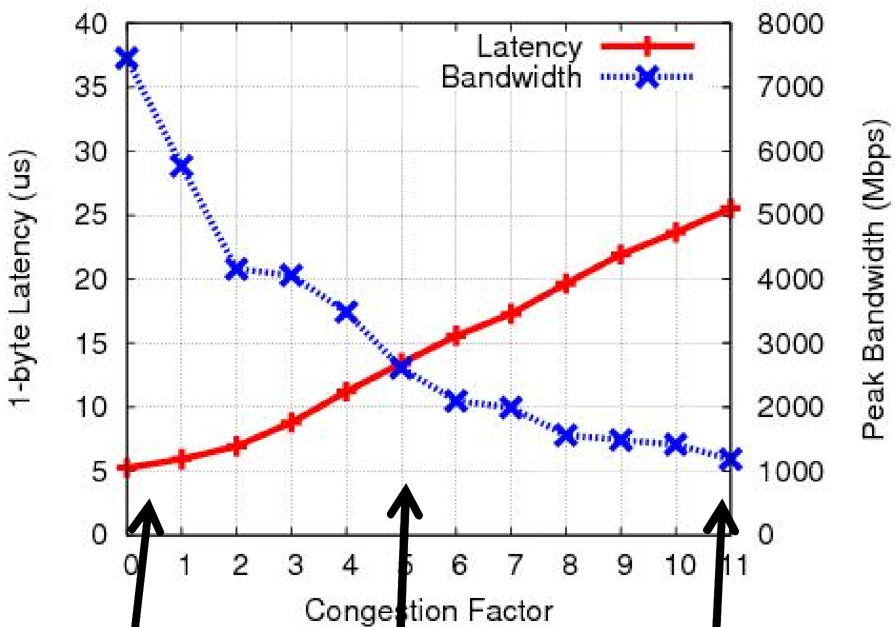
Optimized Routing for Large-Scale InfiniBand Networks

Torsten Hoefler, Timo Schneider, and Andrew Lumsdaine

Open Systems Lab
Indiana University

Effect of Network Congestion



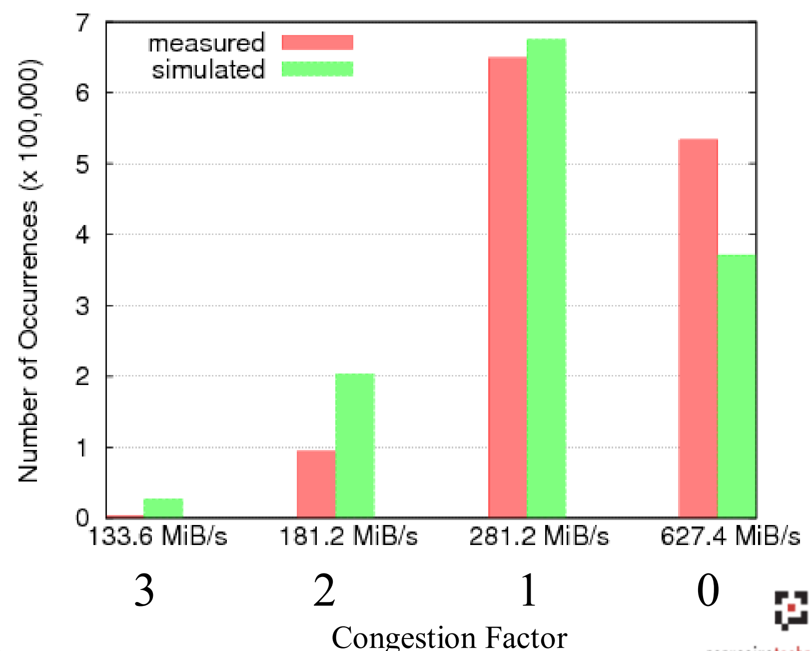
Microbenchmarks
(NetPIPE, IMB ping pong
Netgauge one_one)

Reality?

Lower Bound!

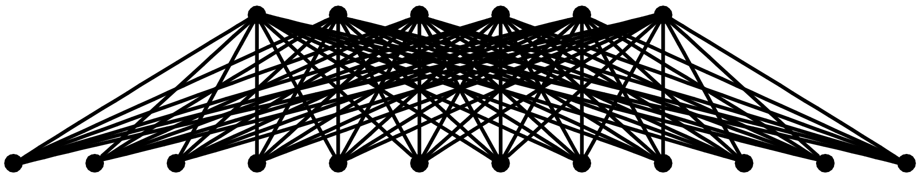
CHiC Supercomputer:

- 566 nodes, full bisection IB fat-tree
- effective Bisection Bandwidth: 0.699



Full Bisection Bandwidth != Full Bandwidth

- expensive topologies do not guarantee high bandwidth
- deterministic oblivious routing cannot reach full bandwidth!
  - see Valiant's lower bound
  - random routing is asymptotically optimal but looses locality



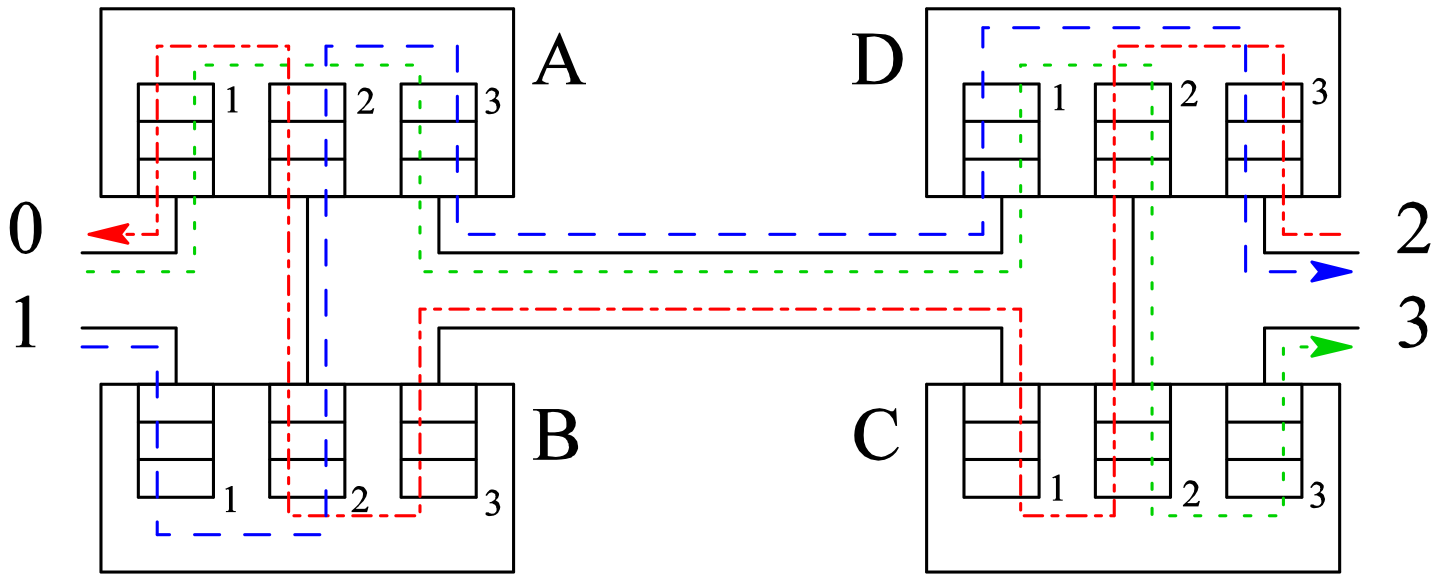
- but deterministic routing has many advantages
  - completely distributed
  - very simple implementation
- InfiniBand routing:
  - deterministic oblivious, destination-based
  - linear forwarding table (LFT) at each switch
  - lid mask control (LMC) enables multiple addresses per port

InfiniBand Routing Continued

- offline route computation (OpenSM)
- different routing algorithms:
  - MINHOP (finds minimal paths, balances number of routes local at each switch)
  - UPDN (uses Up*/Down* turn-control, limits choice but routes contain no credit loops)
  - FTREE (fat-tree optimized routing, no credit loops)
  - DOR (dimension order routing for k-ary n-cubes, might generate credit loops)
  - LASH (uses DOR and breaks credit-loops with virtual lanes)

Why do Credits Loop?

- IB uses credit-based p2p flow-control
  - egress messages sent only if receive-buffer available



- very similar to deadlocks in wormhole-routed systems

How to deal with Credit Loops?

- prevent (UP*/Down*, turn-based routing)
- resolve (LASH, use VLs to break cycles)
- ignore (MINHOP, DOR, not as bad as it sounds, might deadlock but can be “resolved” with packet timeouts)
  - discouraged by IB spec

Some Theoretical Background

- model network as $G=(V_P \cup V_C, E)$
- 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 trivial upper bound to congestion!

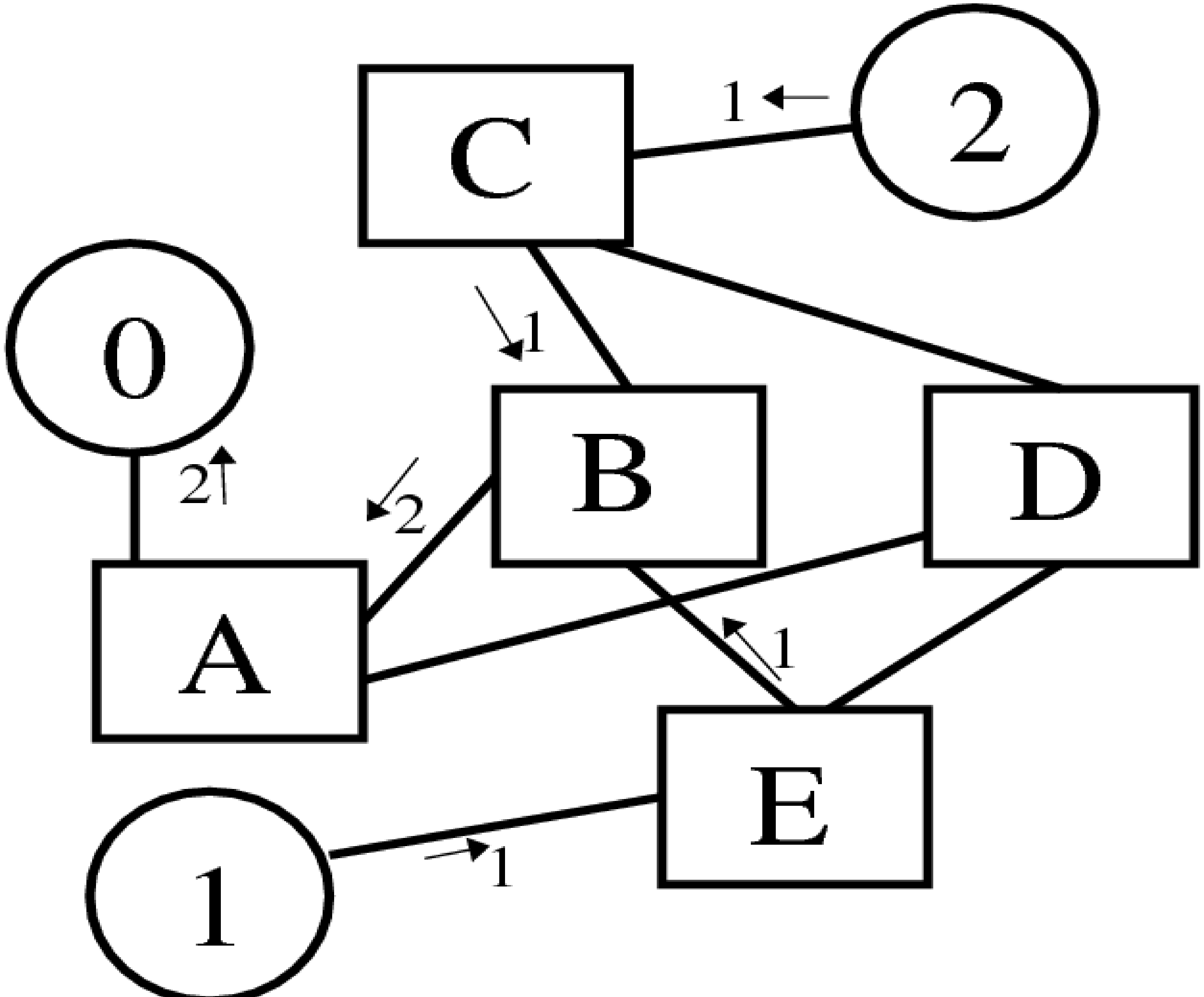
➢ goal is to find R that minimizes $\pi(G,R)$
  - shown to be NP-hard in the general case

Two heuristics based on SSSP

- we propose two heuristics:
  - P-SSSP
  - P^2-SSSP
- P-SSSP 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)

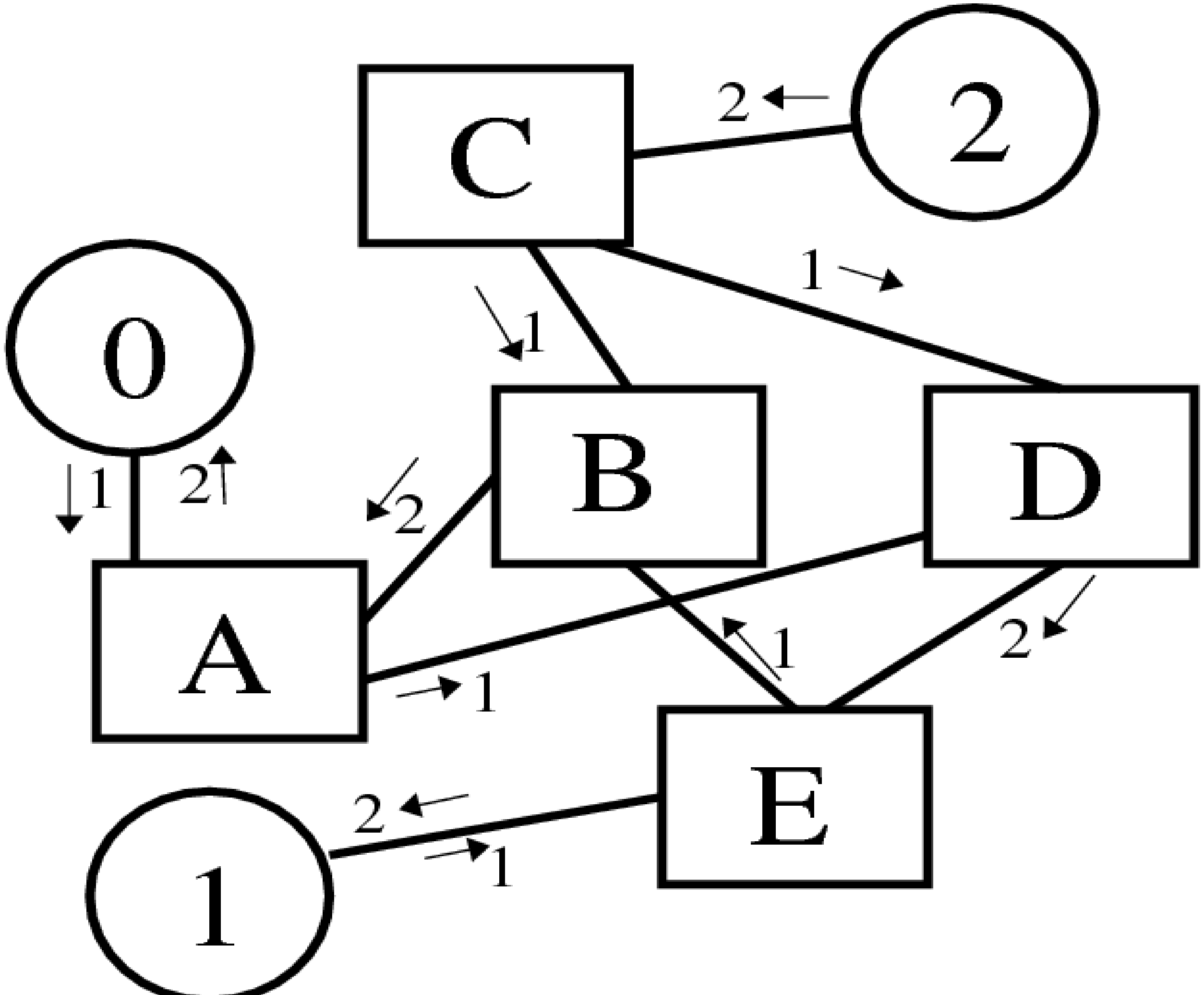
Step 1:
Source-node 0:



P-SSSP Routing (2/3)

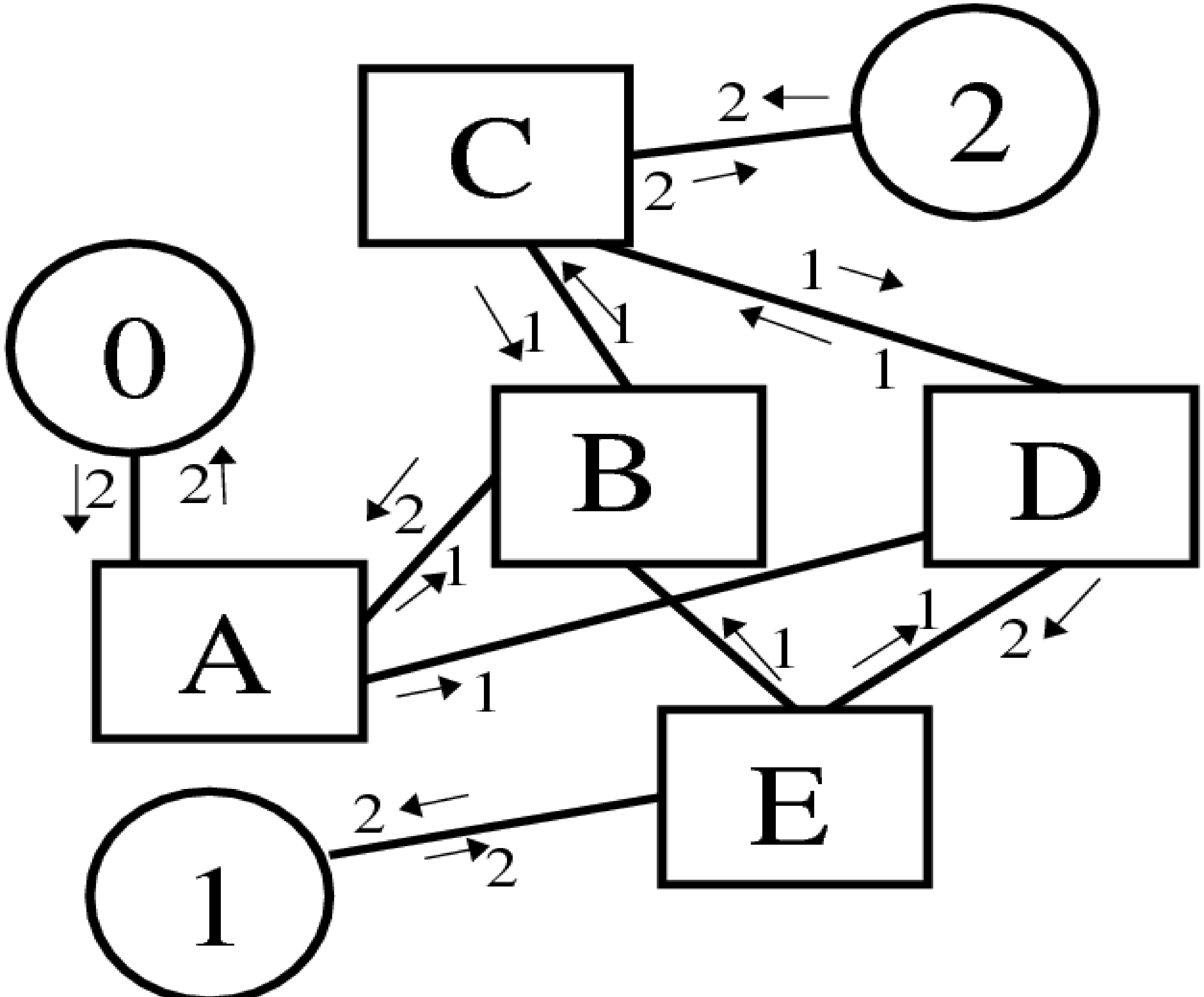
Step 2:

Source-node 1:



P-SSSP Routing (3/3)

Step 3:
Source-node 2:



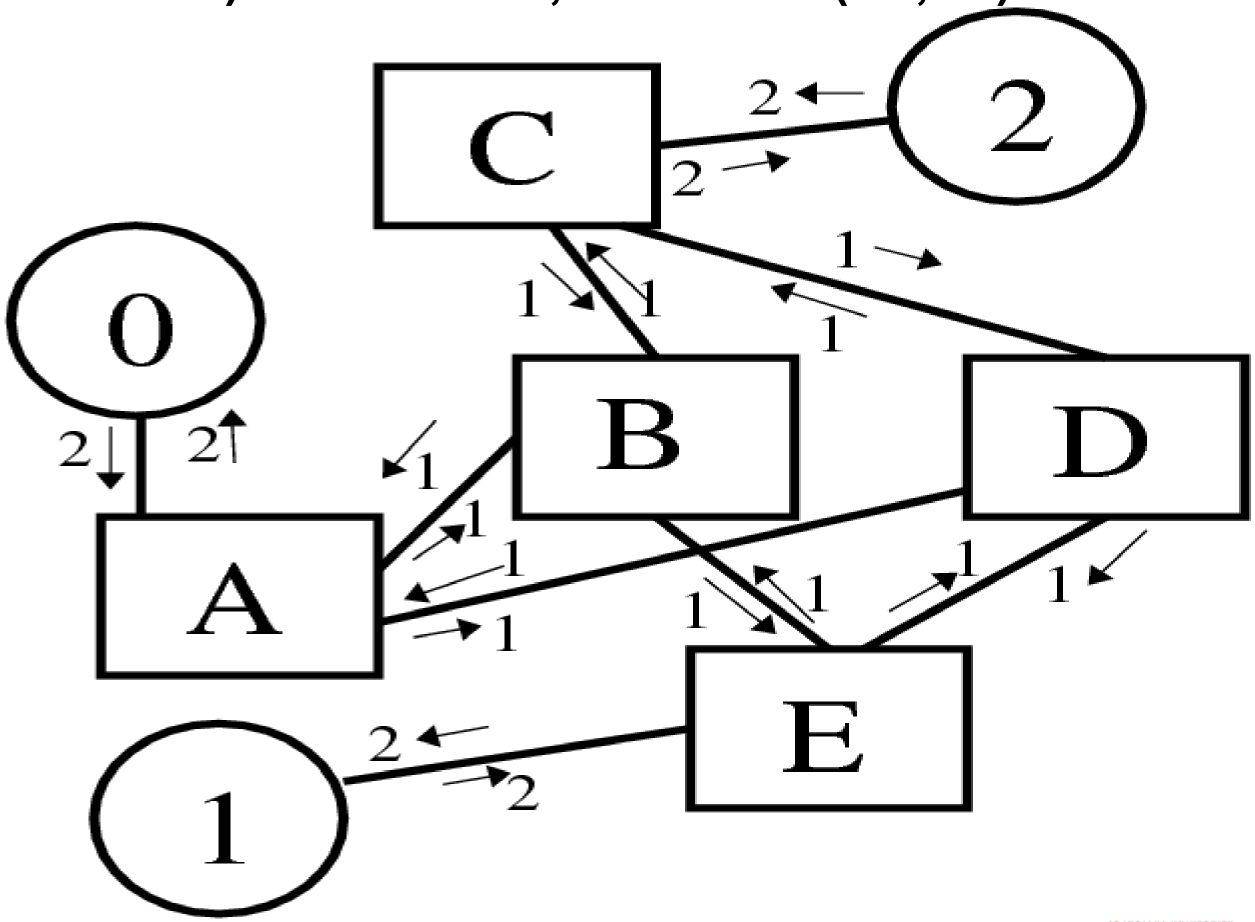
$\pi(G,R)=2$

pervasivetechnologylabs AT INDIANA UNIVERSITY

P^2-SSSP

- simply run a single SSSP for each route
  - better (expensive) heuristic, lower $\pi(G,R)$

$\pi(G,R)=1$



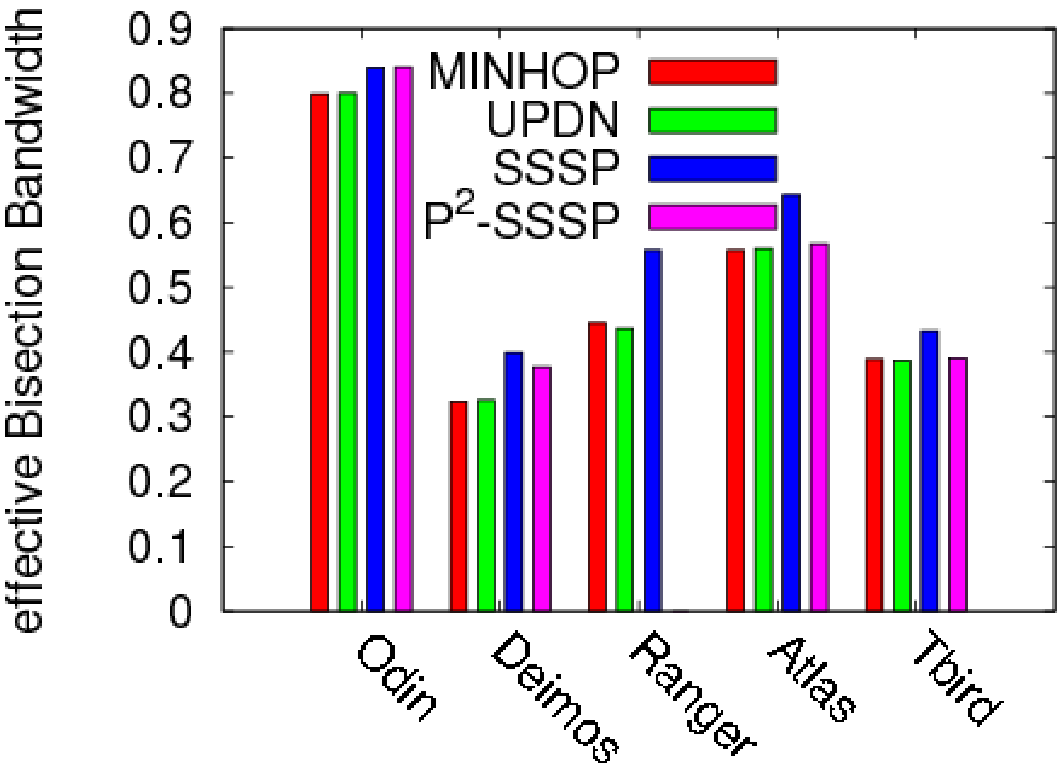
How to Assess a Routing?

- edge forwarding index is a trivial upper bound
- ability to route permutations is more important
  - bisect P into two equally-sized partitions
  - choose exactly one random partner for each node
  - Θ(P!/(P/2)!) combinations!
- our simulation approach:
  - pick N (=5000) random bisections/matchings
  - compute average bandwidth
  - shown to be rather precise (Cluster’08)

Comparison to Real Systems

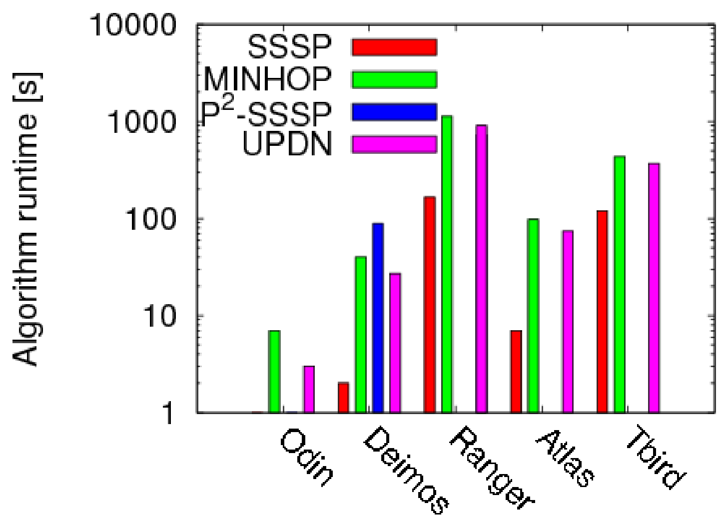
- `ibdiagnet`, `ibnetdiscover`, and `ibsim`
- we extracted topology and routing from:
  - Thunderbird (SNL) – 4390 LIDs
    - thanks to: Adam Moody & Ira Weiny
  - Ranger (TACC) – 4080 LIDs
    - thanks to: Christopher Maestas
  - Atlas (LLNL) – 1142 LIDs
    - thanks to: Len Wisniewsky
  - Deimos (TUD) – 724 LIDs
    - thanks to: Guido Juckeland and Michael Kluge
  - Odin (IU) – 128 LIDs

Real-world Results

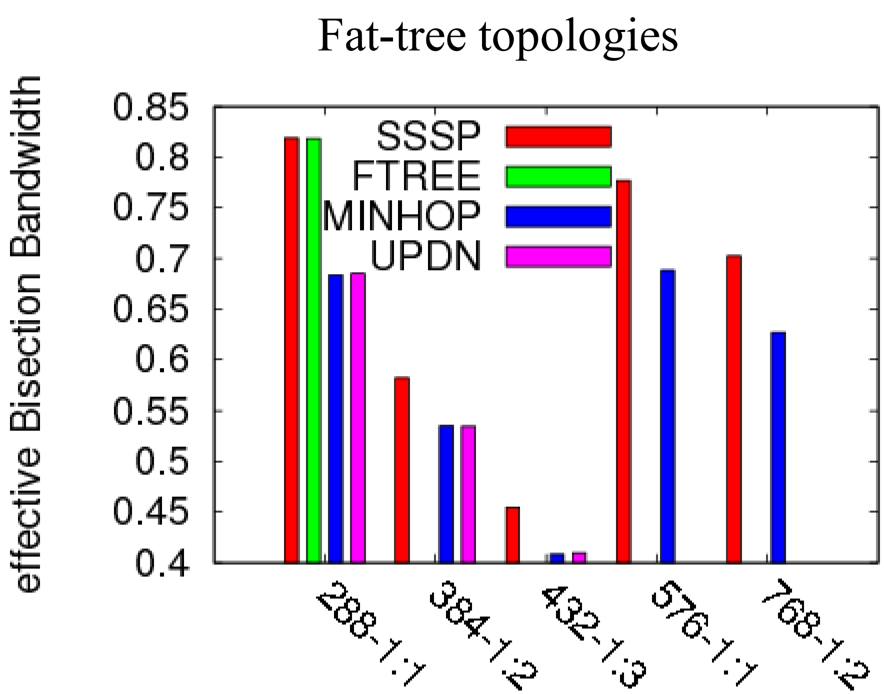


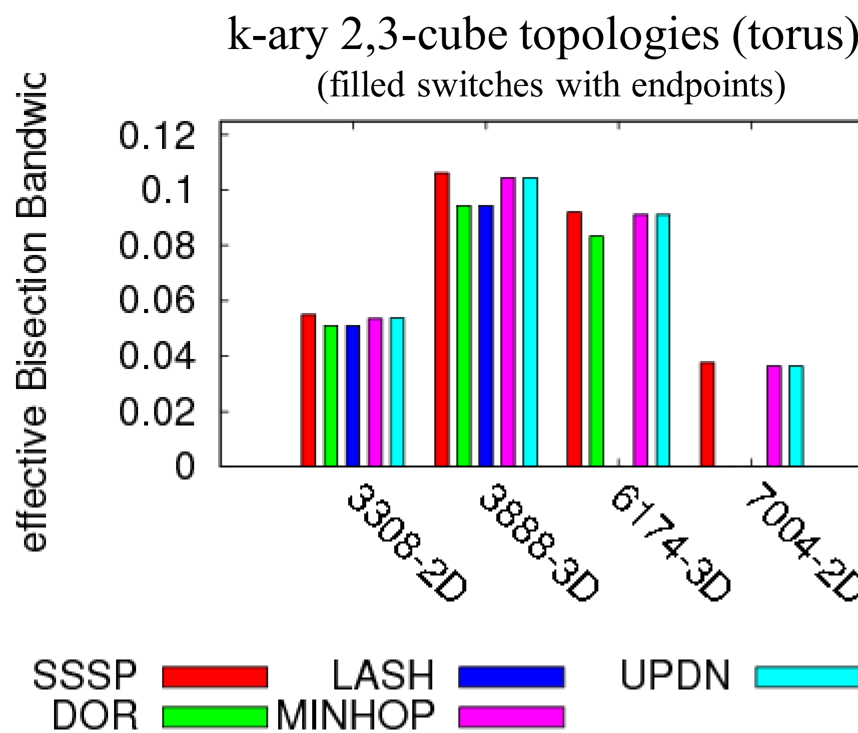
Real-World Bandwidth

Real-World Runtime



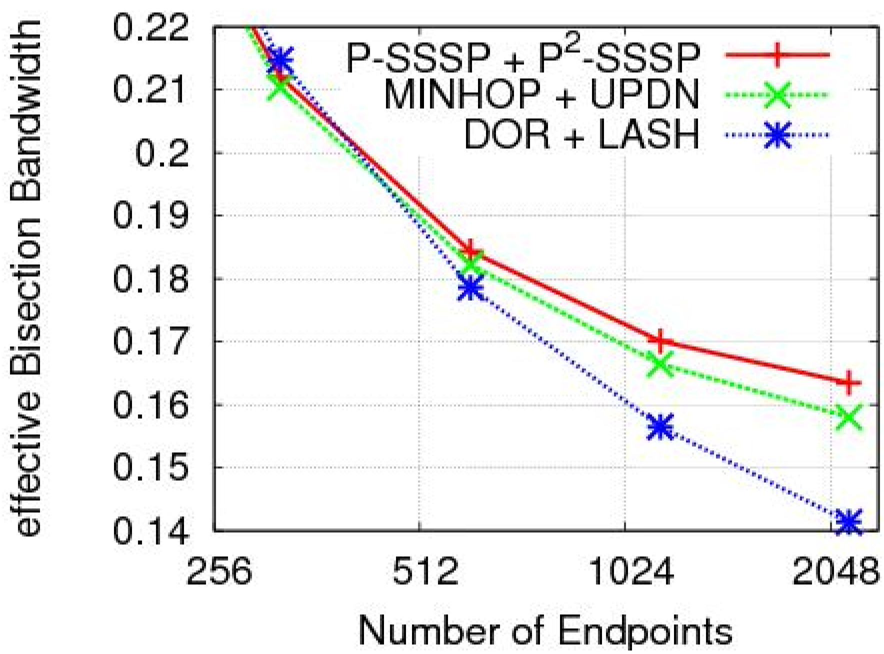
Some more Topologies



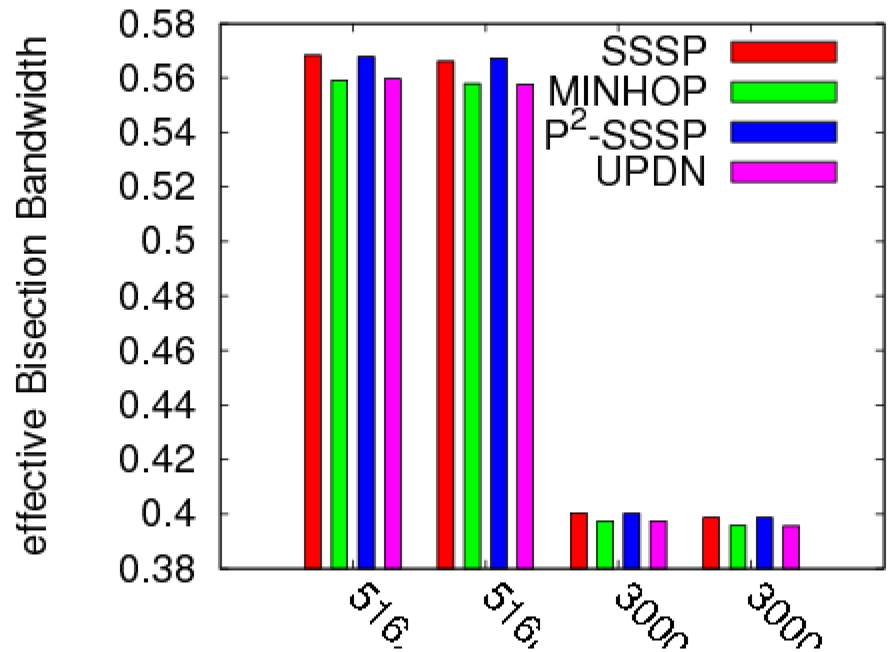


Even more Topologies

2-ary n-cube topologies (hypercube)
(filled switches with endpoints)



random topologies
(12 nodes per switch)



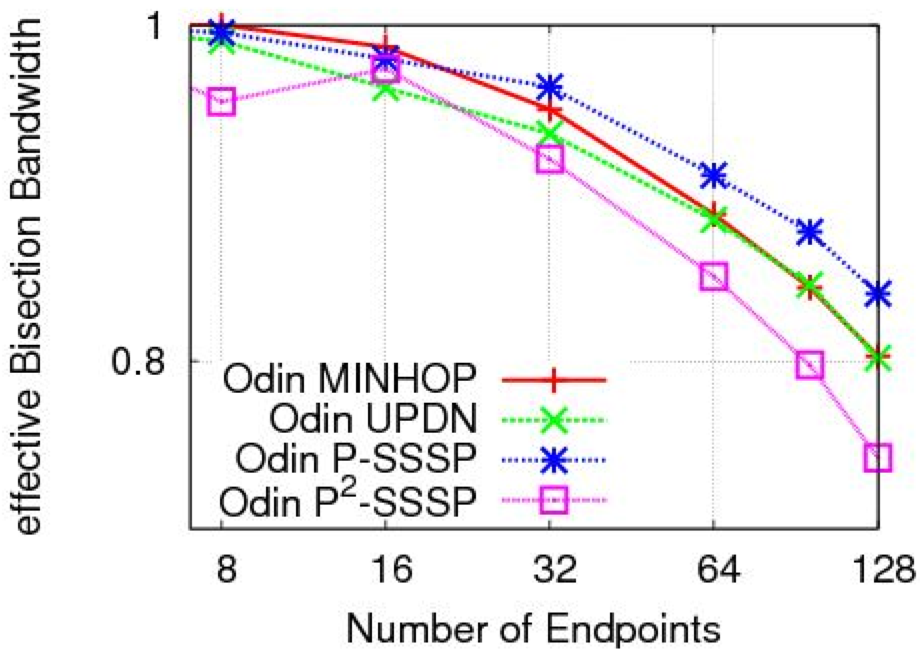
pervasivetechnologylabs
AT INDIANA UNIVERSITY

Simulations are good, but still Simulations

- we implemented our routing with OpenSM's file method
- tested it on the Deimos and Odin clusters (needs exclusive admin access to whole machine – many thanks to Guido Juckeland)
- Odin is standard fat-tree, Deimos' topology:

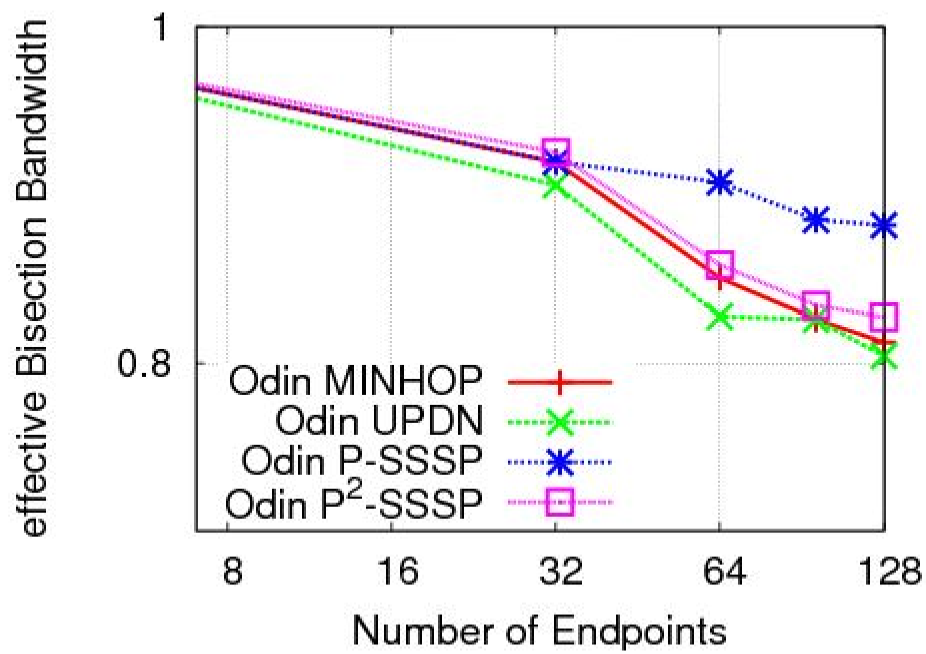
Benchmark Results Odin

Simulation



Simulation predicts 5% improvement

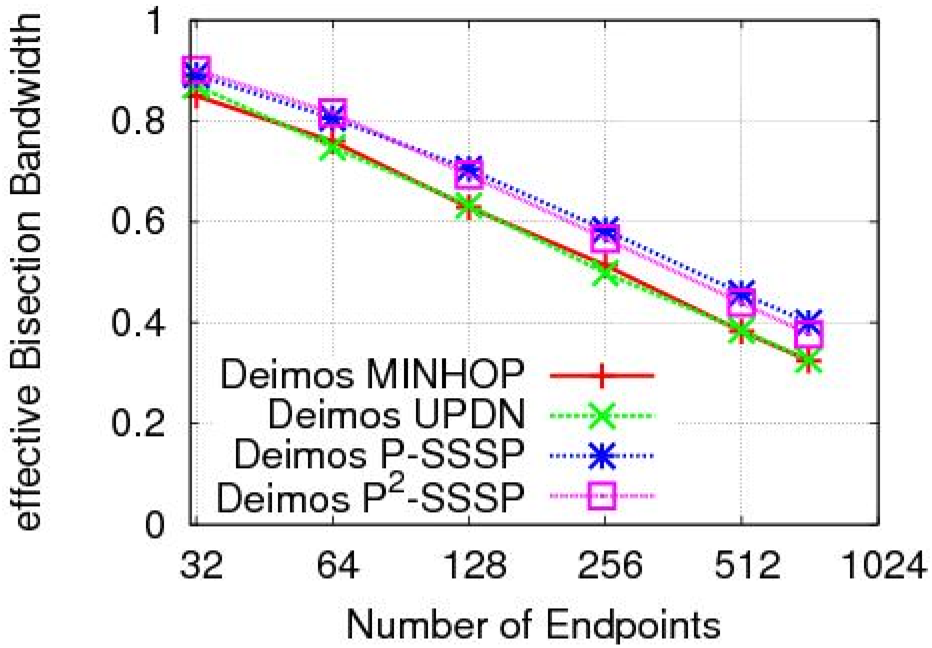
Benchmark
(Netgauge Pattern eBB)



Benchmark shows 18% improvement!

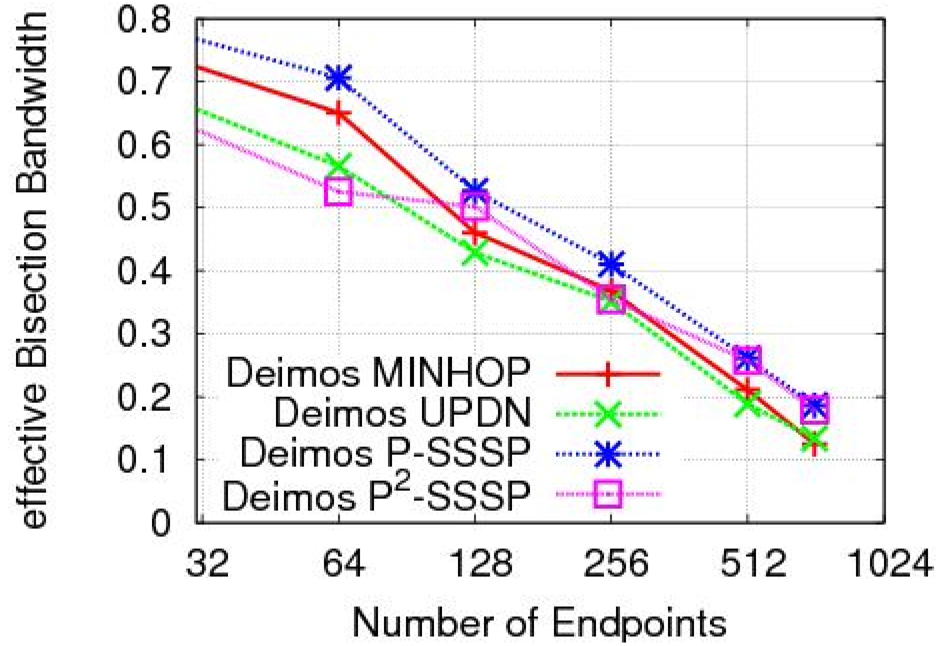
Benchmark Results Deimos

Simulation



Simulation predicts 23% improvement

Benchmark
(Netgauge Pattern eBB)



Benchmark shows 40% improvement!

Summing up and Future Work!

- we proposed two new routing heuristics for deterministic oblivious routing (IB)
- simulation shows increase in effective bisection bandwidth over standard OpenSM routing
  - e.g., Odin 5%, Deimos 23%, Atlas 15%, Thunderbird 6%
- benchmarks show even higher improvements
  - Odin 18%, Deimos 40%
- Credit-loops remain, but solution is obvious (LASH-like VL principle)

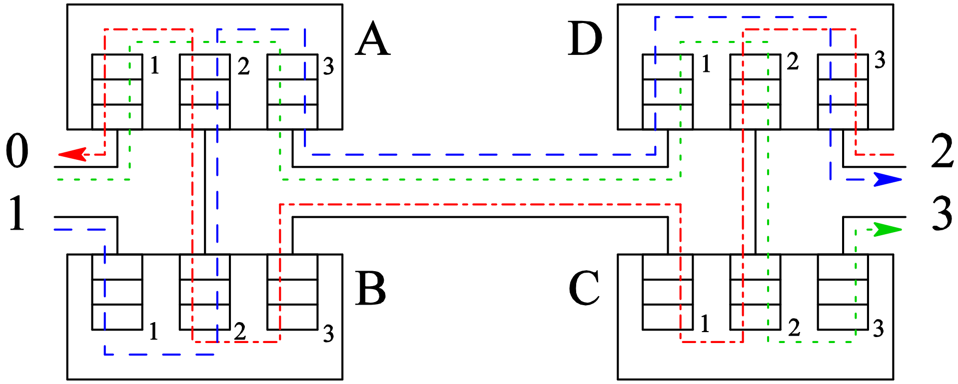
Reproduce our Results!

- talk to us!
- play with our ORCS simulator
  - http://www.unixer.de/ORCS
- benchmark your cluster (and talk to us)
  - Netgauge pattern “ebb”
  - http://www.unixer.de/research/netgauge
- ask questions – now!

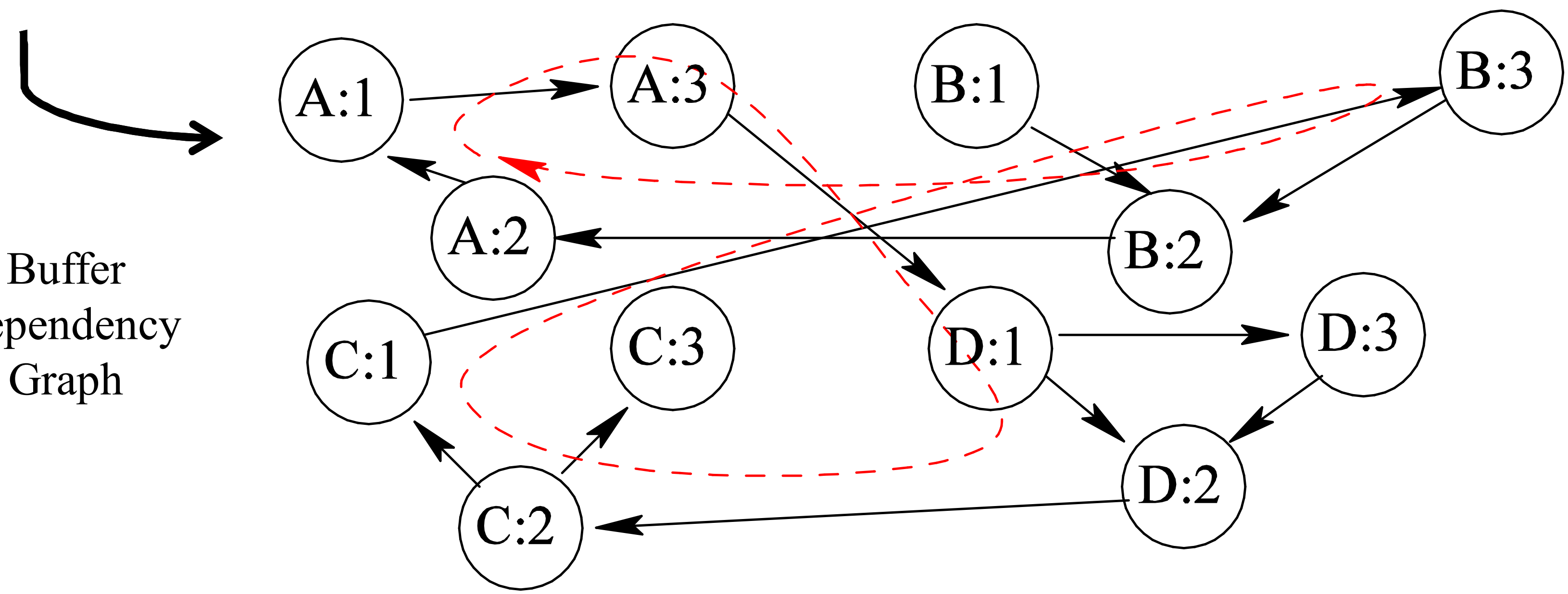
Backup Slides

Backup Slides

Credit Loops Continued …



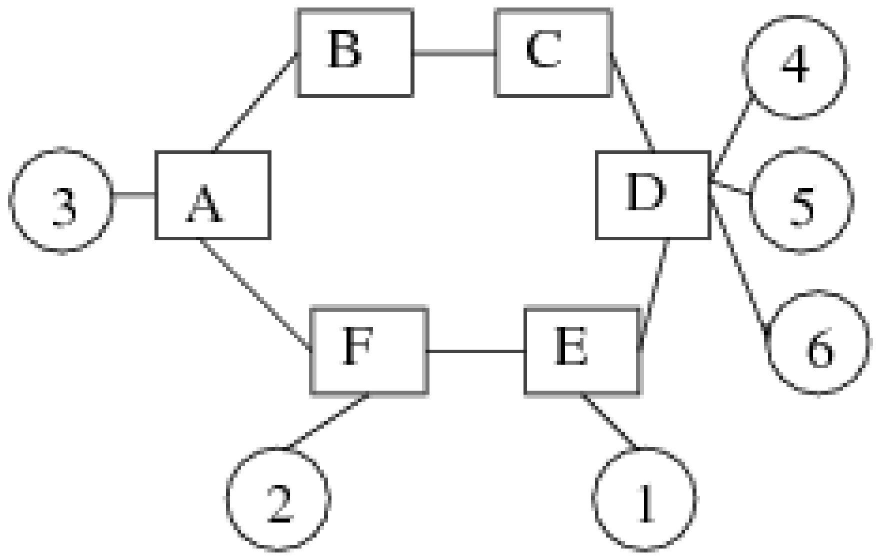
Source Network and Routes



Buffer Dependency Graph

Lower $\pi(G,R)$ and lower bandwidth!?

- Yes!
  - $\pi(G,R)$ is just an upper bound
  - example:



  - no worries, I will not explain it here (refer to article for details)