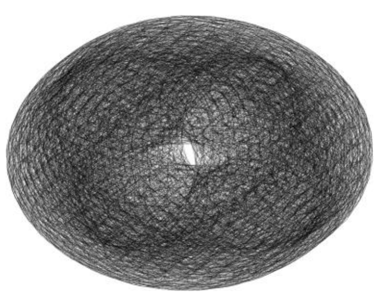
Task-Node Affinity





Nodes allocated for a run with 1K nodes on Jaguar

- Our task assignment for load balancing used a **3-D space filling curve**, assuming that the nodes are predominantly in a few cubic pieces of the machine
  - ✓ This assumption is not accurate
  - ✓ A more general solution will be useful

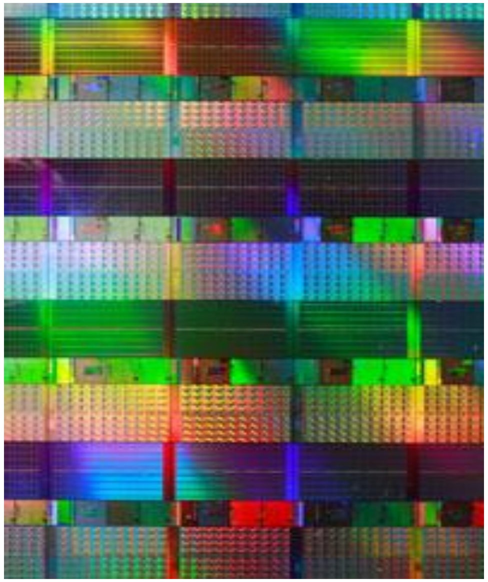


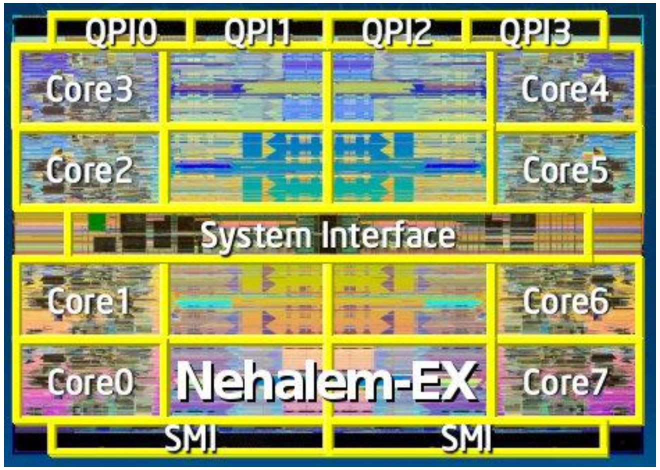
Cray XT5 (NCCS Jaguar)

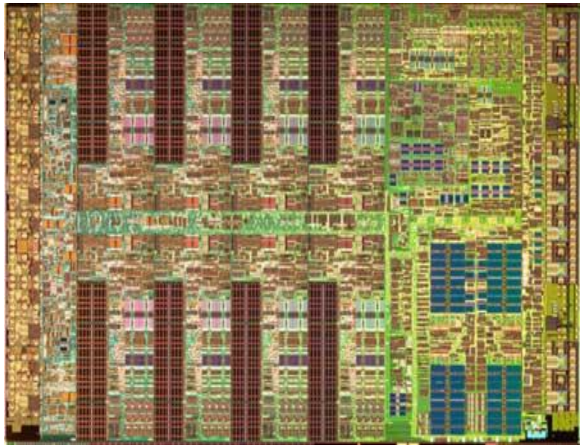
18,688 nodes, SeaStar 2+
25x32x24 **3D-Torus** network

2.595 PF, 532 TB/s interconnect

On-Chip Interconnect networks







Intel Polaris
80-core prototype
2D Mesh

Intel Nehalem EX
Ring

IBM Cell BE
Ring

Sun Niagara
Crossbar

MIT Raw, TRIPs
2-D Mesh Topology

Cell BE Processor Architecture

- **Cell BE**

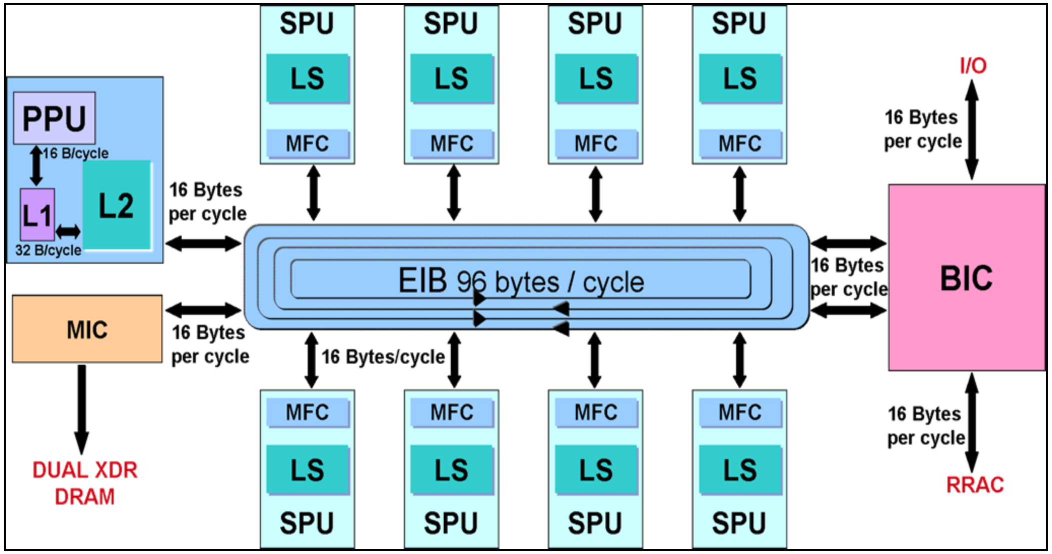
  8 SPEs, 1 PPE

  EIB

- **Inter SPE Communication:**

  EIB theoretical

  peak: 204.8 GB/s

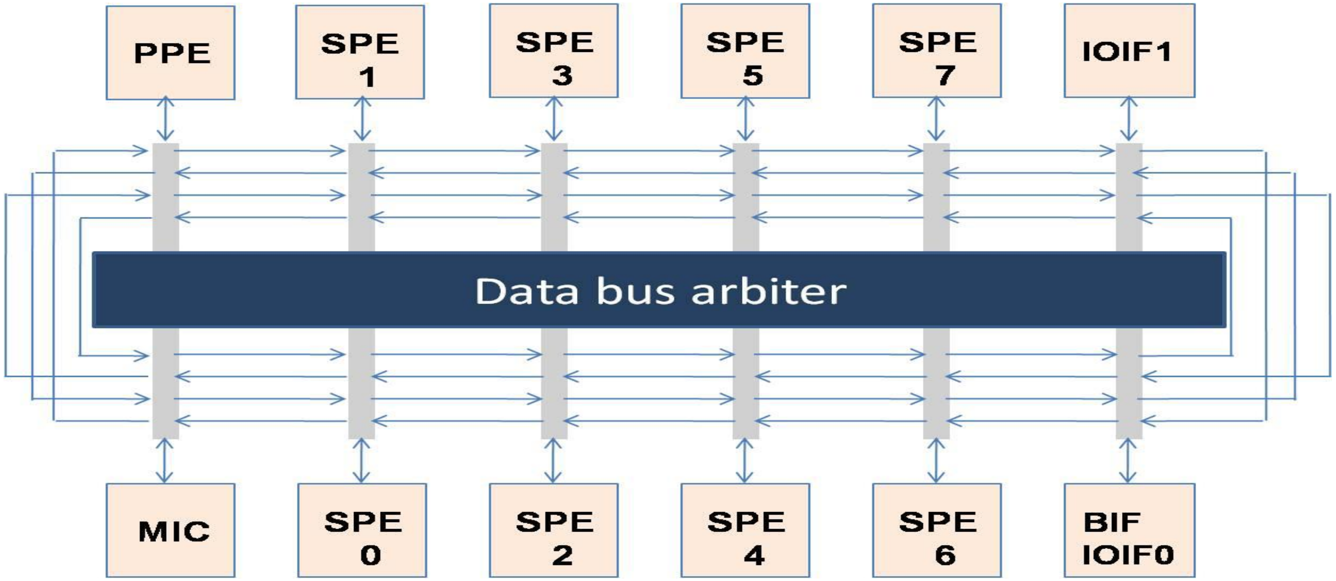


- **Memory Access:**

  MIC 25.6 GB/s

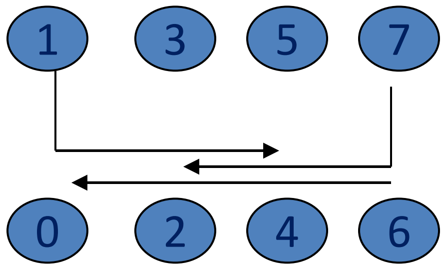
- **Algorithm Design:** Advantageous if SPEs communicate directly over EIB, and have less main memory usage.

Cell BE Topology and Routing



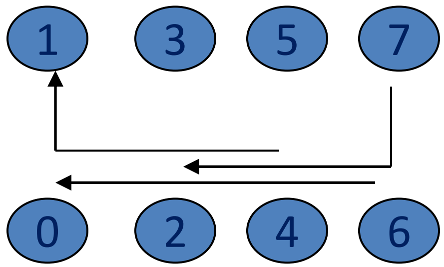
- **Topology:** Four unidirectional rings, two in each direction
- Theoretical peak network bandwidth is 204.8 GB/s
- Worst-case throughput of 50% or even less with adversarial traffic patterns
- **Routing:** Each ring supports 3 transfers when no path overlap
- Only shortest path routes are permitted

Inter-SPE Communication Bandwidth Analysis



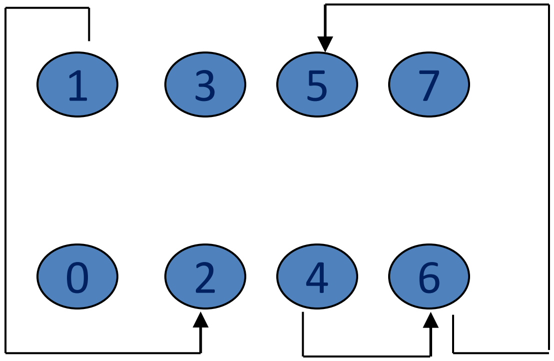
Three Comm. overlap
and not all in same direction

x06=[25.5969];
x27=[25.5969];
x41=[25.5642];



Three – Comm. overlap
and all in same direction

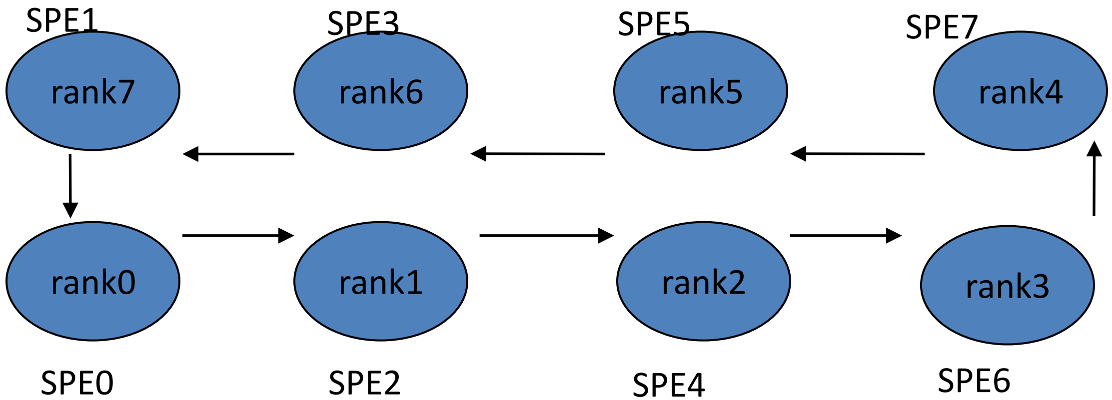
x06=[17.052];
x27=[25.5362];
x14=[17.0661];



Three non overlapping comm.
and in one direction and on the
same row.

x21=[24.54];
x64=[24.15];
x56=[24.3];

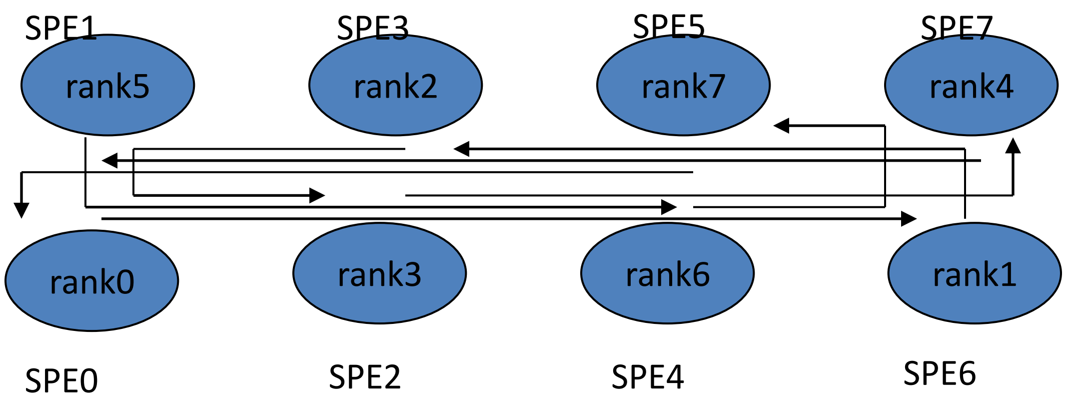
Performance of the Ring Pattern



Ring Mapping

MIN. Bandwidth [14.61 GB/s]; AVG Bandwidth [15.21 GB/s];

NO CONGESTION, LOAD IMBALANCE

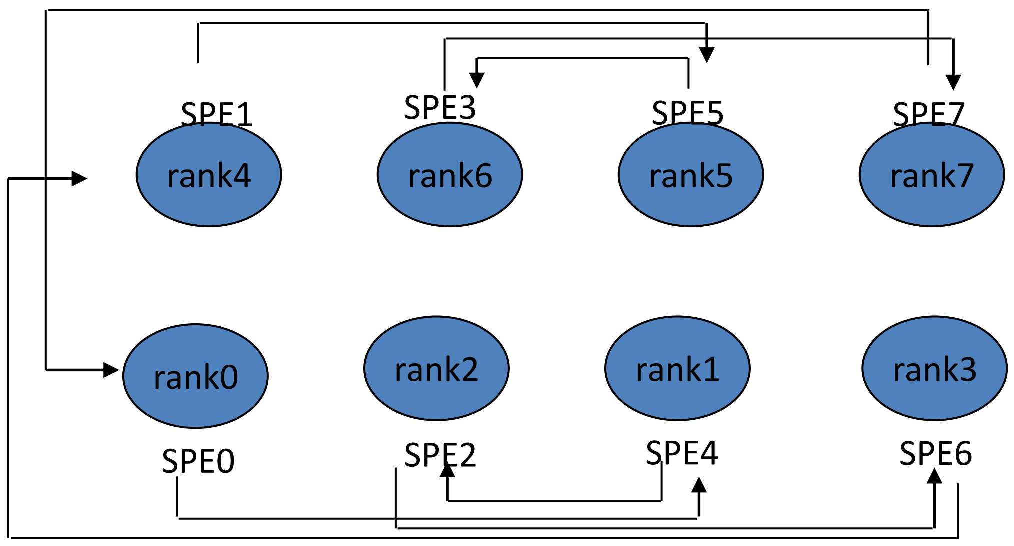


Overlap Mapping

MIN Bandwidth [7.24 GB/s]; AVG Bandwidth [8.8 GB/s];

CONGESTION & LOAD IMBALANCE

Performance of the Ring Pattern



EvenOdd Mapping

MIN. Bandwidth [24.15 GB/s]; AVG Bandwidth [24.33 GB/s];

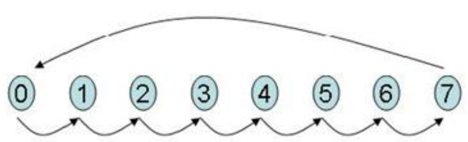
NO CONGESTION, LOAD BALANCED.

Observations:

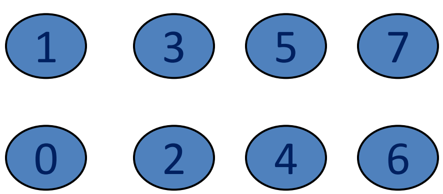
- Avoid overlapping paths for more than two messages in the same direction.
- Minimize the number of messages in any direction by balancing the load in both directions.
- Do not make any assumptions regarding the direction of transfer for messages that travel half-way across the EIB ring.

Performance of the Ring Pattern

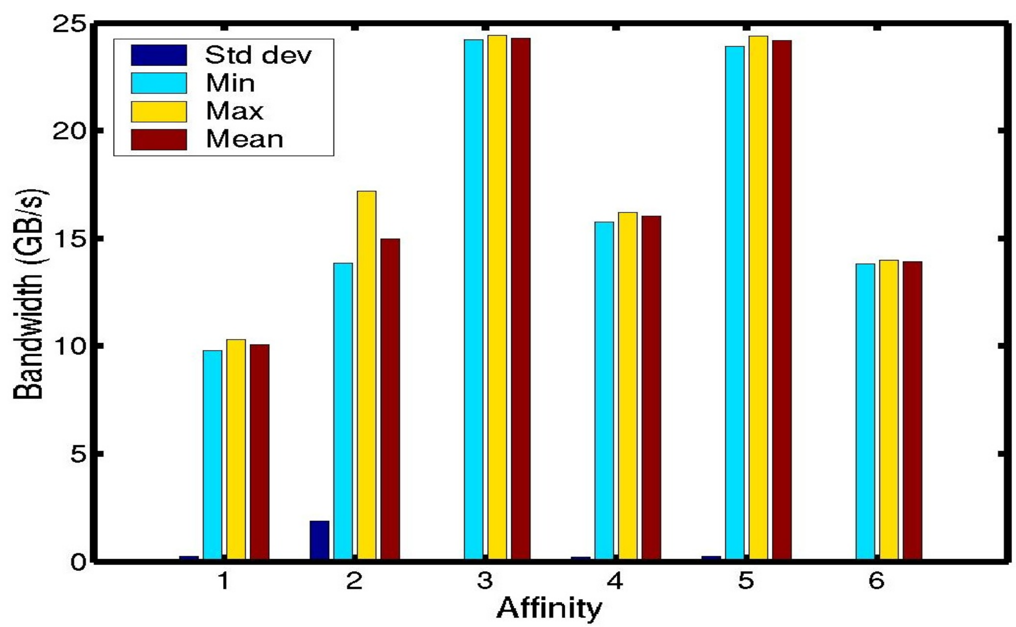
Affinity	(Physical ID, Thread Number) mapping
Overlap	{ (0, 0), (1, 7), (2, 2), (3, 5), (4, 4), (5, 3), (6, 6), (7, 1) }
EvenOdd	{ (0, 0), (1, 4), (2, 2), (3, 6), (4, 1), (5, 5), (6, 6), (7, 2) }
Identity	{ (0, 0), (1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7) }
Leap2	{ (0, 0), (1, 4), (2, 7), (3, 3), (4, 1), (5, 5), (6, 6), (7, 2) }
Ring	{ (0, 0), (1, 7), (2, 1), (3, 6), (4, 2), (5, 5), (6, 3), (7, 4) }



Ring Pattern

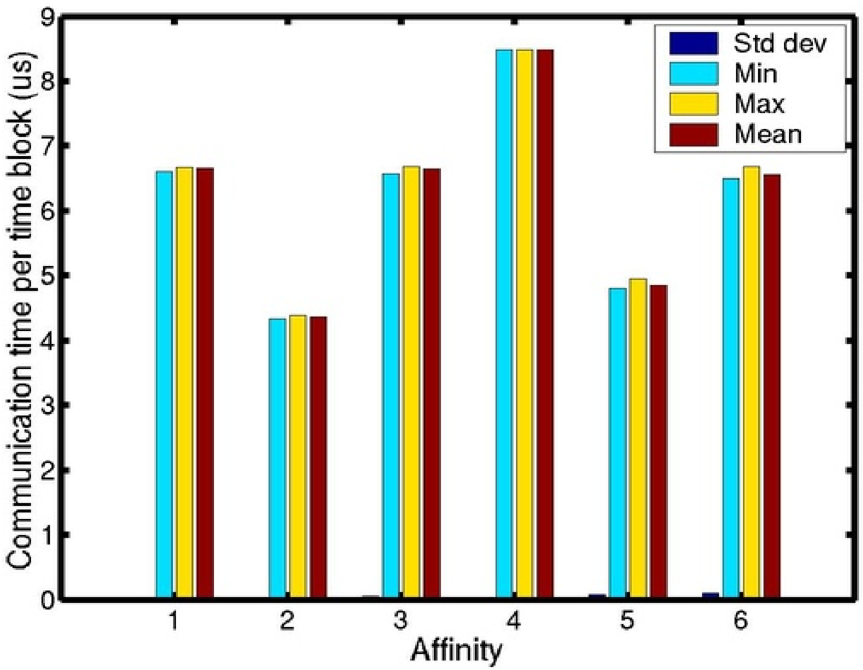


SPE Physical Layout

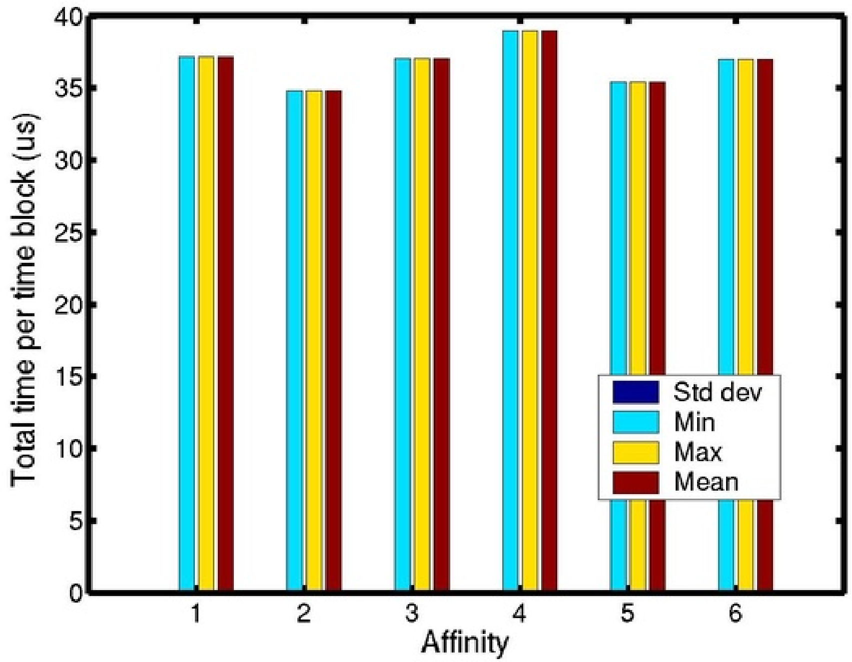


Affinities Tested: 1. Overlap 2. Default 3. EvenOdd 4. Identity 5. Leap2 6. Ring

Performance of Particle Transport Application



Communication Time



Total Application Time

Affinities Tested: 1. Identity 2. EvenOdd 3. Ring 4. Overlap 5. Leap2 6. Default

A factor of 2 difference between the best and worst affinities

10% between the best and worst affinities

Paper published in IEEE IPDPS 2009, PDSEC workshop;
PhD proposal accepted at IEEE TCPP PhD Forum