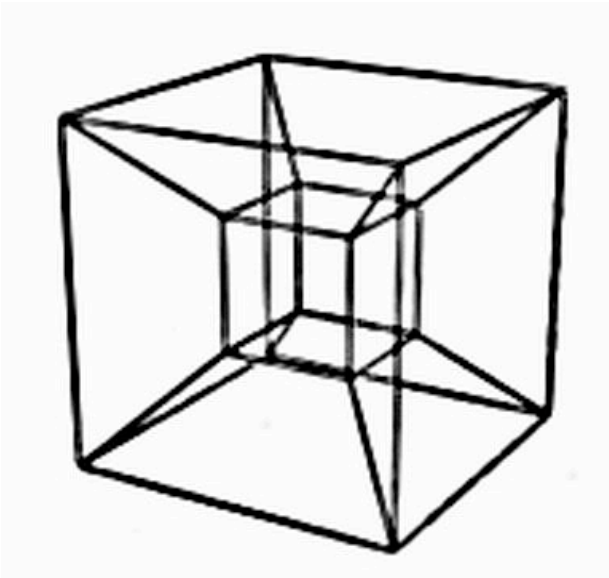


From Hypercubes to Dragonflies

a short history of interconnect

William J. Dally
Computer Science Department
Stanford University

IAA Workshop

July 21, 2008

Outline

- The low-radix era
- High-radix routers and networks
- To ExaScale and beyond
- NoCs the final frontier

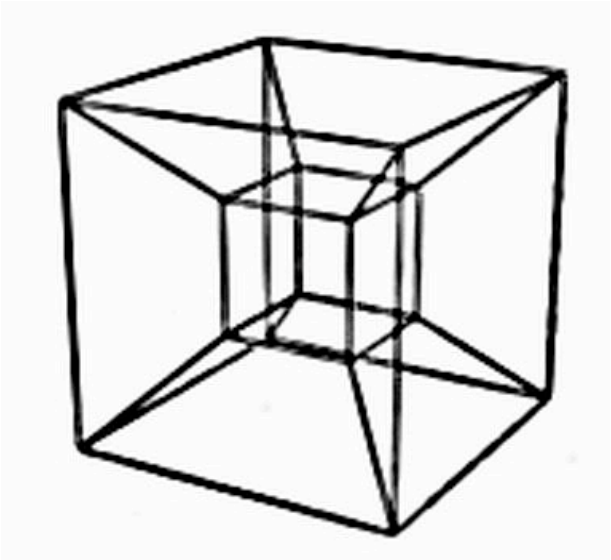
Partial Timeline

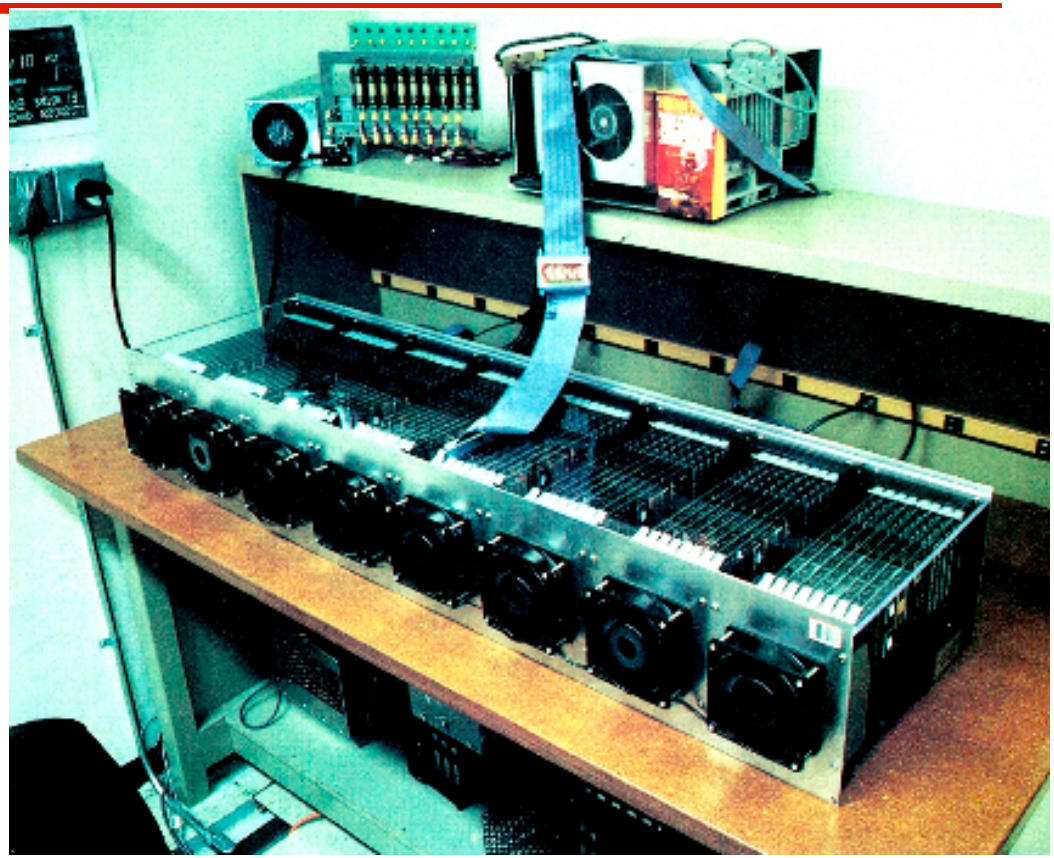
Date	Event	Features
1983	Caltech Cosmic Cube	Hypercube, programmed transfer
1985	Torus Routing Chip	torus, routed, wormhole, virtual channels
1987	iPSC/2	routed hypercube
1990		Virtual-channel flow control
1991	J-Machine	
1992	Paragon, T3D, CM5	
1994	Vulcan	
1995	T3E	
1996	Reliable Router	Link level retry
2000		NoCs
2001	SP2, Quadrics	
2002	X1	
2004		Global adaptive routing
2005		High-Radix Routers
2006	YARC/BW	

Low-Radix Era

The Low-Radix Era

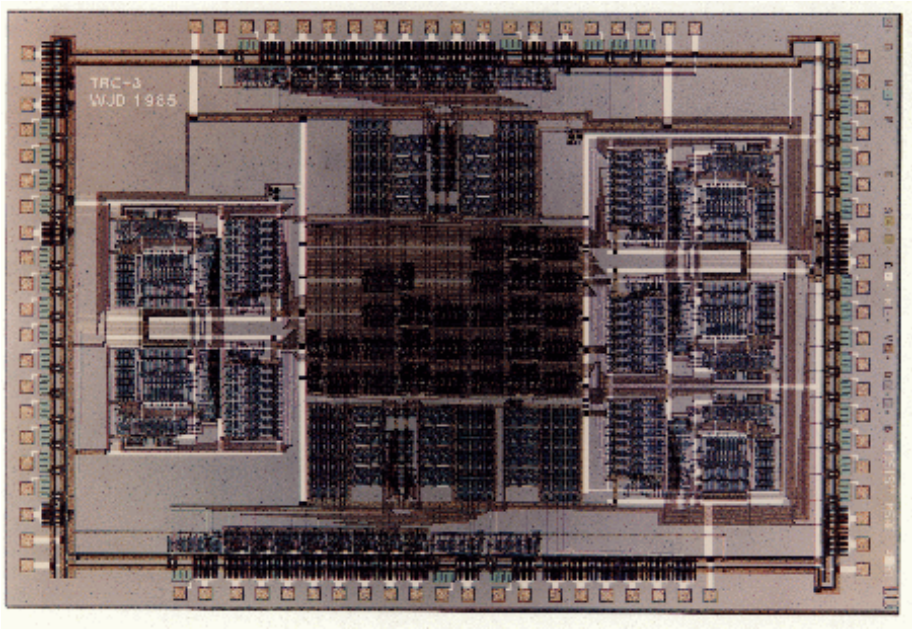
The Cosmic Cube





- Caltech 1983
- Hypercube topology
- No routers – programmed transfers for every hop
- Store and forward: T = L/B x H

Torus Routing Chip



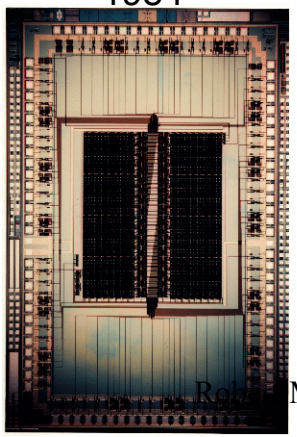
- Caltech 1985, 3μm CMOS
- Torus topology
  - Topology driven by technology constraints (pins, bisection)
- Wormhole routing: T = L/B + H
- Virtual channels to break deadlock

1985-2004
The Low-Radix Era

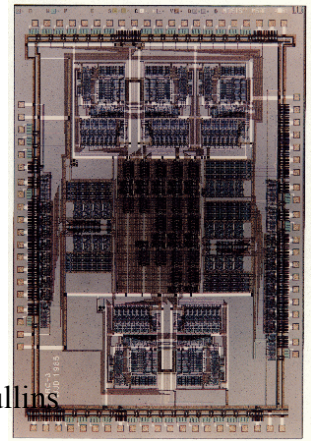
- Low-radix ($4 \leq k \leq 8$) routers
- Torus, mesh, or Clos (fat-tree) topologies
- Wormhole or virtual-cut-through flow control
- Virtual channels for deadlock avoidance and performance
- Almost exclusively minimal routing
- Delta, Paragon, T3D, SP-1, CM-5, T3E, SP-2, …
- … but router bandwidth was increasing exponentially

Some Routers

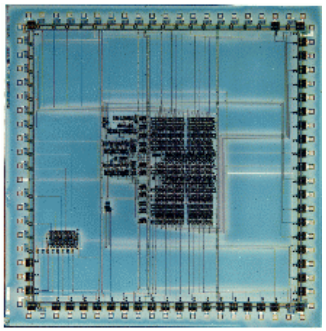
MARS Router
1984



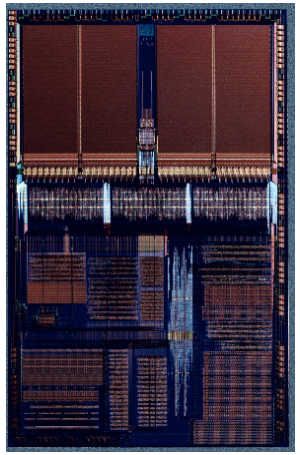
Torus Routing Chip
1985



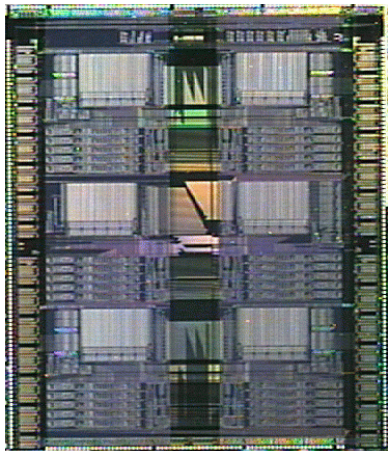
Network Design Frame
1988



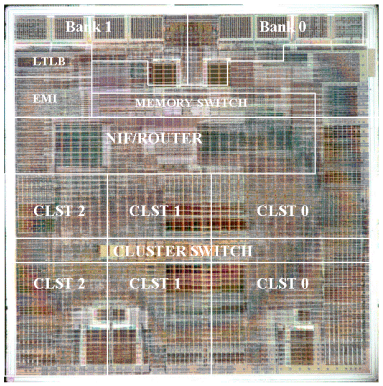
MDP 1991



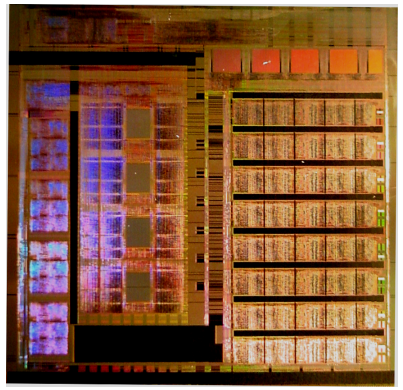
Reliable Router
1994



MAP 1998



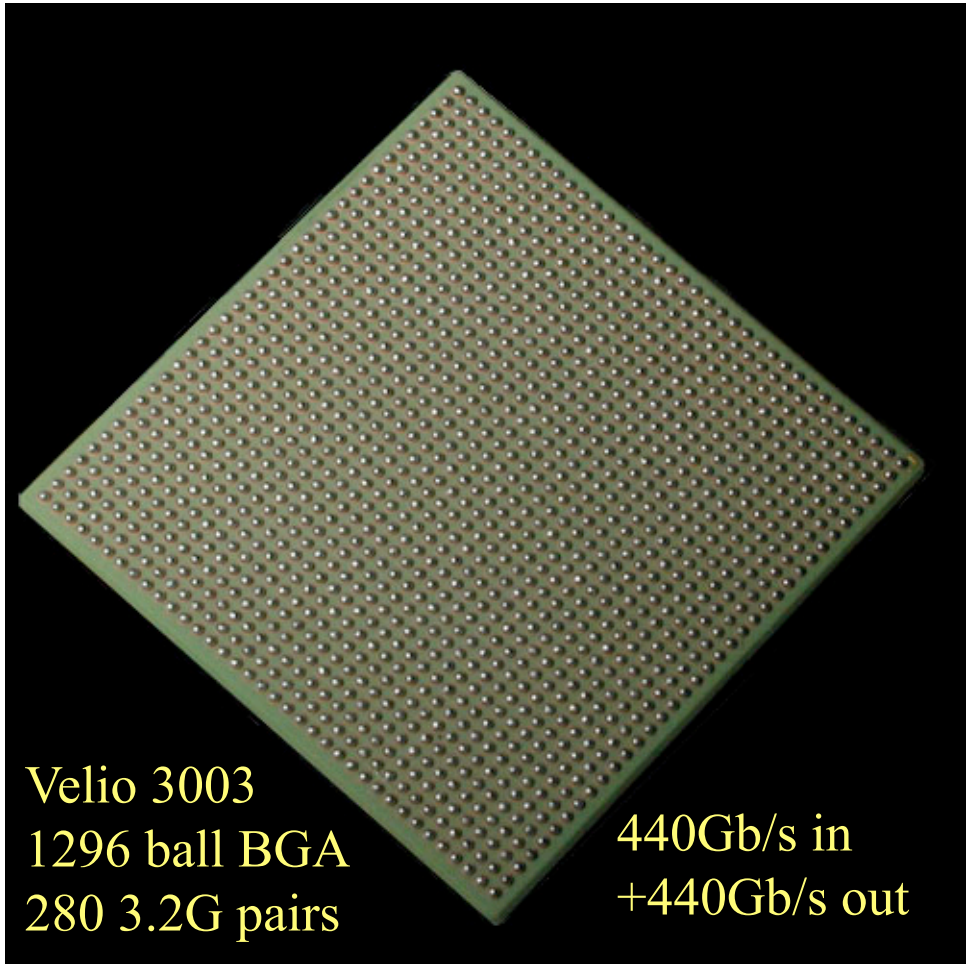
Imagine
2002

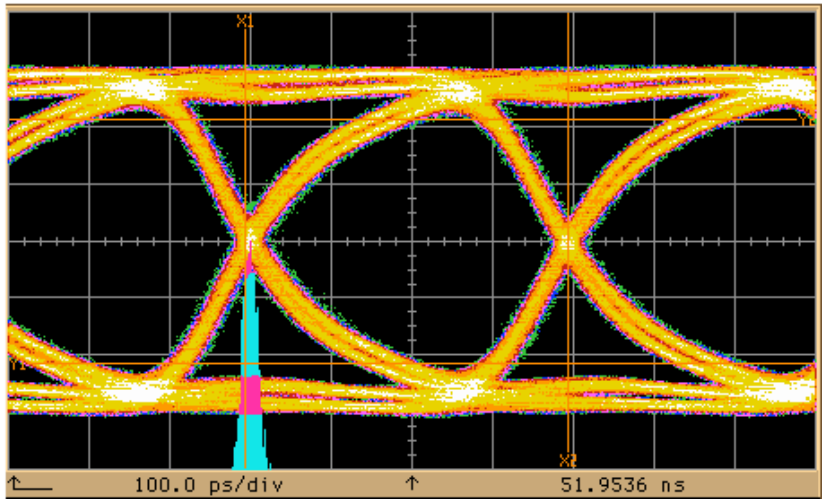


Robert Mullins

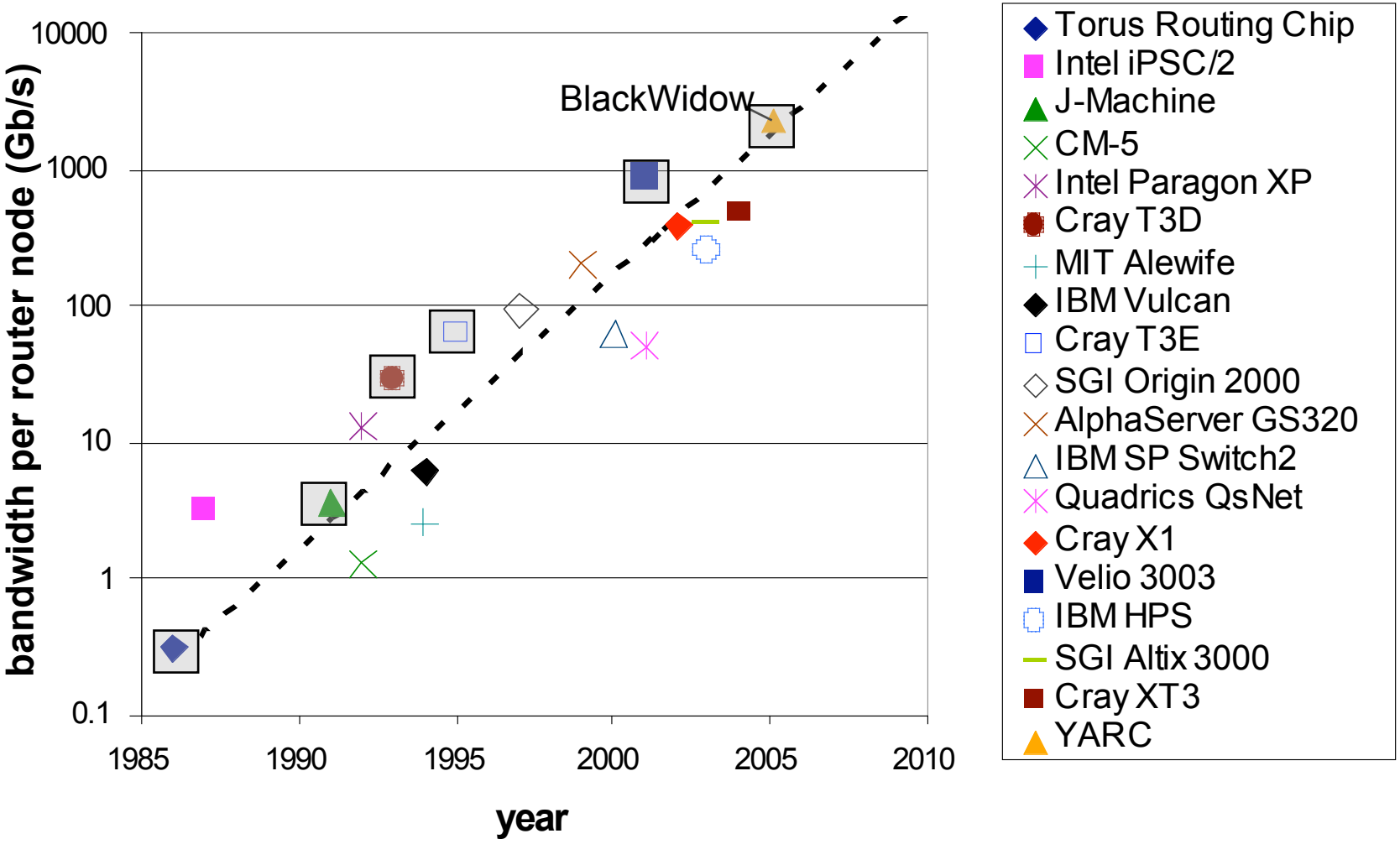
High Radix Routers and Networks

Bandwidth

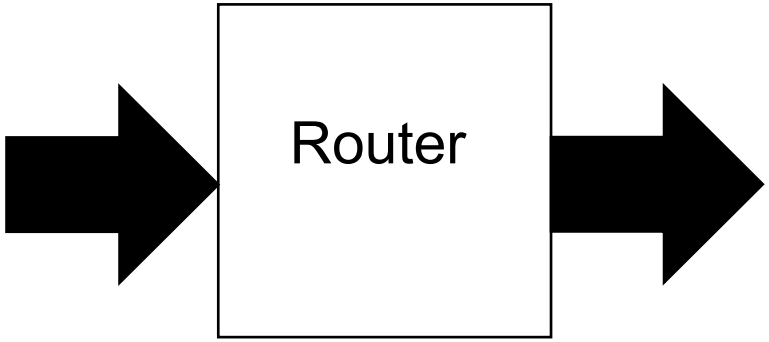

Velio 3003
1296 ball BGA
280 3.2G pairs
440Gb/s in
+440Gb/s out

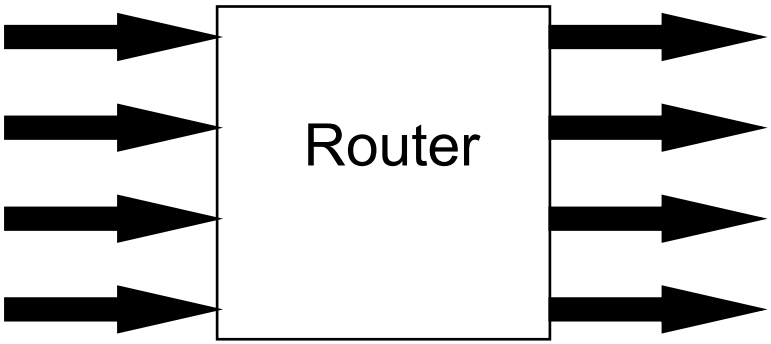

100.0 ps/div
51.9536 ns

Router Bandwidth Scaling
~100x in 10 years

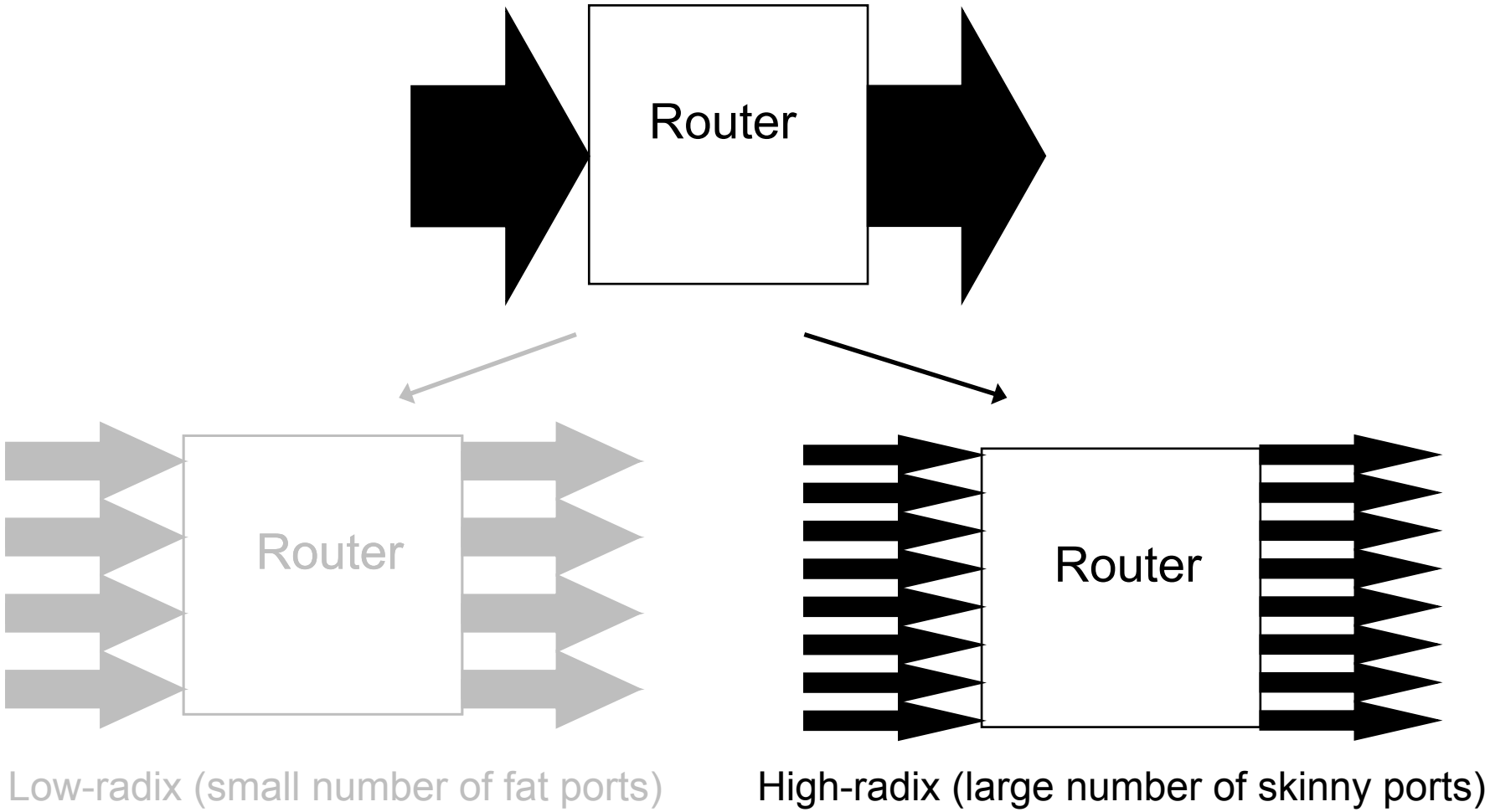


High-Radix Router





High-Radix Router

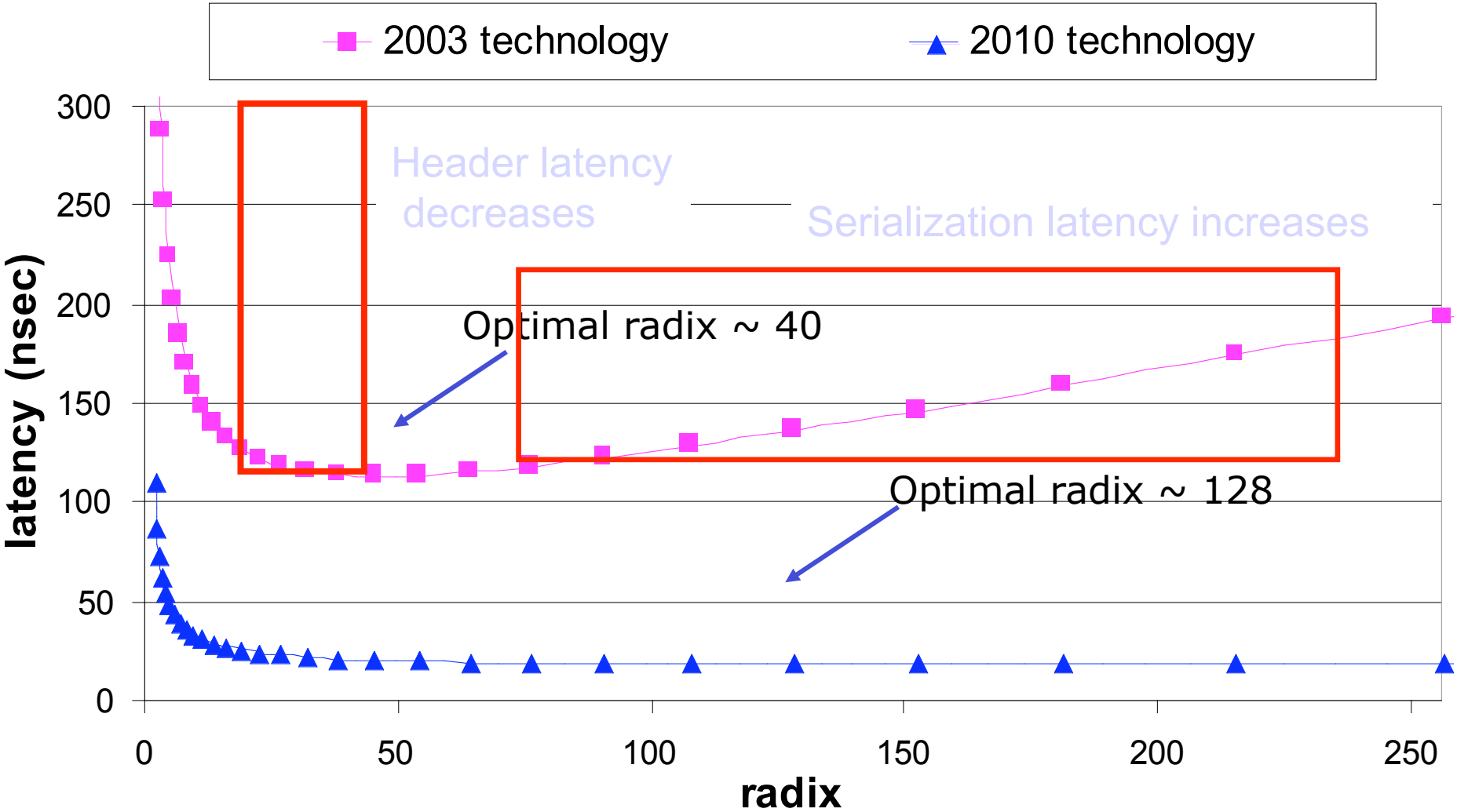


Latency

$$\text{Latency} = H\,t_r + L / b$$
$$= 2t_r \log_k N + 2kL / B$$

where k = radix
B = total Bandwidth
N = # of nodes
L = message size

Latency vs. Radix



Determining Optimal Radix

Latency = Header Latency + Serialization Latency

$= H\, t_r + L / b$

$= 2 t_r \log_k N + 2kL / B$

where k = radix
B = total Bandwidth
N = # of nodes
L = message size

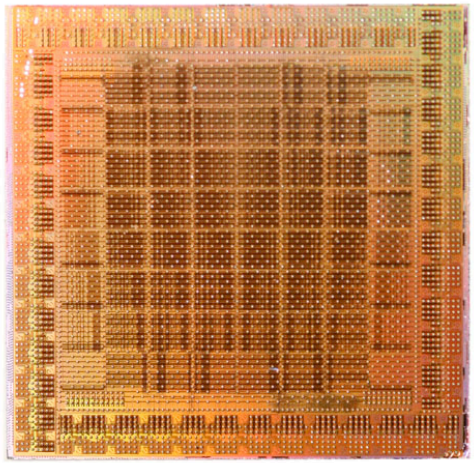
Optimal radix

➔ $k \log_2 k = (B\, t_r \log N) / L$

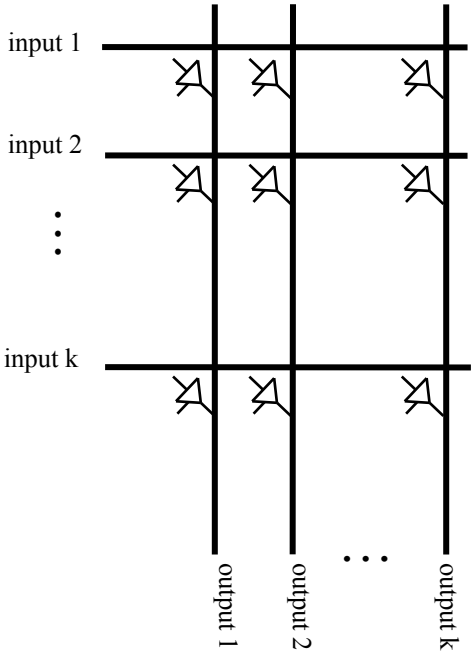
= ***Aspect Ratio***

High-Radix Router

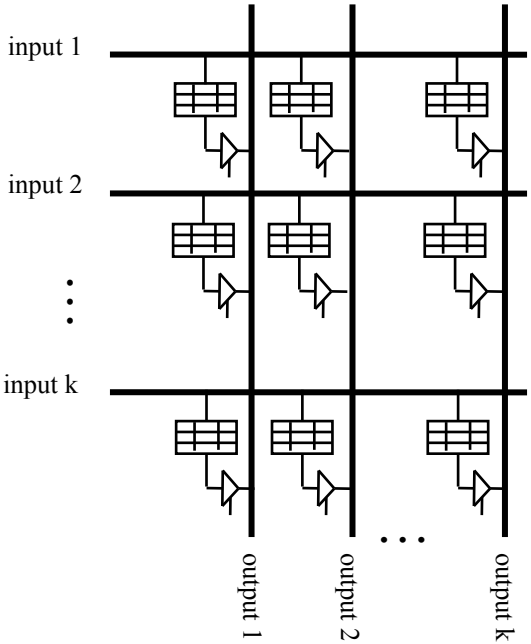
- Many router structures scale as P^2 (or P^2V^2)
  - Allocators particularly difficult
  - Not feasible with P=64 and V~8
- Decompose router
  - Each sub-allocation is feasible
- Put the buffers where they do the most good
- YARC – Cray 2006
  - 64 ports, each 18.75Gb/s (3x6.25)
  - Tiled hierarchical design
    - 8x8 array of 8x8 subswitches
    - Buffering at subswitch inputs



High-Radix Switch Architectures (II)



(a) Baseline design

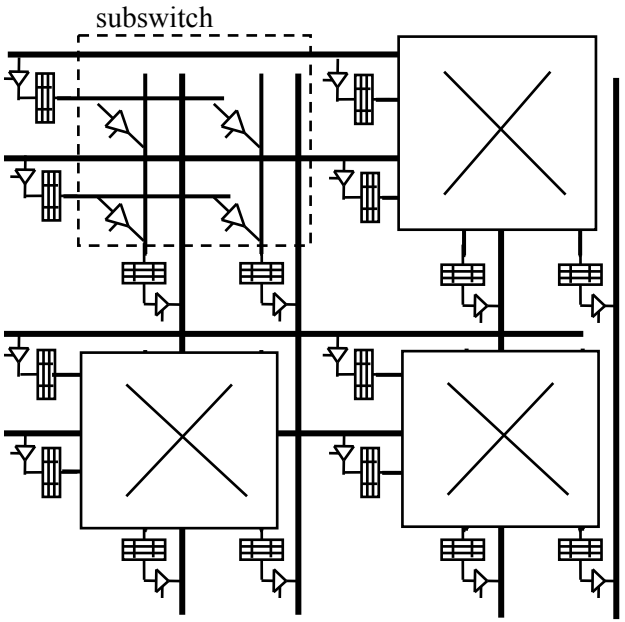


(b) Fully buffered crossbar

High-Radix Switch Architectures (III)

(a) Baseline design

(b) Fully buffered crossbar

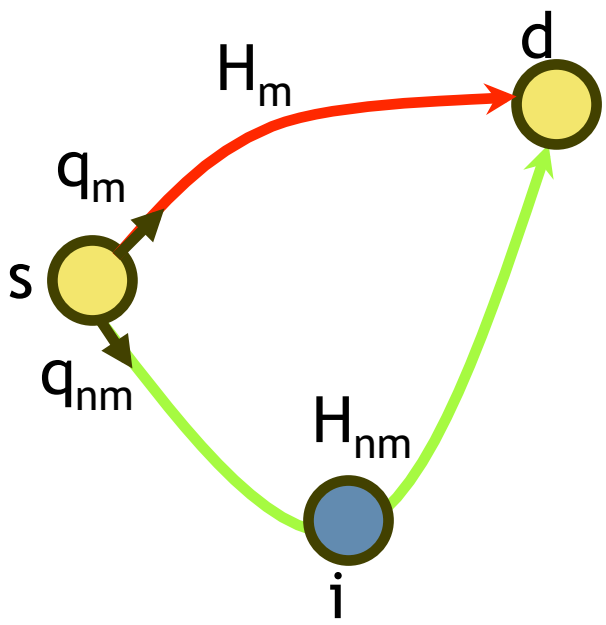


(c) Hierarchical crossbar

Global Adaptive Routing Enables new Topologies

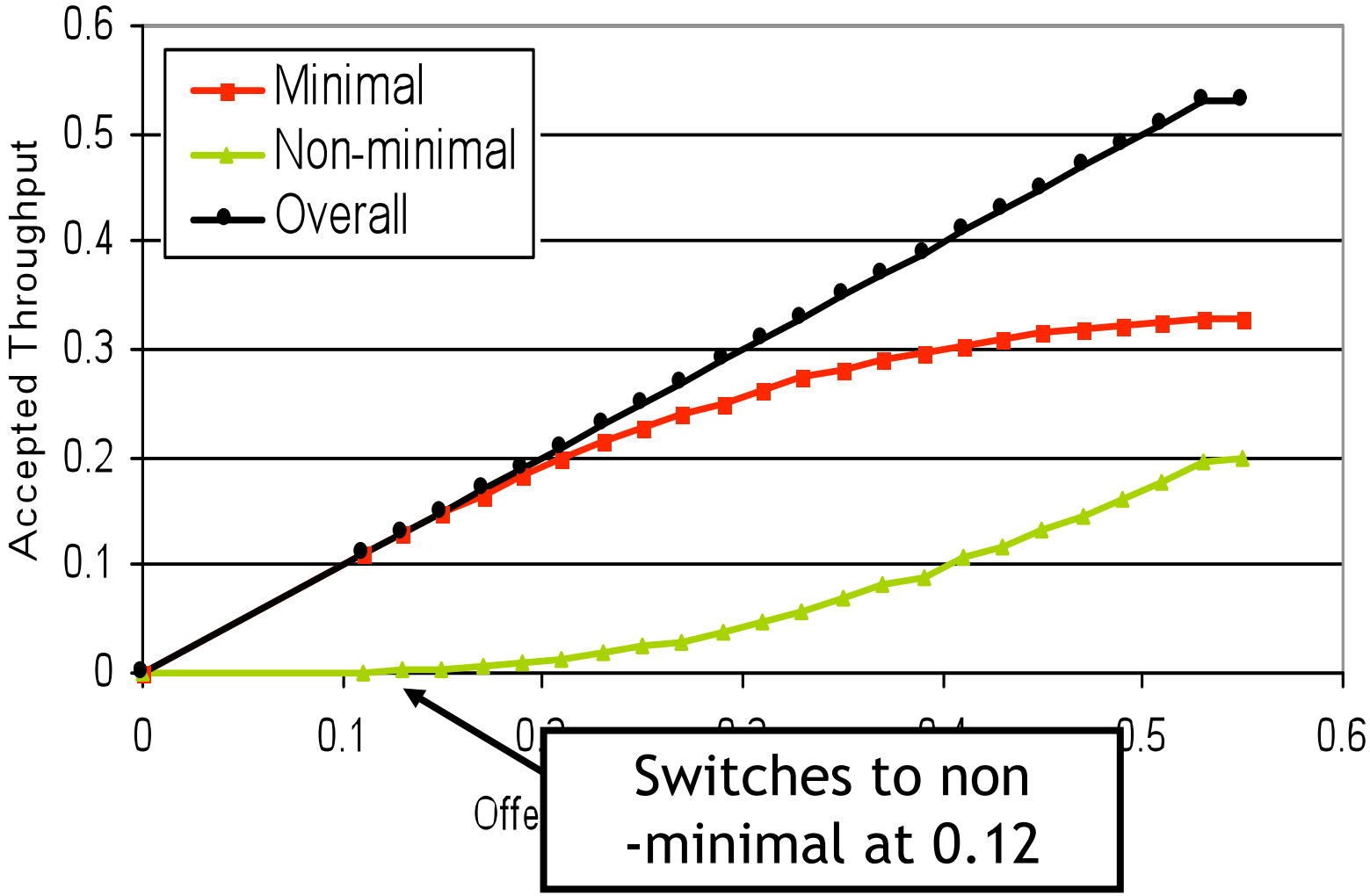
- VAL gives optimal worst-case throughput
- MIN gives optimal benign traffic performance
- UGAL (Universal Globally Adaptive Load-balance)
  - [Singh '05]
  - Routes benign traffic minimally
  - Starts routing like VAL if load imbalance in channel queues
  - In the worst-case, degenerates into VAL, thus giving optimal worst-case throughput

UGAL

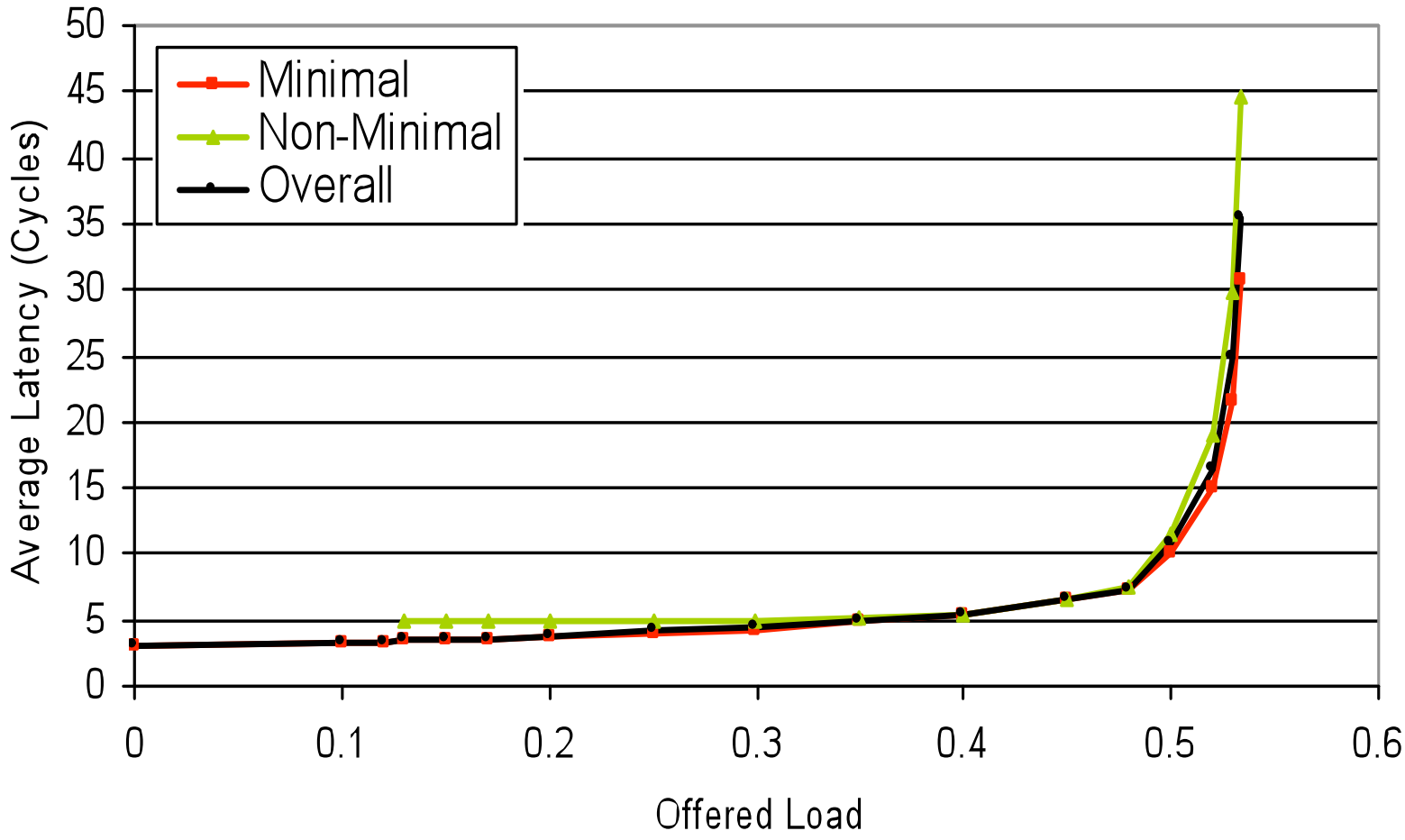


1. H_m= shortest path (SP) length
2. q_m= congestion of the outgoing channel for SP
3. Pick i, a random intermediate node
4. H_{nm}= non-min path (s→i→d) length
5. q_{nm}= congestion of the outgoing channel for s→i→d
6. Choose SP if $H_m q_m \leq H_{nm} q_{nm}$; else route via i, minimally in each phase

CQR: TOR throughput



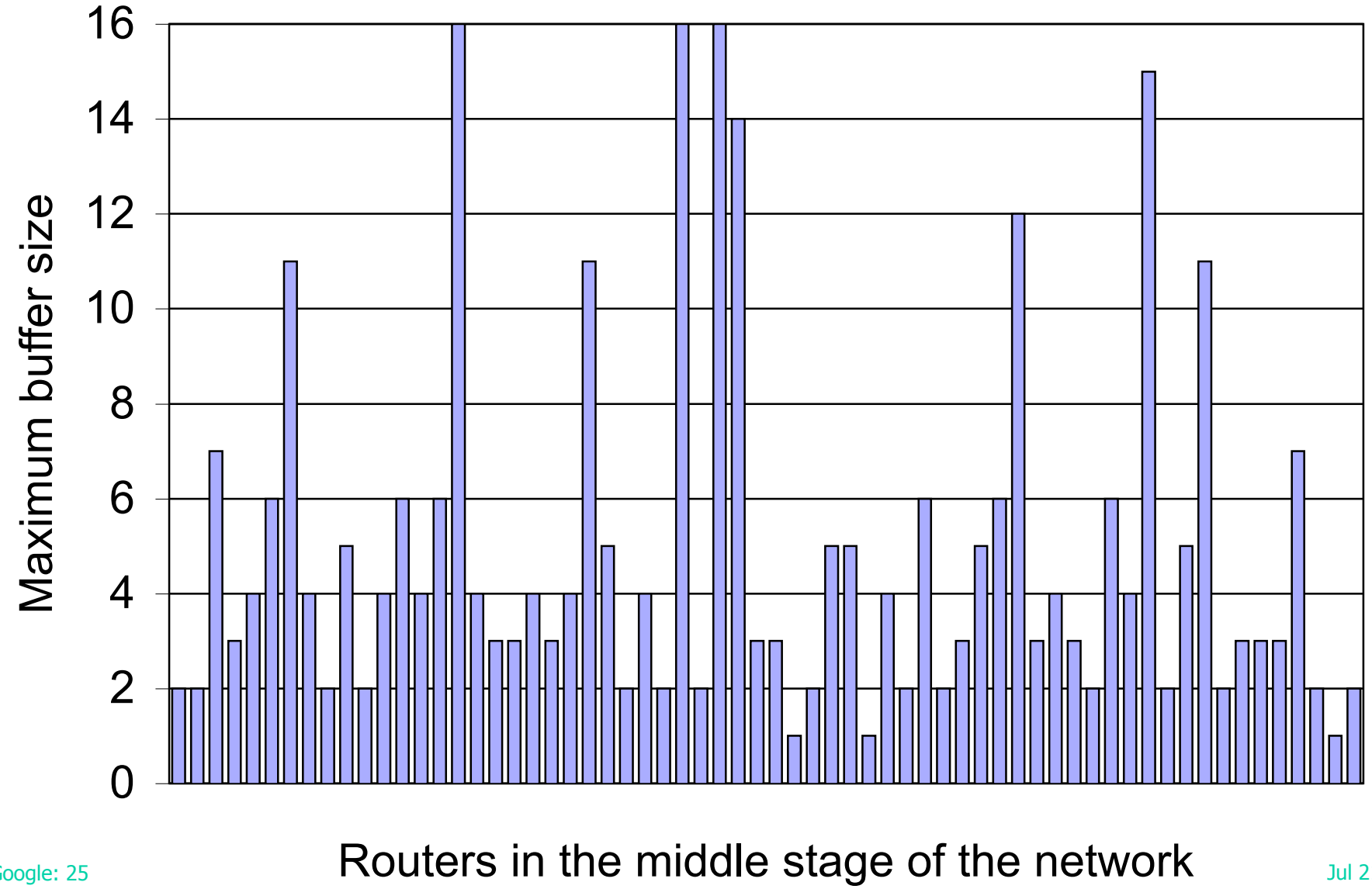
CQR: TOR latency



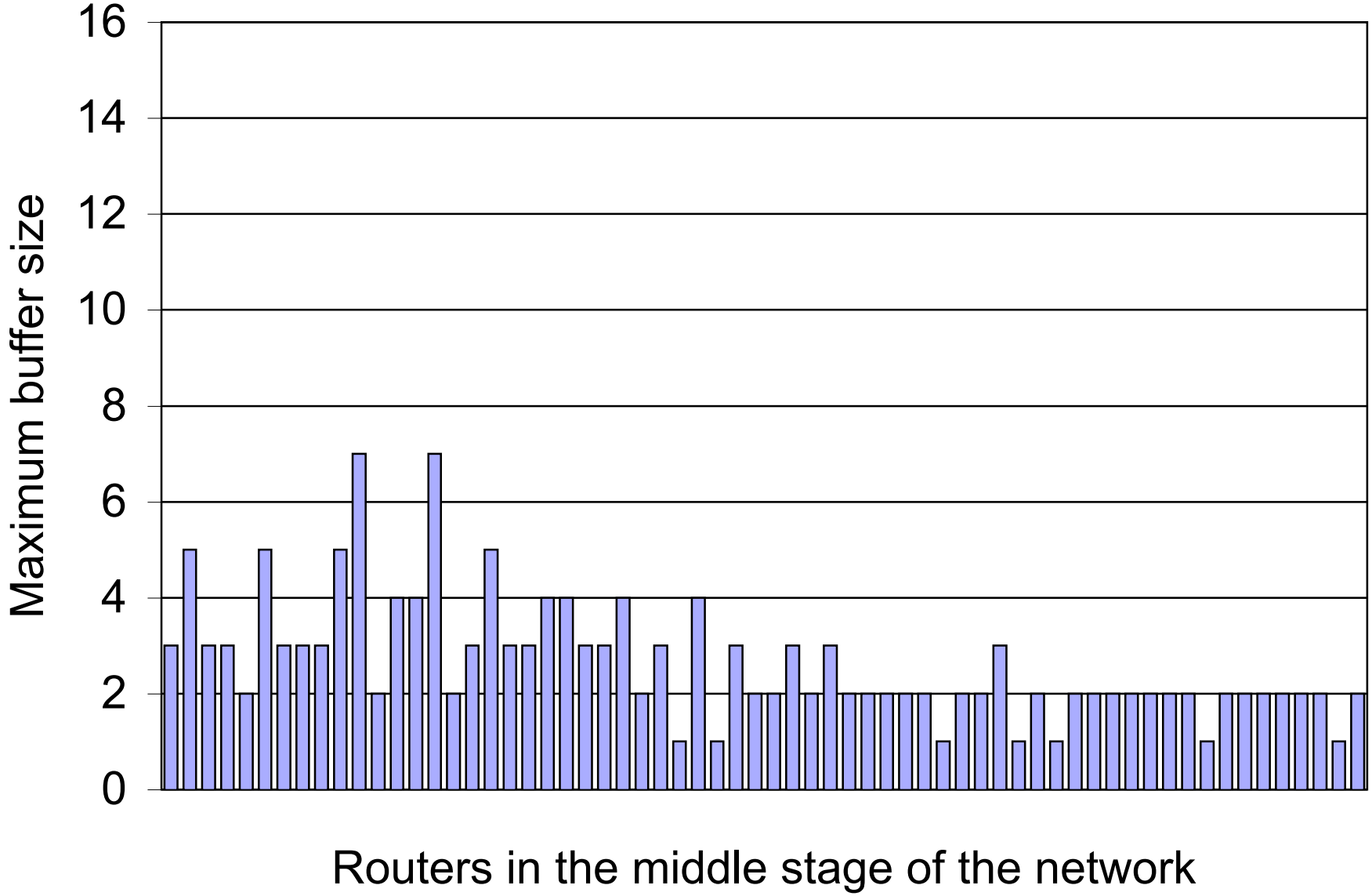
UGAL report card

64 node topology	Throughput (frac of capacity)			
	Algo	Θ benign	Θ adv	Θ avg
K_{64}	VAL	0.5	0.5	0.5
	MIN	1.0	0.02	0.02
	UGAL	1.0	0.5	0.5
8 x 8 torus	VAL	0.5	0.5	0.5
	MIN	1.0	0.33	0.63
	UGAL	1.0	0.5	0.7
64 node CCC	VAL	0.5	0.5	0.5
	MIN	1.0	0.2	0.52
	UGAL	1.0	0.5	0.63

Transient Imbalance



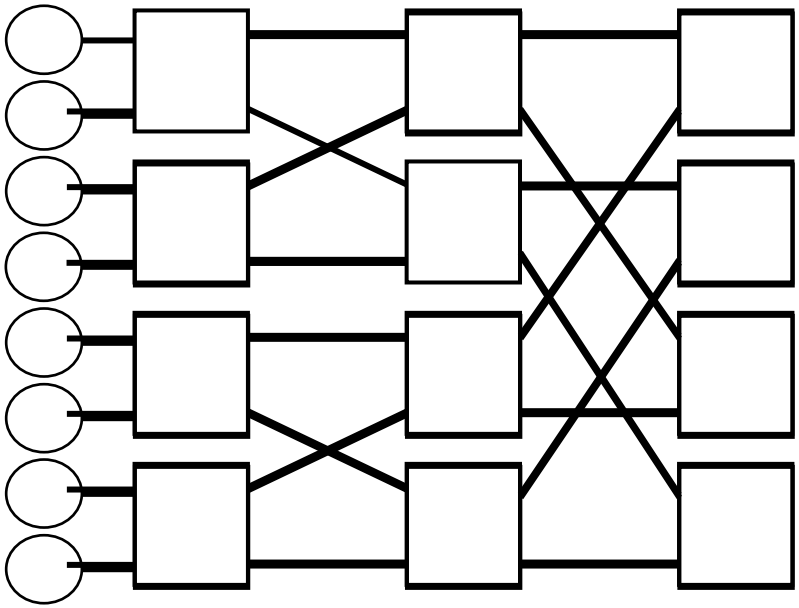
With Adaptive Routing



High-Radix Topology

- Use high radix, k, to get low hop count
  - $H = \log_k(N)$
  - Hop count ~ cost
- Provide good performance on both benign and adversarial traffic patterns
  - Rules out butterfly networks - no path diversity
    - $H = \log_k(N)$ - optimal
    - Dismal throughput on worst-case traffic
  - Clos networks work OK
    - $H = 2\log_k(N)$ - with short circuit
    - But twice the hop count needed on benign traffic
  - Cayley graphs have nice properties but are hard to route

Clos Networks
Delivering Predictable Performance Since 1953



IEEE Spectrum Online, July 16, 2008

Flattened Butterfly Network Lets Data Fly Through Supercomputers and Multicore Processors

By Neil Savage

First Published July 2008

Interconnect architecture allows for the most efficient routing of data, developer says

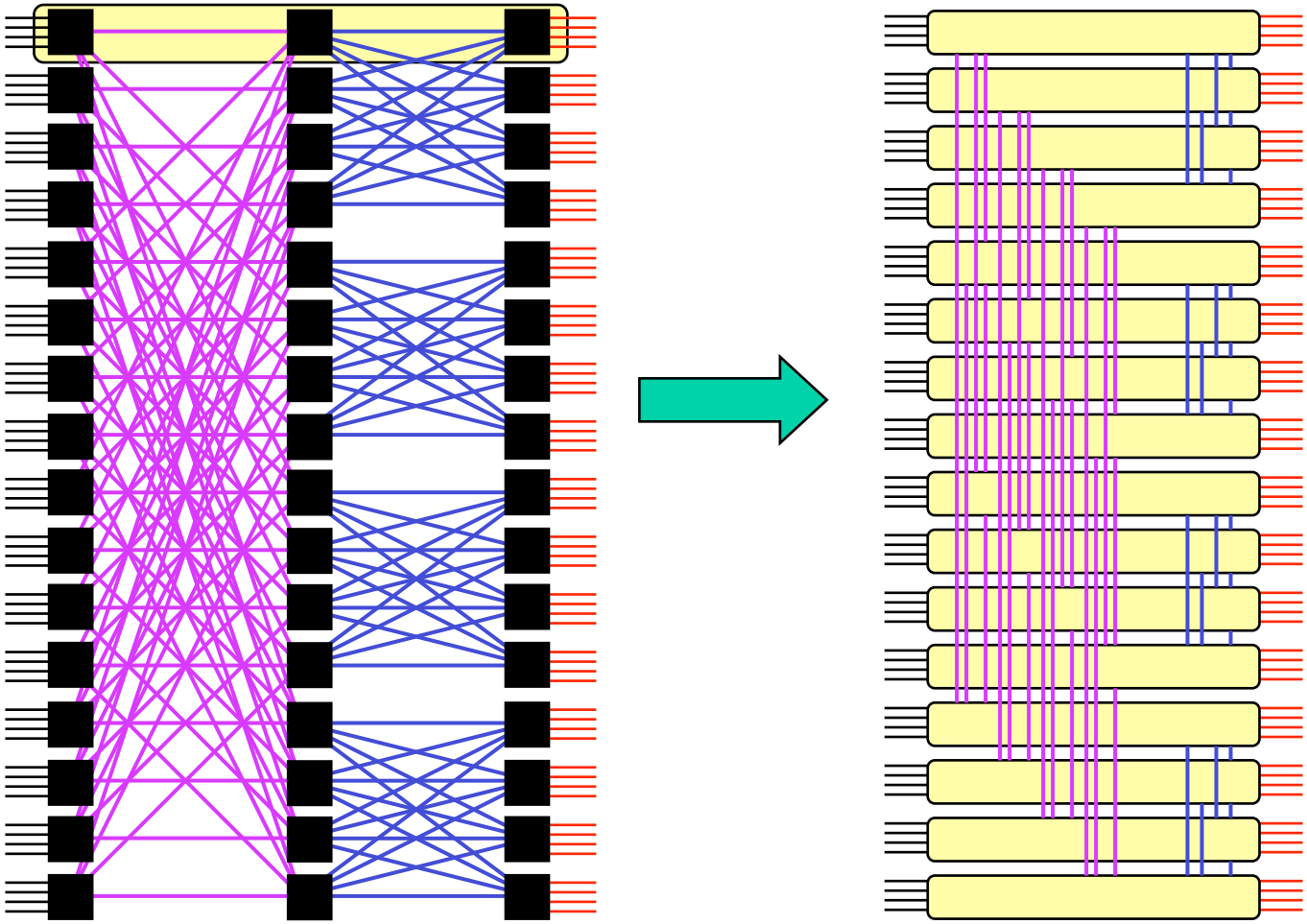
Email Print Comments (1) Reprints Newsletters

Del.icio.us Digg Slashdot

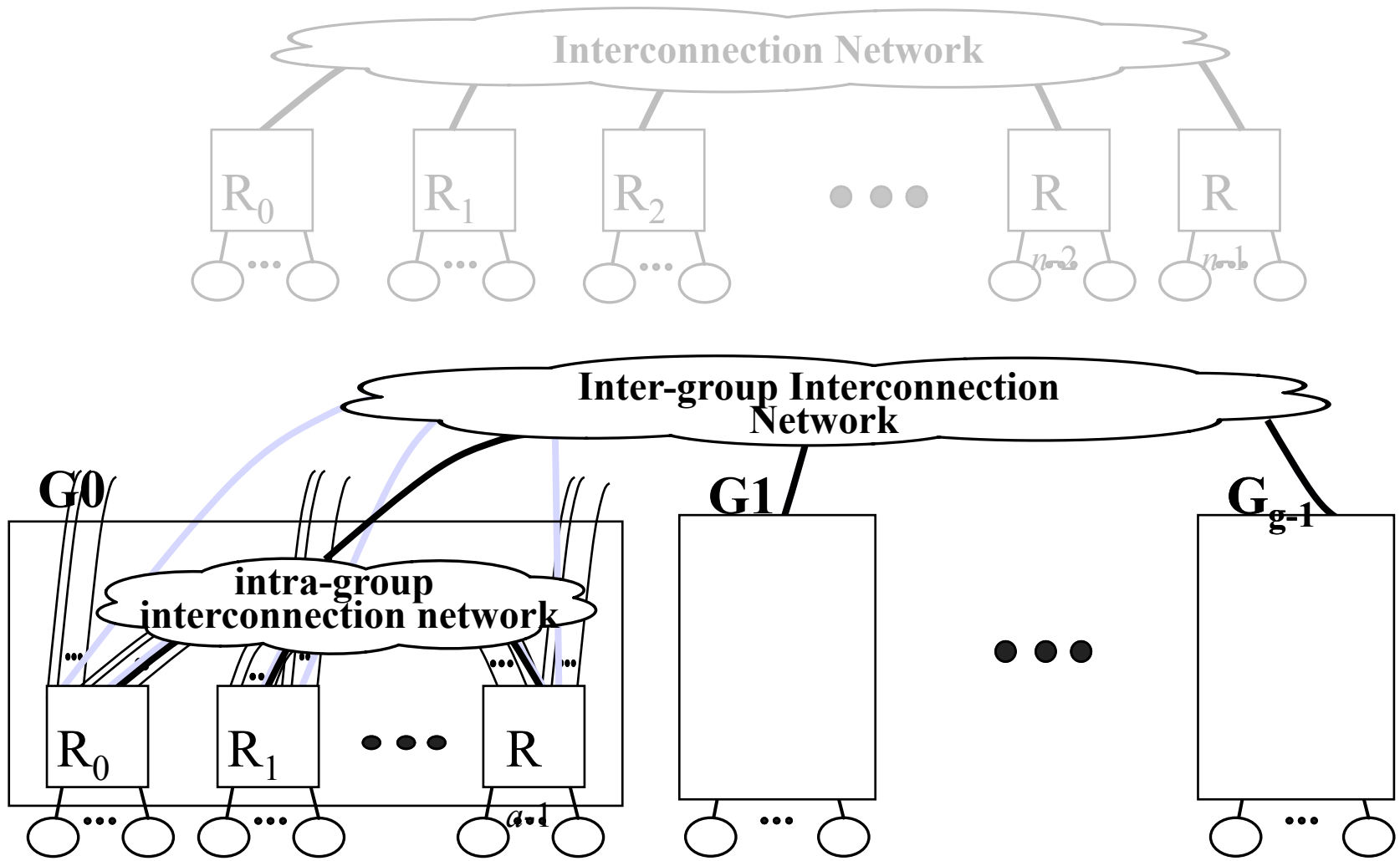
PHOTO: LEE PETTET/ISTOCKPHOTO

16 July 2008—As both computer chips and supercomputers grow more powerful by linking together more and more processors, they risk wasting money, energy, and time by sending data among processors over

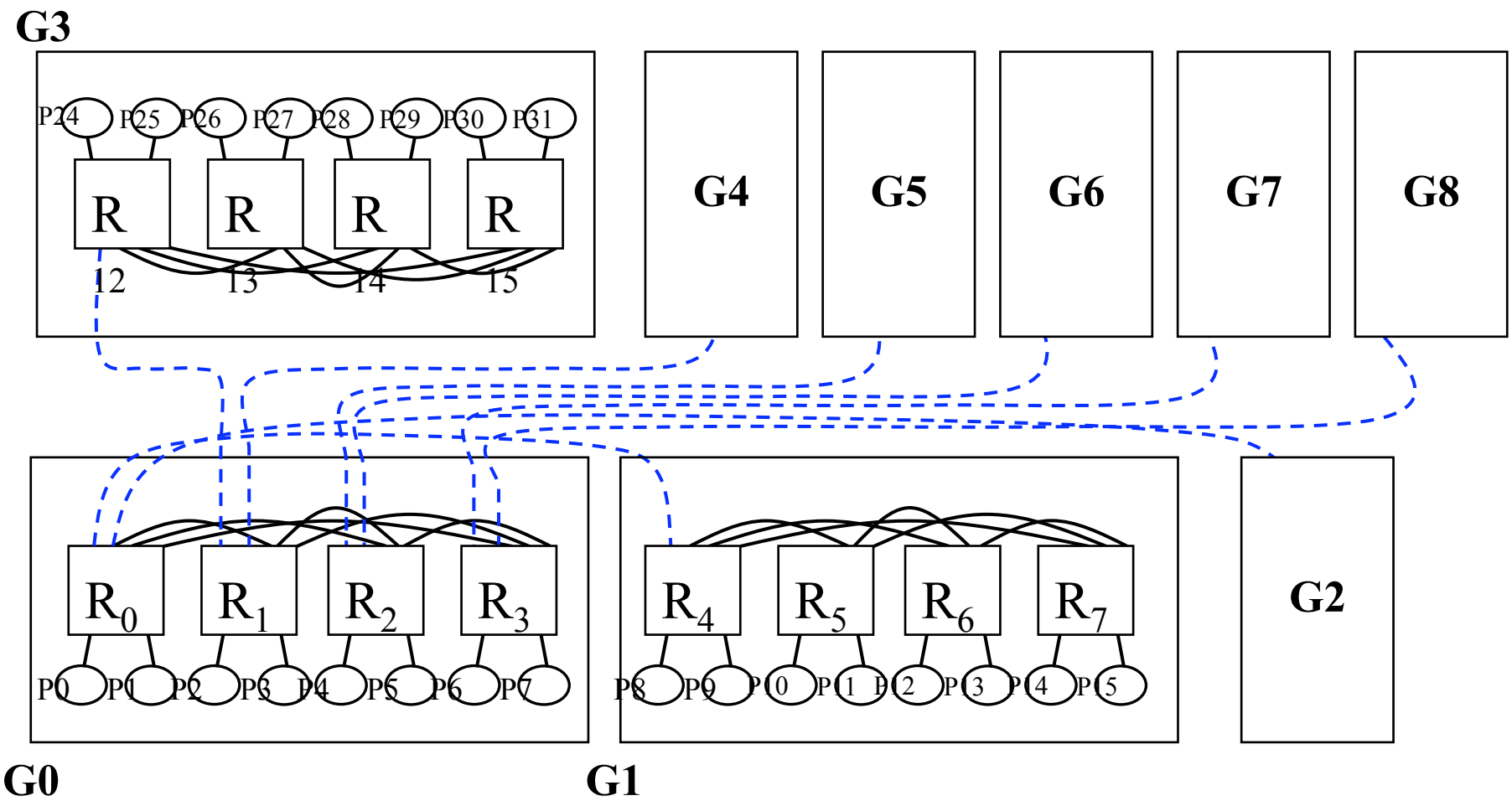
Flattened Butterfly (ISCA'07)



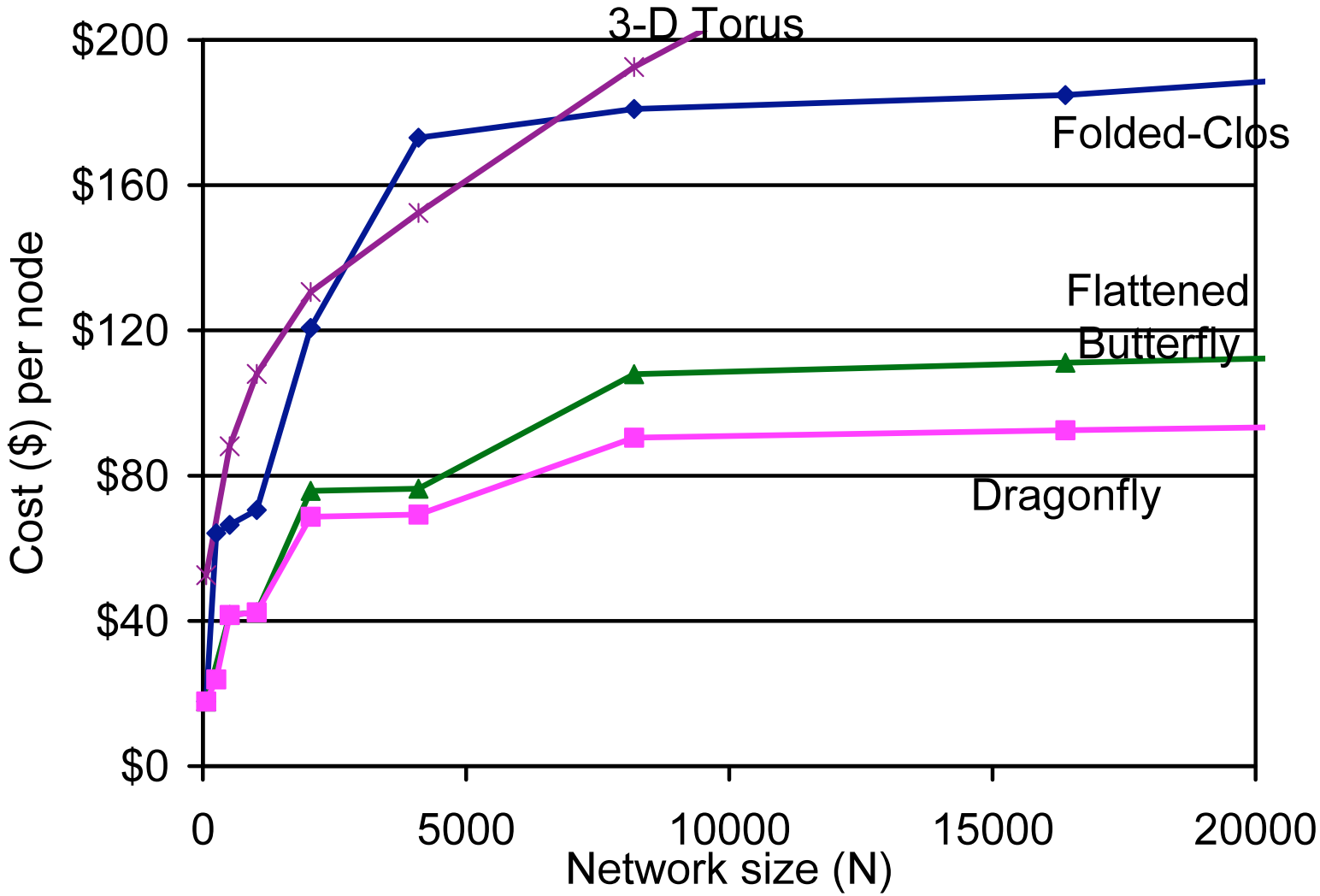
Dragonfly Topology



Dragonfly Topology Example



Topology Cost Comparison



A Summary of Where We Are

- High radix routers and networks
  - Best able to convert high pin bandwidth into ***value***
- Router organization
  - Hierarchical switch and allocator, subswitch buffers
- Routing
  - Global adaptive routing enables aggressive topologies
- Topology
  - Flattened butterfly gives minimal diameter (hence cost)
  - Dragonfly minimizes number of long (expensive) links

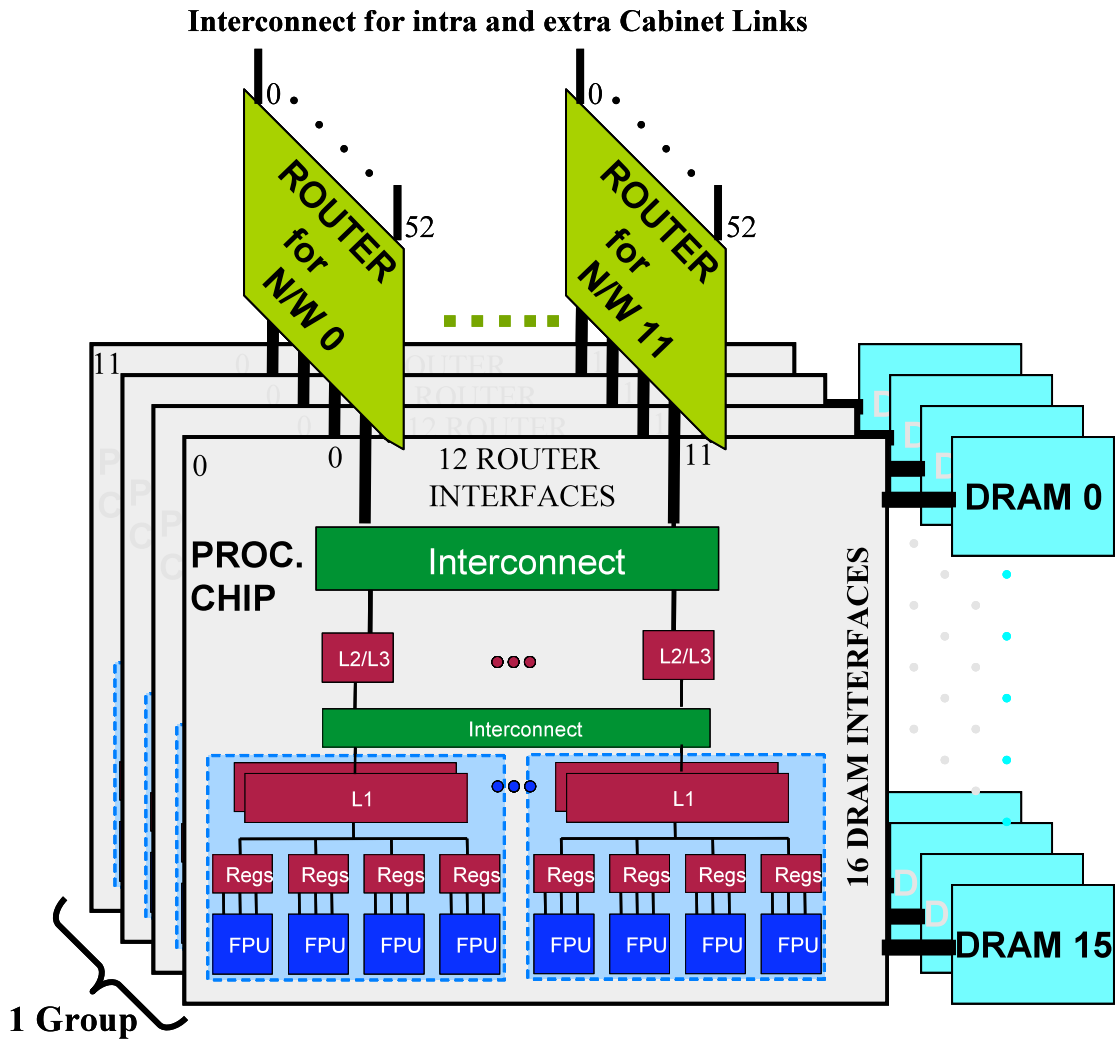
To ExaScale and Beyond

1 EFlop/s Strawman

Sizing done by "balancing" power budgets with achievable capabilities

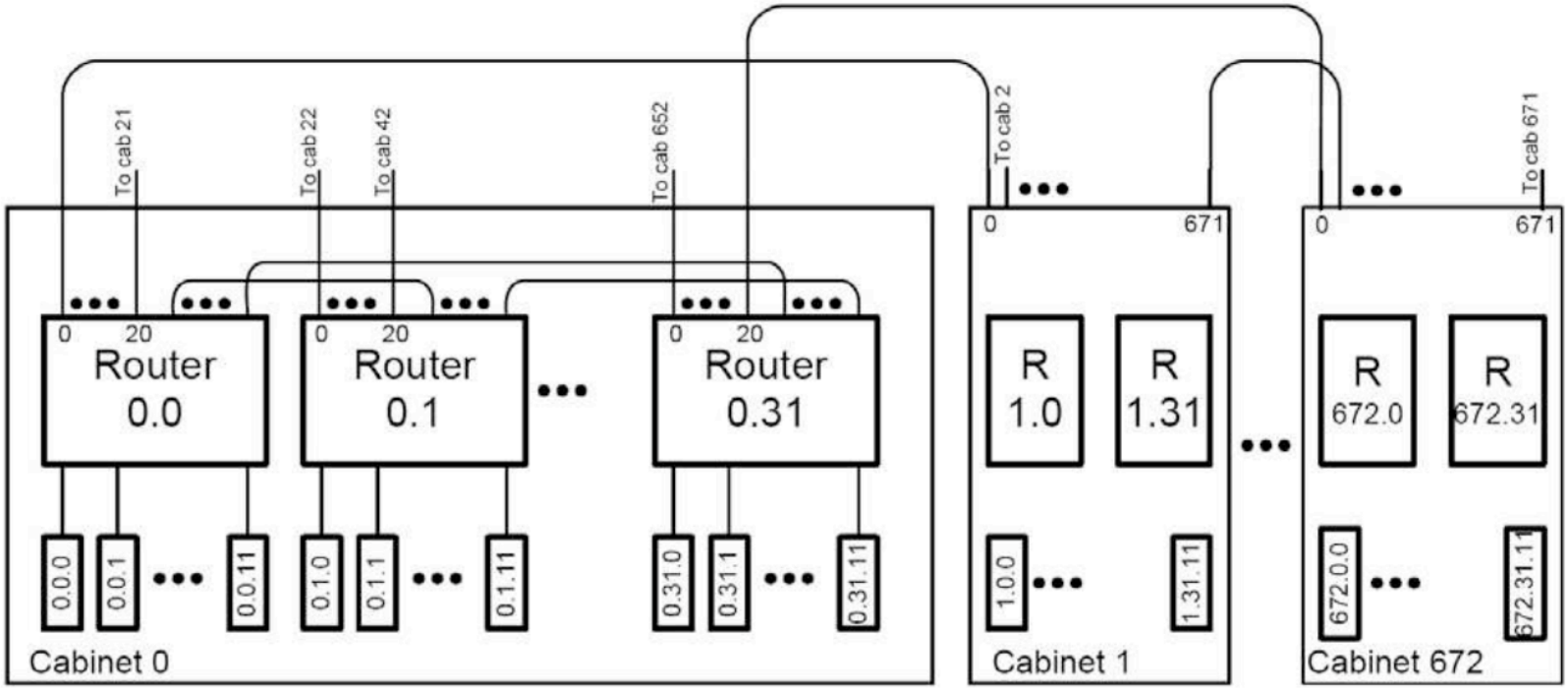
- 4 FPUs+RegFiles/Core (=6 GF @1.5GHz)
- 1 Chip = 742 Cores (=4.5TF/s)
  - 213MB of L1I&D; 93MB of L2
- 1 Node = 1 Proc Chip + 16 DRAMs (16GB)
- 1 Group = 12 Nodes + 12 Routers (=54TF/s)
- 1 Rack = 32 Groups (=1.7 PF/s)
  - 384 nodes / rack
- 3.6EB of Disk Storage included
- 1 System = 583 Racks (=1 EF/s)
  - 68 MW w'aggressive assumptions
  - 166 MILLION cores
  - 680 MILLION FPUs
  - 3.6PB = 0.0036 bytes/flops

	BW: Words/flop/s
Register File	3.0000
L1	0.2500
L2	0.1250
L3	0.0625
Node DRAM	0.0200
Network	0.0044



Some Exa Observations

- Its all about energy (J/bit) – more on this later
- Two levels of network – on-chip (Noc) and off-chip
- System network is a dragonfly
  - 12 CMPs, 31 local, 21 global links per router
  - 12 Parallel networks – enables tuning and tapering



Exa Energy Budget - Fixed

Item	Percent	Watts	Units	BW (GW/s)	Taper	Comments
FPUs	59.0%	88.5	2968	4495	1	Includes 3-port reg and I-mem
L1 Data	10.9%	16.4	742	1124	4	64KB per 4 FPUs
L2	6.9%	10.4	371	562	8	256KB per 2 L1s
L3	7.5%	11.3	189	286	16	Global access to L2s
DRAM	10.0%	15.0	59	89	50	Attached to this chip
Network	5.6%	8.4	13	27	164	Global access
Taper = number of flops executed per access						

- Allocating 60% power to FPUs doesn't leave much for communication
- Leads to a steep BW taper

Exa Energy Budget Adaptive

Item	Percentage	Watts	Units	Bandwidth	Taper
FPUs	84.3%	126.5	4240	6421	1
L1	62.5%	93.7	4240	6421	1
L2	79.1%	118.6	4240	6421	1
L3	99.7%	149.6	2523	3821	1.7
DRAM	100.0%	150.0	592	897	7
Network	100.0%	150.0	234	354	18

- 1060 cores per chip (vs 742) 4 FPUs/core
- Can sustain 1 access per cycle at L1 or L2
  - But nothing else
- Can use all power at L3, DRAM, or globally
- Adapt actual power budget to demand of application
  - Throttle to stay in bounds

Its all about Joules/bit

10pJ/FLOP in 32nm

128FLOPs of energy to read a word from DRAM

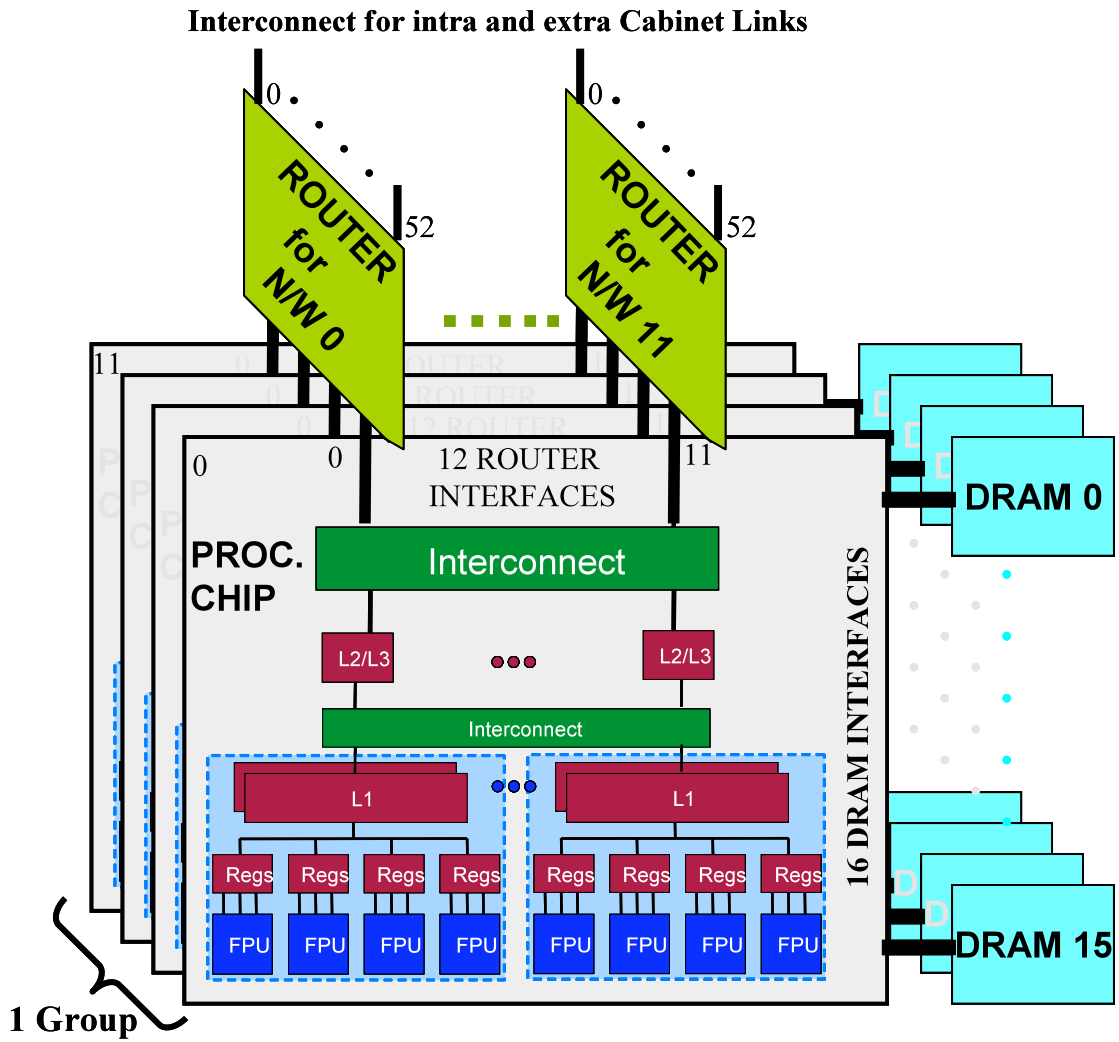
32FLOPs of energy to send word within cabinet

256FLOPs to cross machine

NoCs

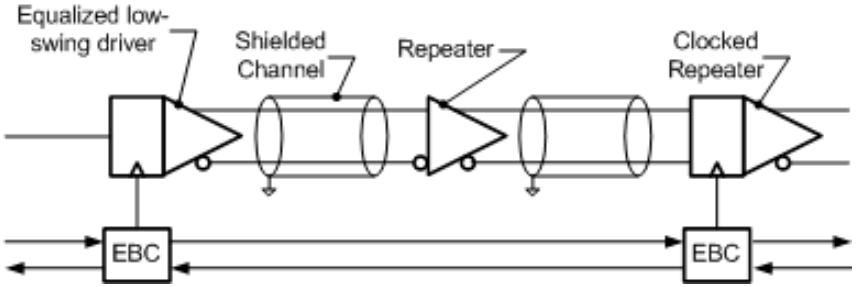
Exa NoC

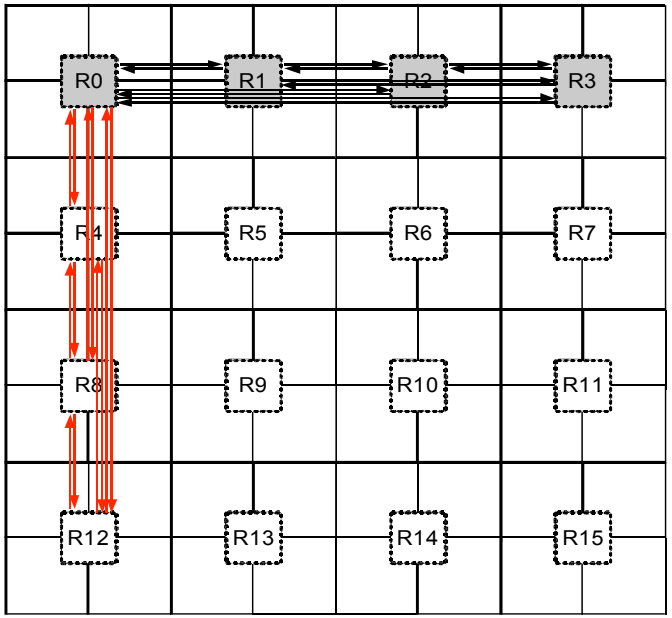
- Much of the communication is on chip
- First 4-levels of storage hierarchy
- The bulk of all bandwidth
- What does this network look like?



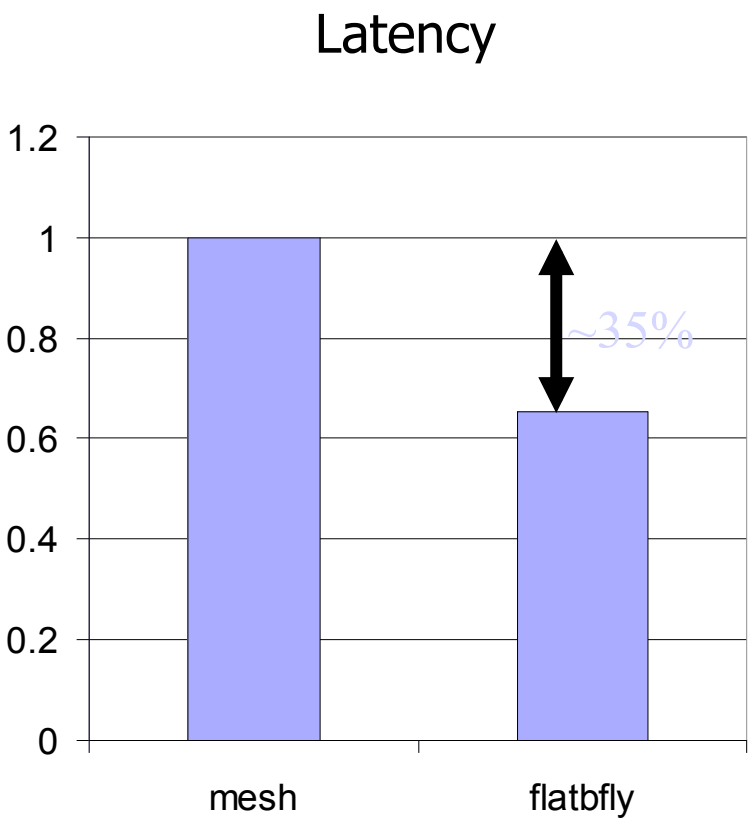
On Chip Interconnect

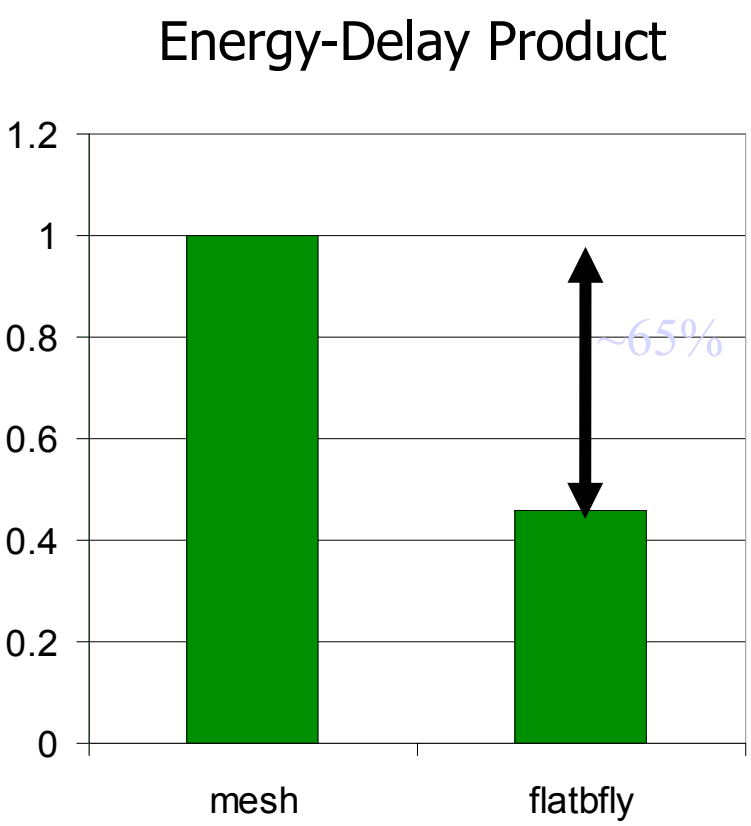
- Enabling circuit technology
  - Links, switches, buffers
  - 10x - 100x improvements in b/J
  - Drives topologies, routing, flow control
- Network design
  - Efficient topology/routing
  - Flow control (trade BW for buffers)
  - Low latency uArch





Evaluation Comparison



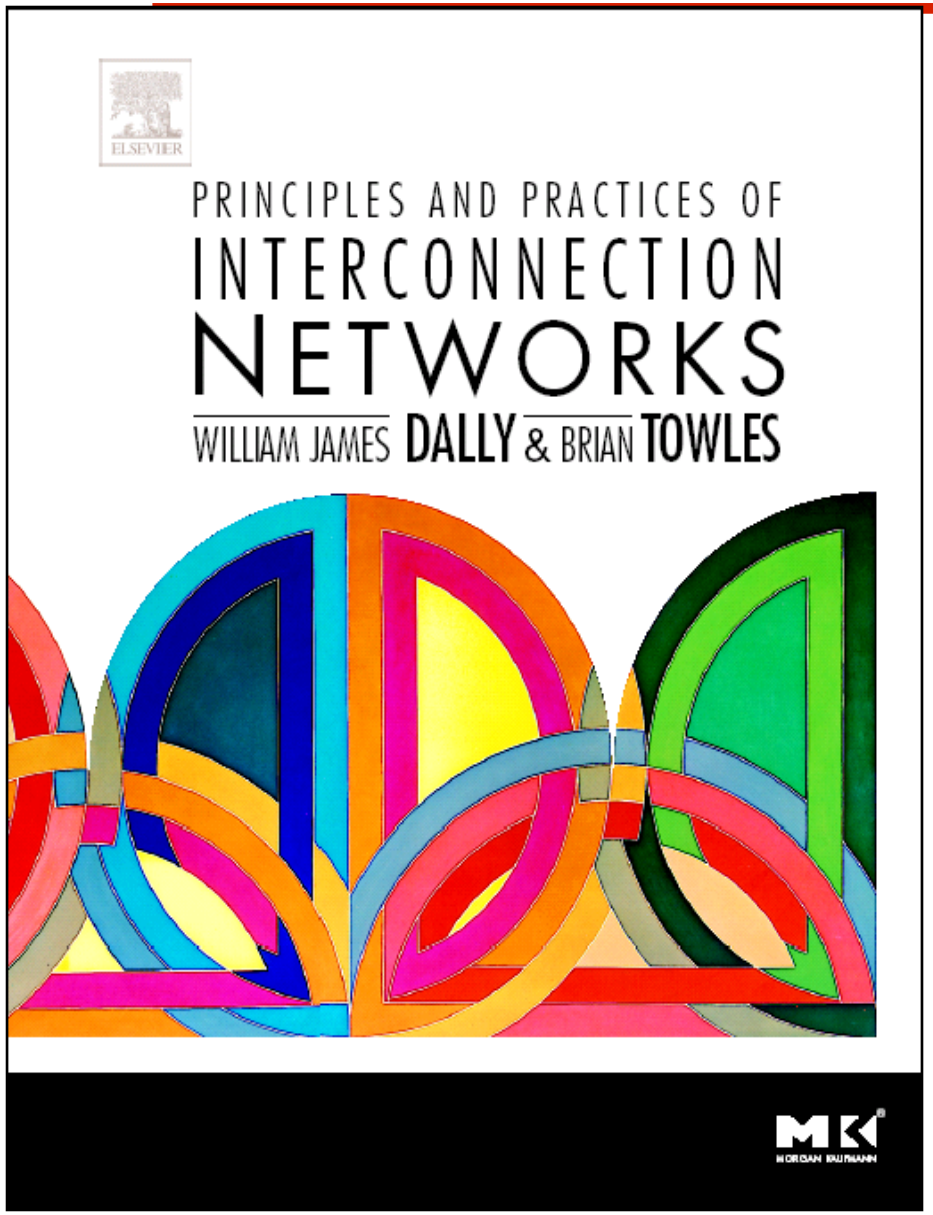
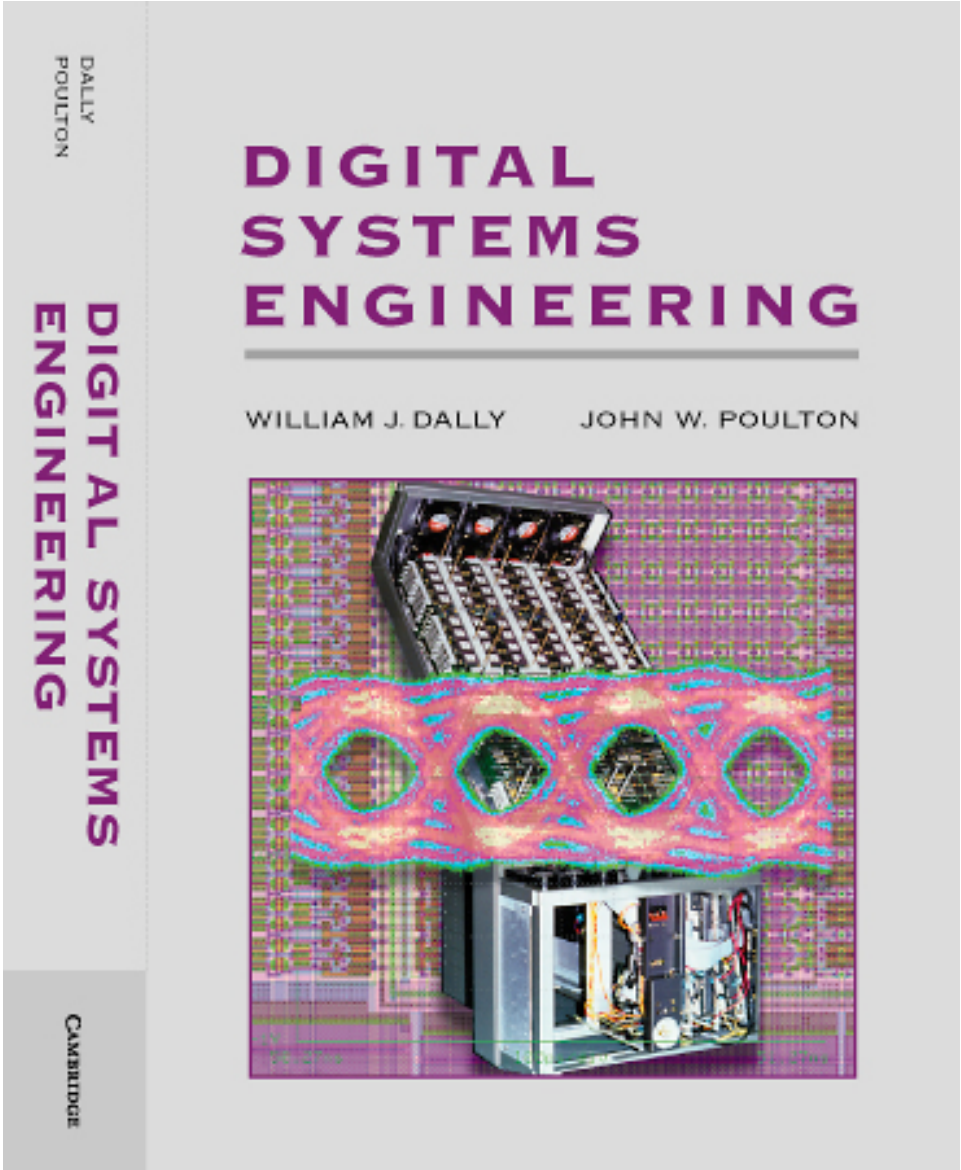


Flattened butterfly can be extended to on-chip networks to achieve lower latency and lower energy.

Summary

- Low-radix era
  - Developed key network technologies
    - Wormhole FC, virtual channels, etc…
    - Matched network design to packaging and signaling constraints
  - $4 < k < 8$ torus, mesh, and Clos networks
- High-radix era
  - Router bandwidth increasing 100x per decade
  - Increasing network radix better able to exploit increased router bandwidth
  - Requires partitioned router organization with subswitch buffers.
  - Global adaptive routing enables efficient topologies
  - Flattened butterflies and dragonflies
  - Data centers now driving these technologies
- NoCs
  - Bulk of bandwidth is on-chip
  - Flattened butterfly is ideal topology here too! – fewer hops
  - Many open questions in NoC design

Challenges

- Power – at all levels
  - Circuits/devices
  - Topology/routing/flow control
  - On and off chip
- Network architecture
  - Indirect adaptive routing
  - Dealing with cable aggregation (cables and WDM)
- On-chip networks – all levels
  - Circuits
  - Topology/routing/flow control
  - On/off chip interfaces
- Network interfaces
  - Can have zero overhead send and receive (J-Machine, M -Machine)
- Programming abstractions
  - Abstract the communication hierarchy (Sequoia)
  - Vertical, not horizontal is what matters
  - Focus on essential issues, not incidentals (like MPI occupancy)
- Device/Circuit Technology
  - Focus on pJ/bit

Some very good books