Introduction to Parallel Computing Architectures

Ramón Beivide

Universidad de Cantabria

Outline

- Parallel Computers
- Basic Parallel Architectures
- Some Paradigmatic Examples
- Interconnection Networks
- A More Detailed Example: IBM BlueGene
- ExaScale Computers
- Conclusions and Questions

Motivation

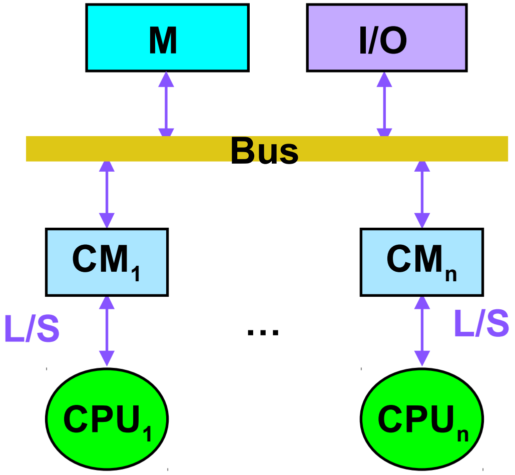
- Some (many) applications require more computational power than the offered by a single microprocessor.
- Any multiprocessor solution should achieve high performance at a sustainable cost (price).
- A *Parallel Computer* can be simply seen as a “collection of communicating processors”
- Examples:
  - CPUs multi-core
  - GPUs
  - Symmetric multiprocessors
  - CC-NUMA multiprocessors
  - Supercomputers

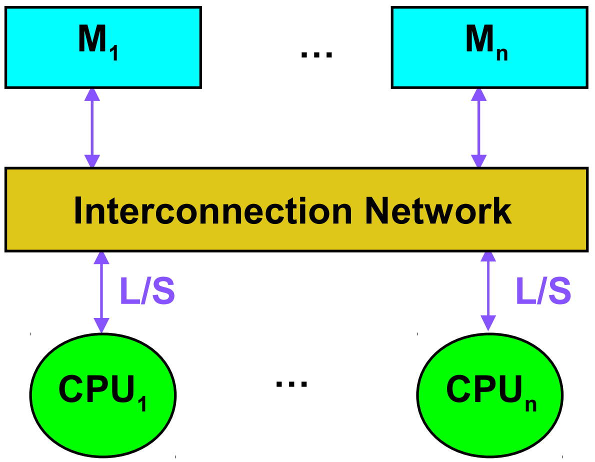
Basic Parallel Architectures (1 of 3)

Shared variables and architectures with shared memory (UMA)

Programs share a single address space
Tasks share data
Each processor can access the global address space
Primitives for synchronization and exclusive access

Natural extension of the sequential programming model

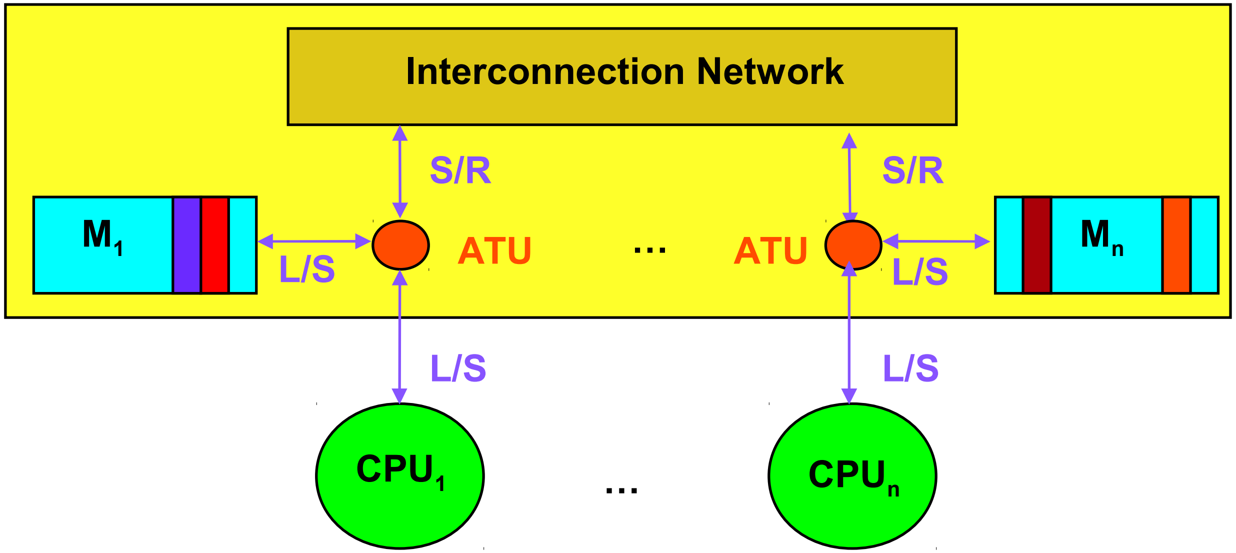




Basic Parallel Architectures (2 of 3)

Shared Virtual Memory with physically local memory (NUMA)

Single and shared virtual address space
Easy programming model with shared variables
Allows porting of applications
Updated old concepts (Cmmp, CM*)



L/S: load/store
S/R: send/receive

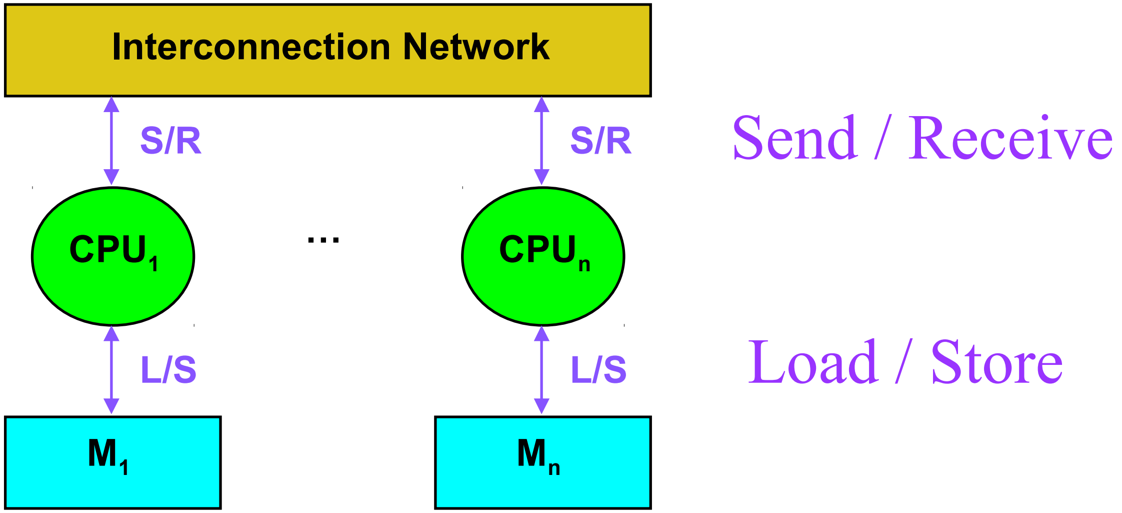
ATU: Address Translation Unit

Ancient and modern systems: KSM, Convex (HP), SGI,…, Alpha EV7, AMD Opteron (no additional hard), Itanium (aditional hard), Nehalem Xeon+QPI

Basic Parallel Architectures (3 of 3)

Message-passing and architectures with local memory (MPPS and Clusters)

Each task has its private data
Task communicate data under agreement
Communication is through message passing
Very close to LAN Networks



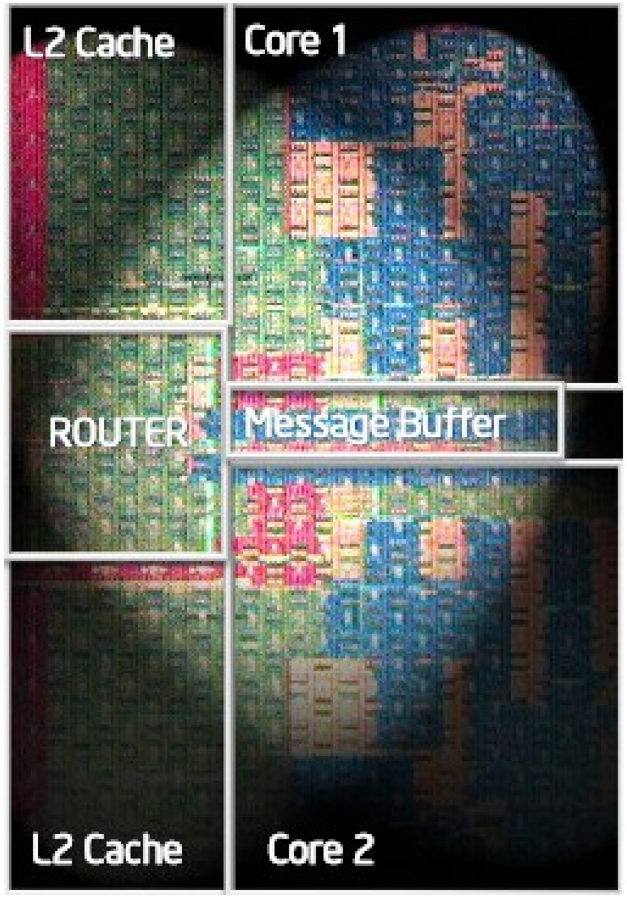
Ancient and modern systems: Transputers, Hypercubes, Paragon, SP2, NOWs, MareNostrum, BlueGene, Jaguar,…

Outline

- Parallel Computers
- Basic Parallel Architectures
- Some Paradigmatic Examples
- Interconnection Networks
- A More Detailed Example: IBM BlueGene
- ExaScale Computers
- Conclusions and Questions

Example: Multicore CPUs and GPUs

- Eager bandwidth nodes
- Intel:
  - 48-core x86 Intel processor
    - 1.3 billion transistors and is built on 45nm CMOS
    - 24 dual-core tiles, arranged in a two dimensional 6-by-4 layout
- Nvidia
  - 512-core Fermi GPU



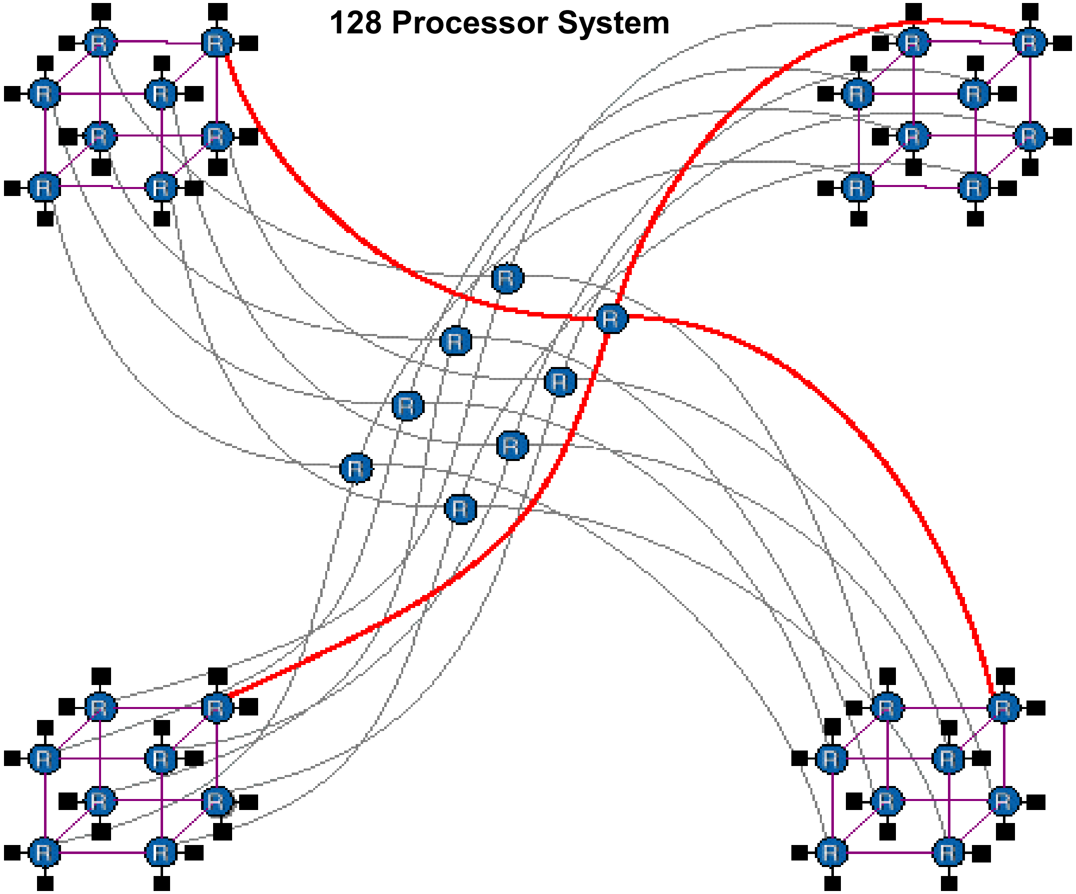
Intel Single-chip Cloud Computer

Example: Tilera

Tilera offers 32- -64 and 100-core version general purpose processors (albeit non-x86)

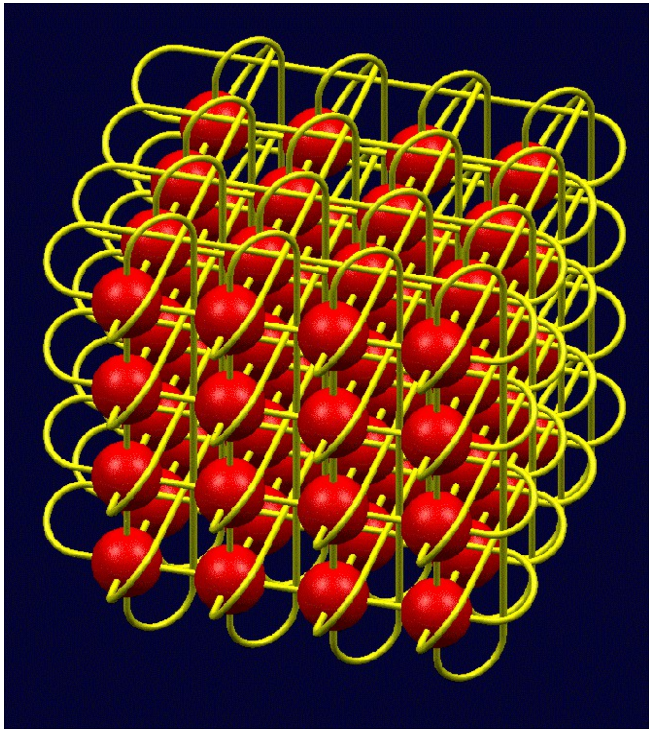
Example: CC-NUMA

SGI Origin 2000



Example: MPP

- **IBM BlueGene**



Outline

- Parallel Computers
- Basic Parallel Architectures
- Some Paradigmatic Examples
- Interconnection Networks
- A More Detailed Example: IBM BlueGene
- ExaScale Computers
- Conclusions and Questions

Interconnection Networks (INs)

- INs are designed to transport:
  - Scalar Variables: Scalar Operand Networks (SON, Raw, Trips)
  - Cache Lines: Remote Loads and Stores (CC-NUMA and Multi/Many-core CMPs)
  - MPI Messages: Supercomputers (Clusters and MPPs)
  - TCP/IP packets (LAN, MAN, WAN)
- INs used at several levels:
  - On Chip
  - Stacked (3D) Inter Chip
  - On Board
  - Intra Cabinet (board-to-board)
  - Inter Cabinet (cabinet-to-cabinet)
  - Local, Campus, Metropolitan,…,World-wide
- INs have a great impact on the performance and cost of parallel (and distributed) applications and architectures.

Network Parameters

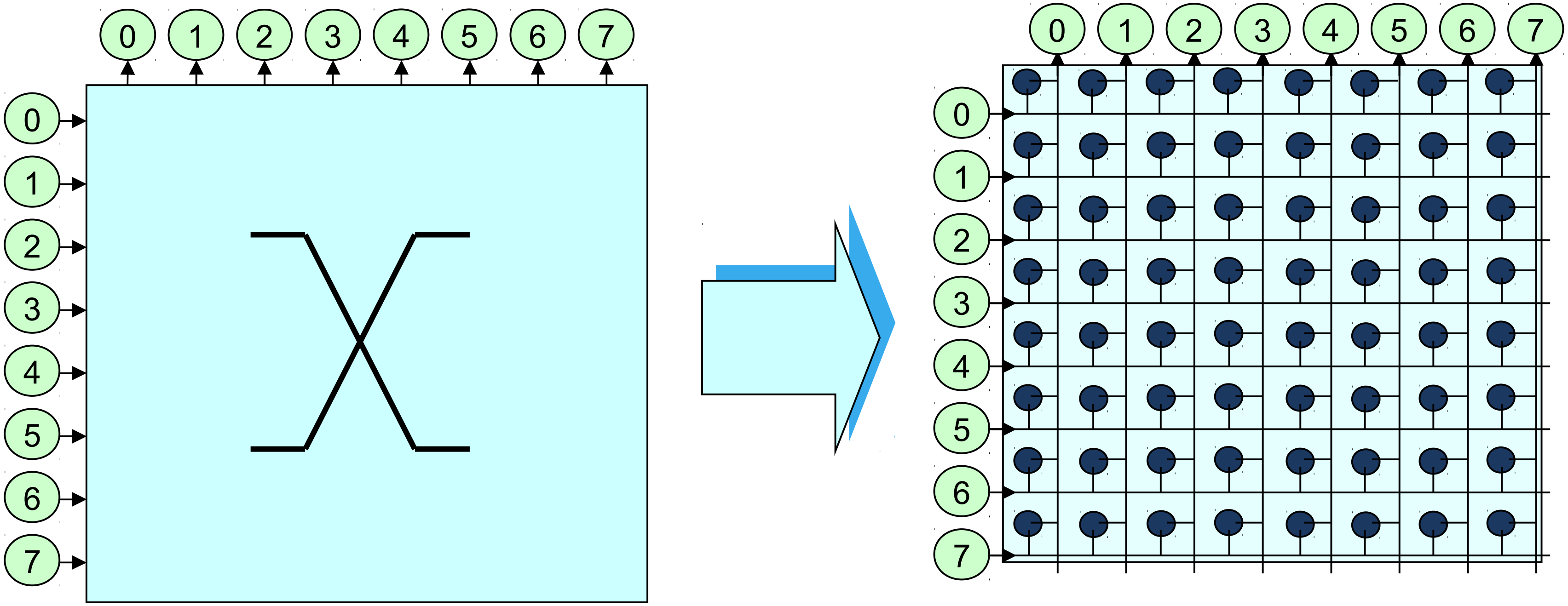
- Cost: NICs, links, switches, transceivers.
- Performance:
  - Latency (small messages, collectives,…)
  - Throughput or Bandwidth (medium to large messages)
- Packaging
- Energy consumption
- Fault-tolerance including ECC
- Partitioning and task allocation

INs Classification

- **Centralized**: Crossbar
- **Direct** Interconnection Networks.
  - Switching elements are part of the compute nodes.
  - Generally implemented using *ad-hoc* technologies.
  - Tori, Hypercubes, Dragonfly and other hierarchical networks.
- **Indirect** Interconnection Networks.
  - Only a subset of switching elements are connected to the compute nodes, by means of NICs.
  - Commonly implemented using *off-the-shelf* technologies (InfiniBand, Myrinet, Convergent Ethernet)
  - Folded Clos (Fat tree), Multi-trees.

Network Topology

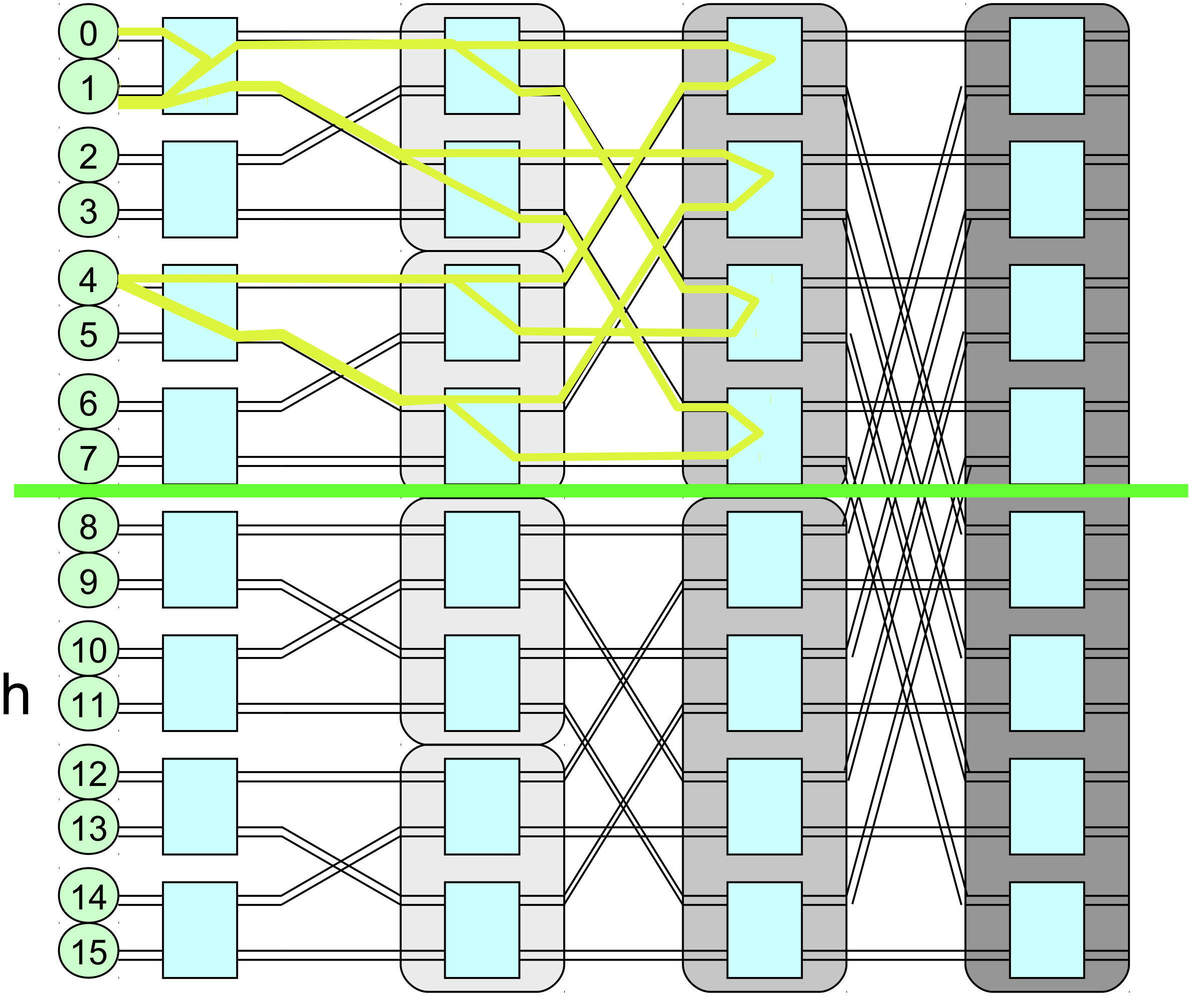
- Centralized Switched Networks
  - *Crossbar network*
    - Crosspoint switch complexity increases quadratically with the number of crossbar input/output ports, *N*, i.e., grows as O(N^2)
    - Has the property of being *non-blocking*



Network Topology

- **Indirect Networks**
  - *Bidirectional MINs*
  - Increase modularity
  - Reduce hop count
  - *Fat tree network*
    - Nodes at tree leaves
    - Switches at tree vertices
    - Total link bandwidth is constant across all tree levels, with *full bisection bandwidth*

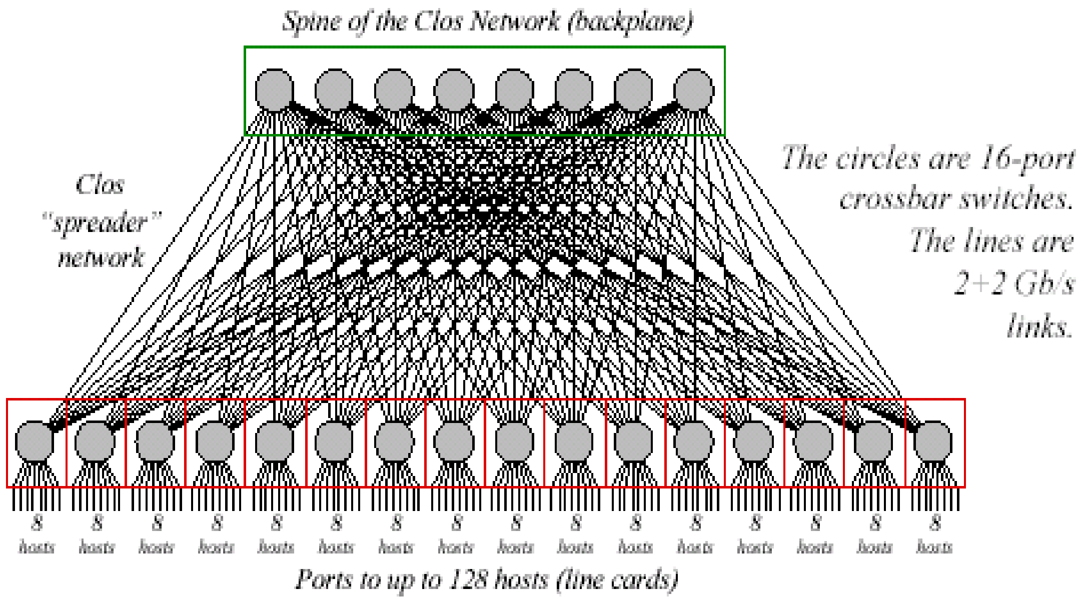
Network Bisection

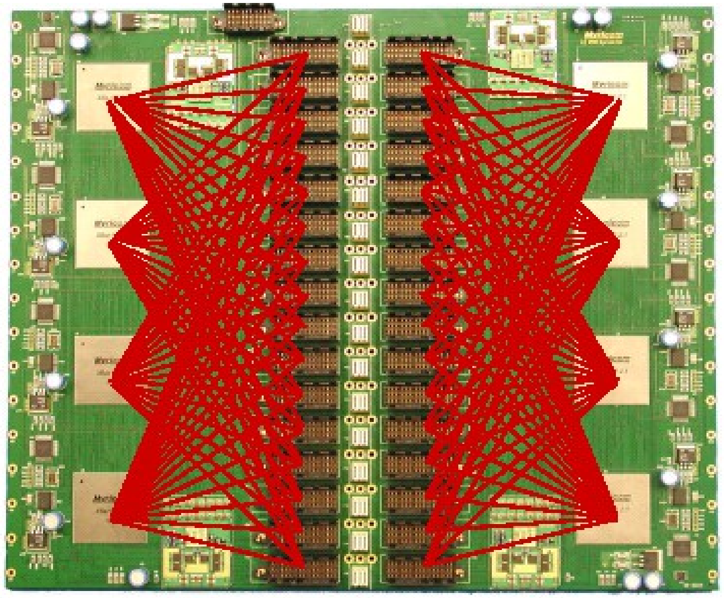


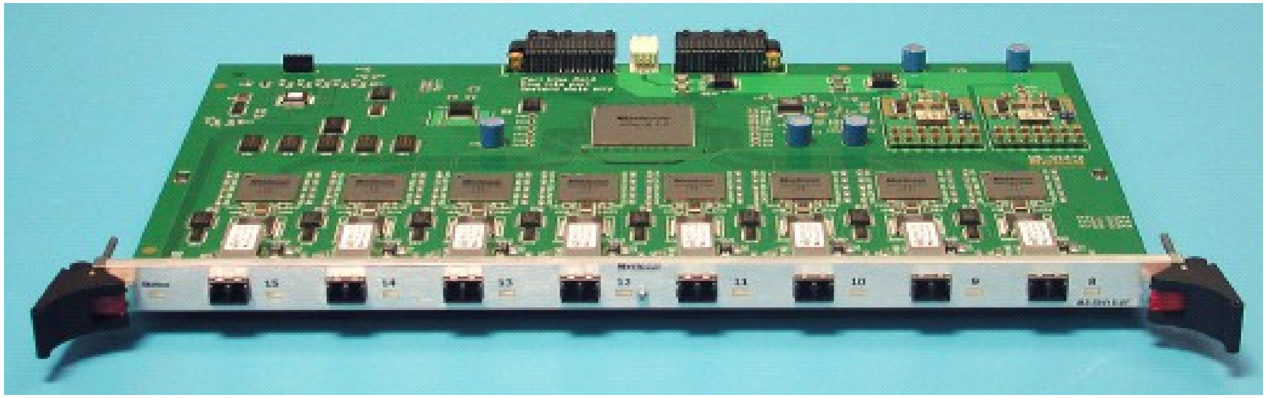
Folded Clos **=** ***Folded Benes*** **=** ***Fat tree*** network

Network Topology

- **Myrinet-2000 Clos Network for 128 Hosts**


Spine of the Clos Network (backplane)
Clos "spreader" network
The circles are 16-port crossbar switches. The lines are 2+2 Gb/s links.
8 hosts
8 hosts
8 hosts
8 hosts
8 hosts
8 hosts
8 hosts
8 hosts
8 hosts
8 hosts
8 hosts
8 hosts
8 hosts
8 hosts
8 hosts
8 hosts
Ports to up to 128 hosts (line cards)

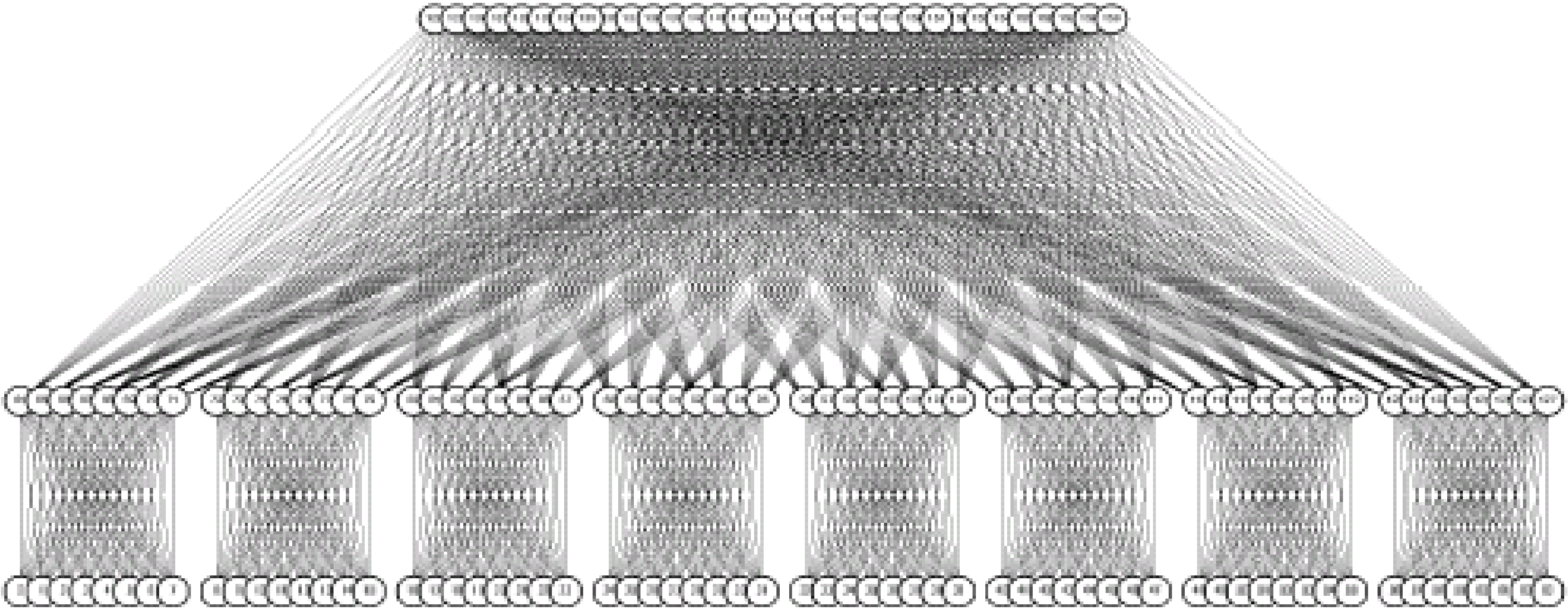




- **Backplane of the M3-E128 Switch**
- **M3-SW16-8F fiber line card (8 ports)**

Network Topology

- **Myrinet-2000 Clos Network Extended to 512 Hosts**



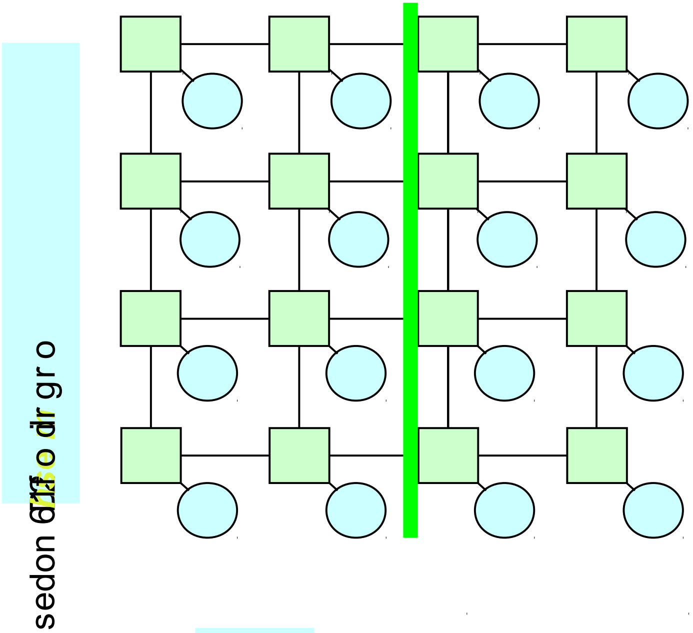
The 512 hosts connect to 8 ports on each of these 64 "leaf" switches

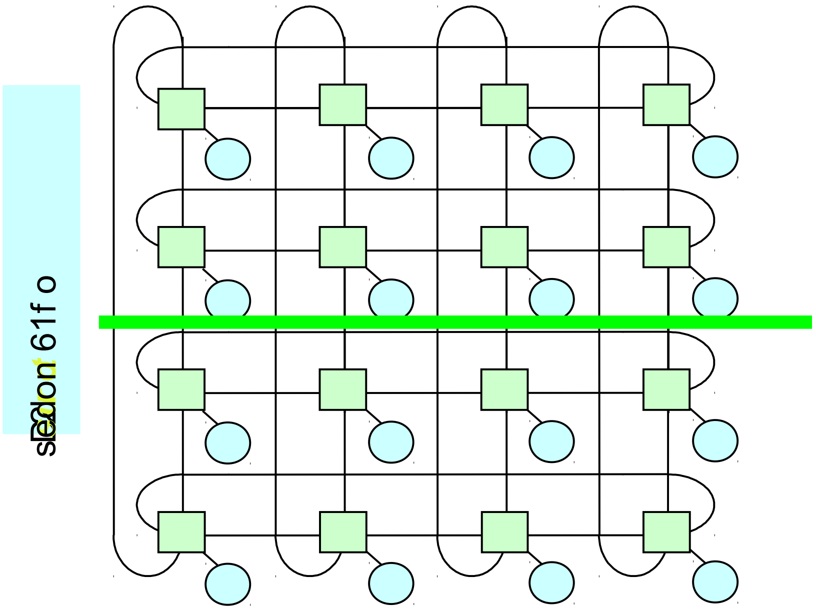
- 160 16-port switches (2,560 switch ports); 1,024 switch-to-switch links; diameter 5.
- The bisection data rate (total throughput) is 1.024 Terabits/s (128 GigaBytes/s).
- This network is <u>routine</u> today, and can scale at a similar cost/host to 8,192 hosts.

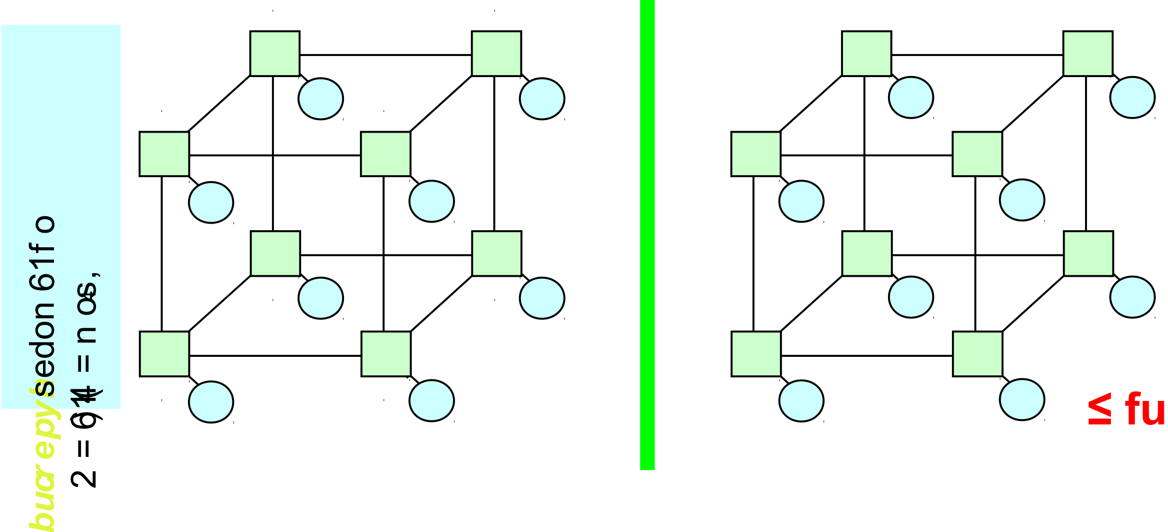
http://myri.com

Network Topology

- **Direct Networks**







Network Bisection

≤ full bisection bandwidth!

Outline

- Parallel Computers
- Basic Parallel Architectures
- Some Paradigmatic Examples
- Interconnection Networks
- A More Detailed Example: IBM BlueGene
- ExaScale Computers
- Conclusions and Questions

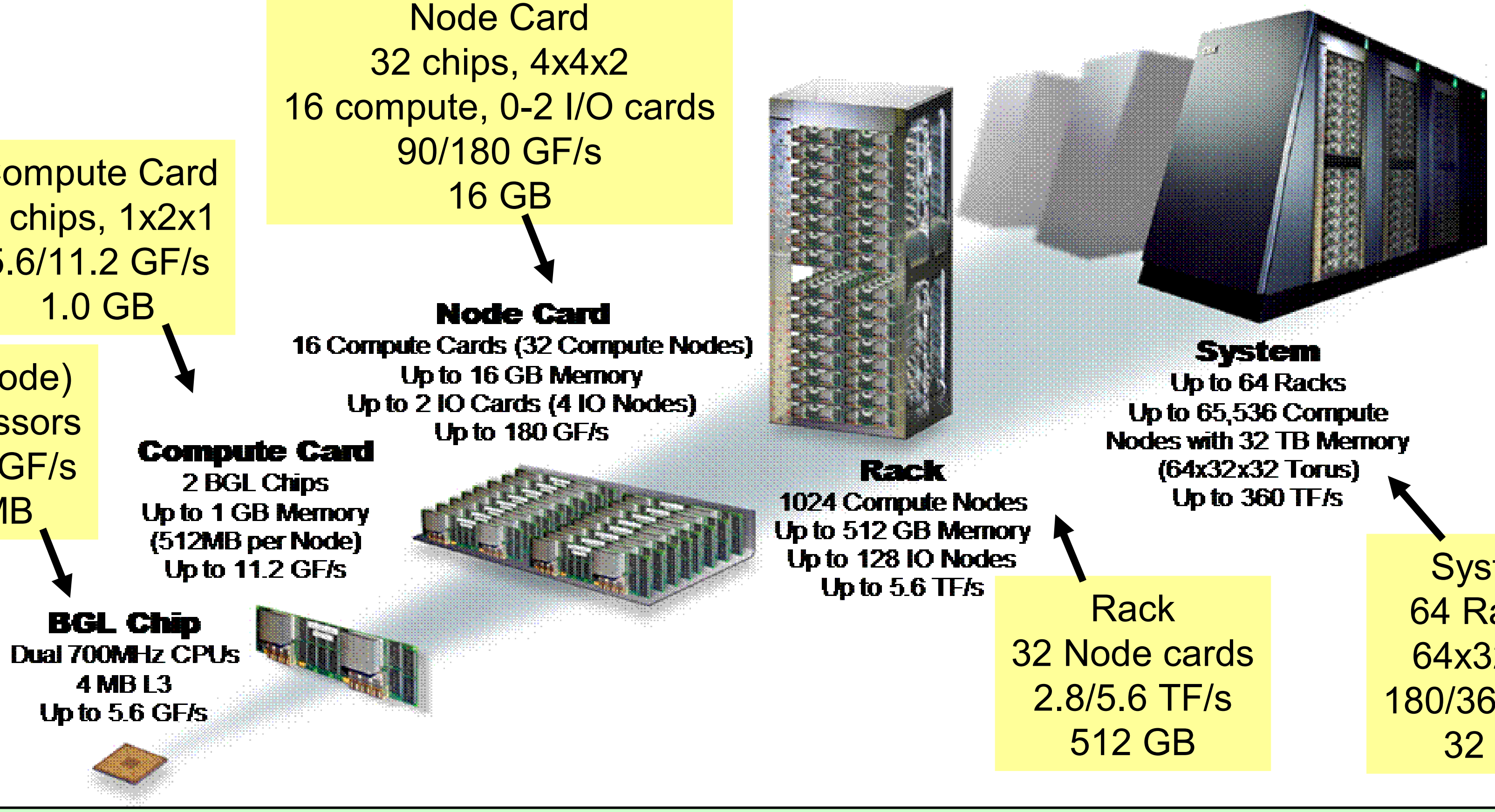
A more detailed example: IBM BlueGene

- Blue Gene/L 3D Torus Network
  - 360 TFLOPS (peak)
  - 2,500 square feet
  - Connects 65,536 dual-processor nodes and 1,024 I/O nodes
    - One processor for computation, other for communication

IBM BlueGene

- **Blue Gene/L 3D Torus Network**

www.ibm.com



Chip (node)
2 processors
2.8/5.6 GF/s
512MB

Compute Card
2 chips, 1x2x1
5.6/11.2 GF/s
1.0 GB

Node Card
32 chips, 4x4x2
16 compute, 0-2 I/O cards
90/180 GF/s
16 GB

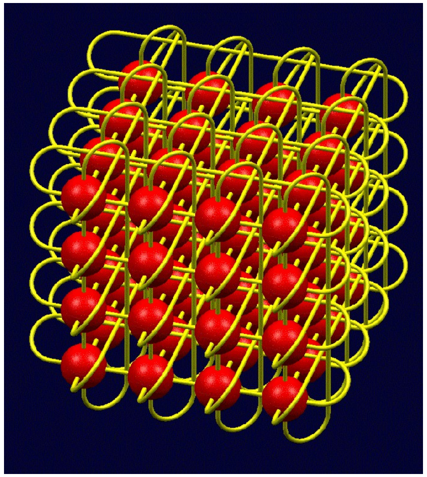
Rack
32 Node cards
2.8/5.6 TF/s
512 GB

System
64 Racks,
64x32x32
180/360 TF/s
32 TB

Node distribution: Two nodes on a 2 x 1 x 1 compute card, 16 compute cards + 2 I/O cards on a 4 x 4 x 2 node board, 16 node boards on an 8 x 8 x 8 midplane, 2 midplanes on a 1,024 node rack, 8.6 meters maximum physical link length

IBM BlueGene torus

- Blue Gene/L 3D Torus Network
  - Main network: 32 x 32 x 64 3-D torus
    - Each node connects to six other nodes
    - Full routing in hardware
  - Links and Bandwidth
    - 12 bit-serial links per node (6 in, 6 out)
    - Torus clock speed runs at 1/4th of processor rate
    - Each link is 1.4 Gb/s at target 700-MHz clock rate (175 MB/s)
    - High internal switch connectivity to keep all links busy
      - External switch input links: 6 at 175 MB/s each (1,050 MB/s aggregate)
      - External switch output links: 6 at 175 MB/s each (1,050 MB/s aggregate)
      - Internal datapath crossbar input links: 12 at 175 MB/s each
      - Internal datapath crossbar output links: 6 at 175 MB/s each
      - Switch injection links: 7 at 175 MBps each (2 cores, each with 4 FIFOs)
      - Switch reception links: 12 at 175 MBps each (2 cores, each with 7 FIFOs)



BlueGene Router

- **Blue Gene/L 3D Torus Network**

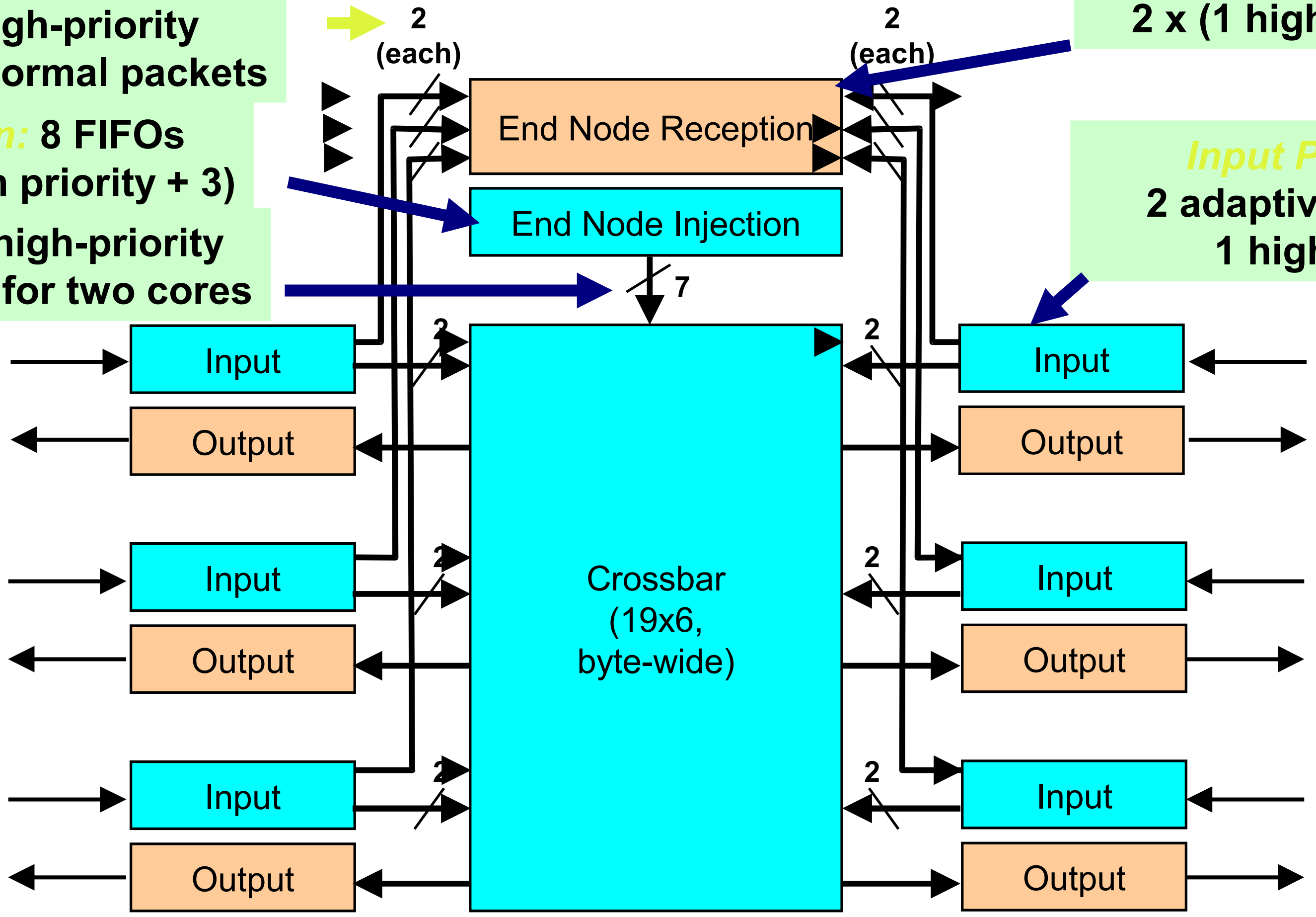
1 for high-priority and 1 for normal packets

***Injection:* 8 FIFOs**
2 x (1 high priority + 3)

1 shared high-priority and 3 each for two cores

***Reception:* 14 FIFOs**
2 x (1 high-priority + 6)

***Input Port:* 4 VCs**
2 adaptive, 1 bubble, 1 high-priority

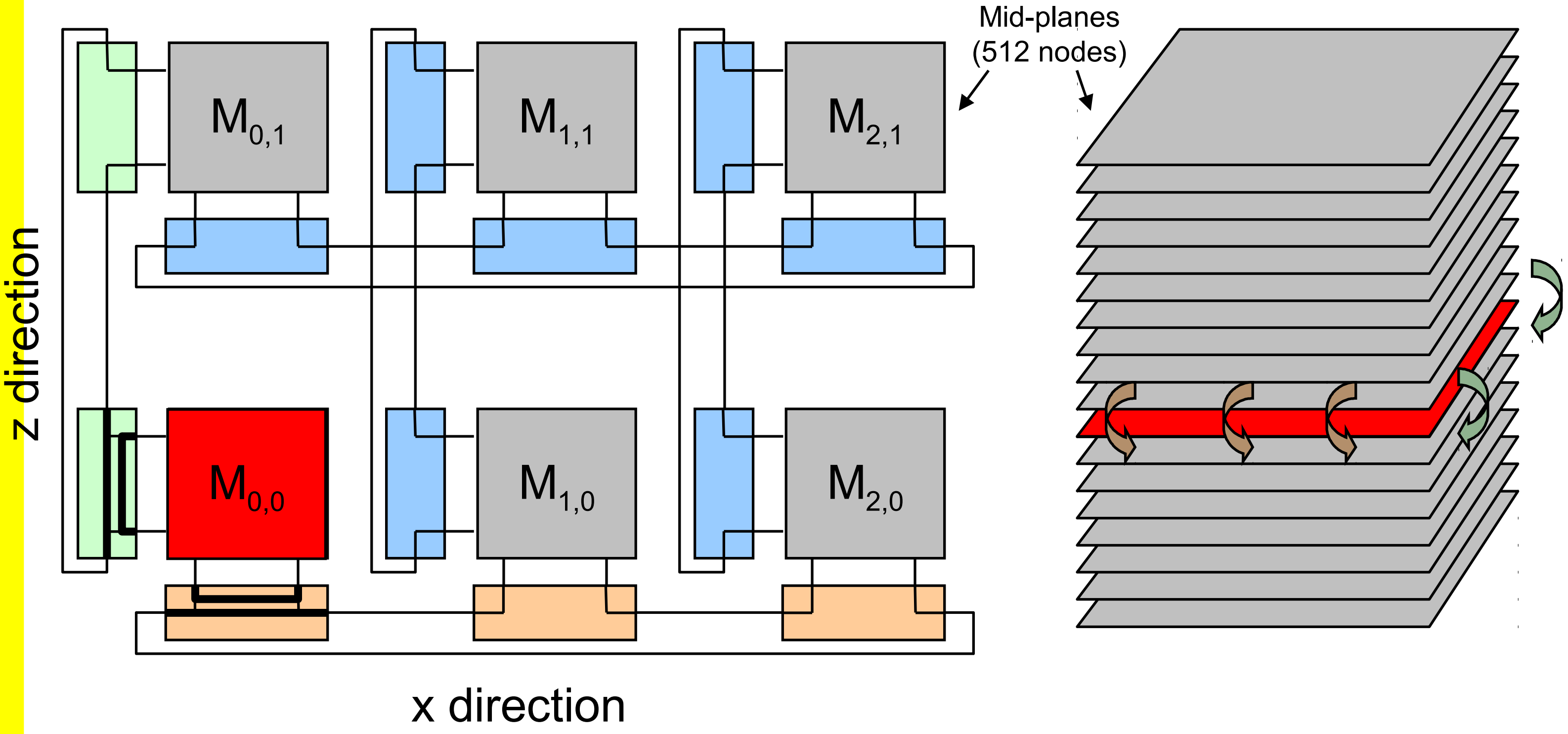


Blue Gene/L Switch

Extracted from "Blue Gene/L Torus Interconnection Network," N. R. Adiga, et al., IBM J. Res. & Dev., Vol. 49, No. 2/3, pp. 265-276, March/May 2005.

BlueGene Fault-tolerance

- Blue Gene/L Torus Network
  - Fault tolerance
    - Static fault model with checkpointing
    - Additional links boards at each rack
      - Each rack can be connected with neighbor racks
      - Internal switches allow skiping one plane (512 nodes)



Inter mid-planes switches and links operating in normal "pass-thru" mode

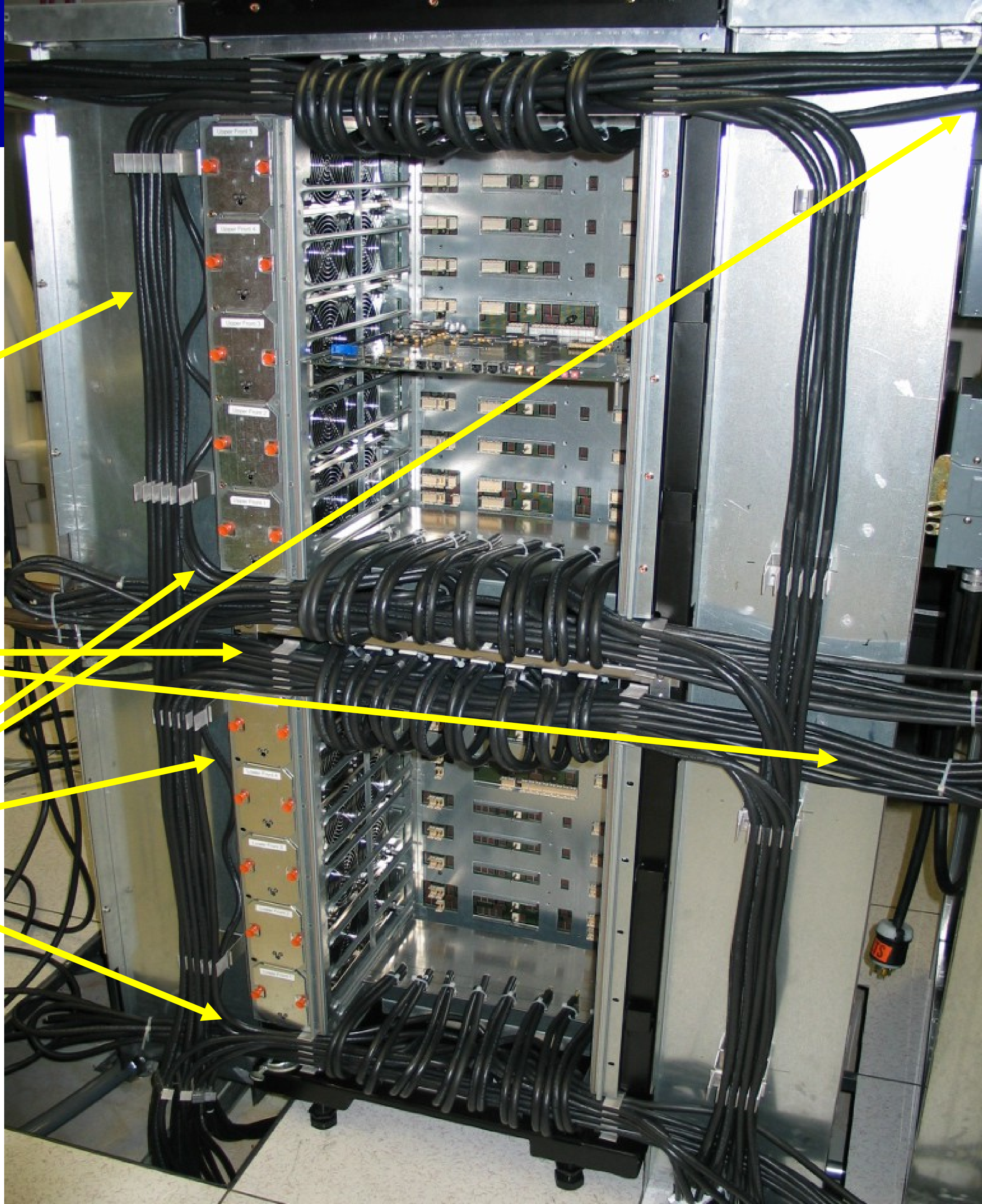
Inter mid-planes switches and links operating in fault "**bypass**" mode

Topology remains the same

IBM BlueGene

- Blue Gene/L 3D Torus Network
  - Routing
    - Fully-adaptive deadlock-free routing based on bubble flow control
      - DOR and bubble mechanism are used for escape path
    - Hint (direction) bits at the header
      - “100100” indicates the packet must be forwarded in X+ and Y-
      - Neighbor coordinate registers at each node
        - A node cancels hint bit for next hop based on these registers
    - A bit in the header allows for broadcast
    - Dead nodes or links avoided with appropiate hint bits

BG/L rack, cabled



X Cables

Y Cables

Z Cables

Examples of Interconnection Networks

- Blue Gene/L 3D Torus Network
  - Flow control
    - Credit-based (token) flow-control per VC buffer
      - A token represents a 32-byte chunk
    - Bubble rules are applied to the escape VC
      - Tokens for one full-sized packet is required for a packet in the escape VC (bubble) to advance
      - Tokens for two full-sized packets are required for
        - A packet entering the escape VC or
        - A packet turning into a new direction
        - An adaptive VC packet enters the escape VC
      - Dimension-ordered routing on the escape VC

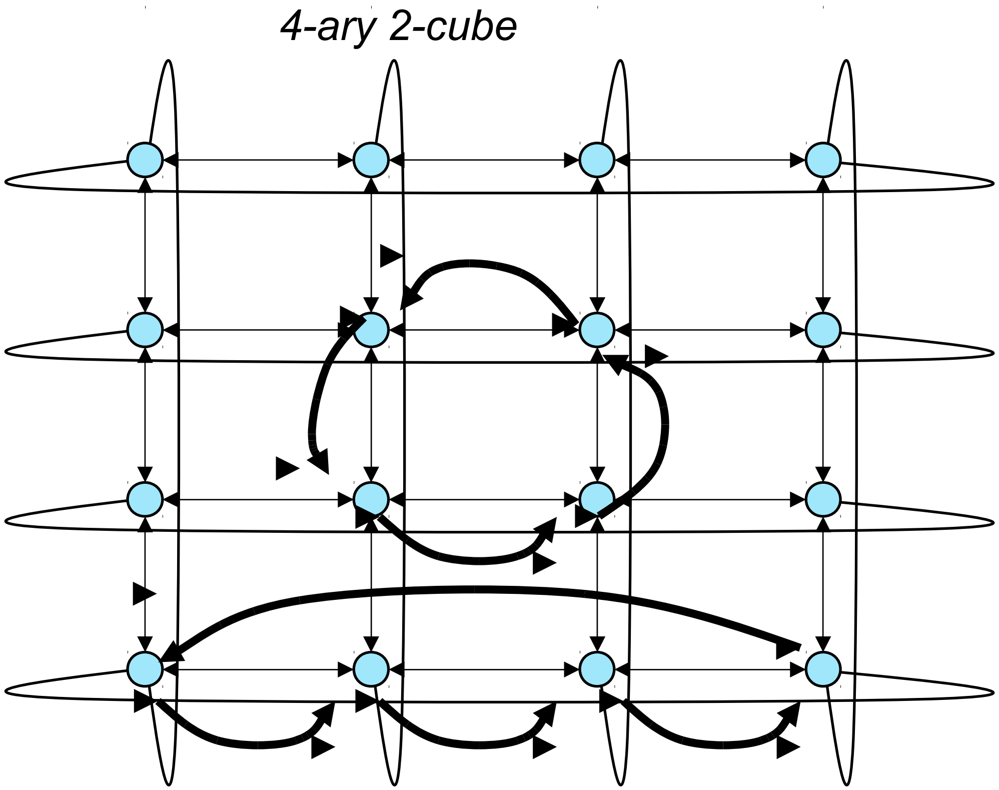
Bubble Switching

- Bubble switching is a combination of a restricted injection flow control mechanism and traditional routing schemes.
- The Bubble Flow Control (BFC) is a deadlock avoidance method proposed by our group (valid for k-ary n-cube networks, but no only).
- The idea: "To maintain a buffer space (a bubble) for at least one packet for any set of physical channels that are involved in any possible static dependency cycle".

Bubble History

- The genesis of the idea: meters in freeways;..; Roscoe (1987).
- From 1990 our group is applying this idea to interconnection networks.
- In 1993 the PhD thesis of A. Arruabarrena shows the first important performance results for torus networks and VCT.
- In 1997, DEC was interested in the solution presented in a HiPC97 paper.
- The adaptive version, as the one presented in ICPP99 (JPDC-2001) was adopted by IBM as the solution for routing packets in the IBM Blue Gene/L.

Deadlocks in *k*-ary *n*-cubes



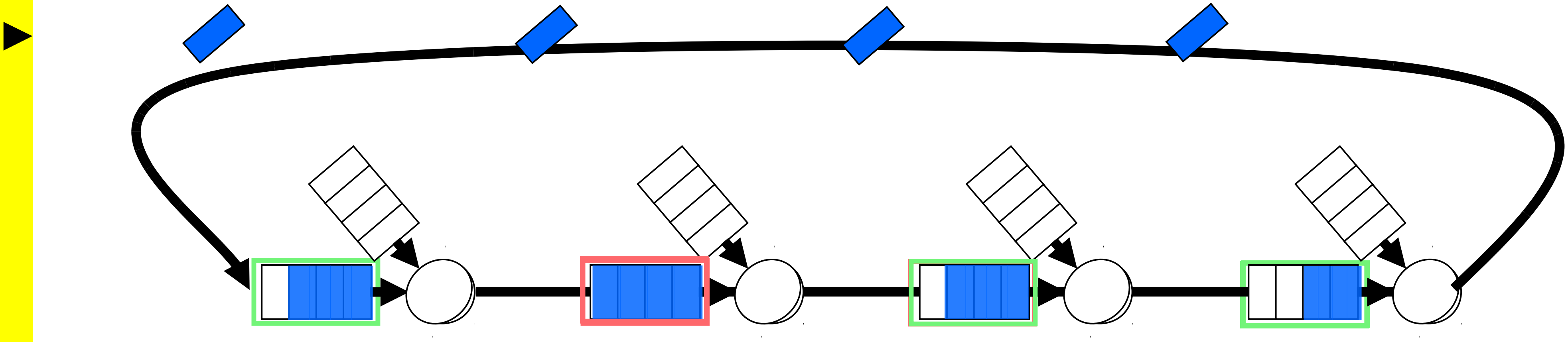
Most common approaches employed

Usually to avoid deadlock is necessary to sacrifice performance or increase resources. Common approaches:

- Reduce routing flexibility
    - Dimension Order Routing (DOR) is enough for meshes
    - Up*/down* in tree networks
- Supplying additional resources
    - Virtual Channels to avoid cyclic dependencies: Dally's approach for k-ary n-cubes an other networks
- Non minimal routing
    - Chaos routing,
- ….

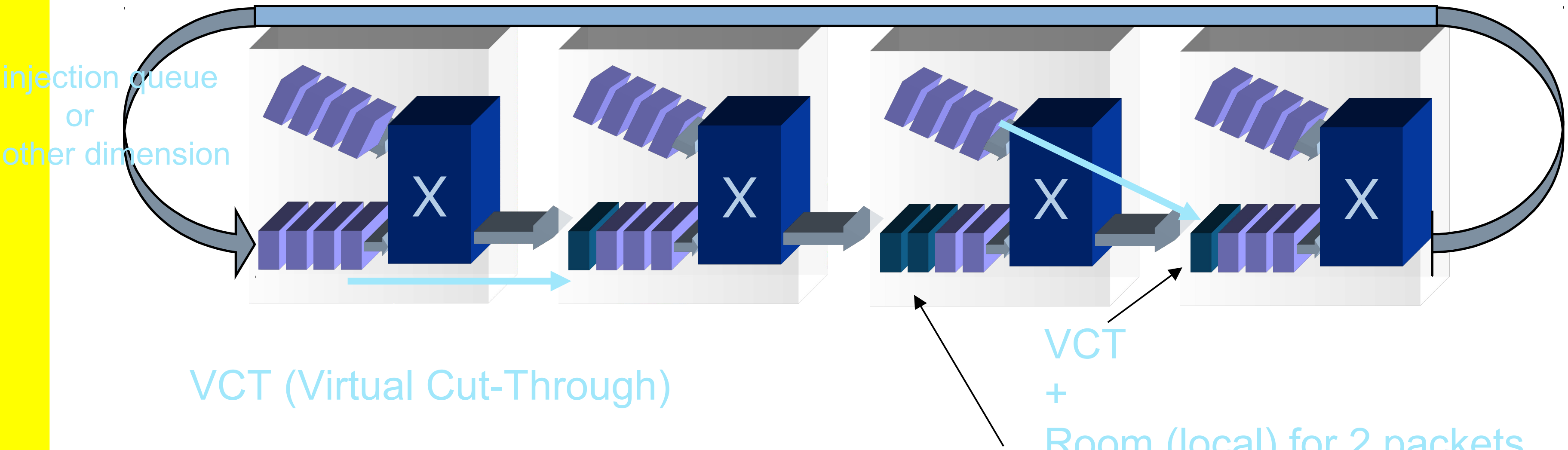
Bubble Flow Control

- Does not exhaust resources in any potential deadlock situation (static cyclic dependencies).
  - Is a method that restricts the injection of packets
  - A packet can be injected only if after the injection there is room at least for one* packet.
  - Decision taken with local available information



BFC

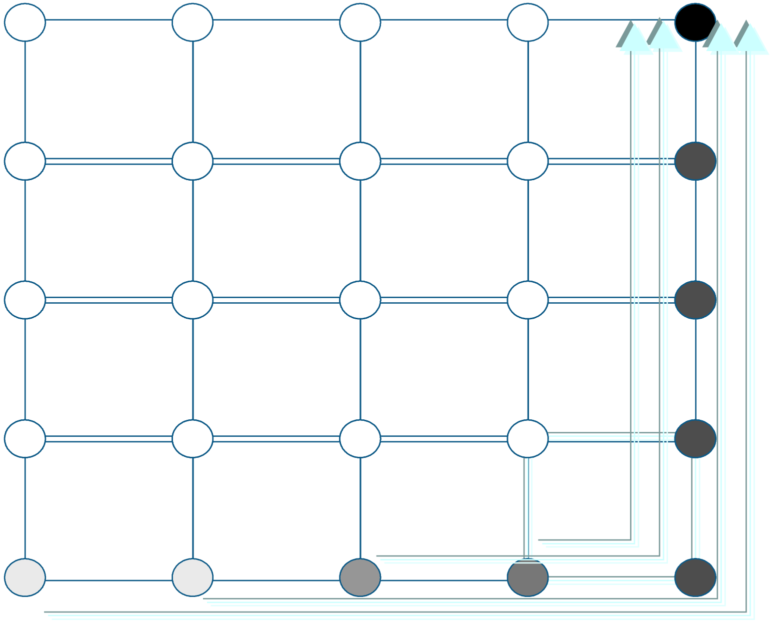
- BFC avoids packet deadlocks in a k-ary n-cube network without using virtual channels.
- Restrictions:
  - Dimension Order Routing: A dimension change is consider as a new injection.
  - Virtual Cut Through



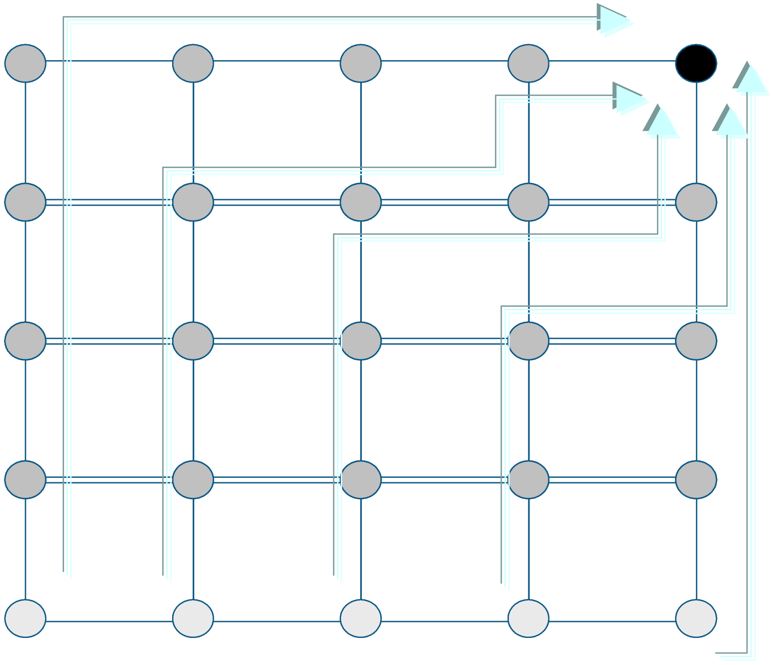
VCT (Virtual Cut-Through)

Adaptive routing

- Not always is true, but lot of times you can get more throughput using an adaptive solution..



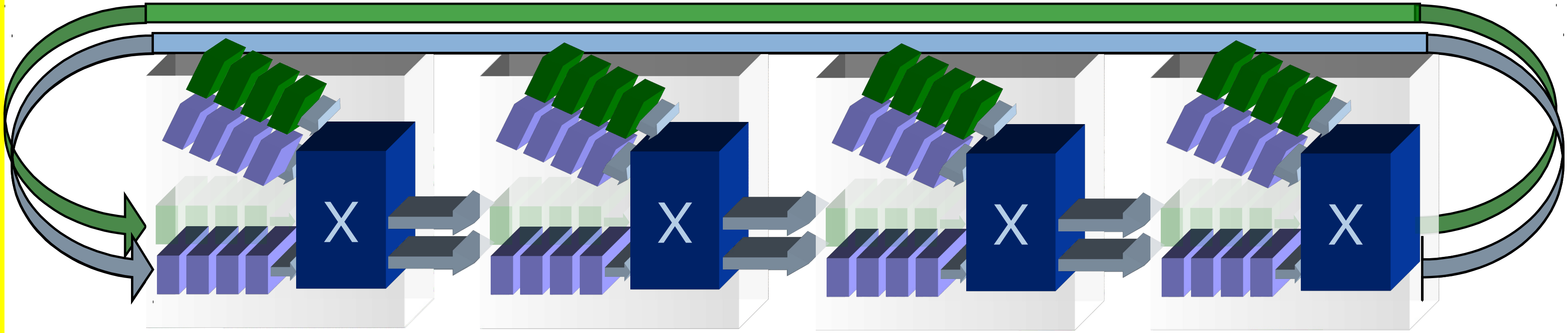
Deterministic Routing (DOR)



Adaptive Routing

ABR (Adaptive Bubble Routing)

- Adding a virtual channel we get a completely adaptive network.
- This new virtual channel can be used without restrictions (just virtual cut-through).
- The “old” channel is used as an “escape” channel. A packet can enter in this escape channel if the bubble condition holds.



Adaptive Bubble Routing

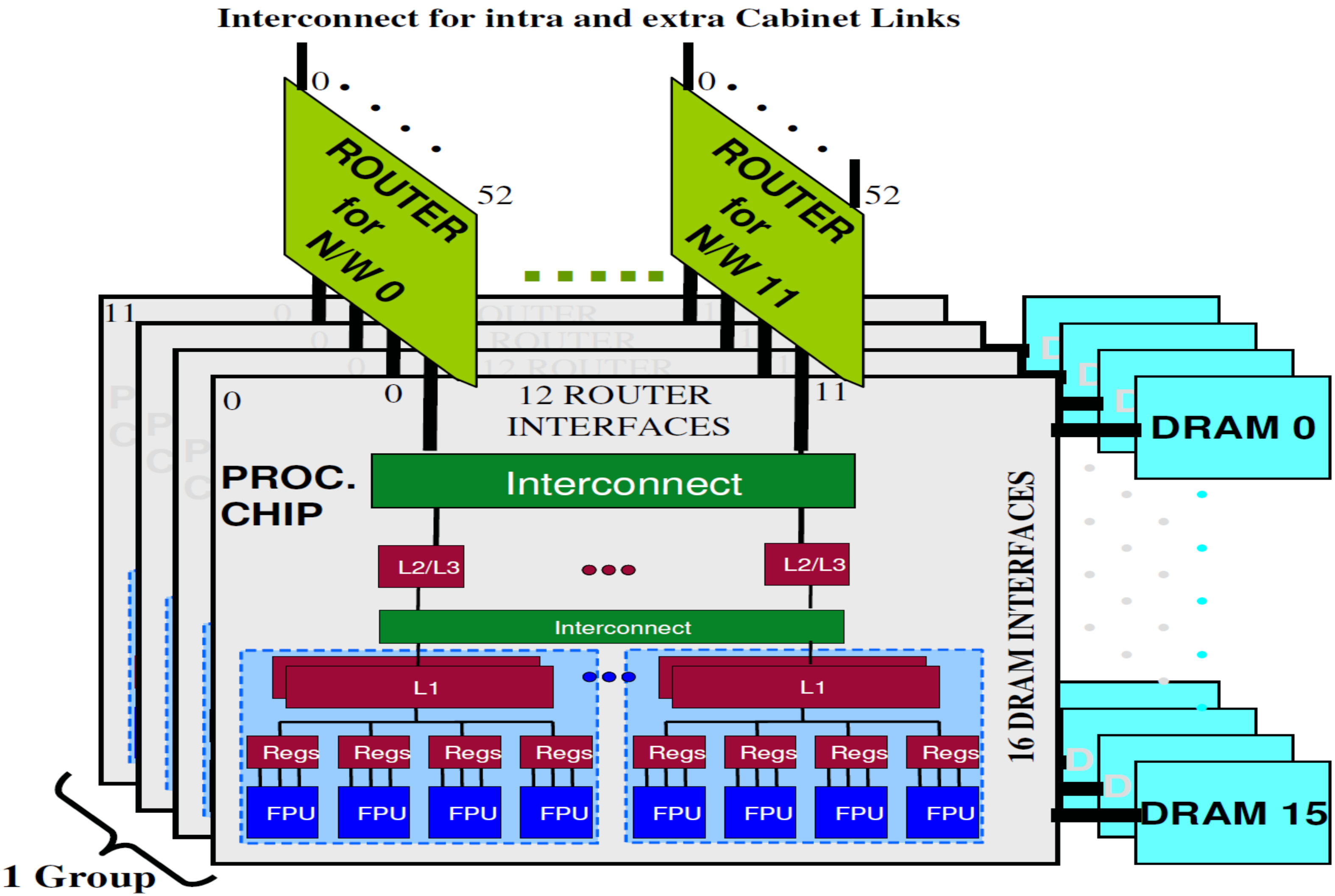
- Low cost adaptative routing
- Good performance
    - Highly stable behavior
    - It overcomes Dally’s+Duato approach (even without take into account the lower cost)
- Bubble Routing can be applied in other contexts
    - Irregular Networks
    - Fault-tolerant Networks
    - Hierarchical Networks
    - Mesh networks and adaptive routing

Outline

- Parallel Computers
- Basic Parallel Architectures
- Some Paradigmatic Examples
- Interconnection Networks
- A More Detailed Example: IBM BlueGene
- ExaScale Computers
- Conclusions and Questions

ExaScale System

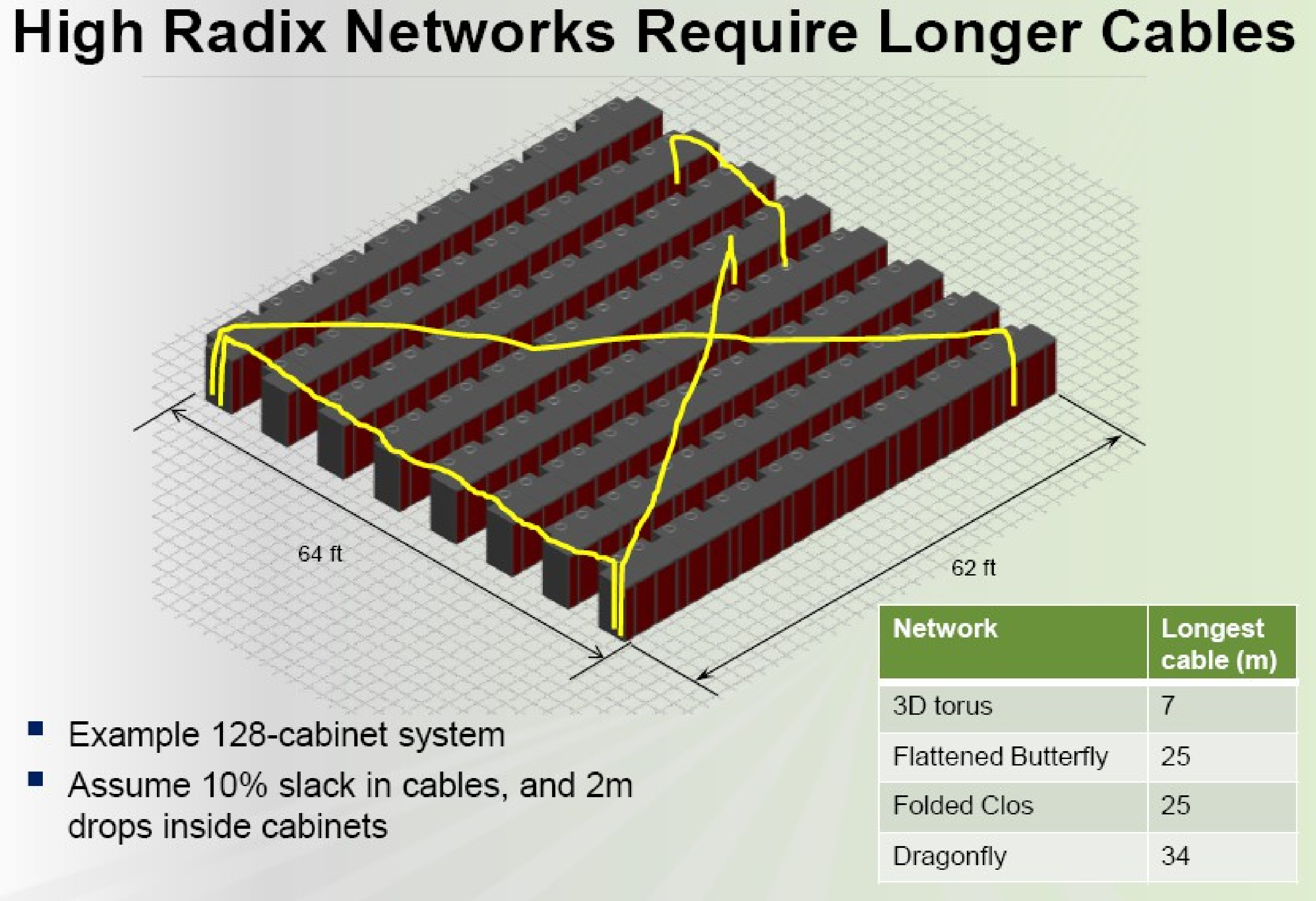
ExaScale System



1 Cabinet Contains 32 Groups on 12 Networks

ExaScale System

ExaScale System



High Radix Networks Require Longer Cables

- Example 128-cabinet system
- Assume 10% slack in cables, and 2m drops inside cabinets

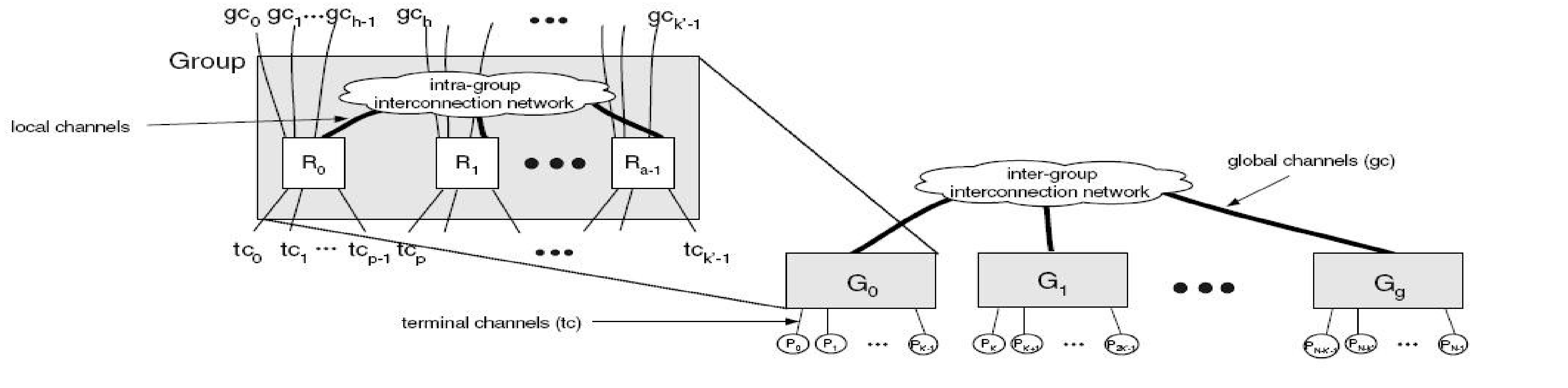
Network	Longest cable (m)
3D torus	7
Flattened Butterfly	25
Folded Clos	25
Dragonfly	34

Conclusions

- Parallel computing is everywhere
- Good Science and Engineering need supercomputing
- Interconnection networks are pervasive
- A field in continuous change
- How to program these exotic architectures
- How to obtain sustainable performance
  - Some of these questions will be answer next

- Any question?
- Thanks

Dragonfly (ISCA 2008)
3 levels: Router, Group, System



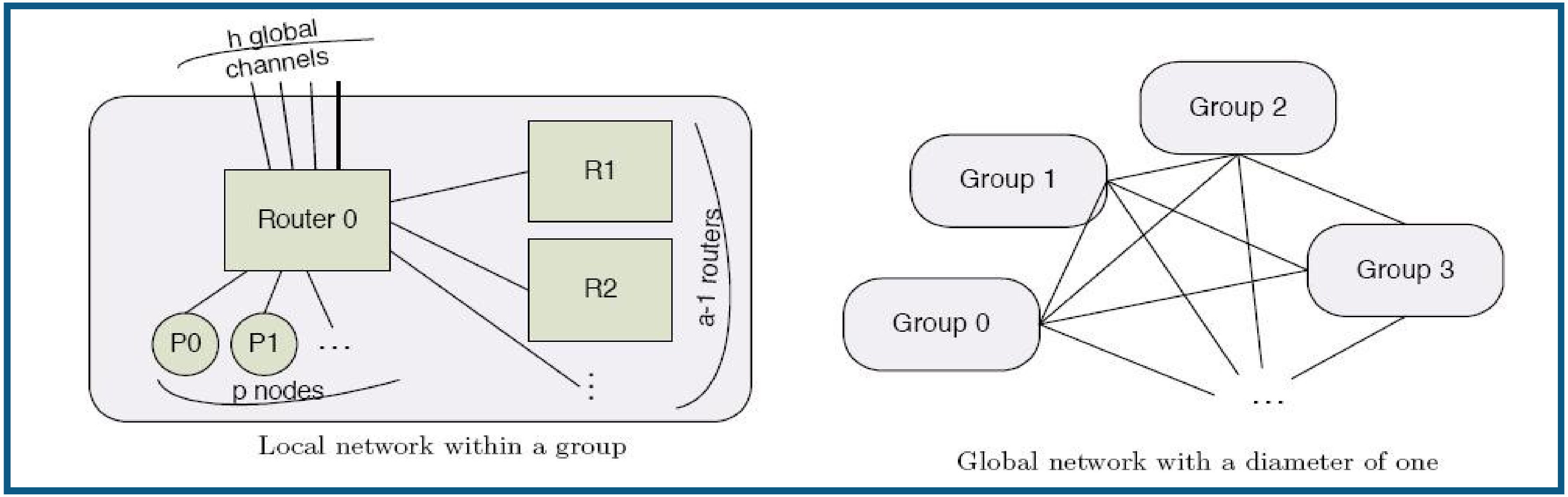
A high-level block diagram of a dragonfly topology and a diagram of a group or a virtual router.

- N **Number of network terminals**
- p **Number of terminal connected to each router**
- a **Number of routers in each group**
- k **Radix of routers**
- k' **Effective radix of group (or the virtual router)**

- h **Number of channels within each router used to connect to other groups**
- g **Number of groups i the system**
- q **Queue depth of an out put port**
- q_{vc} **Queue depth of an individual output VC**
- H **Hop count**
- Out_i **Router output port i**

Dragonfly

- To balance channel load on load-balanced traffic → $a=2p=2h$ → 2:1 ratio (packet traverses 2 local channels for 1 global and 1 terminal channel)
- Deviations from 2:1 ratio → global channels should remain fully utilized → $a \geq 2h,\ 2p \geq 2h$
- Arbitrary networks can be utilized for the inter/intra-group network

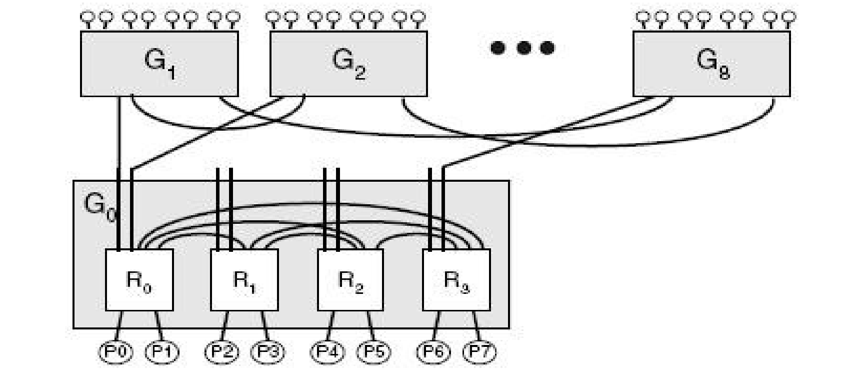


Local network within a group

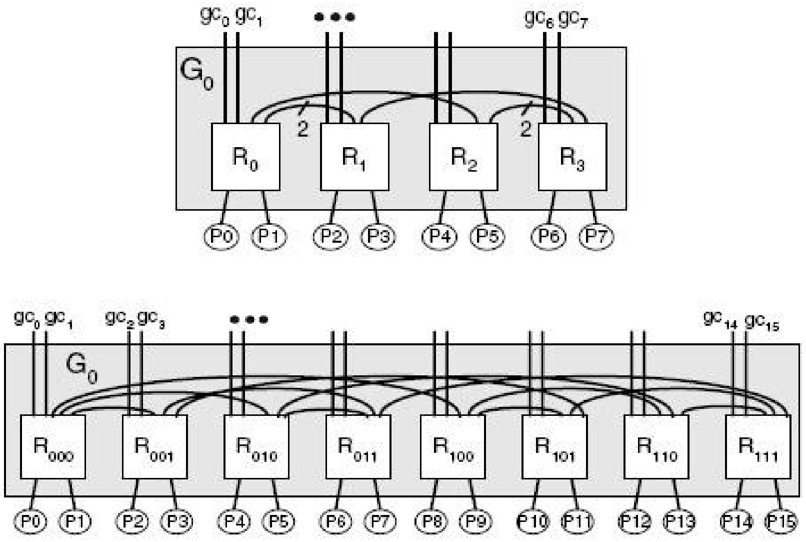
Global network with a diameter of one

Dragonfly

- Dragonfly maximum size: $N = a{\cdot}p{\cdot}(ah+1)$ → exactly 1 connection between each pair of groups
- Smaller dragonflies → more global connections out of each group than there are other groups
- Excess global connection distributed over the groups with each pair of groups connected by at least $(ah+1)/g$ channels
- a, p, h can have any values



An example block diagram of a dragonfly topology with $N = 72$.



Alternative organization of a group in dragonfly.

Dragonfly Routing

- MIN: minimal paths l-g-l
- Valiant: randomized paths l-g-l-g-l
  - Routing each packet first to a randomly-selected intermediate group *Gi* and then to its final destination *d*
- Universal Globally-Adaptive Load-balanced (UGAL)
  - Chooses between MIN and VAL on a packet-by-packet basis to load balance the network
  - Choice is made using queue length and hop count to estimate network delay and choose path with minimum delay
    - UGAL-L
      - Uses local queue information at the current node
    - UGAL-G
      - Uses queue information of all the global channels in Gs (represents an ideal but very difficult to implement)

Universal Global Adaptive Routing (UGAL)

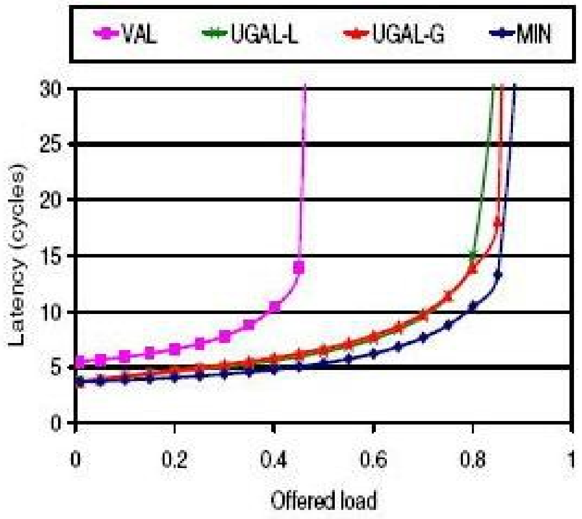
- Delay of a route estimated by the product of path queue length (Q)
- Routes minimally if:

$$Q_{min} \times H_{min} \leq Q_{val} \times H_{val} + T$$

- T: routing threshold constant → added to original UGAL's algorithm to balance between benign and adversarial traffic patterns

Dragonfly routing (uniform)

- Evaluation
  - Dragonfly of size 1K node: p = 4, h = 4, a = 8
  - Benign synthetic traffic: Uniform random
    - MIN: sufficient to provide low latency and high throughput
    - VAL: load-balancing doubles the load on the global channels → achieves half of the network capacity
    - UGAL-L and UGAL-G: approach the throughput of MIN but with slightly higher latency near saturation

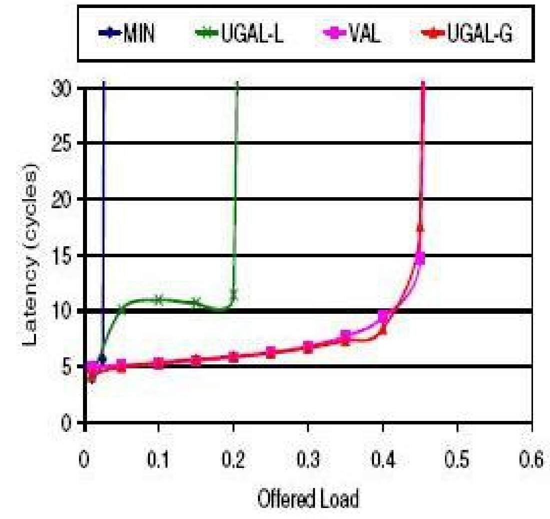


(a)

Routing algorithm comparison on the dragonfly for uniform random traffic

Dragonfly routing (worst case)

- Worst-case traffic pattern: Each node in G_i send traffic to a randomly selected node in G_i+1
  - – MIN: forwards all traffic across the single global channel to group G_i+1 → Throughput limited to $1/(a \cdot h)$
  - – VAL: Achieves slightly under 50% throughput (maximum possible with WC traffic)
  - – UGAL-G similar throughput as VAL
  - – UGAL-L limited throughput &high avg. packet latency at intermediate node



(b)

Routing algorithm comparison on the dragonfly for adversarial traffic pattern.

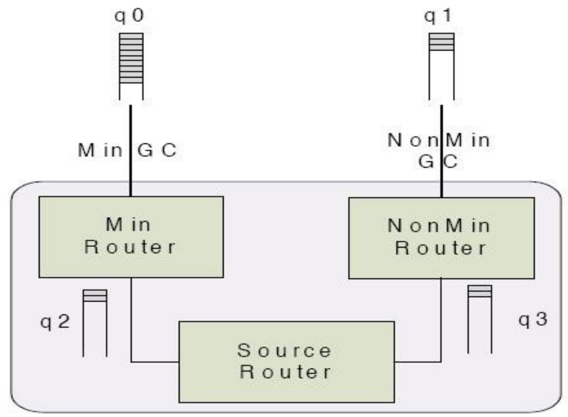
How the decission is made

A packet in R1 is making its adaptive routing decision of routing either minimally through *gc*0 or non-minimally through *gc*7

The routing decision needs to load balance global channels and ideally, the channel utilization can be obtained from the queues associated with global channels, *q*0 and *q*3

However, *q*0 and *q*3 queue informations are only available at R0 and R2 and not at R1

In this example, *q*1 reflects the state of *q*0 and *q*2 the state of *q*3. When either *q*0 or *q*3 is full, the flow control provides backpressure to *q*1 and *q*2 as shown with the arrows



Congestion on the global channels not directly connected to the source router ($q0$,$q1$) must be inferred through local channels ($q2$,$q3$). $q2$ does not reflect congestion until $q0$ is filled.

Dragonfly adaptive routing

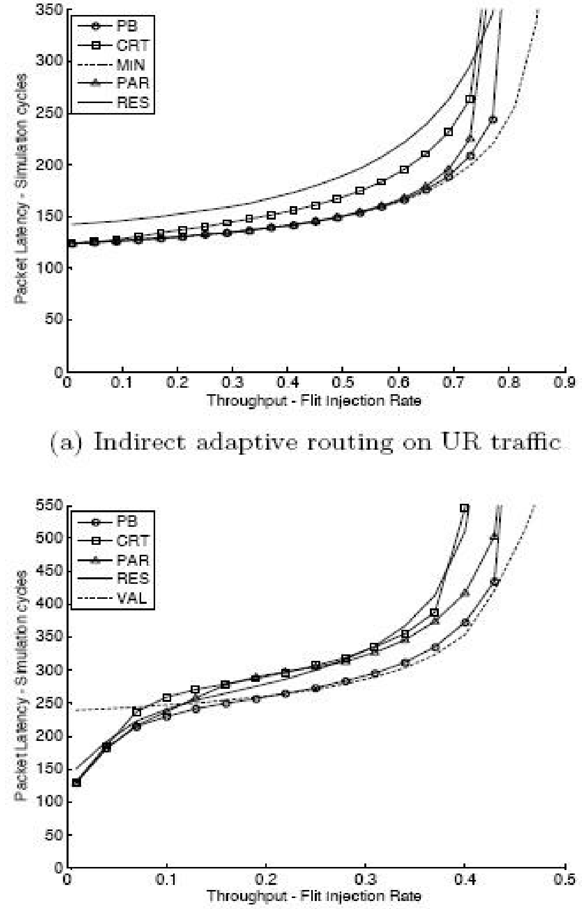
- Global channels, not router outputs, need to be balanced
- Each router must pick a global channel using only local information that depends indirectly on the state of global channels
- With the dragonfly topology local queues only sense congestion on global channels via backpressure over the local channels
- With local channels overprovisioned many packets are enqueued on the overloaded minimal route before source router sense congestion → Degradation in throughput and latency

Indirect Adaptive Routing (ISCA 2009)

- Four IAR methods:
  - Credit Round trip (CRT)
  - Progressive Adaptive Routing (PAR)
  - Piggyback (PB)
  - Reservation (RES)
- Each method decides whether to route a packet minimally or non-minimally using information not directly available at the source router

Some IAR Results

- Steady State Traffic Performance
  - UR:
    - PB & PAR closely match performance of MIN
    - CRT deviates form ideal performance earlier but maintains stable
    - RES performance comparable to other methods. Higher latency at low injection rates due to reservation flit round-trip latency
  - WC*n*:
    - All start with lower latency than VAL and then converge to VAL’s performance as load increases
  - PB
    - Broadcast link information → more accurate routing decision → lowest latency under load



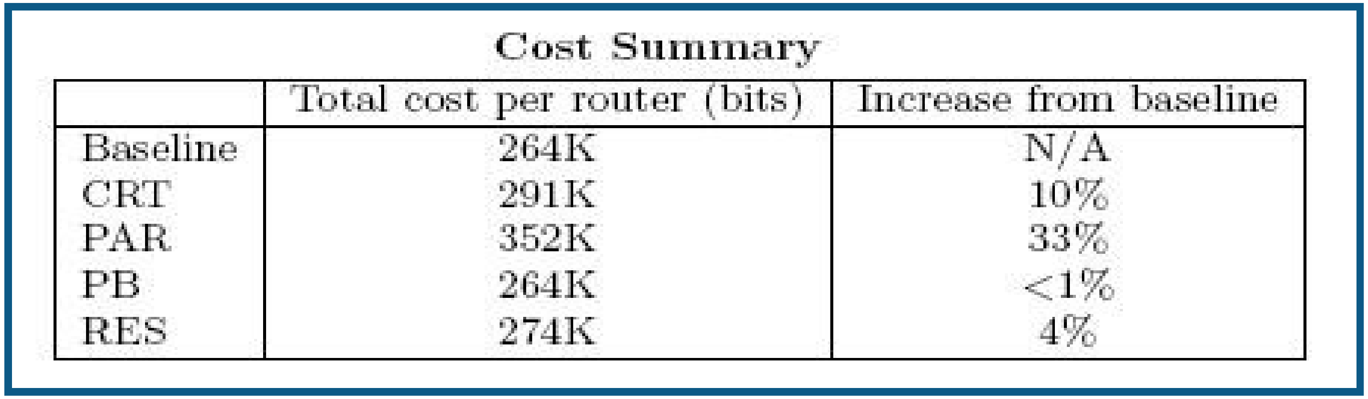
(a) Indirect adaptive routing on UR traffic

(b) Indirect adaptive routing on WC traffic

Four indirect adaptive routing methods in a dragonfly network with 1D local networks

Implementation cost

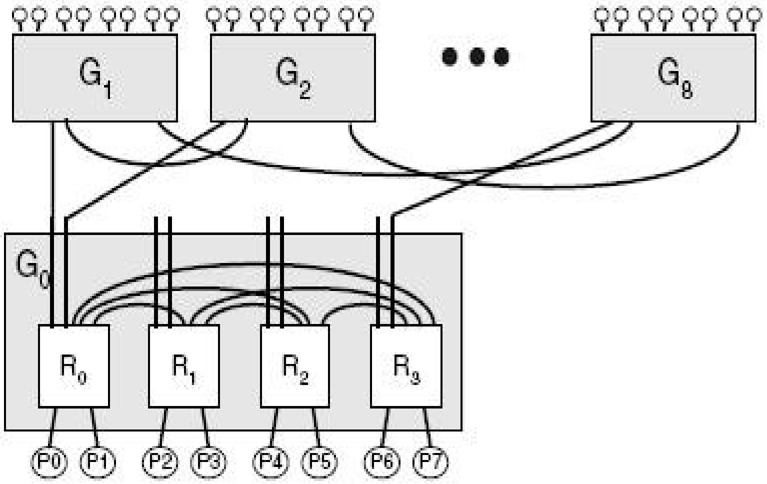
- Baseline router
  - UGAL route computation implemented on it
  - Flit size = 64 bits
  - Packet size = 10 flits
  - 15 ports, 3 VCs per input port, 256 flits buffer size per global VC, 32 flits per local VC → 264K bits per router
- PB is the most cost-effective IAR method
  - PAR has the highest cost because of additional VCs per input port



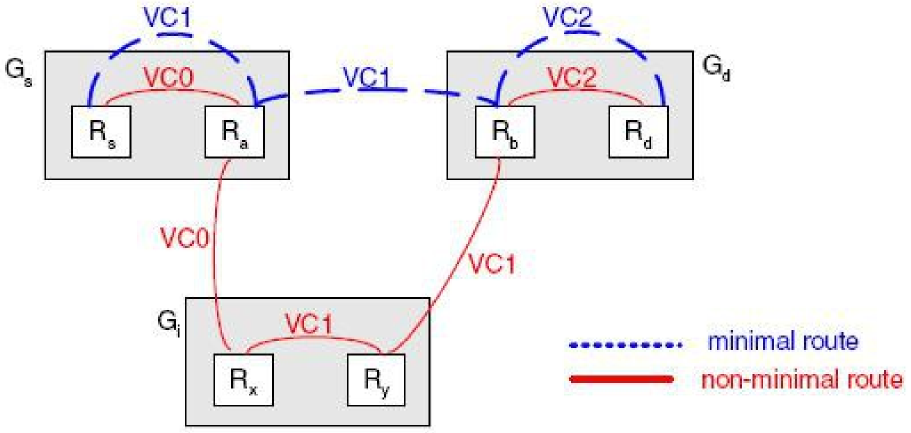
Cost Summary

	Total cost per router (bits)	Increase from baseline
Baseline	264K	N/A
CRT	291K	10%
PAR	352K	33%
PB	264K	<1%
RES	274K	4%

Virtual Channels



An example block diagram of a dragonfly topology with $N = 72$.



Virtual channel assignment to prevent routing deadlock in a dragonfly topology with both minimal and nonminimal routing.