CS 498
Hot Topics in High Performance Computing

Networks and Fault Tolerance

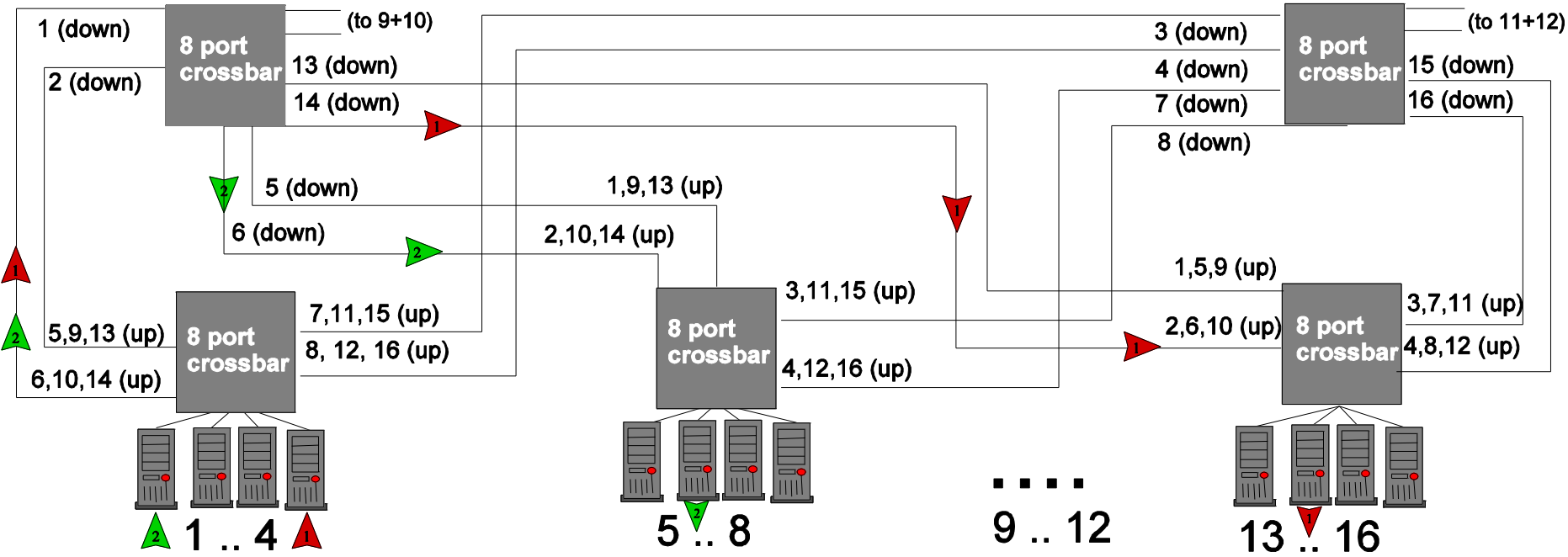
10. Routing and Flow Control

Intro

- What did we learn in the last lecture
  - Some more topologies
  - Routing (schemes and metrics)

- What will we learn today
  - Routing practical examples
  - Flow control
  - Blue Waters topology and routing (if time)

Deterministic Routing: InfiniBand



- Full bisection bandwidth fat-tree
- Communications 1 → 6 and 4 → 14

Deterministic Routing: InfiniBand

- Bisection (band)width is not a good metric!
  - It is only an upper bound on worst-case performance!
  - Routing can lower effective bandwidth
    - Deterministic routing *WILL* (see Valiant's proof)
- Introduce "Effective Bisection Bandwidth"
  - Expected bandwidth for a random permutation pattern (observable empirically)
  - Optimal: full bandwidth, check impact by simulation

Deterministic Routing: InfiniBand

- Thunderbird @ Sandia
  - 4096 compute nodes
  - dual Xeon EM64T 3.6 Ghz CPUs
  - 6 GiB RAM
  - ½ bisection bandwidth fat tree
  - 4390 active LIDs while queried

Deterministic Routing: InfiniBand

- Atlas @ LLNL
  - 1152 compute nodes
  - dual 4-core 2.4 GHz Opteron
  - 16 GiB RAM
  - full bisection bandwidth fat tre
  - 1142 active LIDs while queried

Deterministic Routing: InfiniBand

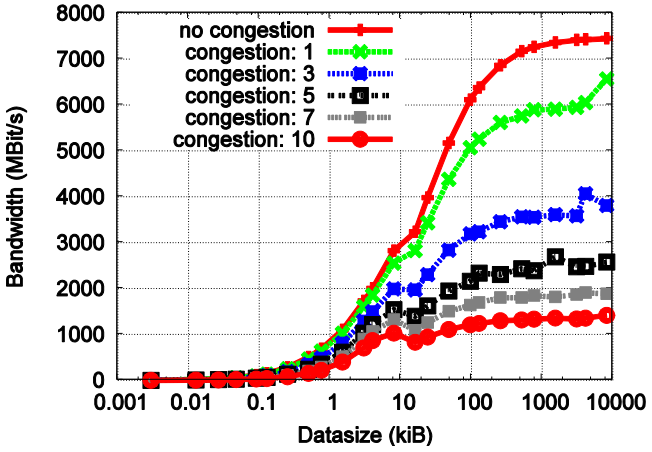
- Ranger @ TACC
  - 3936 compute nodes
  - quad 4-core 2.3 GHz Opteron
  - 32 GiB RAM
  - full bisection bandwidth fat tree
  - 3908 active LIDs while queried

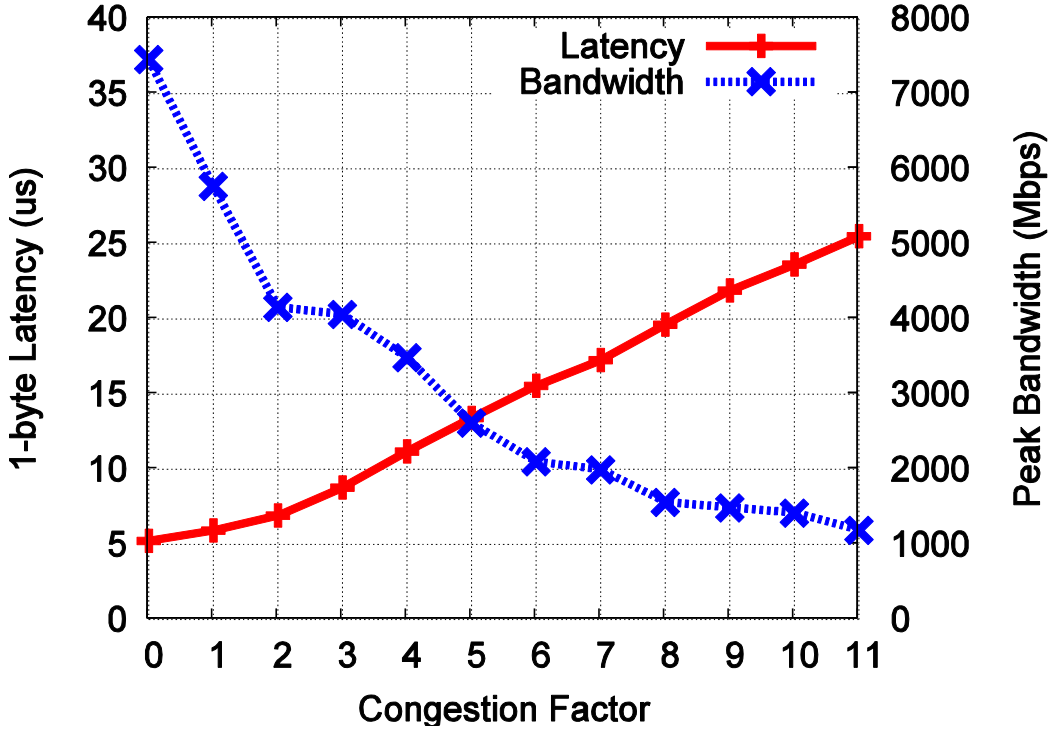
Deterministic Routing: InfiniBand

- CHiC @ TUC
  - 542 compute nodes
  - dual 2-core 2.6 GHz Opteron
  - 4 GiB RAM
  - full bisection bandwidth fat tre
  - 566 active LIDs while queried

Deterministic Routing: InfiniBand

- Impact of backend congestion on bandwidth
  - Provoked congestion
  - 24-port switches
  - Fat-tree
  - Max. cong: 12!





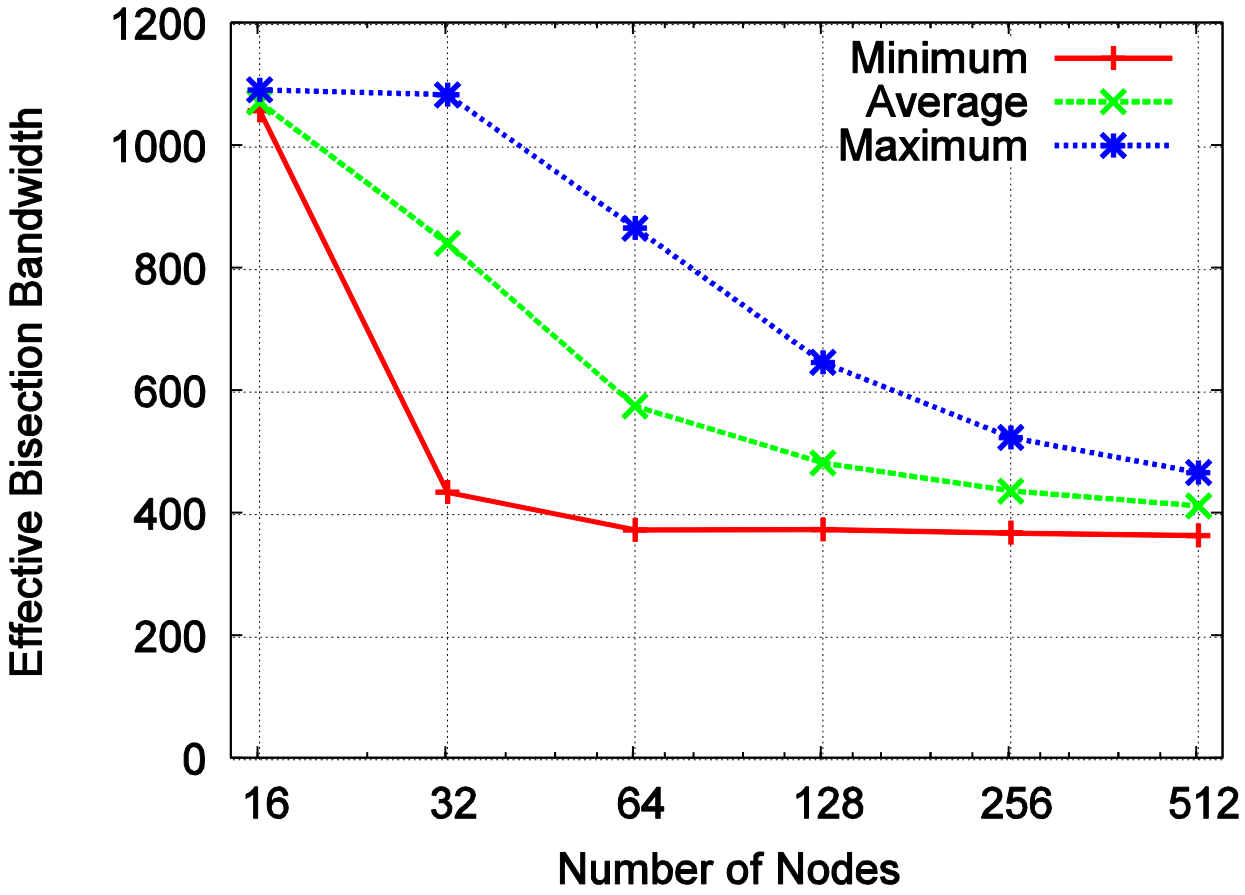
Deterministic Routing: InfiniBand

- Effective bisection bandwidth:
  - Ranger: 57.6%
  - Atlas: 55.6%
  - Thunderbird: 40.6% (1/2 bisection bandwidth)
- Obvious questions:
  - Does it make sense to build networks with full bisection bandwidth?
  - What is the tradeoff between cost and bandwidth?

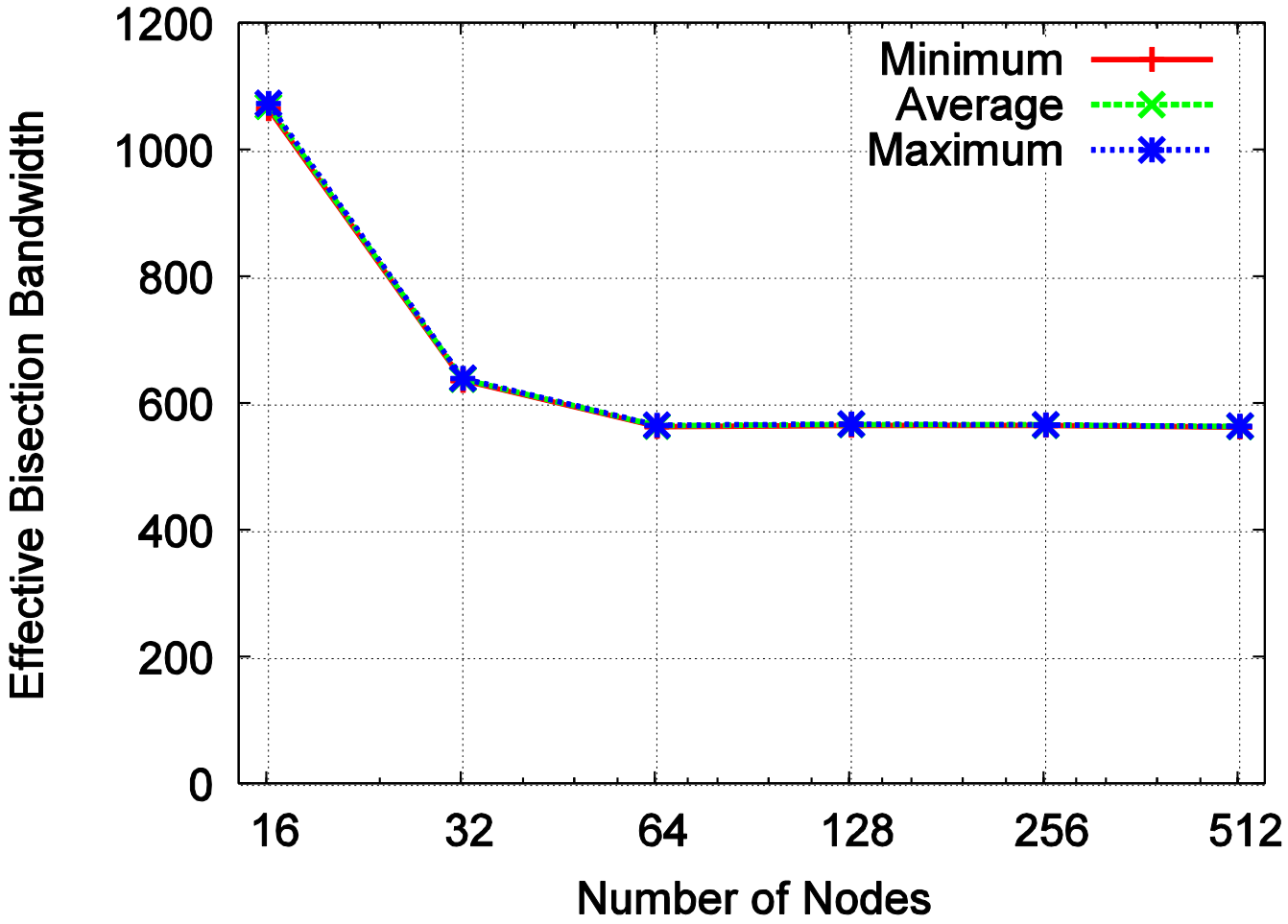
Adaptive Routing: Myrinet

- Implemented and benchmarked optimal deterministic routing (will be defined later), random routing, and local and global adaptive routing in Firmware
- 512 Myrinet MX nodes connected as full-bisection bandwidth Clos (similar to fat-tree)
- Running effective bisection bandwidth benchmark, reporting min, avg, max bandwidth

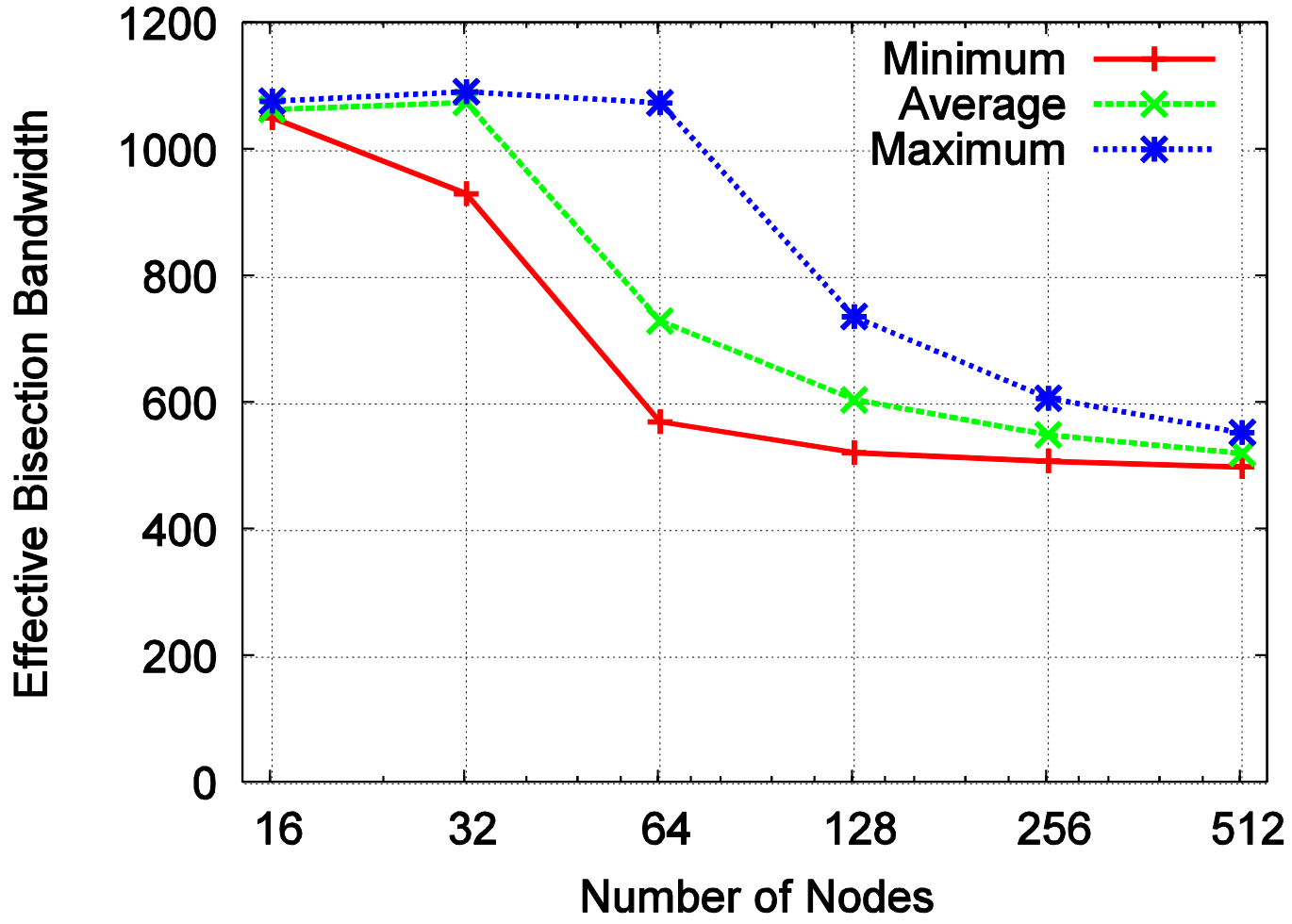
Deterministic Routing: Myrinet



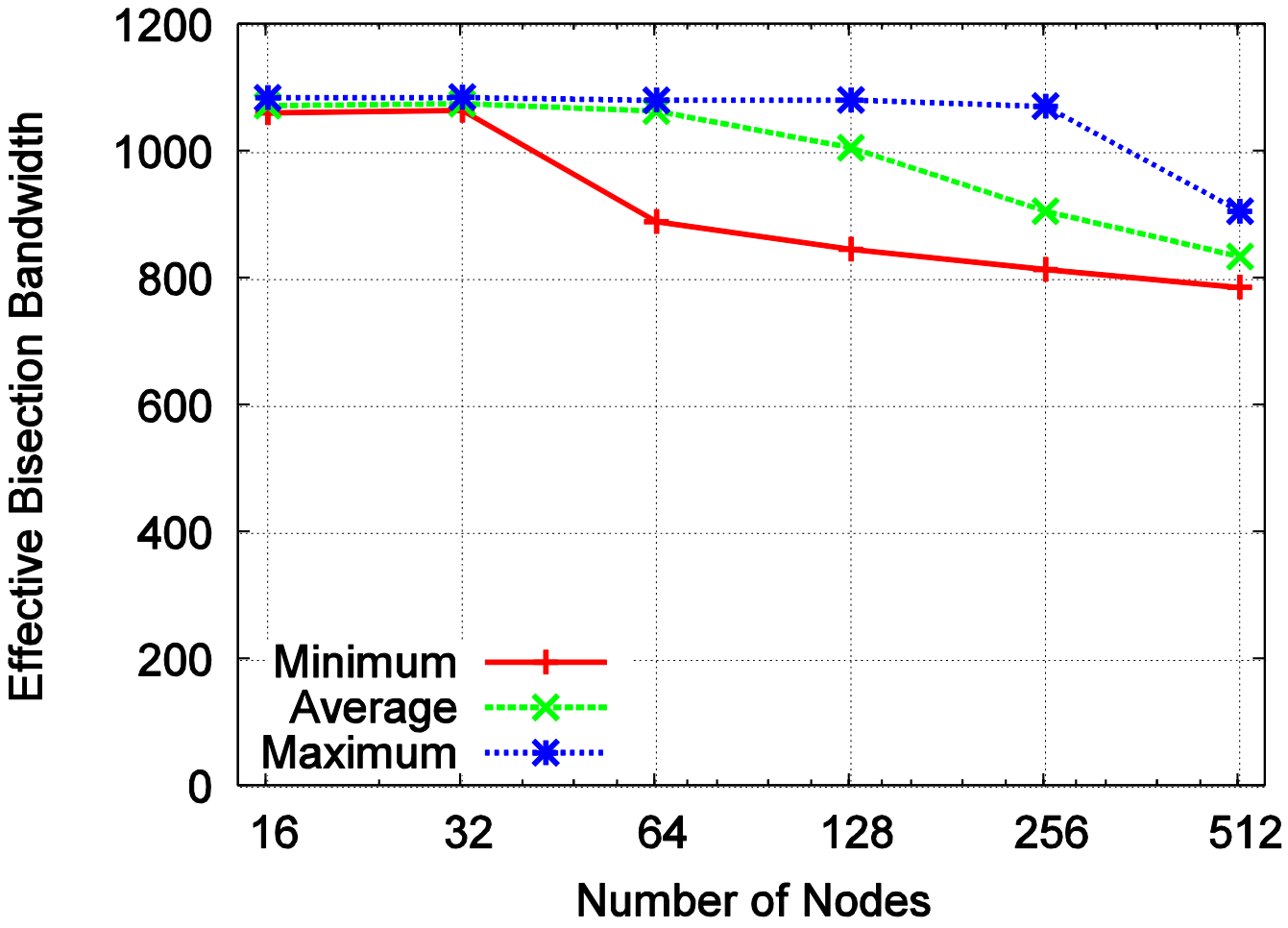
Random Routing: Myrinet



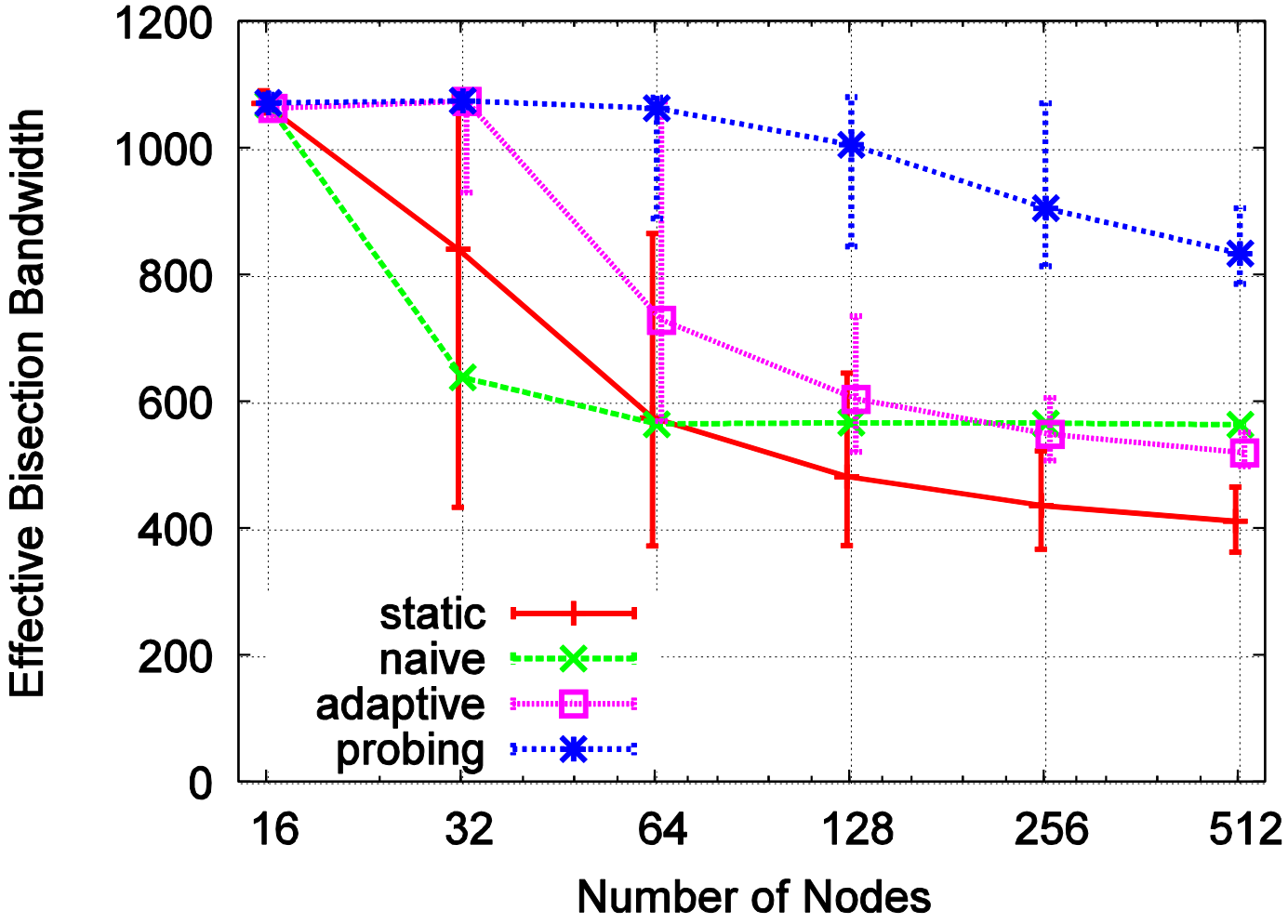
Local Adaptive Routing: Myrinet



Global (Probing) Adaptive Routing: Myrinet



Overview of Techniques: Myrinet



Routing Implementations

- Source routing
  - All routing decisions are made entirely at the source node and encoded in packet
- Distributed (table) routing
  - Routes are distributed across the switches (each switch only stores the parts it needs)
- Algorithmic routing
  - Routes can be determined algorithmically (e.g., butterfly, DOR, …)

Routing on Arbitrary Topologies

- General routing schemes needed for general topologies (e.g., InfiniBand, Ethernet)
  - Standard Ethernet uses spanning tree (bad)
- What metric to optimize for?
  - Application-optimized routing (possible if communication topology is known)
  - Arbitrary permutations seem reasonable for general case! → effective bisection bandwidth

Routing Metrics

- Model network as G=($V_P \dot{\cup} V_C$, E)
- Path r(u,v) is a path between $u,v \in V_P$
- Routing R consists of P(P-1) paths
- Edge load $l(e)$ = number of paths on $e \in E$
- Edge forwarding index $\pi(G,R)=max_{e \in E}\, l(e)$
  - $\pi(G,R)$ is a upper bound to congestion of permutation routing!

➢ Goal is to find R that minimizes $\pi(G,R)$

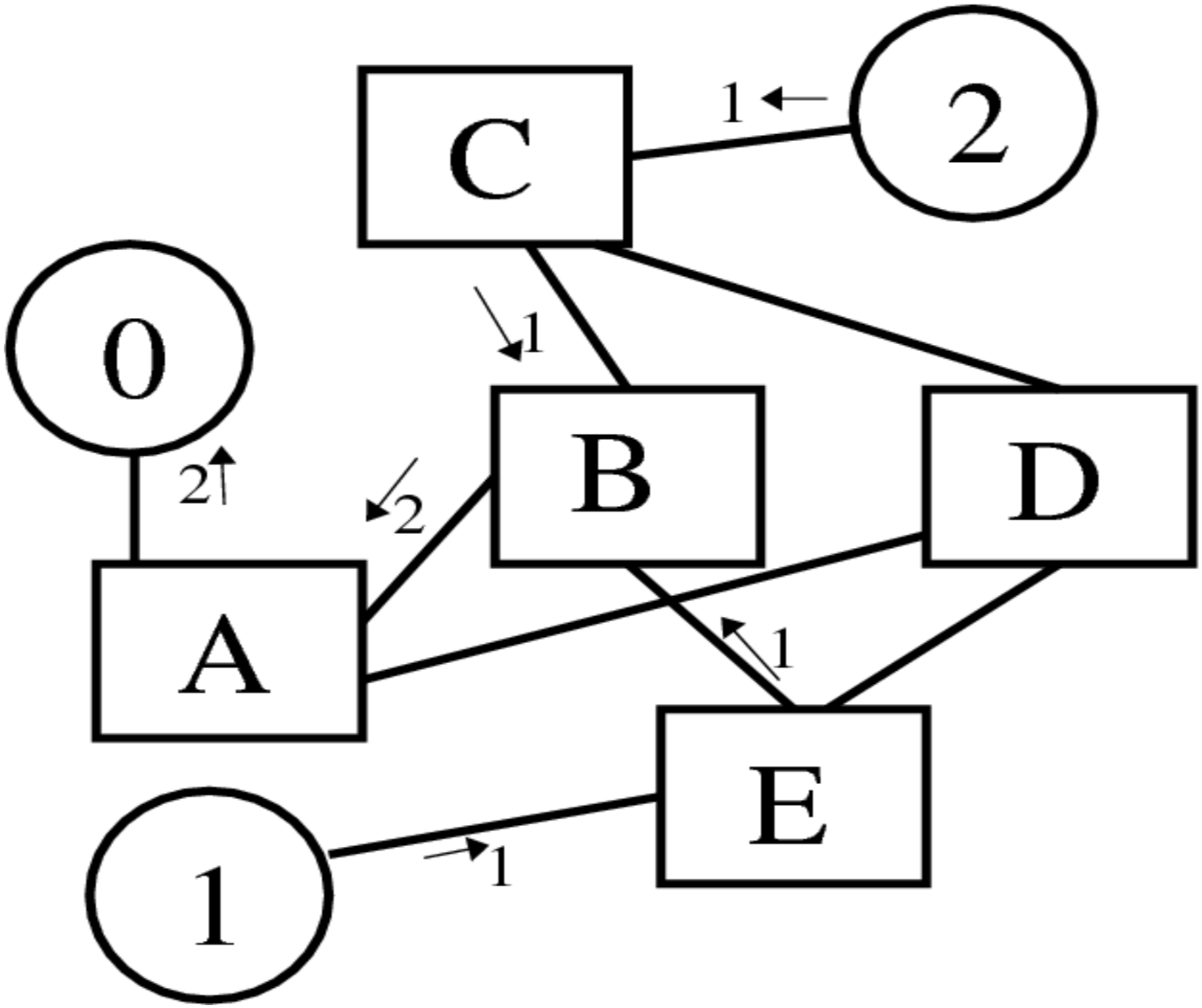
  - shown to be NP-hard in the general case

A Greedy Heuristic

- P-SSSP routing starts a SSSP run at each node
  - finds paths with minimal edge-load $l(e)$
  - updates routing tables in reverse
    - essentially SDSP
  - updates $l(e)$ between runs
- let's discuss an example …

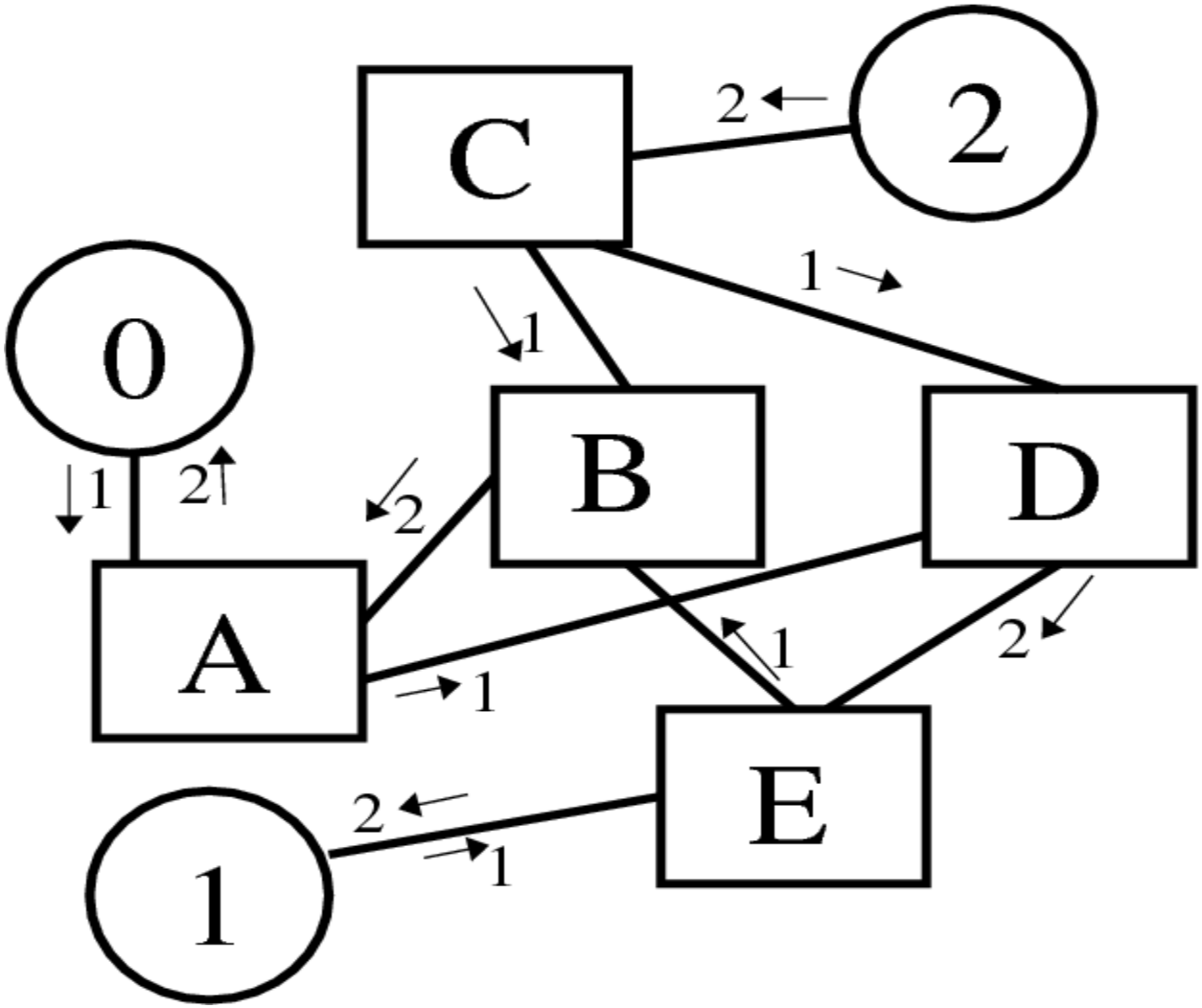
P-SSSP Routing (1/3)

Step 1:
Source-node 0:



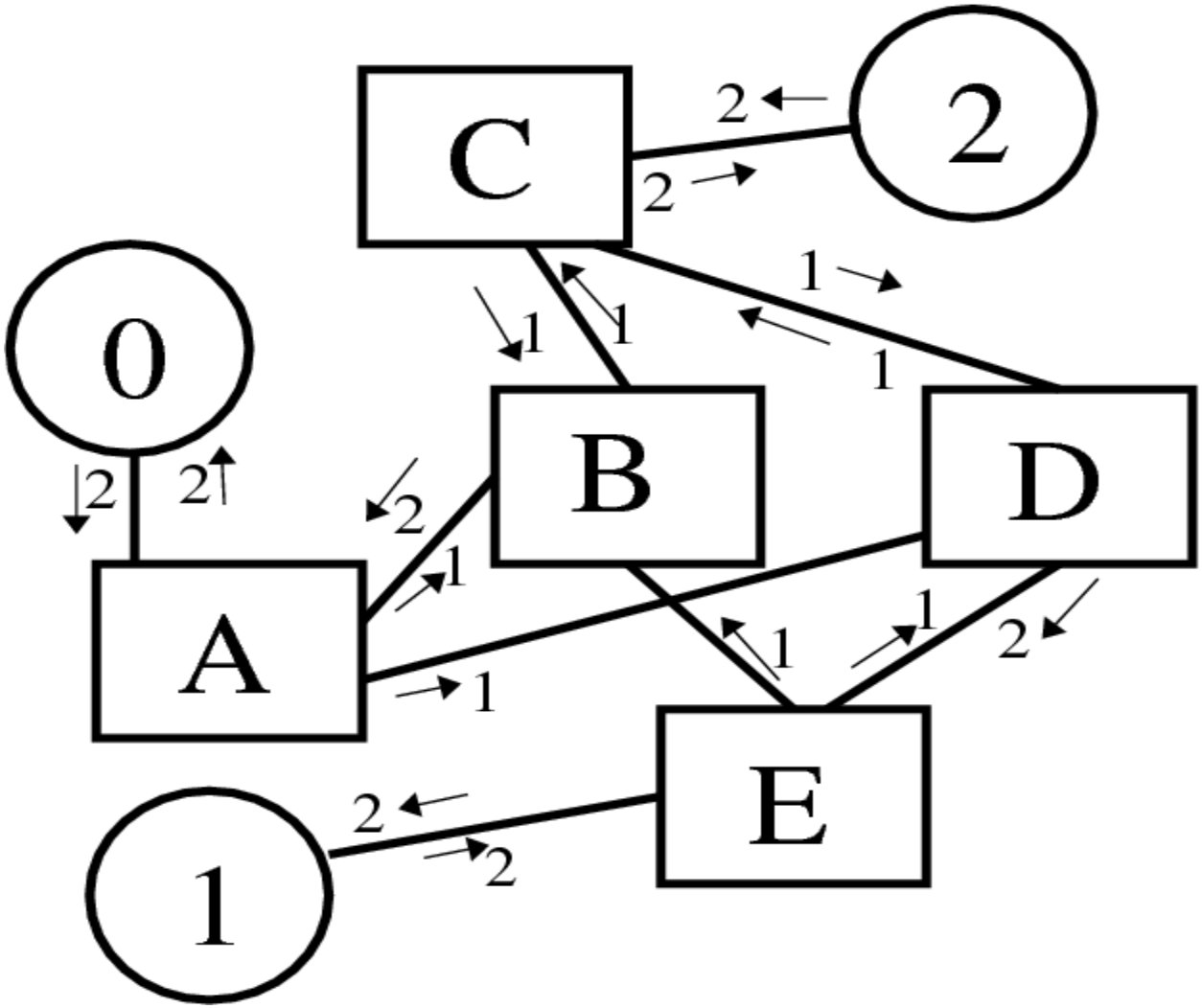
P-SSSP Routing (2/3)

Step 2:
Source-node 1:



P-SSSP Routing (3/3)

Step 3:
Source-node 2:



$\pi(G,R)$=2

Routing Example: InfiniBand

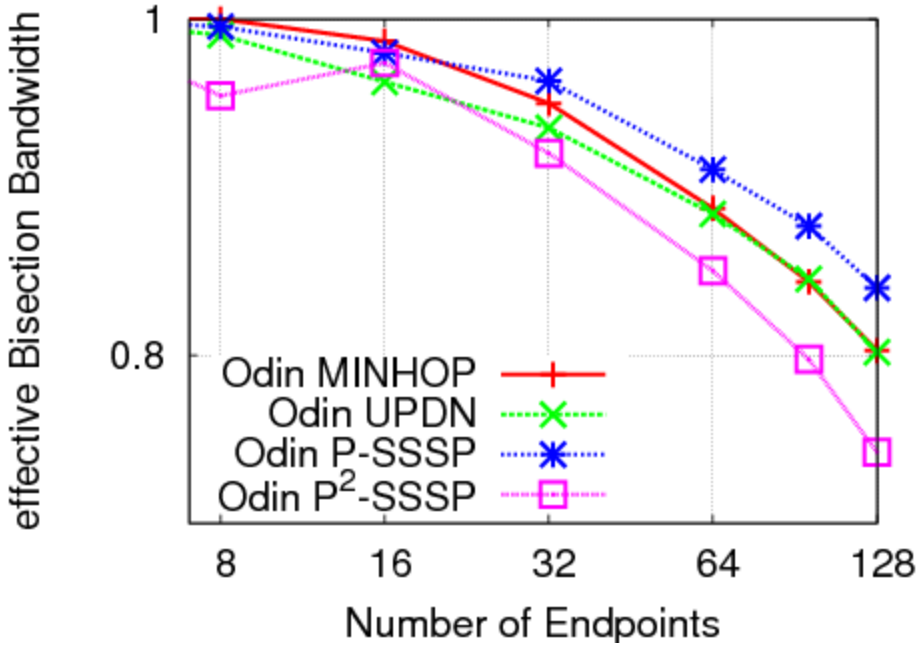
- Example: InfiniBand
  - InfiniBand uses deterministic distributed routing
  - OpenSM configures the local network (subnet)
    - implements different routing schemes
  - Linear forwarding table (LFT) at each switch
  - Lid mask control (LMC) enables multiple addresses per port

Routing Example: InfiniBand

- Different routing algorithms:
  - MINHOP (finds minimal paths, balances number of routes local at each switch)
  - UPDN (uses Up*/Down* turn-control, very similar to MINHOP)
  - FTREE (fat-tree optimized routing)
  - DOR (dimension order routing for k-ary n-cubes)
  - LASH (uses DOR and breaks credit-loops with virtual lanes)

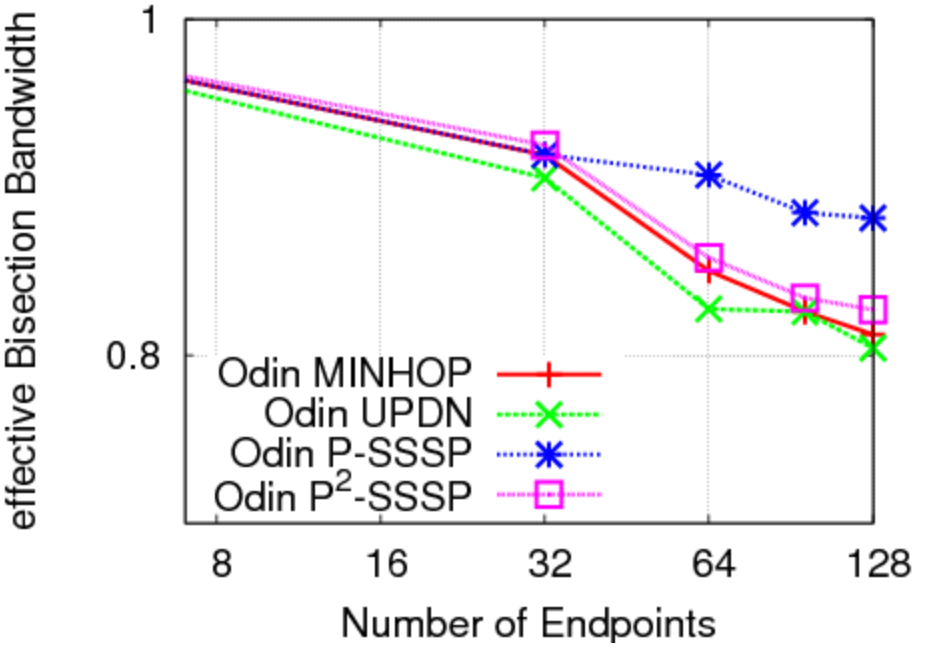
Benchmark Results Odin

Simulation



Simulation predicts 5% improvement

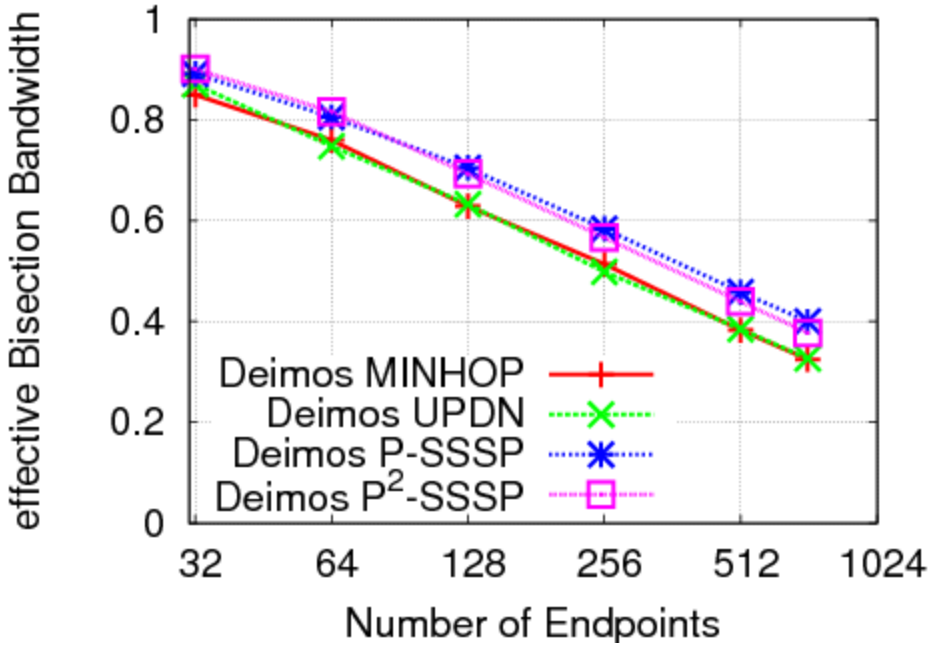
Benchmark
(Netgauge Pattern eBB)



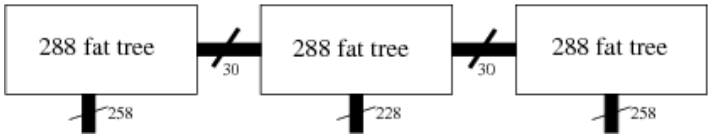
Benchmark shows 18% improvement!

Benchmark Results Deimos

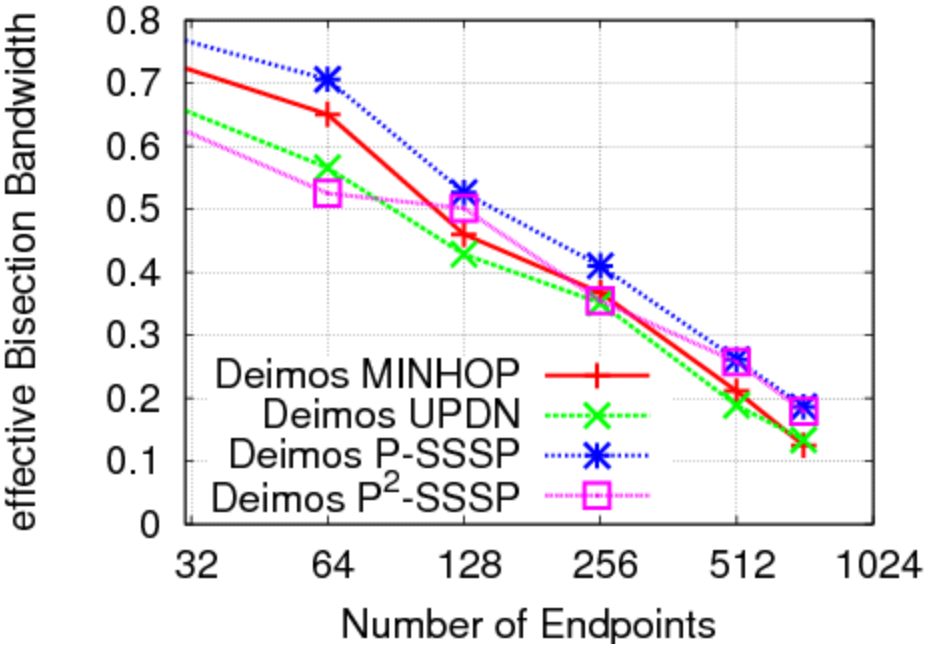
Simulation



Simulation predicts 23% improvement



Benchmark
(Netgauge Pattern eBB)



Benchmark shows 40% improvement!

Flow Control

- Determines how network resources are allocated
  - Can either be viewed as resource allocation or contention resolution problem
- Mechanisms:
  - bufferless: drop or misroute if channel is blocked
  - circuit switching: only headers are buffered and reserves path, waits if path is not available
  - buffered: decouples channel allocation in time

Flits

- Packets are MTU-sized
  - Typically several 1000 bytes
  - Switch buffering and forwarding often works at smaller granularity (several bytes)
- Flit – FLow Control UnIT
  - Packets are divided into flits by the hardware
  - Typically no extra headers

Buffered Flow Control

- Store-and-forward
  - Receives full packets into buffer and forwards them after they have been received
  - High latency (each switch waits for full packet)
- Cut-through
  - Forwards packet as soon as first (header) flit arrives and outgoing resources are available
  - Low latency but blocks a channel for the duration of a whole packet transmission

Buffered Flow Control

- Wormhole
  - Like cut-through but channels and buffers are allocated on a per-flit basis
  - Header flit allocates virtual channel, data and tail flits follow, tail flit frees channel
    - VC might have pointers to a packet's flits
  - Most flexible, high priority packets can intercept lower-priority packets "in the middle"
    - Most efficient use of buffer space
    - Also allows for buffers much smaller than MTU!

Buffer Management and Backpressure

- Flow control methods need to communicate availability of buffers to downstream nodes
- Common types in use today:
  - Credit-based
    - Router keeps count of number of free flit buffers of peer routers, decrements when sending, stops when zero reached, peers send “credits” back
  - On/Off
    - Binary decision, neighbor routers send “on” or “off” flag to start or stop incoming flit streams

Buffer Management and Backpressure

- Common types in use today (continued):
  - Ack/Nack
    - No state at routers, each flit will be ack'd if it fits and nack'd if not (resend)
  - Drop
    - No real flow control, packets are dropped if they do not fit and upper layer will re-send (common in Internet)

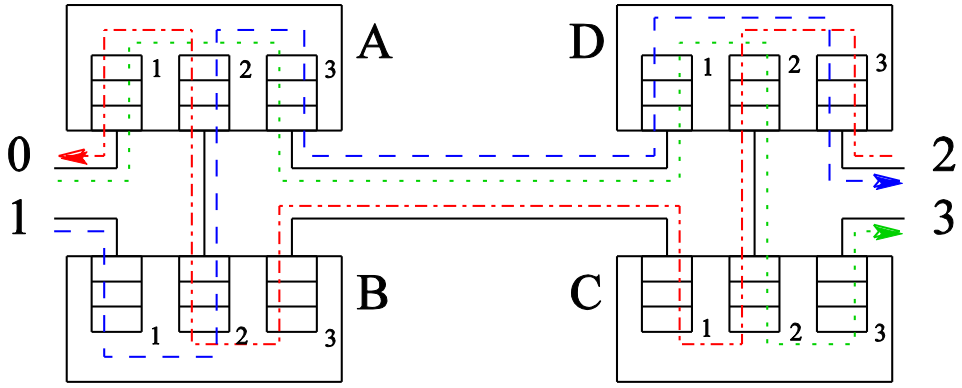
Livelock

- If packets can take non-minimal routes, they could be routed in circles forever
  - Good example: Internet
  - Simple workaround: TTL ☺
- Is not typical for HPC settings though
  - Will not be further discussed here

Deadlock

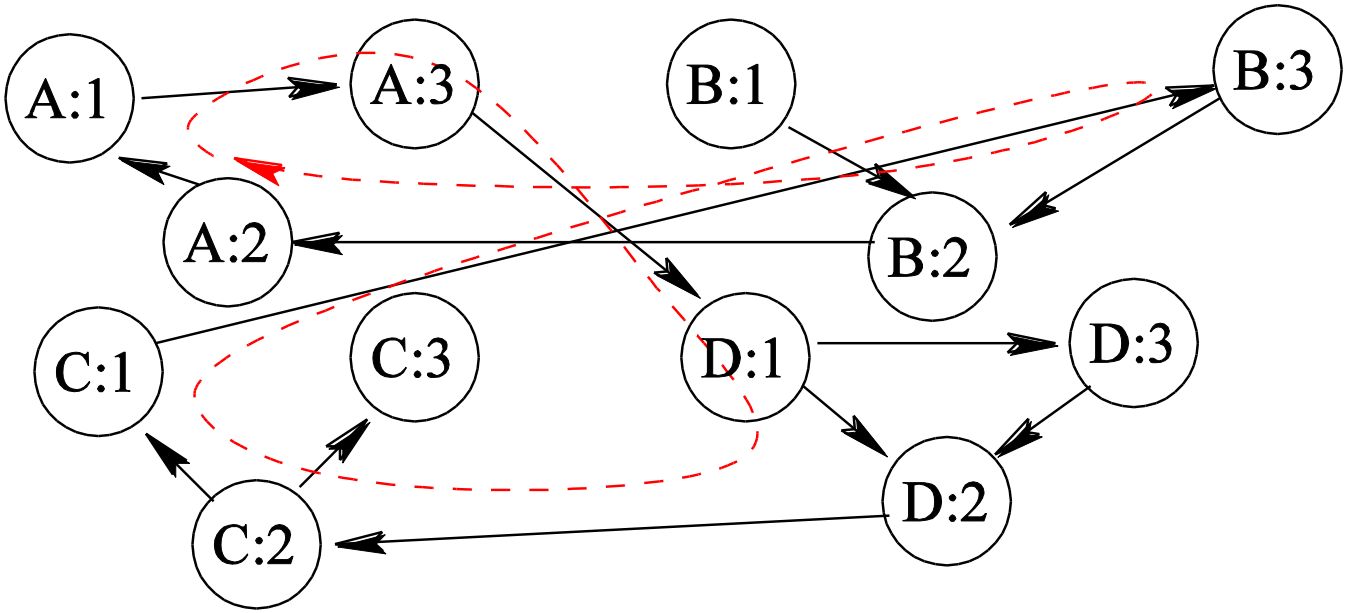
- A group of packets cannot progress because they wait for each other to free resources
  - Cyclic dependency (deadlock criterion)
- Deadlock is catastrophic (networks clogs up)
  - Deadlock avoidance
    - E.g., loop-free routing
  - Deadlock resolution
    - E.g., packet timeouts

Complex Deadlock Example



Source Network and Routes

Buffer Dependency Graph

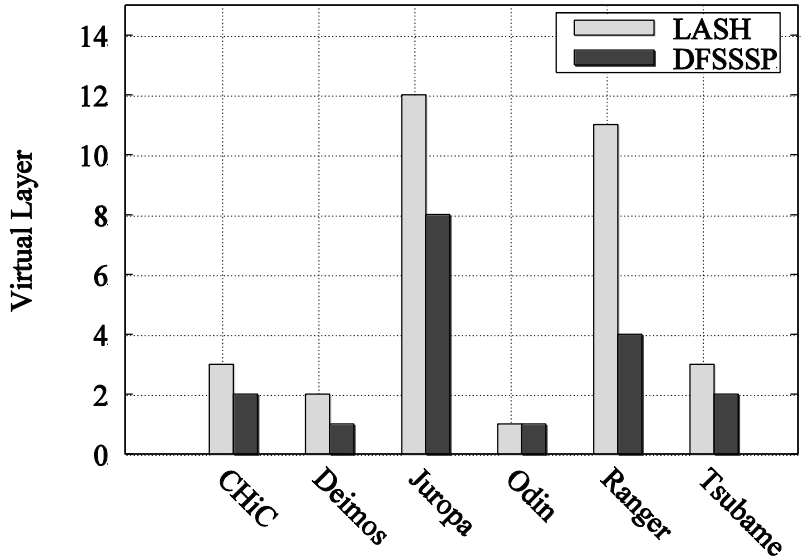


Deadlock Avoidance

- Limit route selection function
  - Up*/Down* routing
  - Identify a tree (root) in the topology
  - Allow only a single “turn” (relative to root)
  - Lower bandwidth due to limited routes
- Virtual Channels
  - VCs are independent, different block each other
  - Can be used to create cycle-free layers
  - Used in OpenSM’s LASH algorithm

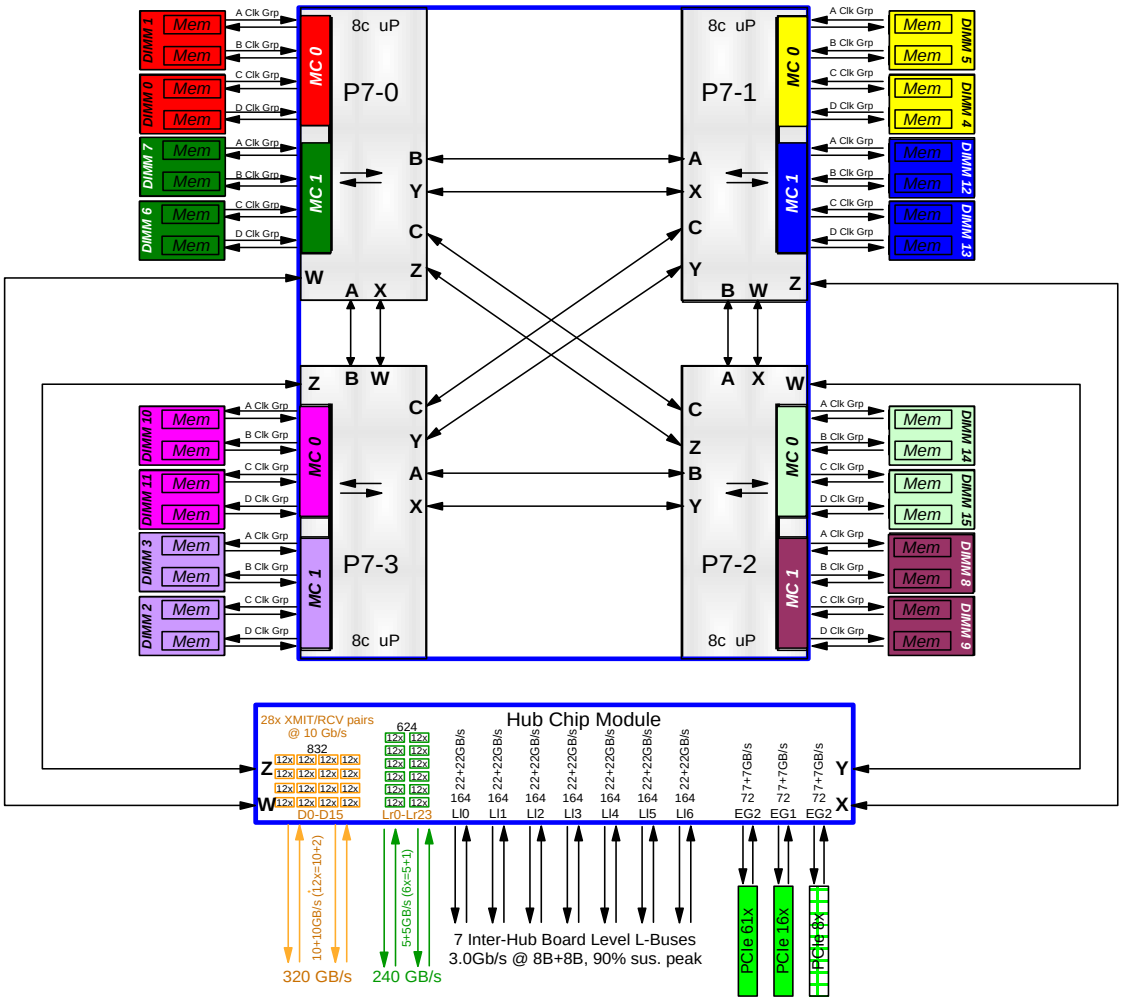
Layered Routing

- Each VL requires physical resources
  - Only 16 in InfiniBand (8 in practice)
- Layered routing needs to minimize needed VLs
  - Minimization is NP-complete for arbitrary channel dependency graphs
    - [see Domke, Hoefler, Nagel @IPDPS'11]



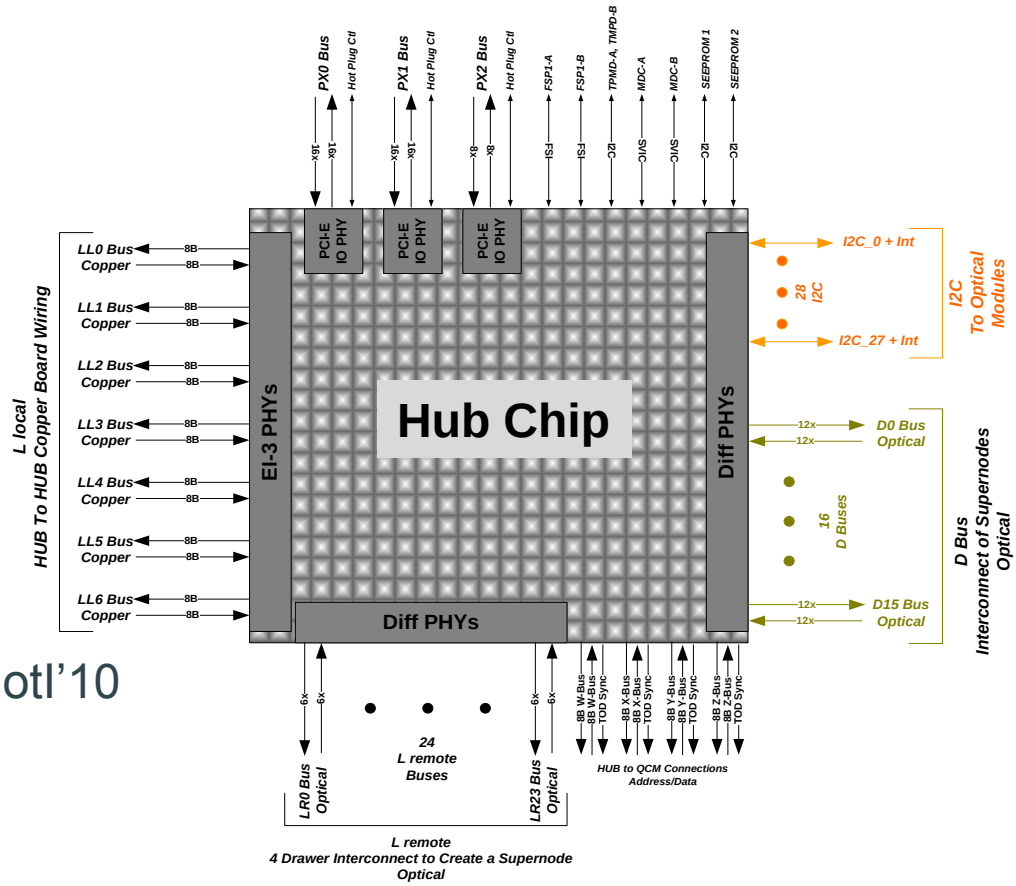
Topology Example: Blue Waters

- Four POWER 7 chips
- One IBM Hup Chip per node



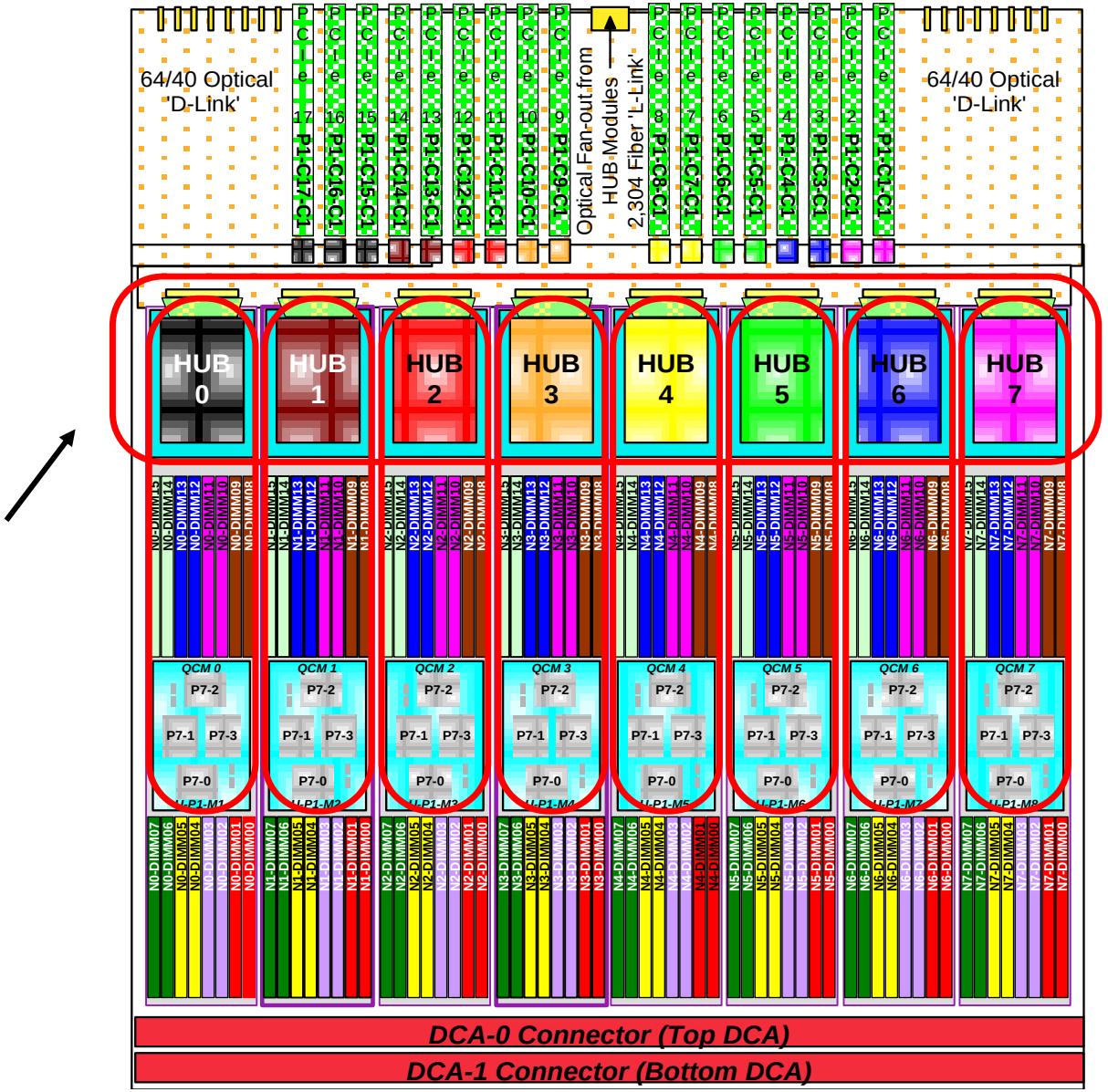
1.1 TB/s HUB

- 192 GB/s Host Connection
- 336 GB/s to 7 other local nodes
- 240 GB/s to local-remote nodes
- 320 GB/s to remote nodes
- 40 GB/s to general purpose I/O
- cf. “The PERCS interconnect” @HotI’10



Drawer

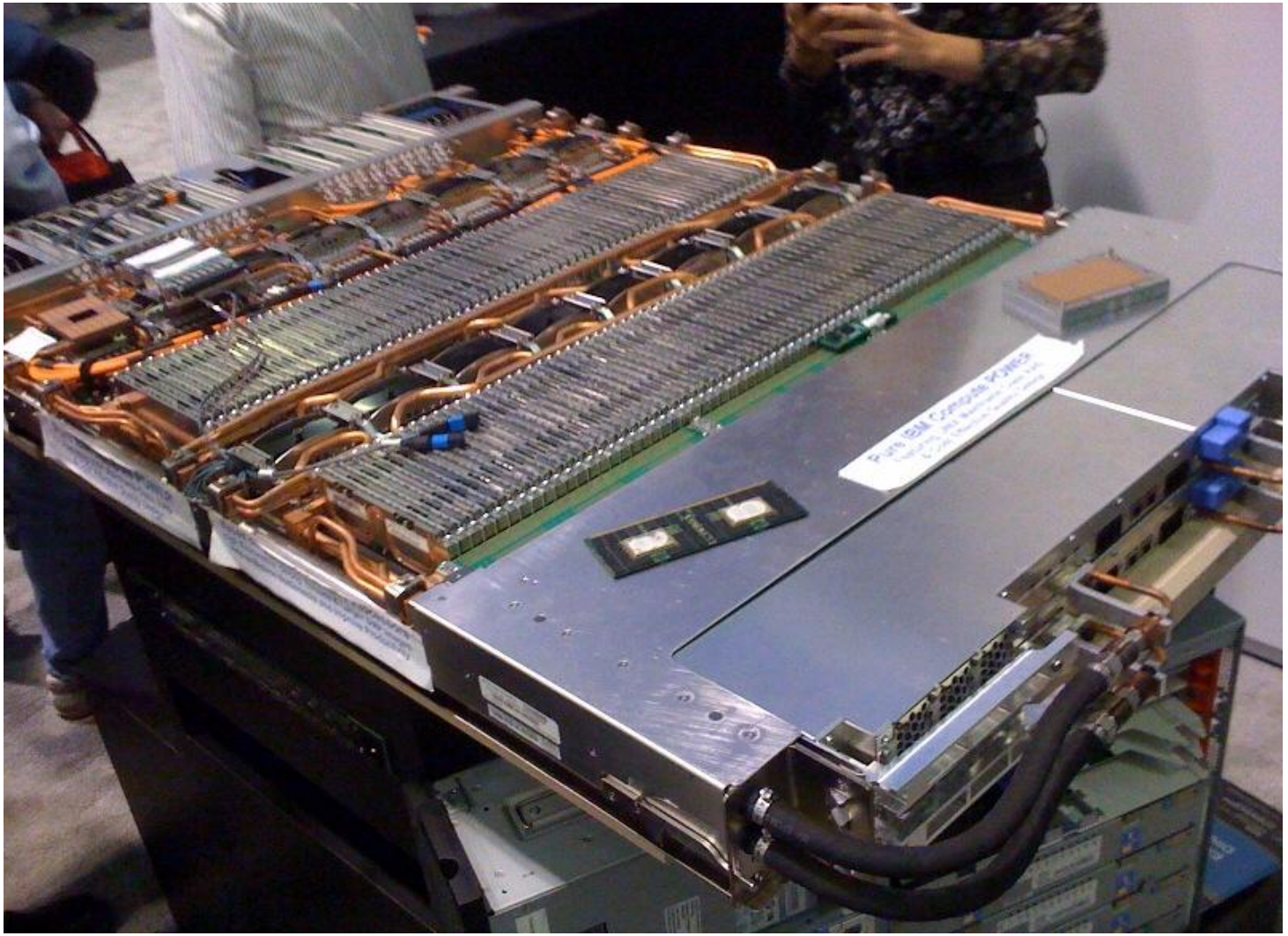
- 8 nodes
  - 32 chips
  - 256 cores



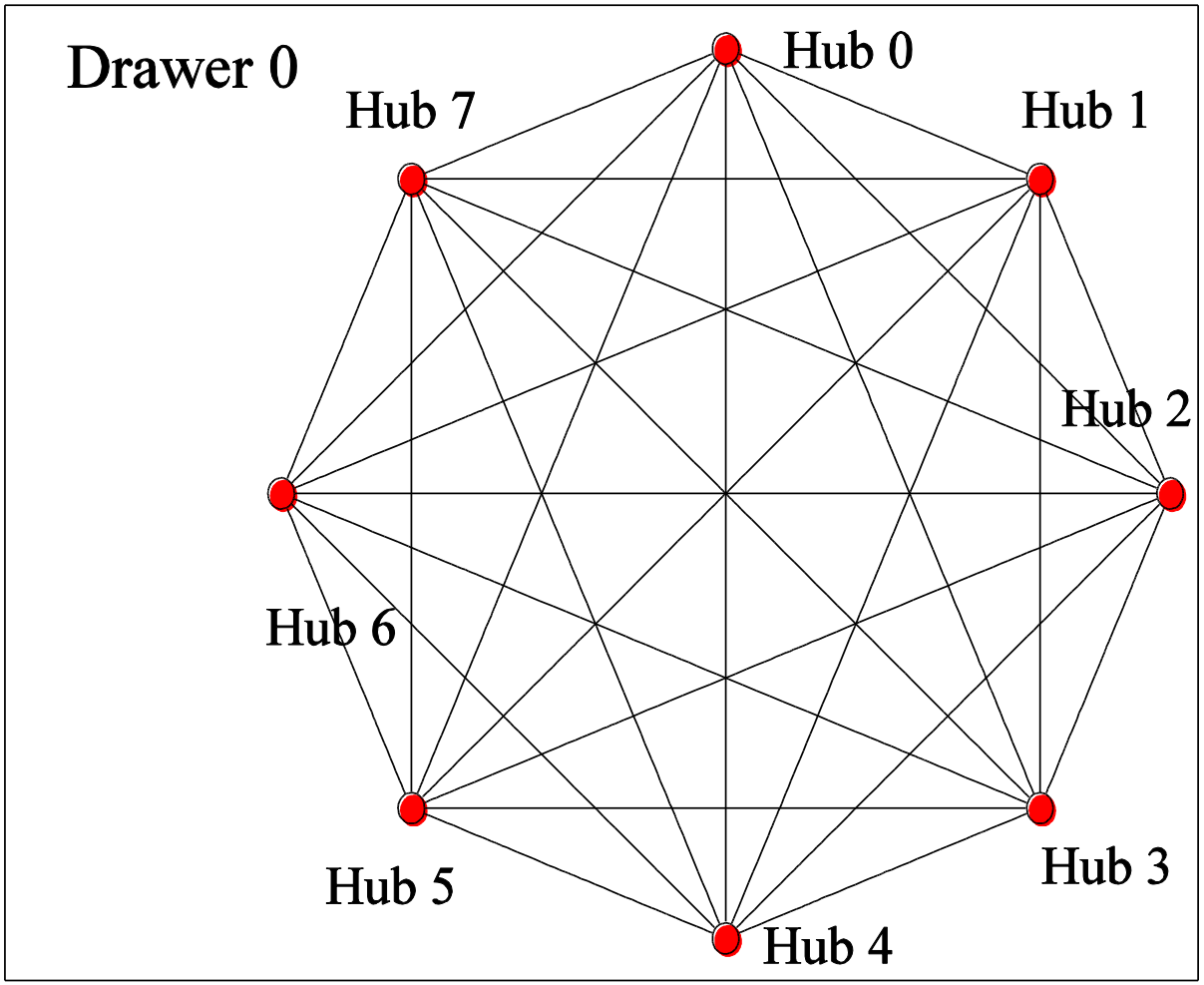
First Level Interconnect

- L-Local
- HUB to HUB Copper Wiring
- 256 Cores

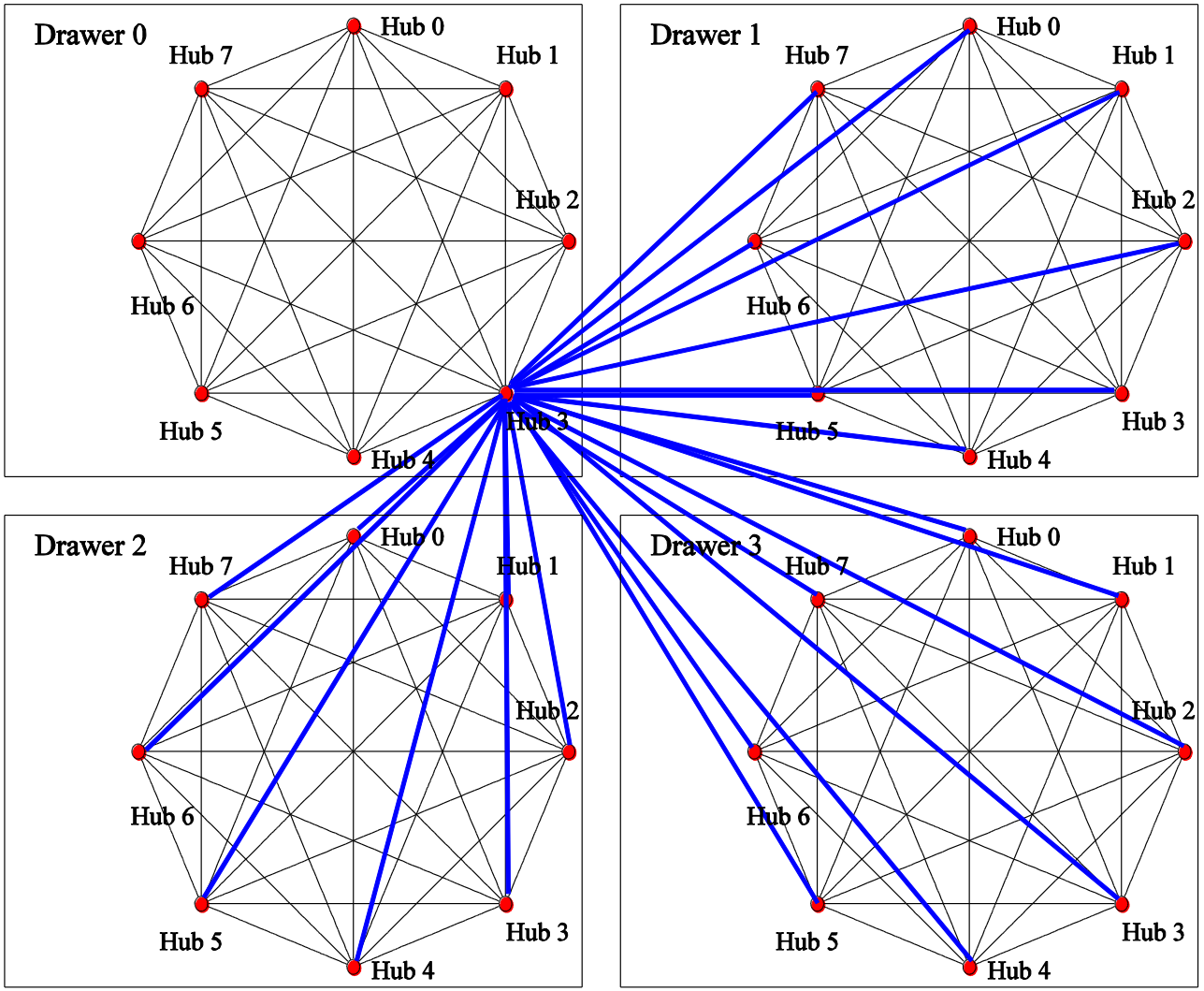
Topology Example: Blue Waters



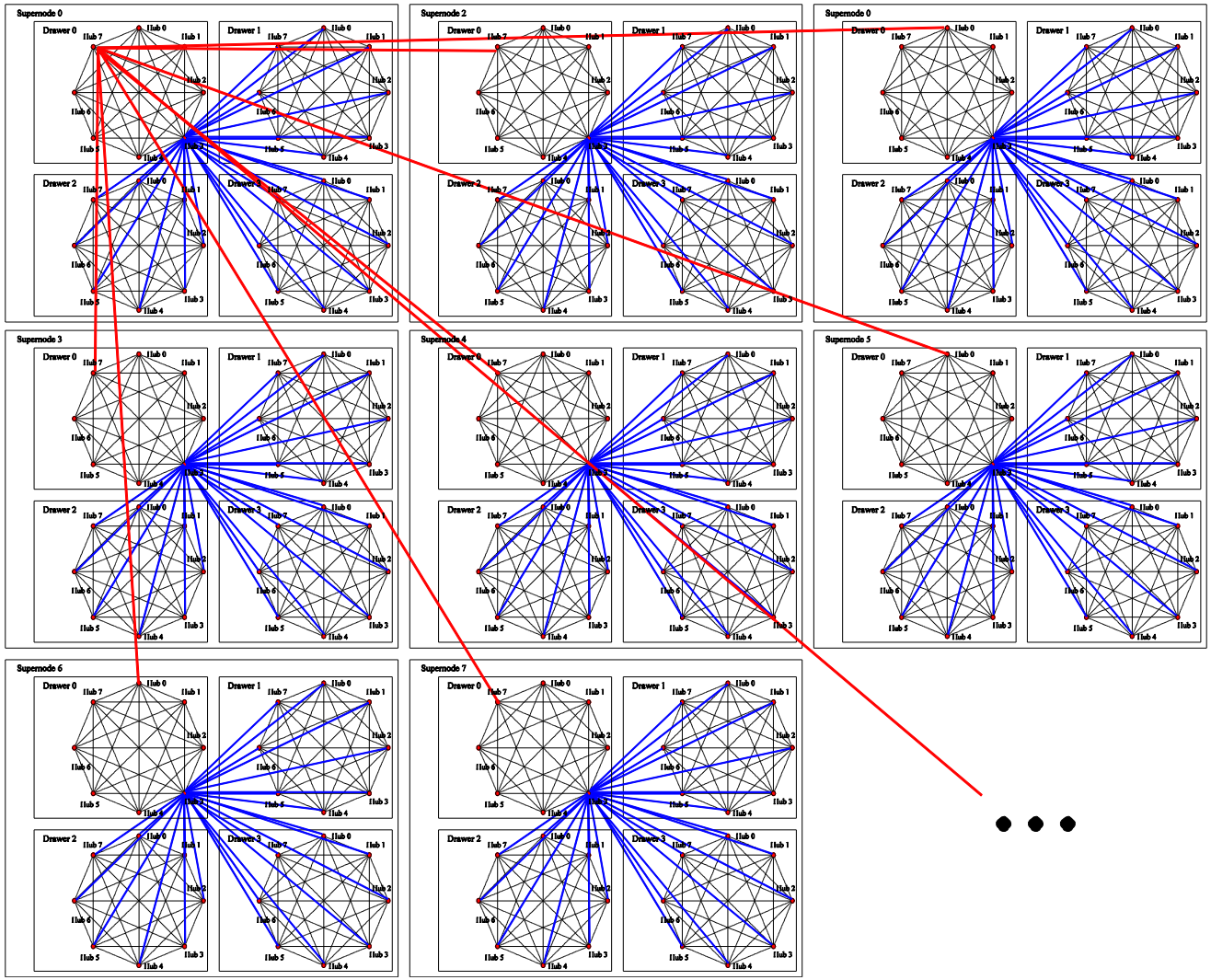
- LL Topology
  - 24 GB/s
  - 7 links/Hub
  - Fully connected
  - 8 Hubs



- LR Topology
  - 5 GB/s
  - 24 links/Hub
  - Fully connected
  - 4 Drawers
  - 32 Hubs



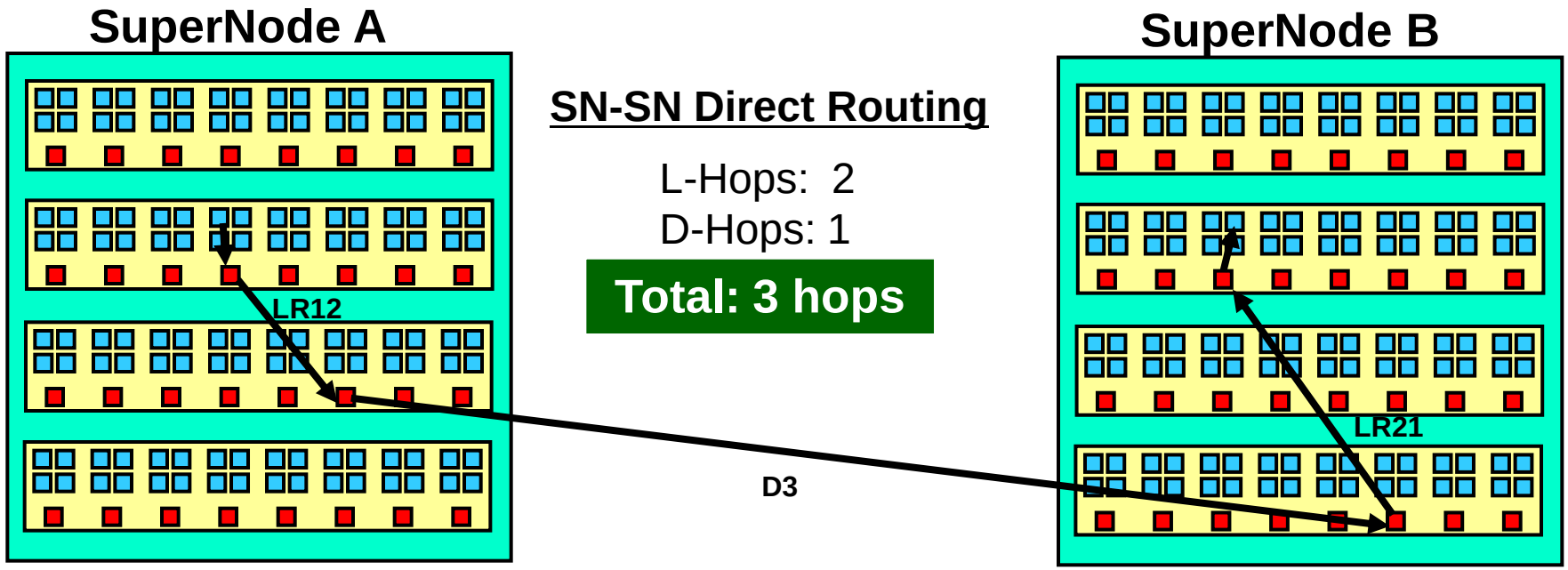
- D Topology
  - 10 GB/s
  - 16 links/Hub
  - Fully connected
  - 512 SNs
  - 2048 Drawers
  - 16384 Hubs

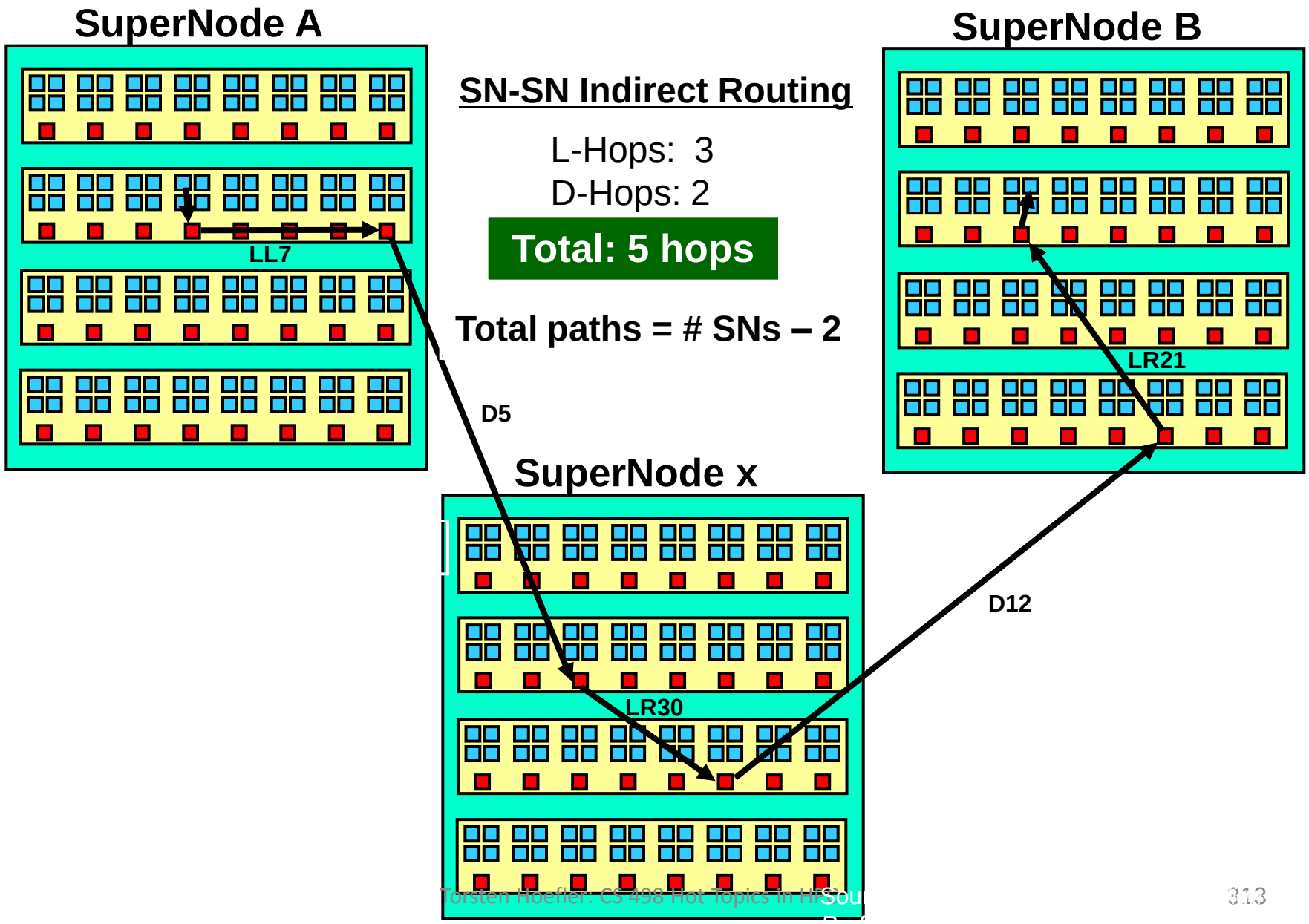


Routing

- Deterministic routing
  - Hits Borodin-Hopcroft bound ($c = \Theta(\sqrt{P})$)
  - Bad worst-case performance

- Indirect (random) routing
  - Increases latency
  - Effectively halves (global) bandwidth
  - Worst-case becomes very unlikely


SuperNode A
SuperNode B
SN-SN Direct Routing
L-Hops: 2
D-Hops: 1
Total: 3 hops
LR12
D3
LR21


SuperNode A
SuperNode B
SN-SN Indirect Routing
L-Hops: 3
D-Hops: 2
Total: 5 hops
Total paths = # SNs – 2
LL7
D5
SuperNode x
LR30
D12
LR21

Example Model: BW Global Alltoall Bandwidth

- Assume all communications happen simultaneously
- Derive upper (expected) bound for direct routing
  - Simple counting argument
- Each CN can be reached through series of LL, LR, D
  - Not more than one D link or LL-LR, LR-LL, LL-LL, LR-LR
- Denote e(P) as number of nodes reachable through path P
  - e(LL) =7, e(LR)=24, e(D)=d (variable number of D-links)
- # nodes reachable in two hops:
  - e(LL-D)=e(D-LL)=7d
  - e(LR-D)=e(D-LR) = 24d

Example: Alltoall Bandwidth Continued …

- # nodes reachable in three hops:
  - e(LL-D-LL) = 49d
  - e(LL-D-LR)=e(LR-D-LL) = 168d
  - e(LR-D-LR)=576d
- Number of paths from each source: 31+1024d
- Now count the number of paths (i.e., congestion) through each LL, LR, D link c(L): (omitted uninteresting parts of summation)
  - c(LL) = 1+64d
  - c(LR) = 1+64d
  - c(D) = 1024

Example: Alltoall Bandwidth Continued …

- Effective (expected) bandwidths for each link-type:
  - bandwidth = link bandwidth / congestion
  - b(LL) = 24 GB/s / (1+64d)
  - b(LR) = 5 GB/s / (1+64d)
  - b(D) = 10 GB/s / 1024
- If d<8, D is the bottleneck, otherwise LR
- Bandwidth per PE: 5 GB/s / 1+64d * total # paths
  - d=9 (294912 cores): 80.13 GB/s
  - d=10 (327680 cores): 80.11 GB/s total ~ 0.8 PB/s ☺
- Connection to P7 chips: 4*24 GB/s = 96 GB/s ~ 83% ☺