OPENFABRICS ALLIANCE

Adaptive Routing on CEE for HPC

Ronald P. Luijten / IBM Research - Zurich
16 March 2010 (Sonoma 2010 workshop)

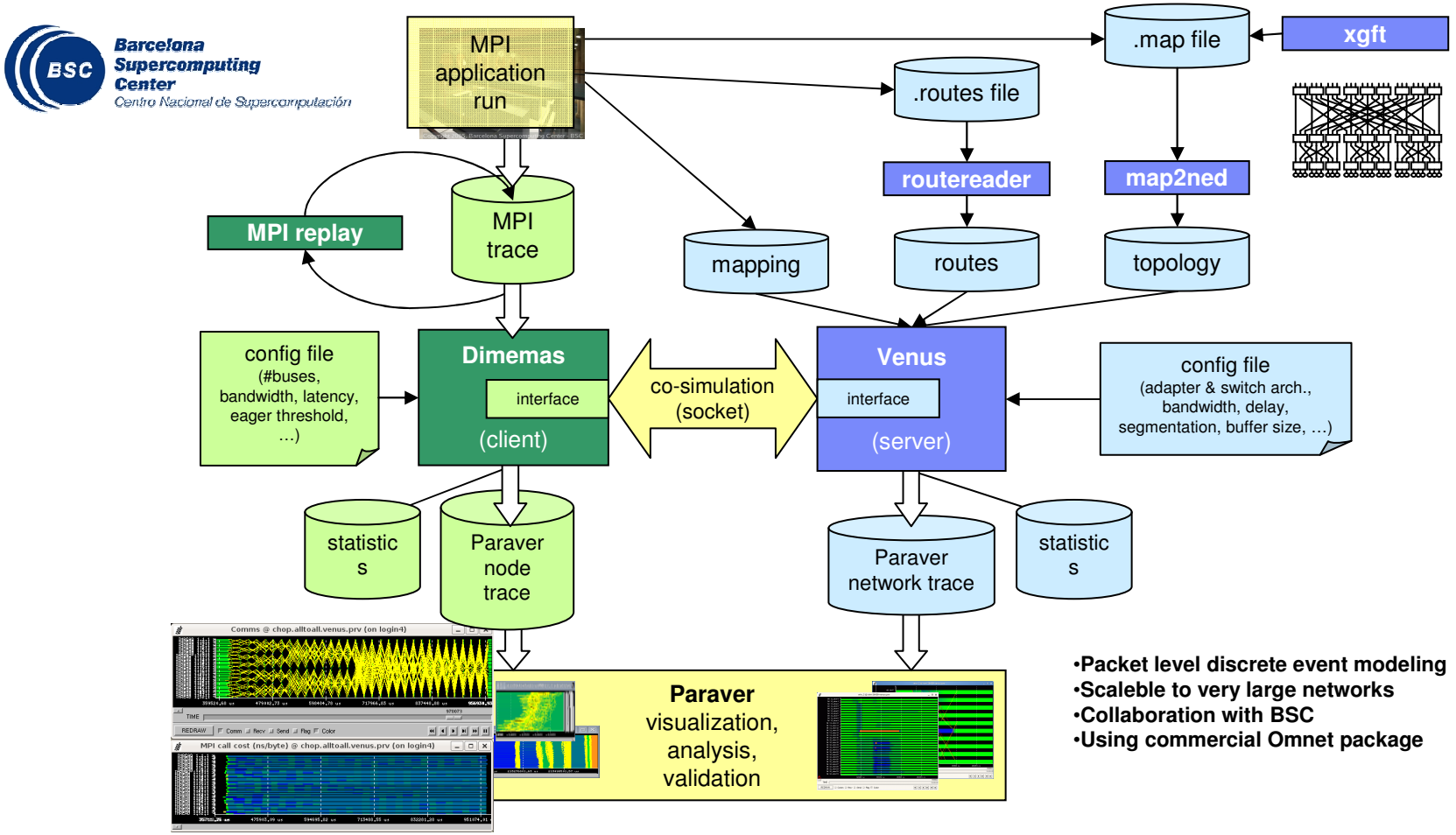
Why do we care about Routing

- Interconnects becoming ever more important towards exascale (SP, Sunday)
- The challenge of any interconnect is to solve a global problem with local information and decisions
  - Simple for internet traffic with independent arrivals
  - Not so simple for highly dependent traffic on HPC interconnects
- Routing potential has not been explored much
  - Can mitigate higher order blocking problems for multistage
  - We must put routing in our bag of tricks for HPC

The team…

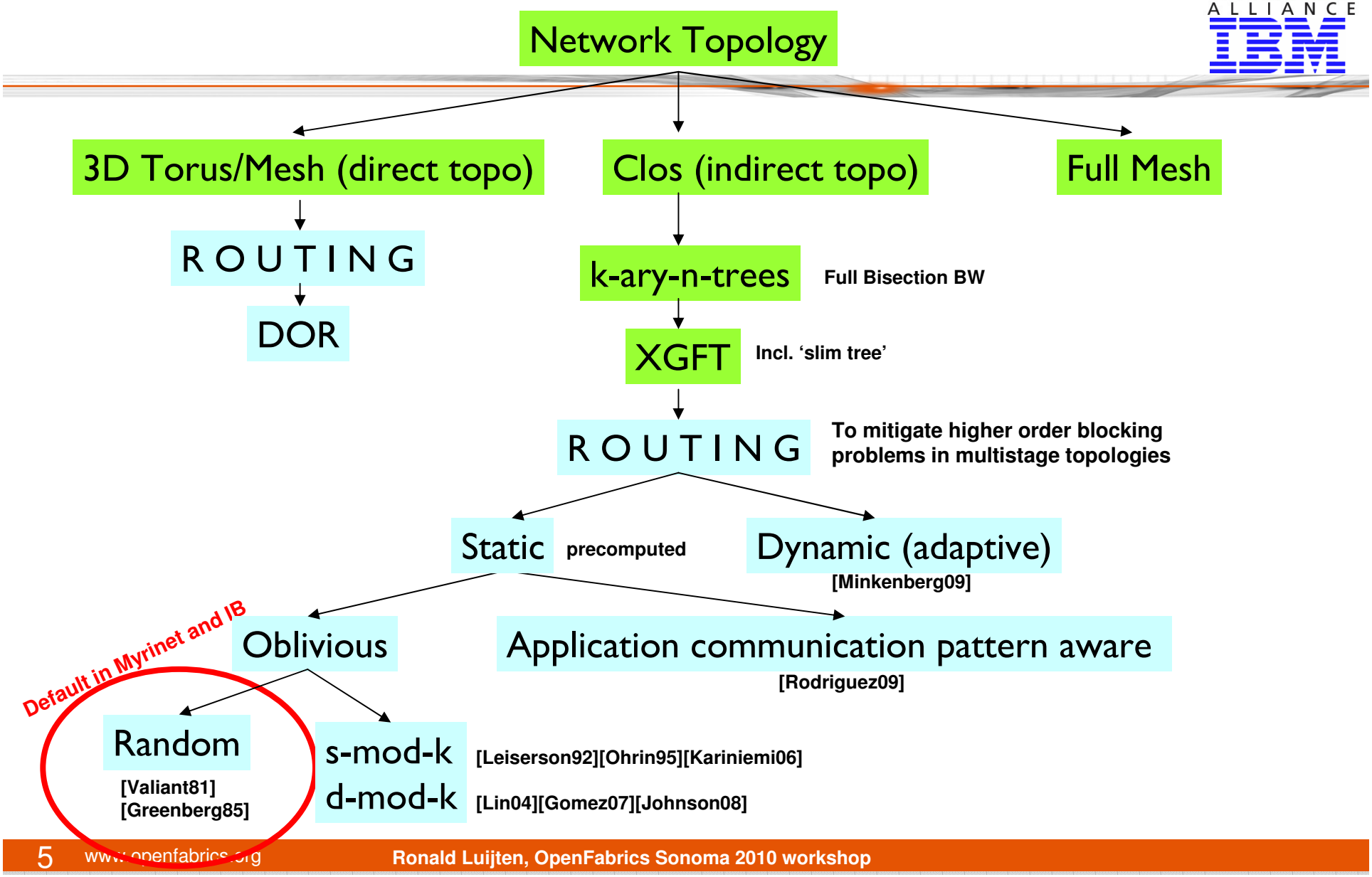

This presentation contains some charts from
Mitch Gusat, Cyriel Minkenberg, German Rodrıguez Herrera

Our simulation environment



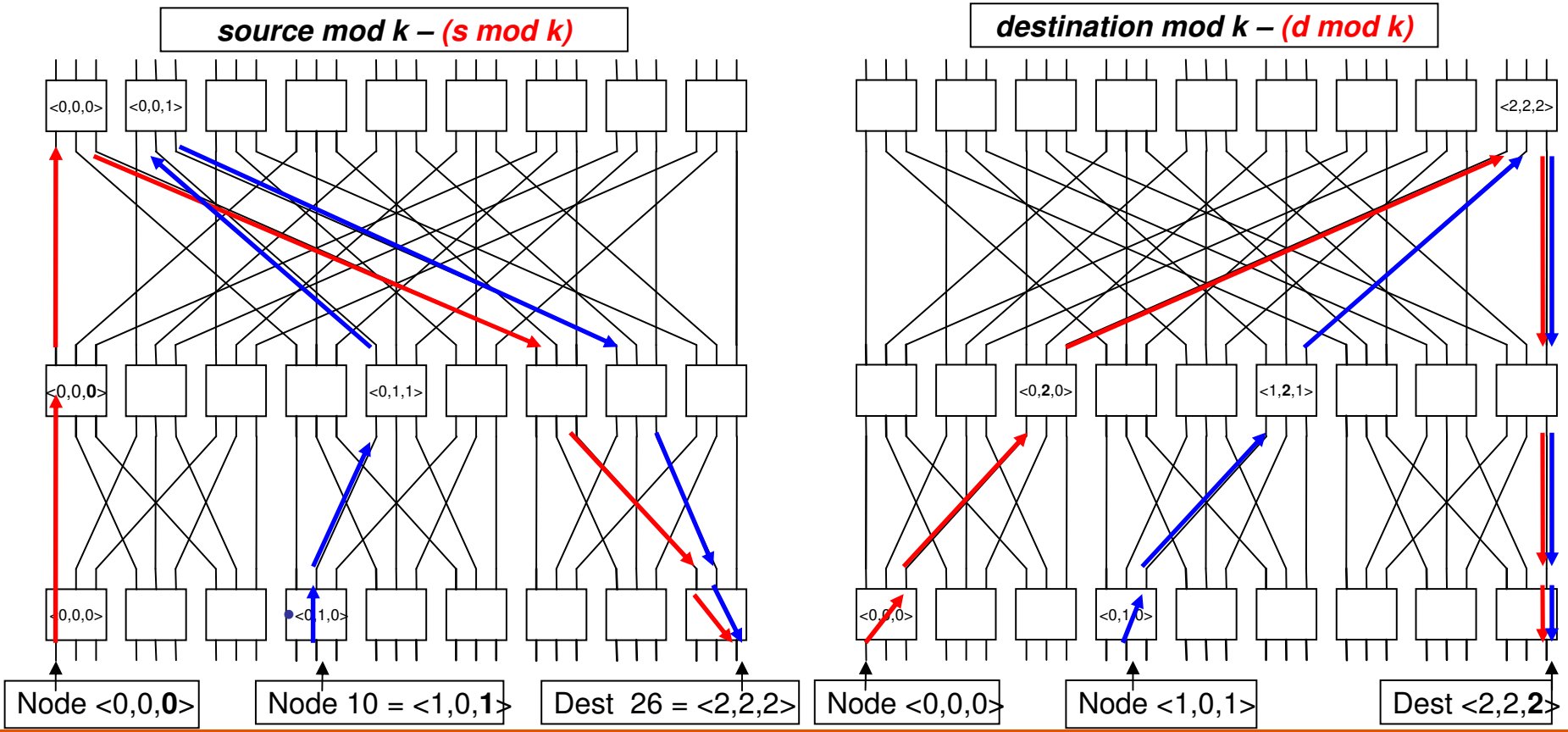
- **Packet level discrete event modeling**
- **Scaleble to very large networks**
- **Collaboration with BSC**
- **Using commercial Omnet package**

topologies and routing…



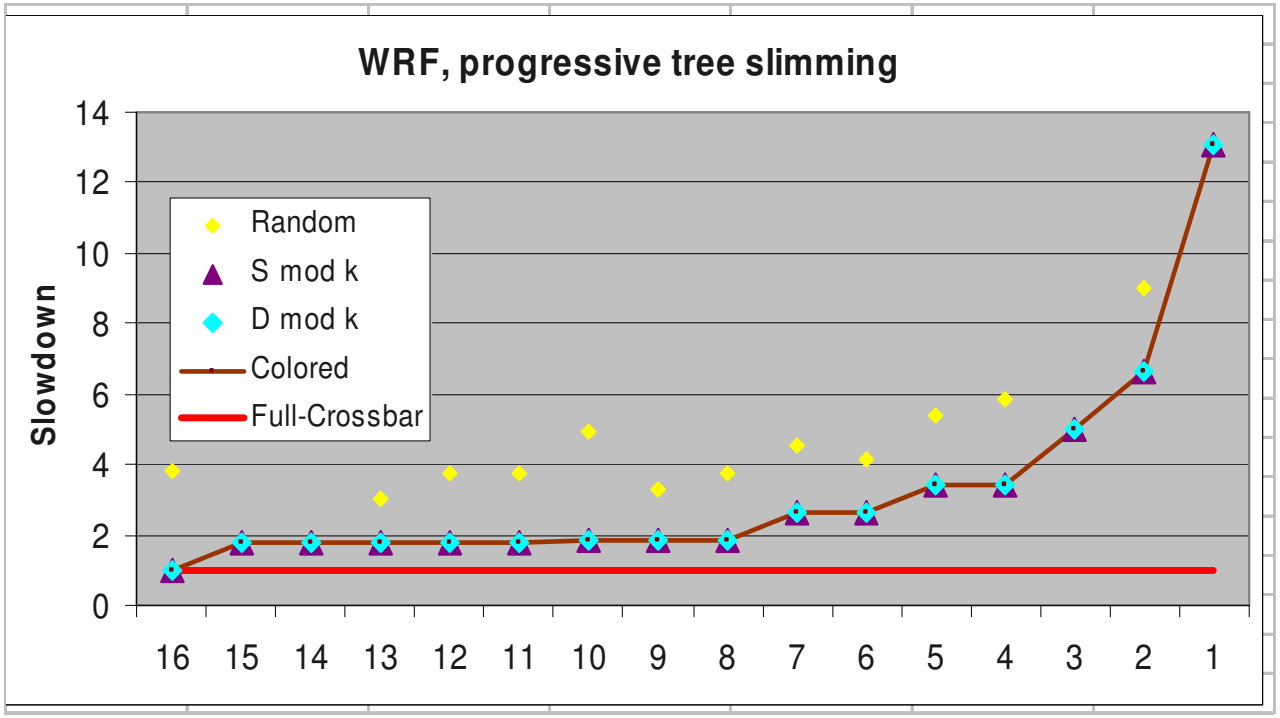
Oblivious, regular routing patterns

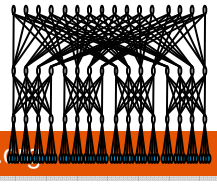
- "Self-routing" approach
  - At each step, choose the parent by getting doing a modulo operation (k)
  - Difference: The label of the source or destination is used to go up to the tree only

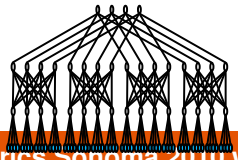


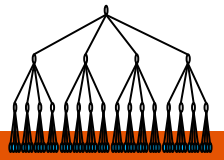
WRF app with Progressive tree slimming

- Removing a single switch degrades the performance by 2
  - Removing 7 more middle switches has no impact for 3 routing schemes
- Regular modulo routing work very well (as good as the baseline), while Random does not.



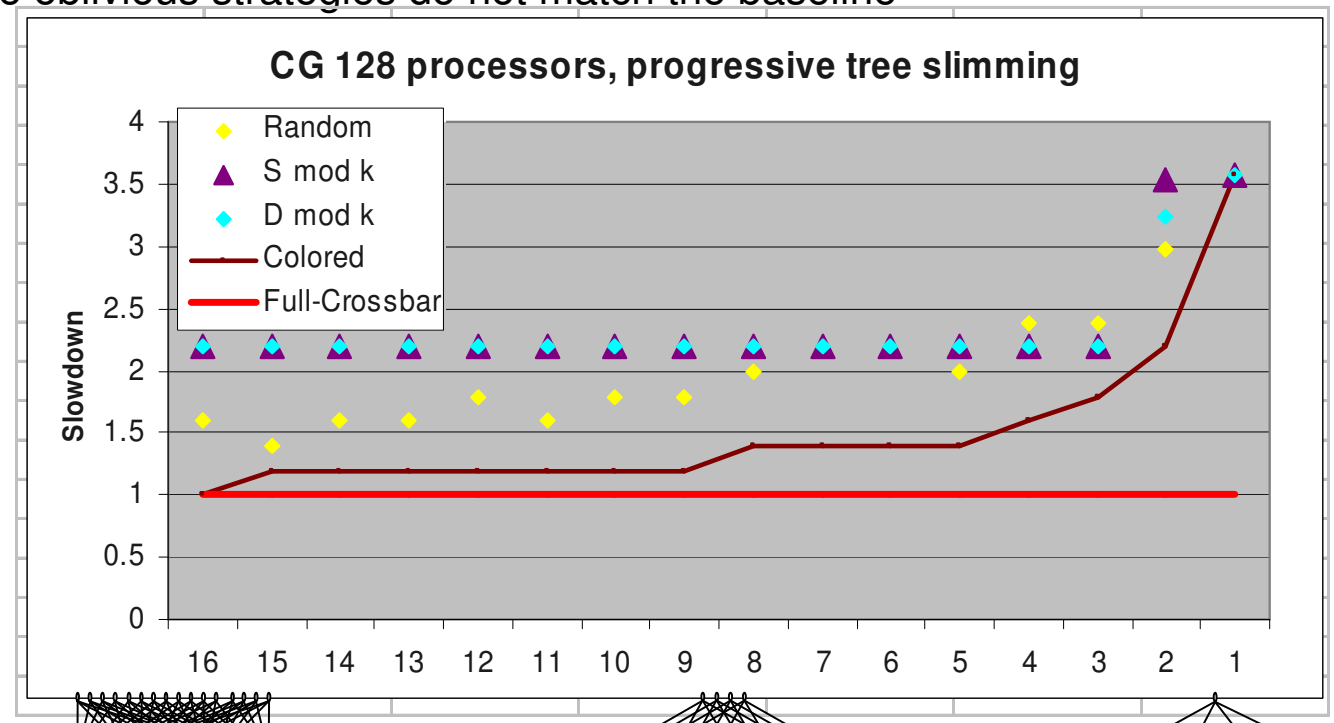






CG app with Progressive tree slimming

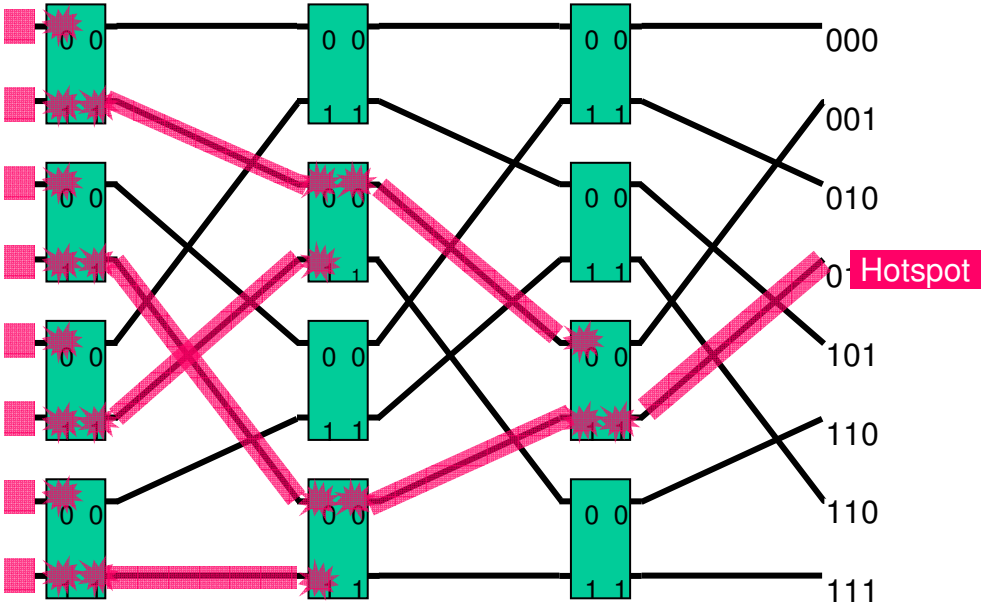
- Oblivious routings cannot achieve the best performance
  - It's a pathological case for modulo-based oblivious algorithms
  - Random routing does not achieve good performance
  - The oblivious strategies do not match the baseline

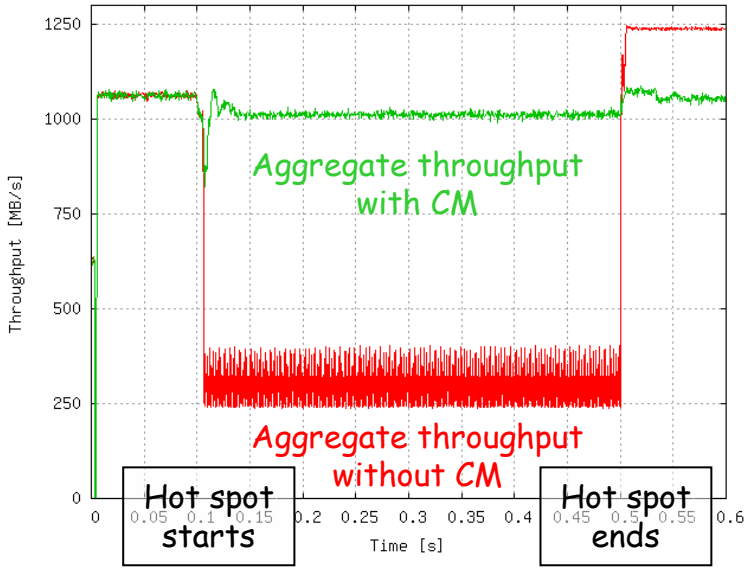


The need for L2 congestion management

- Network congestion can lead to severe performance degradation
  - Lossy networks suffer from the “avalanche” effect: High load ➔ drops ➔ retransmissions ➔ increased load ➔ even more drops
  - Lossless networks suffer from *saturation tree congestion:* Link-level flow control (LL-FC) can cause congestion to roll back from switch to switch

- Congestion management (CM) is needed to deal with long-term (sustained) congestion
  - Dropping and LL-FC are ill suited to this task, dealing only with short-term (transient) congestion
  - Push congestion from the core towards the edge of the network



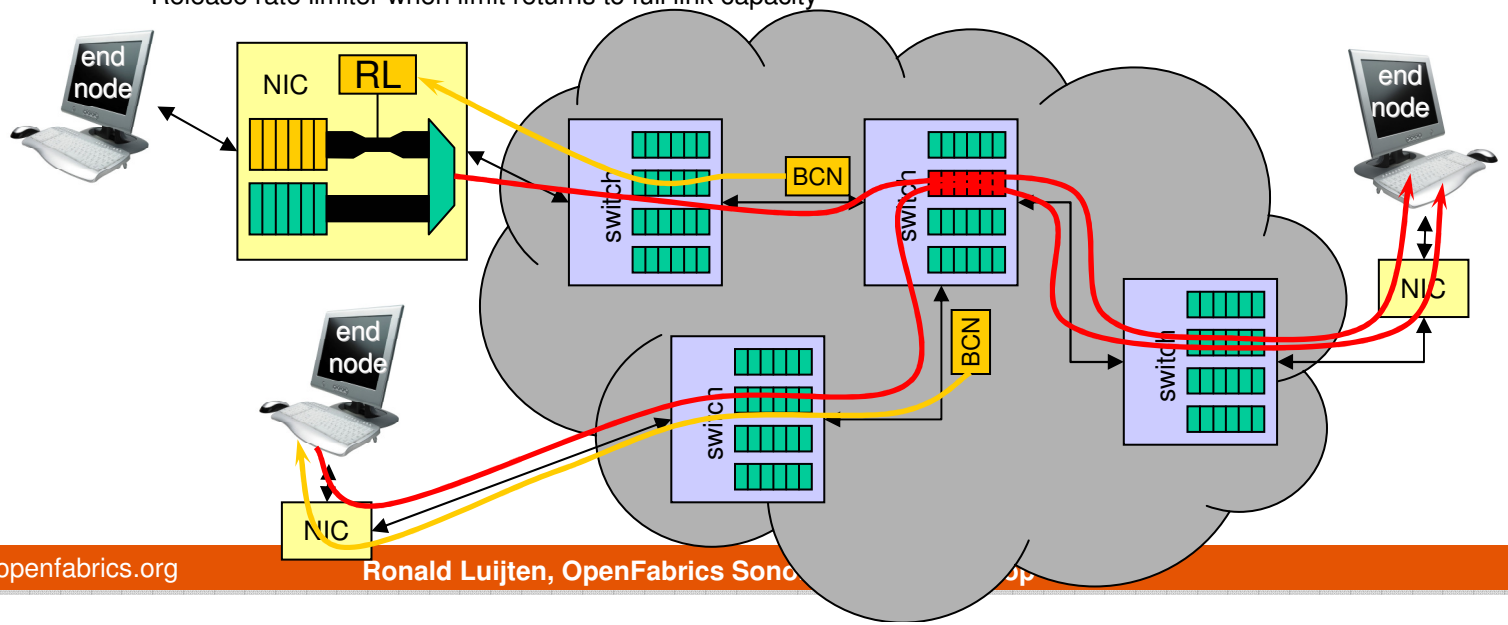


IEEE 802.1Qau congestion management framework

1. Congestion point (CP) = switch output queue
   - Sample output queue length every *n* bytes received
   - Equilibrium length Q_{eq}
   - Compute feedback: $F_b = Q_{off} - W*Q_{delta}$, where $Q_{off} = Q_{eq} - Q_{now}$ and $Q_{delta} = Q_{old} - Q_{now}$

2. Feedback channel
   - Convey backward congestion notifications (BCN) from CP to sources of "offending" traffic
   - Notifications contain congestion information: source address = switch MAC, destination address = source MAC of sampled frame, feedback: Q_{offset} and Q_{delta} , quantized with respect to $F_{b,max} = (1+2W)*Q_{eq}$ using 6-8 bits

3. Reaction point (RP)
   - Use rate limiters (RL) at the edge to shape flows causing congestion; separately enqueue rate-limited flows
   - Reduce rate limit multiplicatively when $F_b < 0$
   - Autonomously increase rate limit based on byte counting or timer (QCN; similar to BIC-TCP)
   - Release rate limiter when limit returns to full link capacity



Congestion management vs. adaptive routing

- CM solves congestion by reducing injection rate
  - Useful for saturation tree congestion, where many “innocent” flows suffer because of backlog of some hot flows
  - Does not exploit path diversity
  - Typical data center topologies offer high path diversity
    - Fat tree, mesh, torus

- Adaptive routing (AR) approach
  - Allow multi-path routing
  - By default route on shortest path (latency, d-mod-k)
  - Detect downstream congestion by means of BCN
  - In case of congestion
    - First try to reroute hot flows on alternative paths
    - Only if no uncongested alternative exists, reduce send rate

Adaptive routing based on 802.1Qau CM

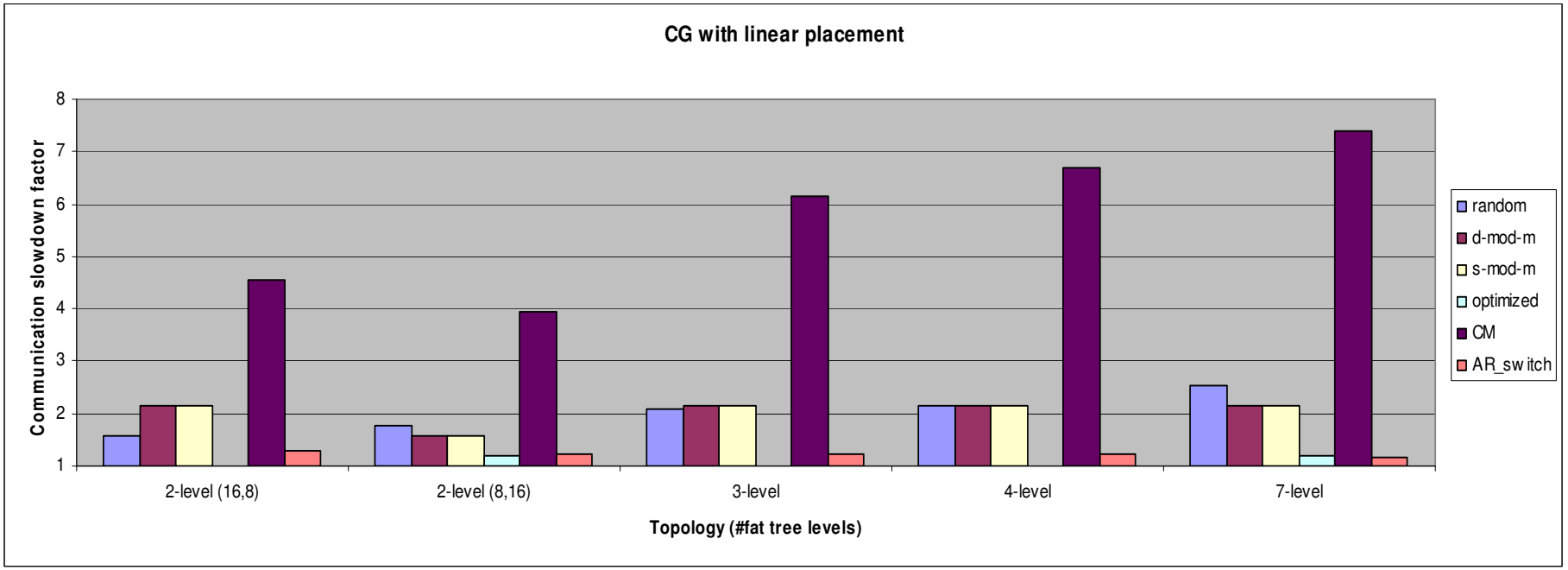
- Concept
  - Upstream switches **snoop** congestion notifications,
  - **annotate** routing tables with congestion information, and
  - **modify routing decisions** to route to the **least congested** port among those enabled for a given destination
- Routing table
  - Maps a destination MAC to one or more switch port numbers, listed in order of preference, e.g., shortest path first
- Congestion table
  - Maps a key <destination MAC, switch port number> to a congestion entry comprising the following information:
- Receiver checks frame order and performs resequencing if needed

field	type	meaning
congested	boolean	Flag indicating whether port is congested
local	boolean	Flag indicating whether congestion is local or remote
fbCount	integer	Number of notifications received
feedback	integer	Feedback severity value

Simulated applications

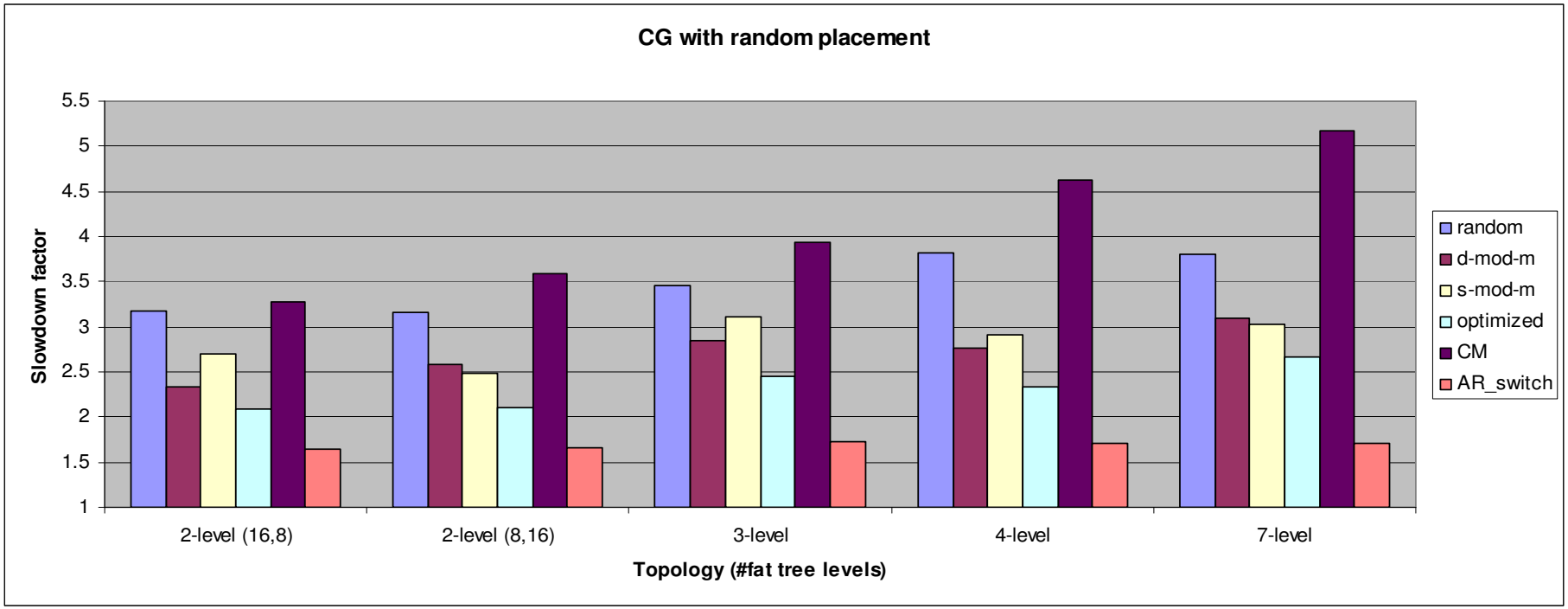
- Simulated two applications from the NAS Parallel Benchmarks
  - Conjugate Gradient (CG), 128 nodes
  - Fast Fourier Transform (FT), 128 nodes
  - Characterized by heavy network loading during communication phases
    - Large messages (CG: 750 KB, FT: 130 KB)
    - Multiple sub-phases (CG: 5, FT: 127)
    - Symmetric permutation patterns in each sub-phase (each node sends to a distinct peer node in a pairwise fashion)
- Simulation methodology
  - Collect MPI trace of application run on real machine (MareNostrum)

Results – CG with linear task placement



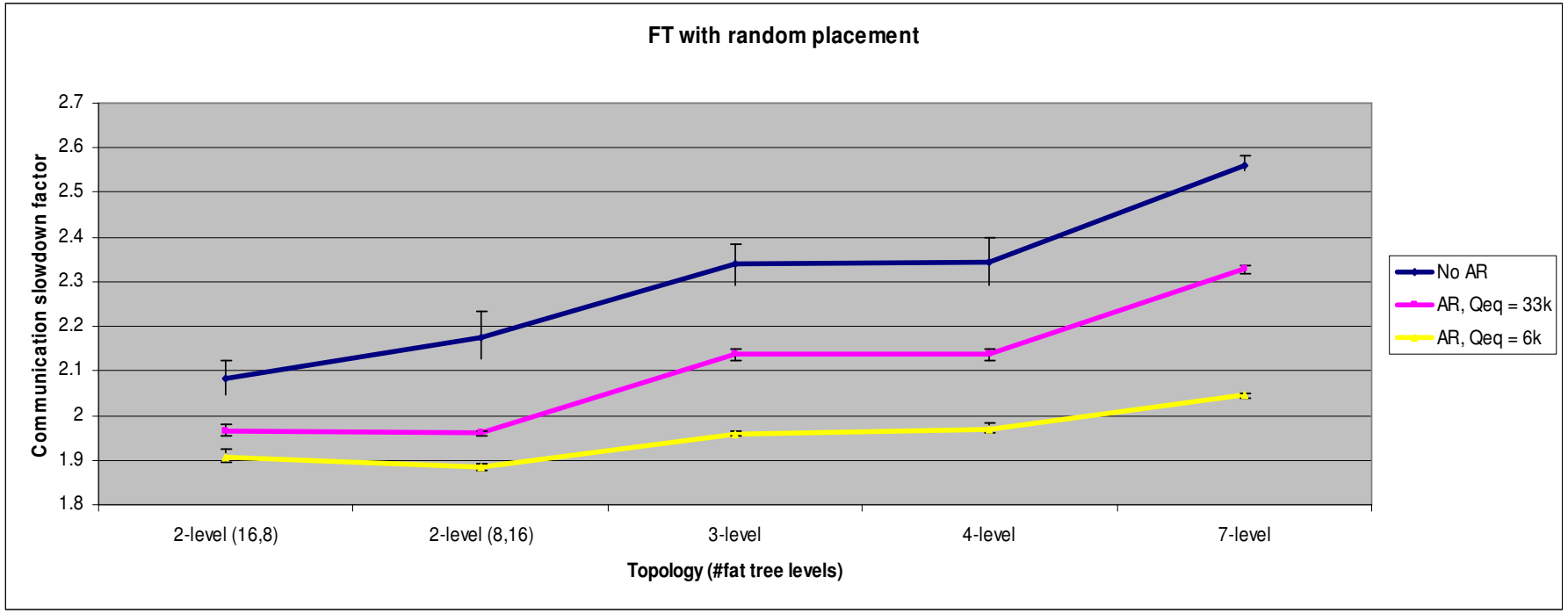
- CG with linear task placement
  - Random, d-mod-, s-mod-m all perform about equally well
  - Colored ('optimized') is almost ideal
  - CM by itself is terrible: rate limiting online delays without solving congestion
  - AR performs significantly better than oblivious algorithm (25 to 45% better than d-mod-m; within 25% of ideal)

Results – CG with random task placement



- CG wih random task placement
  - CM again performs worst
  - Random clearly worse than oblivious schemes
  - AR performs best; even better than offline optimization

Results – FT with random task placement



- FT with linear task placement is entirely contention-free (d-mod-k routing)
- FT with random task placement
  - Five different random placements per data point; error bars indicate min/max
  - Smaller Q_{eq} performs better; about 10 to 25% better than with d-mod-k

Bonus material

- AR on CEE makes most sense when
  - Leveraging TRILL
  - Leveraging RDMA (to deal with out of order delivery)
  - (don't go L3!)
- AR on CEE Route selection when congested path detected is round robin

Conclusions

- Thou shalt model and simulate
- Exploiting routing for HPC is a good idea
- Default random routing on IB is a bad idea
- Using 802.1Qau congestion management for HPC apps is a bad idea
- Using 802.1Qau hooks to do adaptive routing is a good idea

literature

- **"A Framework for End-to-end Simulation of High Performance Computing Systems",** Wolfgang E. Denzel, Jian Li, Peter Walker, Yuho Jin, Simutools conference, March 2008

- **"Trace-driven Co-simulation of High-Performance Computing Systems using OMNeT++",** Cyriel Minkenberg, Germán Rodriguez Herrera, Simutools Conference, March 2009

- **"Oblivious Routing Schemes in Extended Generalized Fat Tree Networks",** German Rodriguez, Cyriel Minkenberg, Ramon Beivide, Ronald P. Luijten, Jesus Labarta, Mateo Valero, HPIDC 2009

- **"Adaptive Routing in Data Center Bridges",** Cyriel Minkenberg, Mitchell Gusat, German Rodrıguez Herrera, HOTi Conference, NY, Aug 2009