18-447
Computer Architecture
Lecture 33: Interconnection Networks

Prof. Onur Mutlu
Carnegie Mellon University
Spring 2015, 4/27/2015

Logistics

- Lab 7 and Lab 8
- Final Exam
- Midterm II scores
- Course grades so far
- Course evaluations
- 740 next semester
- Plans for Wed and Fri lectures

Lab 6 Extra Credit

- 2.5% Ashish Shrestha (ashresth)
- 2.5% Amanda Marano (amarano)
- 2.5% Pete Ehrett (wpe)
- 2.0% Jared Choi (jaewonch)
- 2.0% Akshai Subramanian (avsubram)
- 2.0% Sohil Shah (sohils)
- 2.0% Raghav Gupta (raghavg)
- 1.5% Kais Kudrolli (kkudroll)

Lab 7

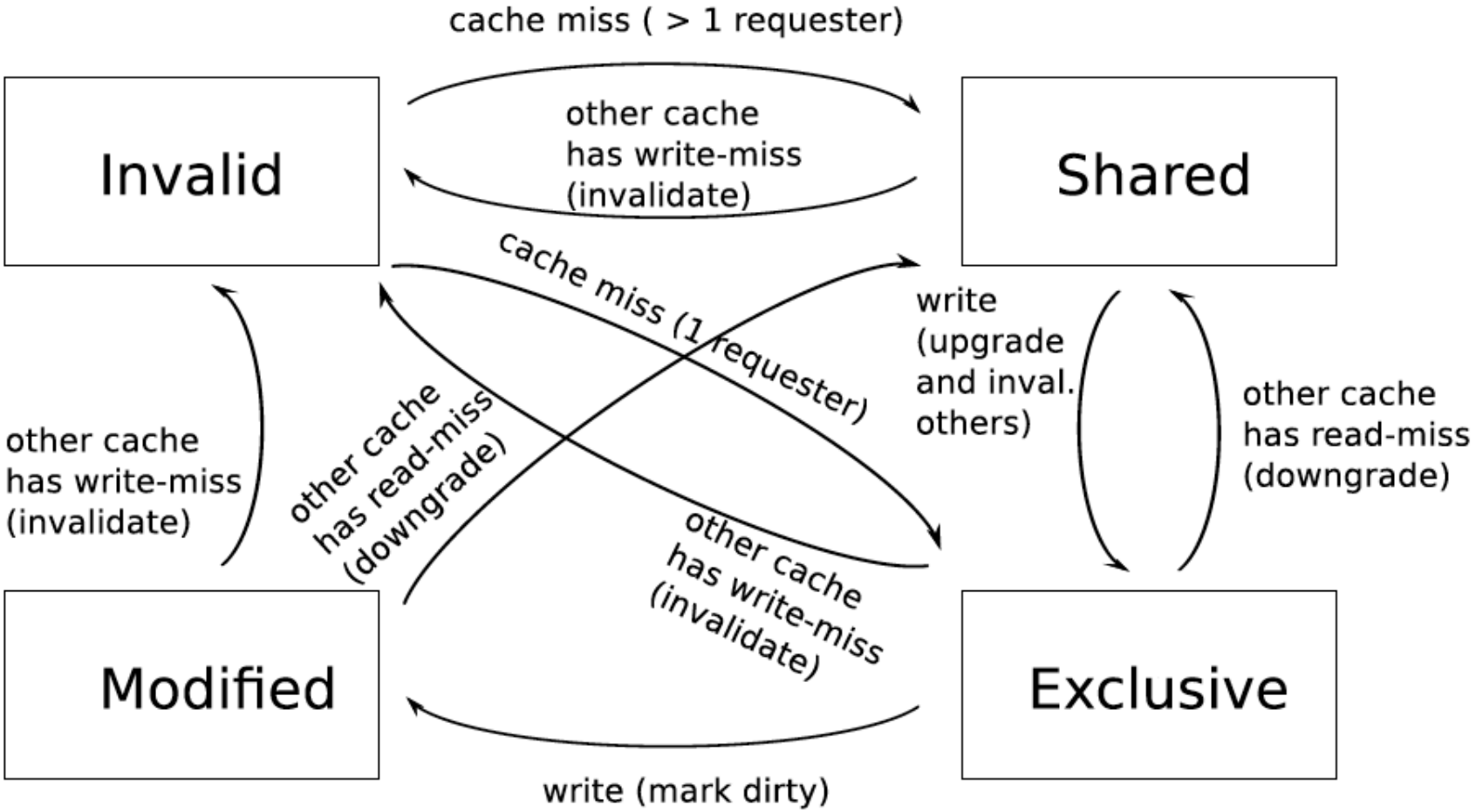
- Abstract, cycle-level modeling of the memory hierarchy
  - L2 Cache and Main Memory Controllers
  - A key part of all modern computing systems today

- You can submit until May 1 and still get full credit
- Feel free to submit the Extra Credit portion as well
  - Prefetching
  - You can get up to 2% of course grade as extra credit

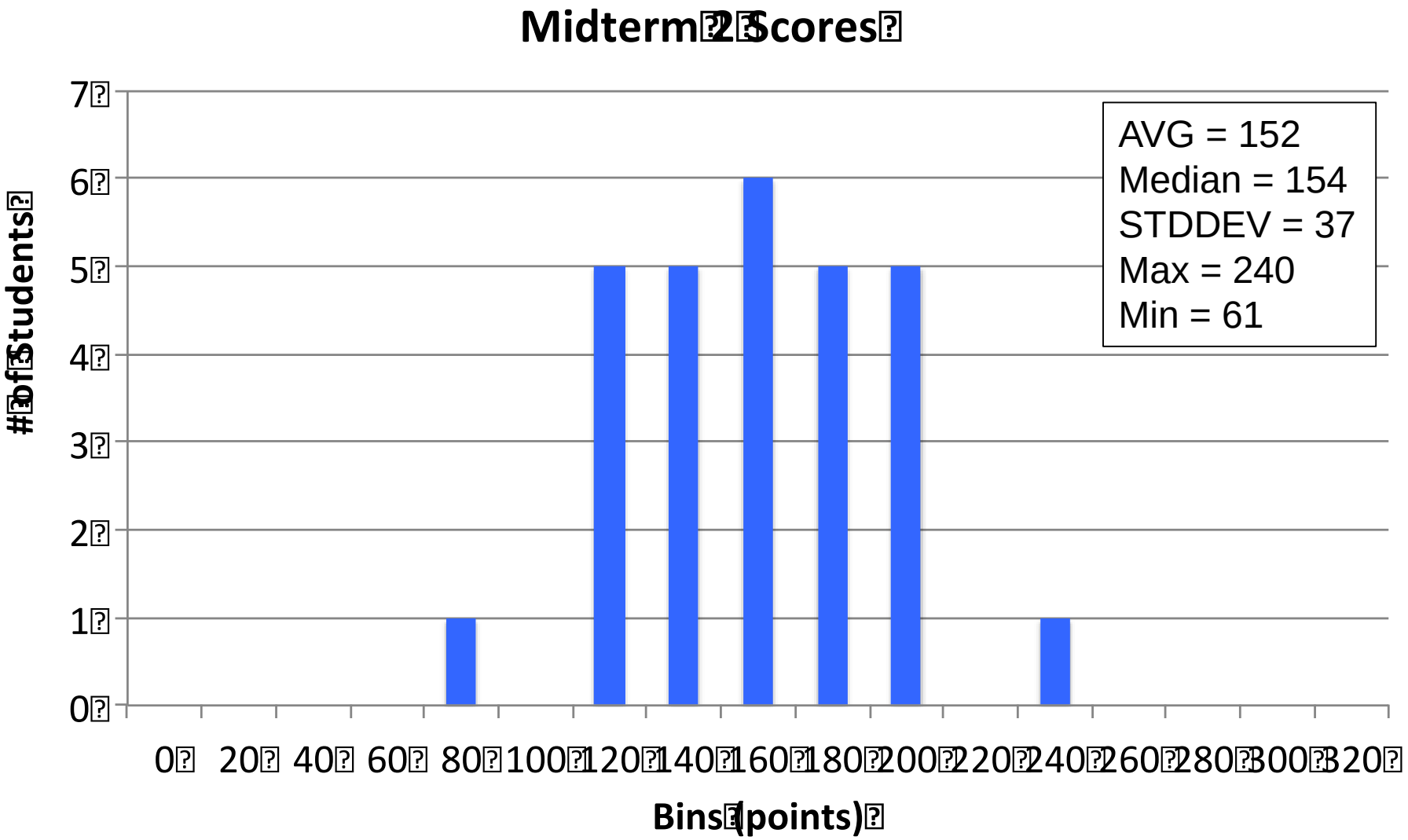
- Remember: The goal is for you to learn…

Extra Credit Lab 8: Multi-Core Cache Coherence

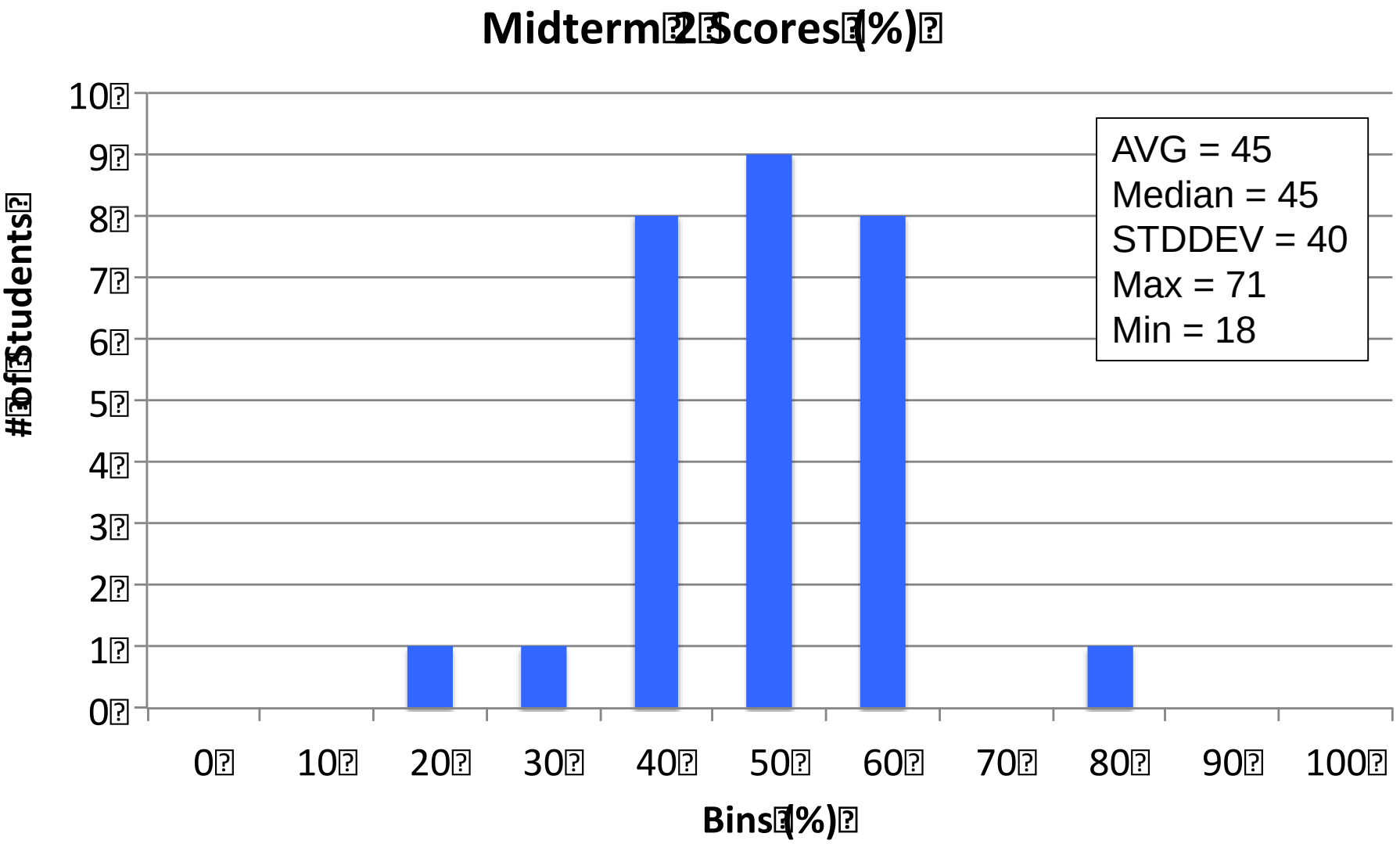
- Completely extra credit (all get 5% for free; can get 5% more)
- Last submission accepted on May 10, 11:59pm; no late submissions
- Cycle-level modeling of the MESI cache coherence protocol



Midterm 2 Grade Distribution



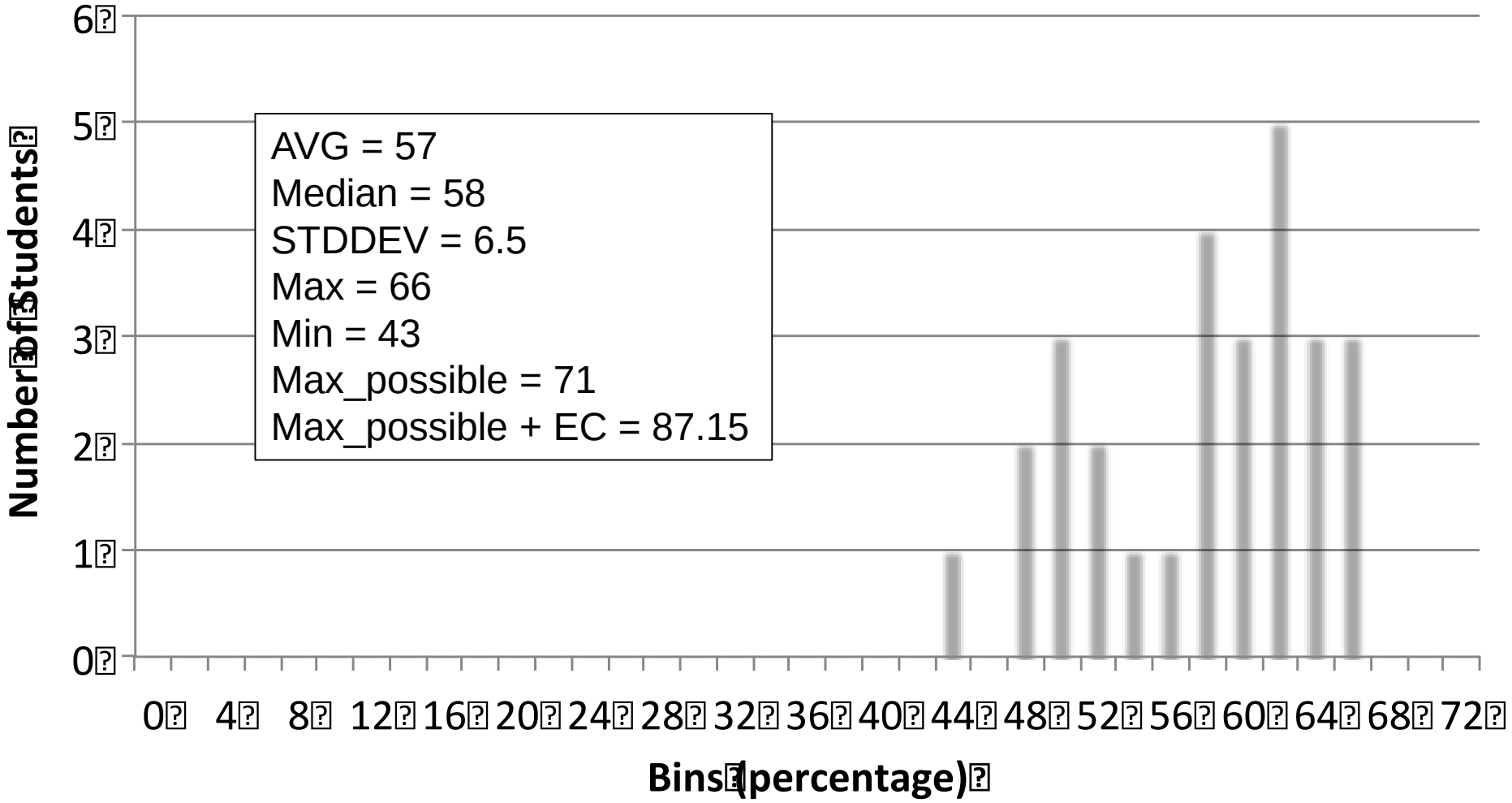
Midterm 2 Grade Distribution (%)



Final Exam: May 5

- May 5, 5:30-8:30pm, Location: BH A51
- Comprehensive (over **all topics** in course)
- Three cheat sheets allowed
- We will (likely) have a review session on Friday
- Remember this is 22% of your grade
  - I will take into account your improvement over the course
  - Know all concepts, especially the previous midterm concepts
  - Same advice as before for Midterms I and II

Course Grades So Far



A Note on 740, Research, Jobs/Internships

- I am teaching 740 next semester (Fall 2015)
  - Lectures M, W 7:30-9:20pm
  - Recitations T 7:30-9:20pm

- If you are enjoying 447 and are doing well, you can take it
  → feel free to talk with me

- If you are excited about Computer Architecture research or looking for a job/internship in this area
  → talk with me

More on 740

- 740 is the next course in sequence
- Time: Lect. MW 7:30-9:20pm, Rect. T 7:30-9:20pm
- Content:
  - Lectures: More advanced, with a different perspective
  - Recitations: Delving deeper into papers, advanced topics
  - Readings: Many fundamental and research readings; will do many reviews
  - Project: More open ended research project. Proposal → milestones → final poster and presentation
    - Done in groups of 1-3
    - Focus of the course is the project and critical reviews of readings
  - Exams: lighter and fewer
  - Homeworks: None

Course Evaluations (due May 11)

- Due May 11, 11:59pm
- Please do not forget to fill out the course evaluations
  - http://www.cmu.edu/hub/fce/
- Your feedback is very important

- I read these very carefully, and take into account every piece of feedback
  - And, improve the course for the future
- Please take the time to write out feedback
  - State the things you liked, topics you enjoyed, what you think the course contributed to your learning, what we can improve on
  - Please don't just say "the course is hard and fast paced"
    - Because you knew that from the very beginning!

Extra Credit for Course Evaluations

- 0.25% extra credit for everyone in the class if more than 90% (i.e., 25) of you fill out the evaluations

Plan for Wed and Fri Sessions This Week

- Wednesday: Informal Q&A Session
  - Location: Porch
  - Tentative format: Fun, information, food
  - Come with questions (about comp arch/systems, and anything else)
  - We will have food

- Friday: Final Exam Review
  - Location: HH 1107
  - Tentative format: TAs will go over Midterms I and II and answer your questions

Where We Are in Lecture Schedule

- The memory hierarchy
- Caches, caches, more caches
- Virtualizing the memory hierarchy: Virtual Memory
- Main memory: DRAM
- Main memory control, scheduling
- Memory latency tolerance techniques
- Non-volatile memory

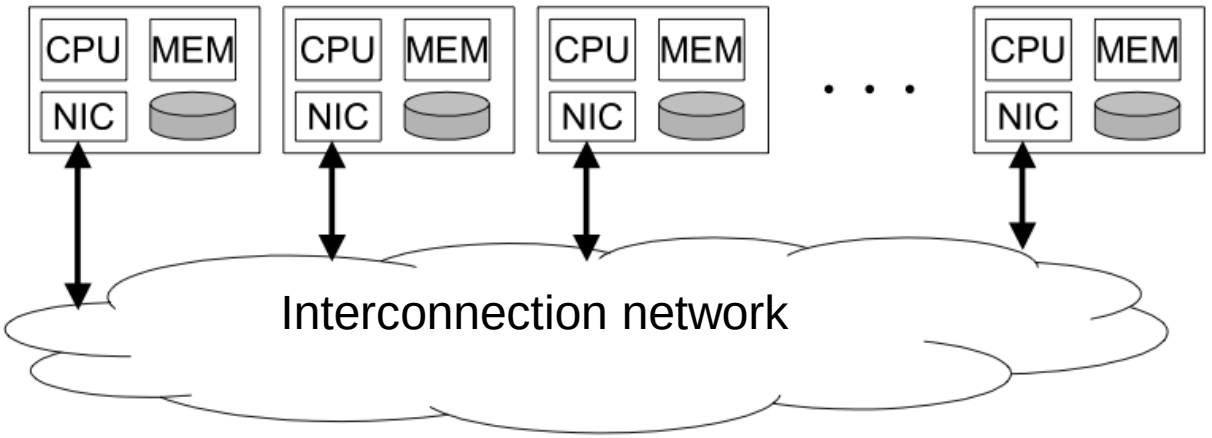
- Multiprocessors
- Coherence and consistency
- In-memory computation and predictable performance
- Multi-core issues (e.g., heterogeneous multi-core)
- Interconnection networks

Interconnection Network Basics

Where Is Interconnect Used?

- To connect components

- Many examples
  - Processors and processors
  - Processors and memories (banks)
  - Processors and caches (banks)
  - Caches and caches
  - I/O devices



Why Is It Important?

- Affects the scalability of the system
  - How large of a system can you build?
  - How easily can you add more processors?

- Affects performance and energy efficiency
  - How fast can processors, caches, and memory communicate?
  - How long are the latencies to memory?
  - How much energy is spent on communication?

Interconnection Network Basics

- Topology
  - Specifies the way switches are wired
  - Affects routing, reliability, throughput, latency, building ease
- Routing (algorithm)
  - How does a message get from source to destination
  - Static or adaptive
- Buffering and Flow Control
  - What do we store within the network?
    - Entire packets, parts of packets, etc?
  - How do we throttle during oversubscription?
  - Tightly coupled with routing strategy

Topology

- Bus (simplest)
- Point-to-point connections (ideal and most costly)
- Crossbar (less costly)
- Ring
- Tree
- Omega
- Hypercube
- Mesh
- Torus
- Butterfly
- …

Metrics to Evaluate Interconnect Topology

- Cost
- Latency (in hops, in nanoseconds)
- Contention

- Many others exist you should think about
  - Energy
  - Bandwidth
  - Overall system performance

Bus

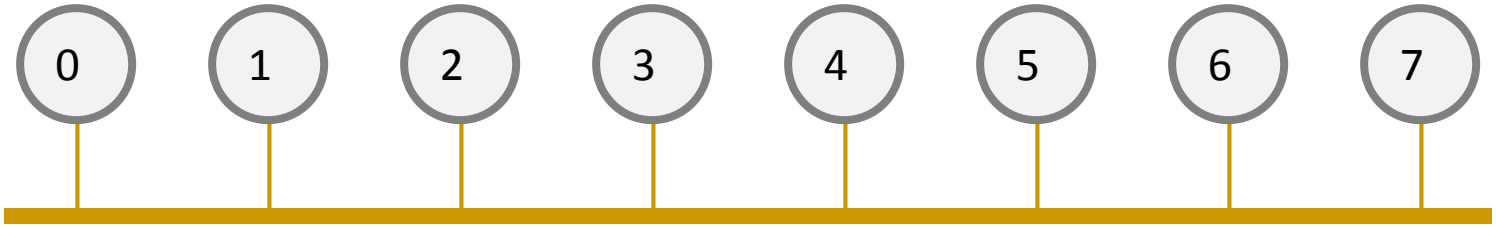
All nodes connected to a single link

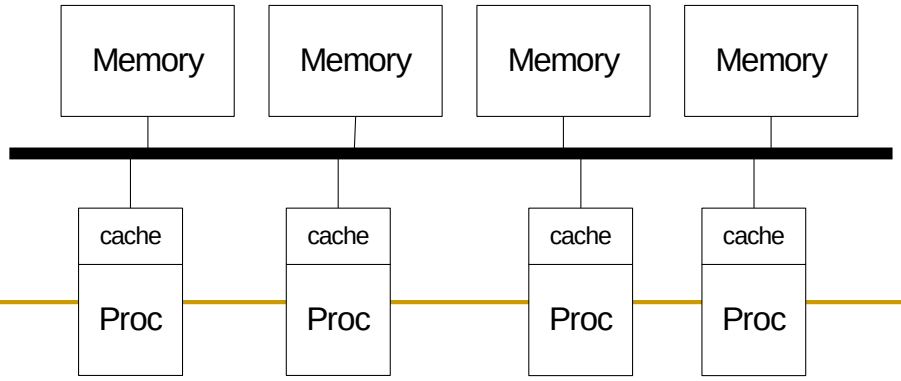
+ Simple + Cost effective for a small number of nodes

+ Easy to implement coherence (snooping and serialization)

- Not scalable to large number of nodes (limited bandwidth, electrical loading → reduced frequency)

- High contention → fast saturation



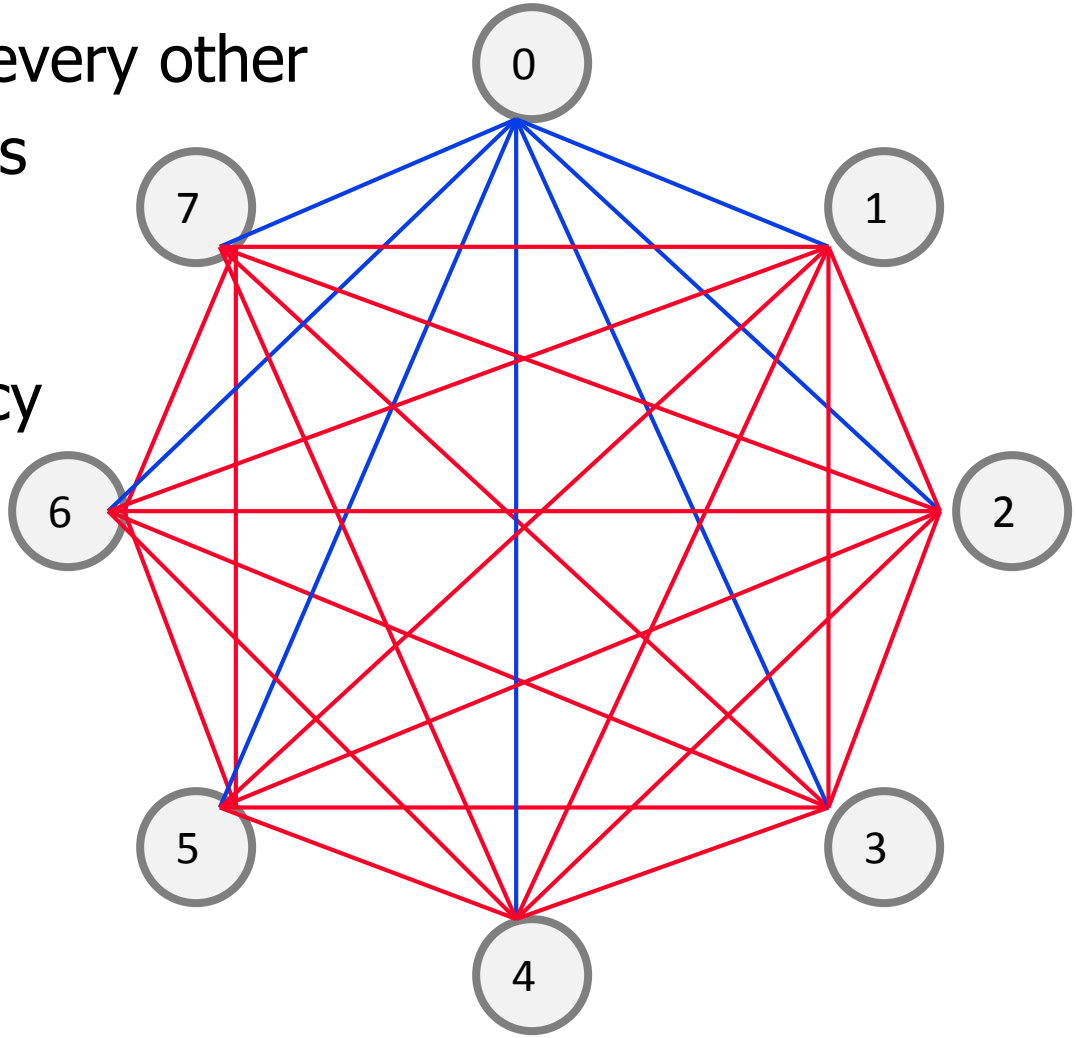


Point-to-Point

Every node connected to every other
with direct/isolated links

+ Lowest contention
+ Potentially lowest latency
+ Ideal, if cost is no issue

-- Highest cost
O(N) connections/ports per node
$O(N^2)$ links
-- Not scalable
-- How to lay out on chip?

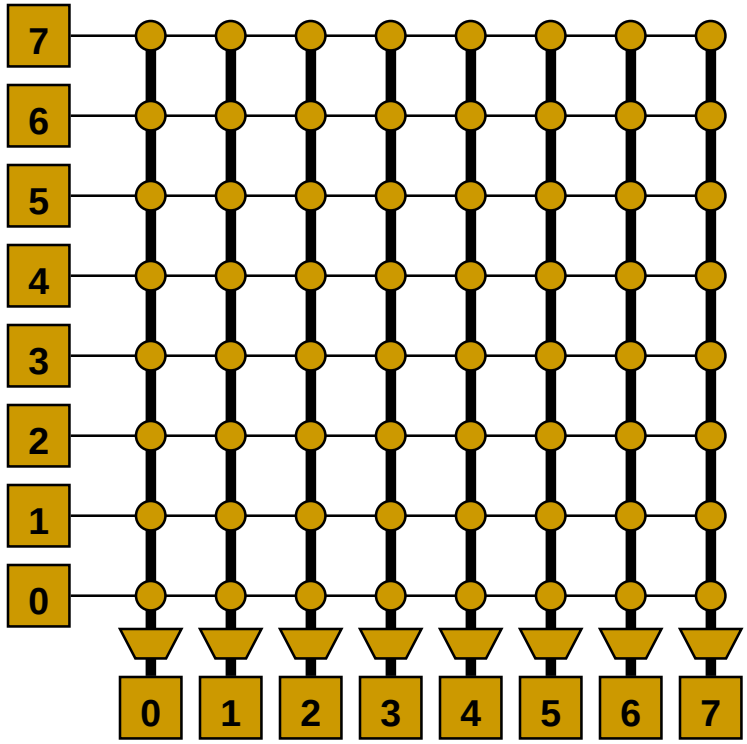


Crossbar

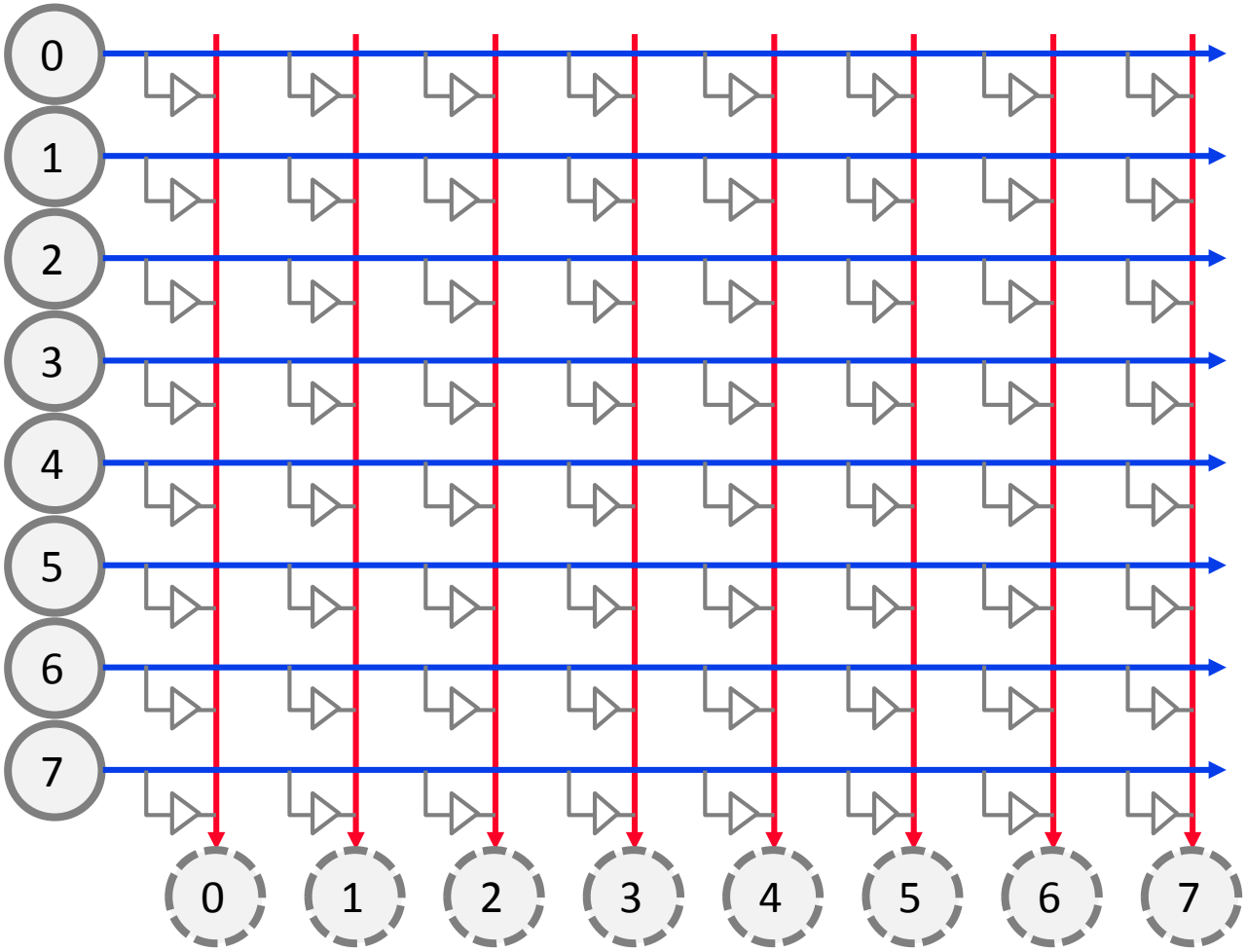
- Every node connected to every other with a shared link for each destination
- Enables concurrent transfers to non-conflicting destinations
- Could be cost-effective for small number of nodes

+ Low latency and high throughput
- Expensive
- Not scalable → $O(N^2)$ cost
- Difficult to arbitrate as N increases

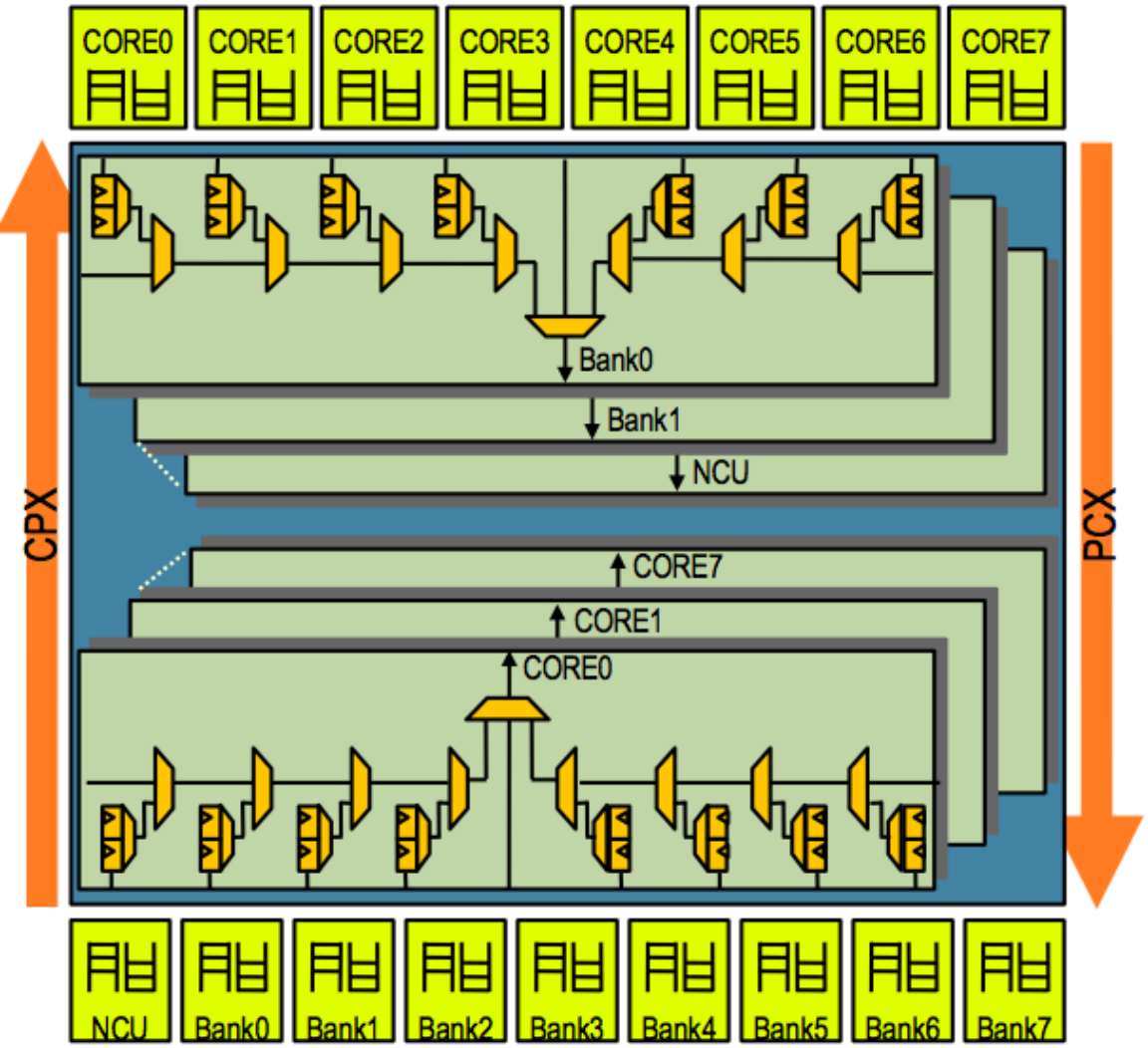
Used in core-to-cache-bank networks in
- IBM POWER5
- Sun Niagara I/II



Another Crossbar Design

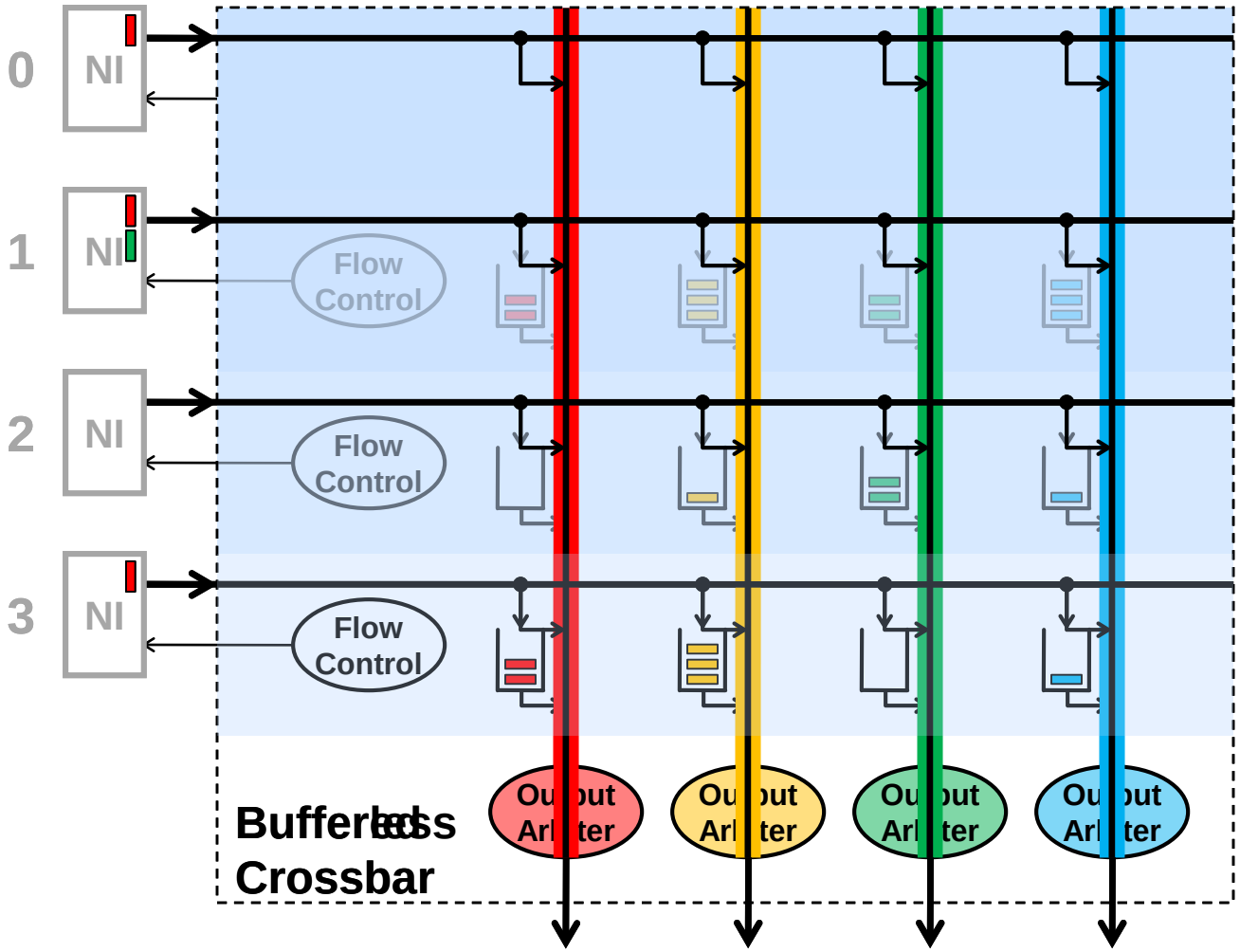


Sun UltraSPARC T2 Core-to-Cache Crossbar



- High bandwidth interface between 8 cores and 8 L2 banks & NCU
- 4-stage pipeline: req, arbitration, selection, transmission
- 2-deep queue for each src/dest pair to hold data transfer request

Bufferless and Buffered Crossbars



+ Simpler arbitration/ scheduling

+ Efficient support for variable-size packets

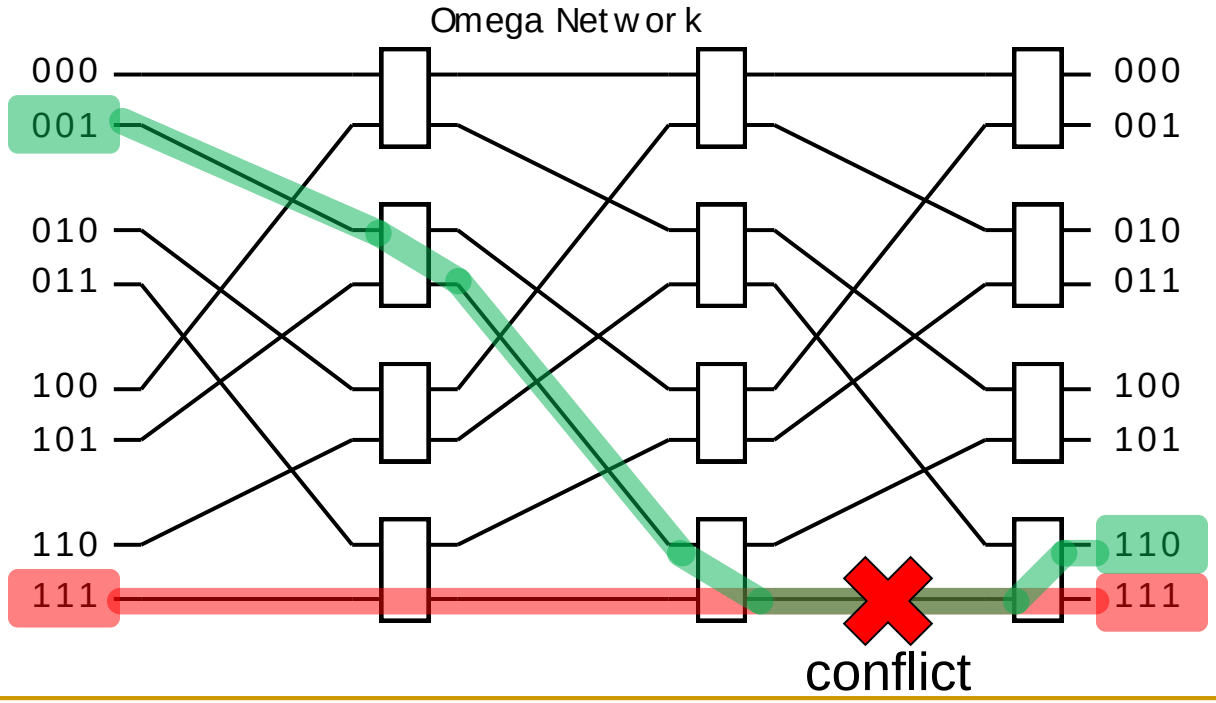
- Requires N^2 buffers

Can We Get Lower Cost than A Crossbar?

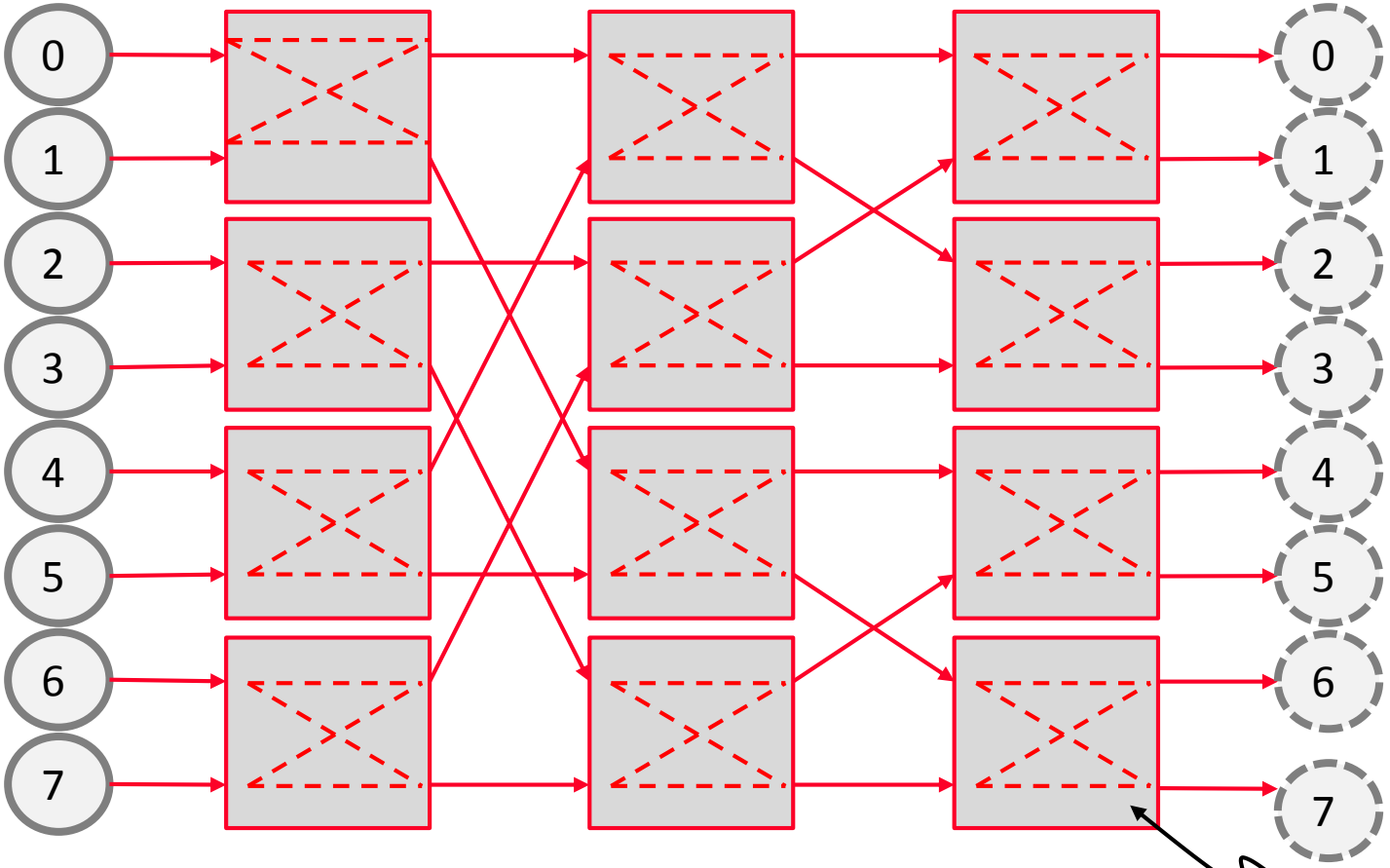
- Yet still have low contention compared to a bus?
- Idea: Multistage networks

Multistage Logarithmic Networks

- Idea: Indirect networks with multiple layers of switches between terminals/nodes
- Cost: O(NlogN), Latency: O(logN)
- Many variations (Omega, Butterfly, Benes, Banyan, …)
- Omega Network:

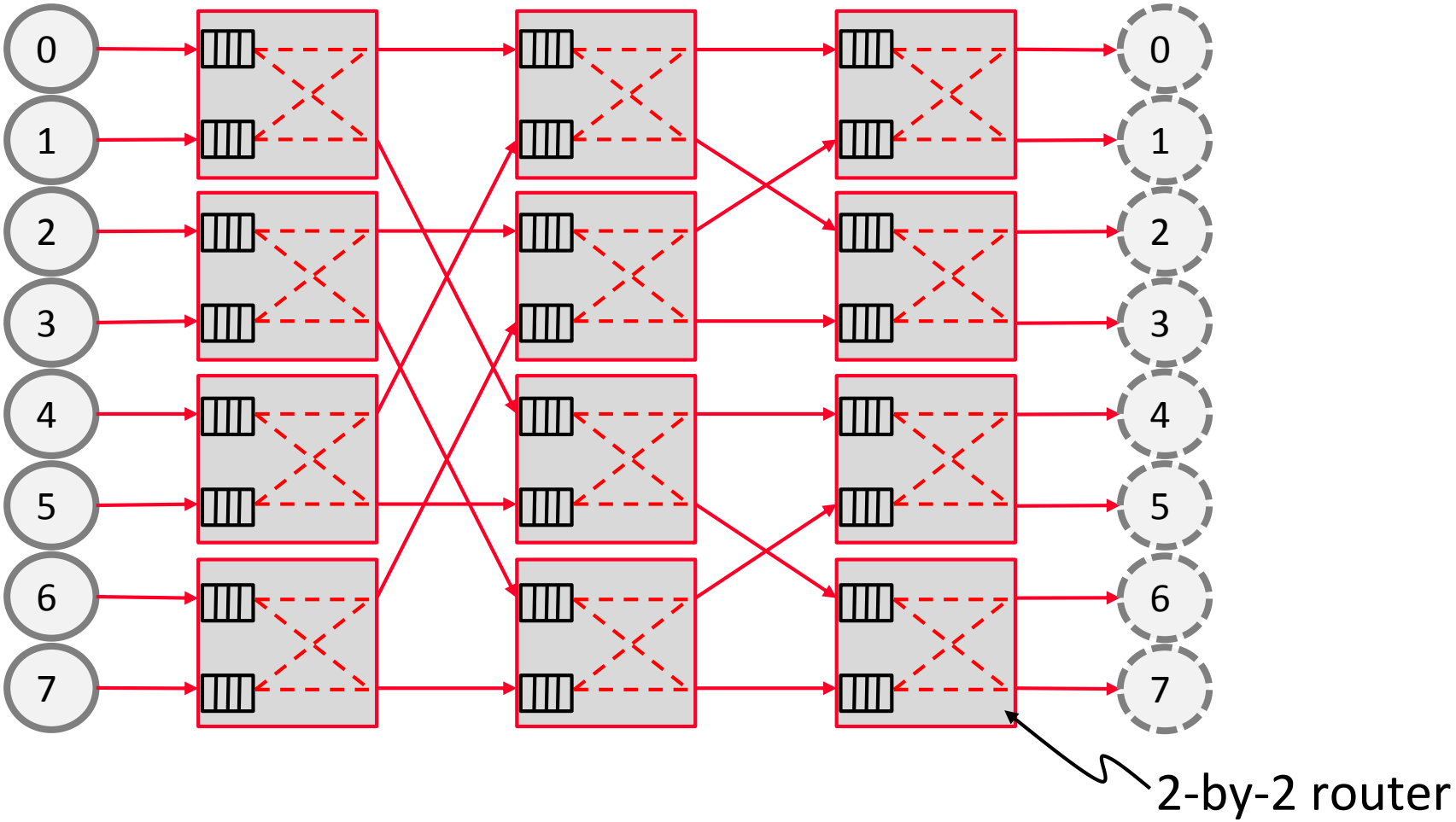


Multistage Networks (Circuit Switched)



- A multistage network has more restrictions on feasible concurrent Tx-Rx pairs vs a crossbar
- But more scalable than crossbar in cost, e.g., O(N logN) for Butterfly

Multistage Networks (Packet Switched)



- Packets "hop" from router to router, pending availability of the next-required switch and buffer

Aside: Circuit vs. Packet Switching

- Circuit switching sets up full path before transmission
  - Establish route then send data
  - Noone else can use those links while "circuit" is set

  + faster arbitration

  -- setting up and bringing down "path" takes time

- Packet switching routes per packet in each router
  - Route each packet individually (possibly via different paths)
  - If link is free, any packet can use it

  -- potentially slower --- must dynamically switch

  + no setup, bring down time

  + more flexible, does not underutilize links

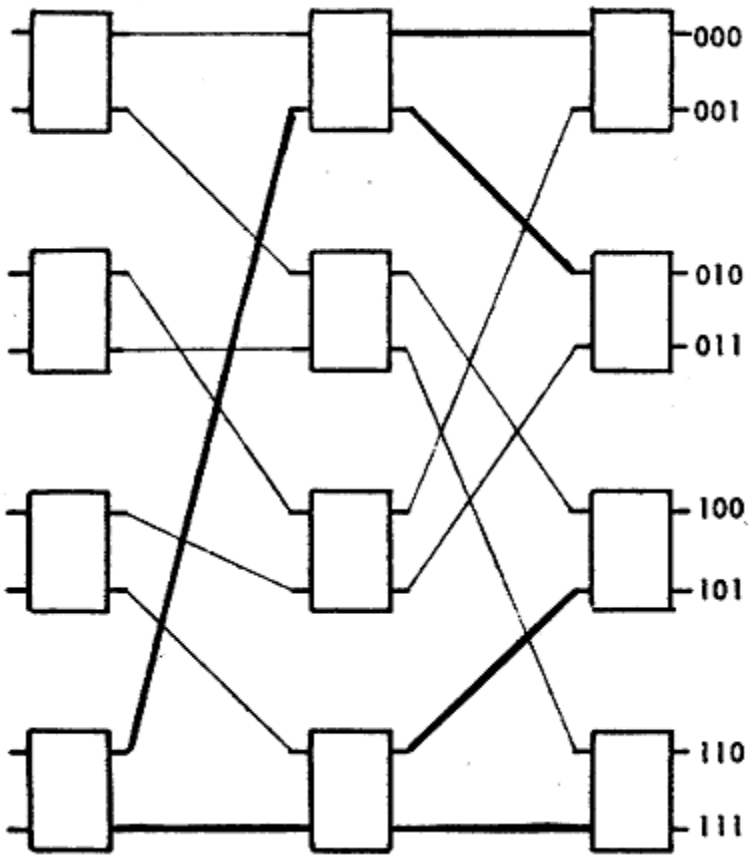
Switching vs. Topology

- Circuit/packet switching choice independent of topology
- It is a higher-level protocol on how a message gets sent to a destination

- However, some topologies are more amenable to circuit vs. packet switching

Another Example: Delta Network

- Single path from source to destination
- Each stage has different routers
- Proposed to replace costly crossbars as processor-memory interconnect
- Janak H. Patel ,"Processor-Memory Interconnections for Multiprocessors," ISCA 1979.



8x8 Delta network

Another Example: Omega Network

- Single path from source to destination
- All stages are the same
- Used in NYU Ultracomputer
- Gottlieb et al. "The NYU Ultracomputer - Designing an MIMD Shared Memory Parallel Computer," IEEE Trans. On Comp., 1983.

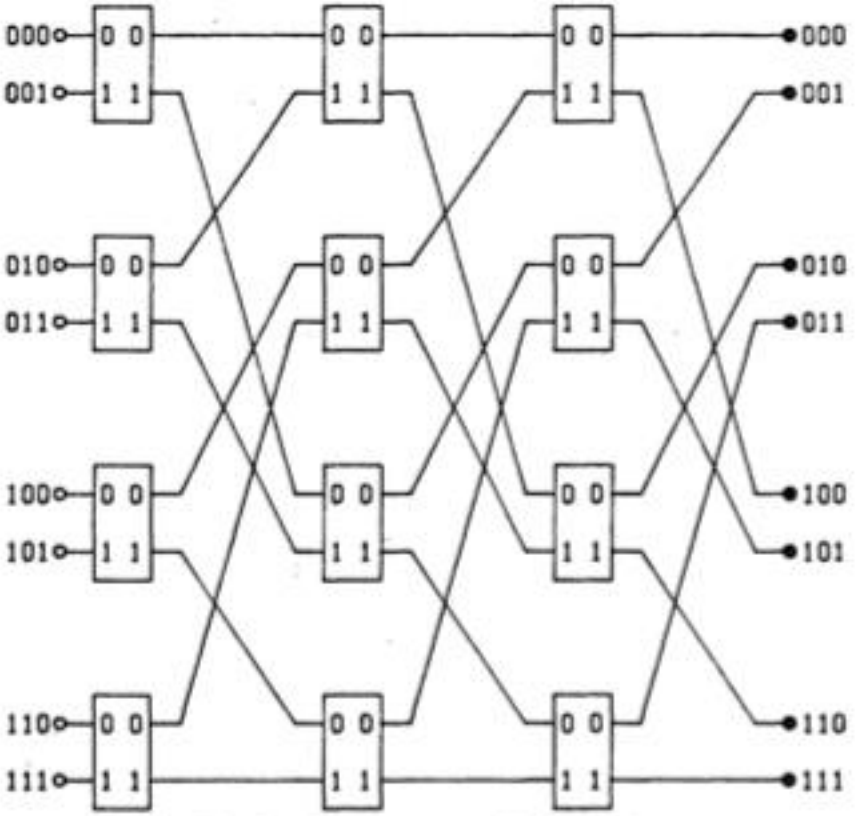


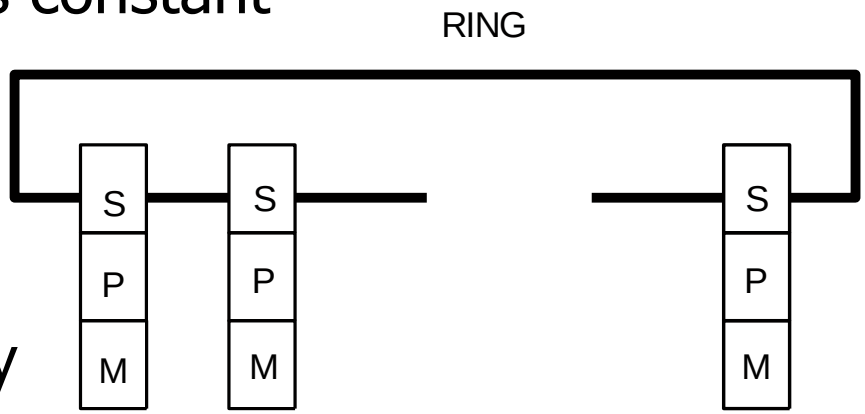
Fig. 2. Omega-network ($N = 8$).

Ring

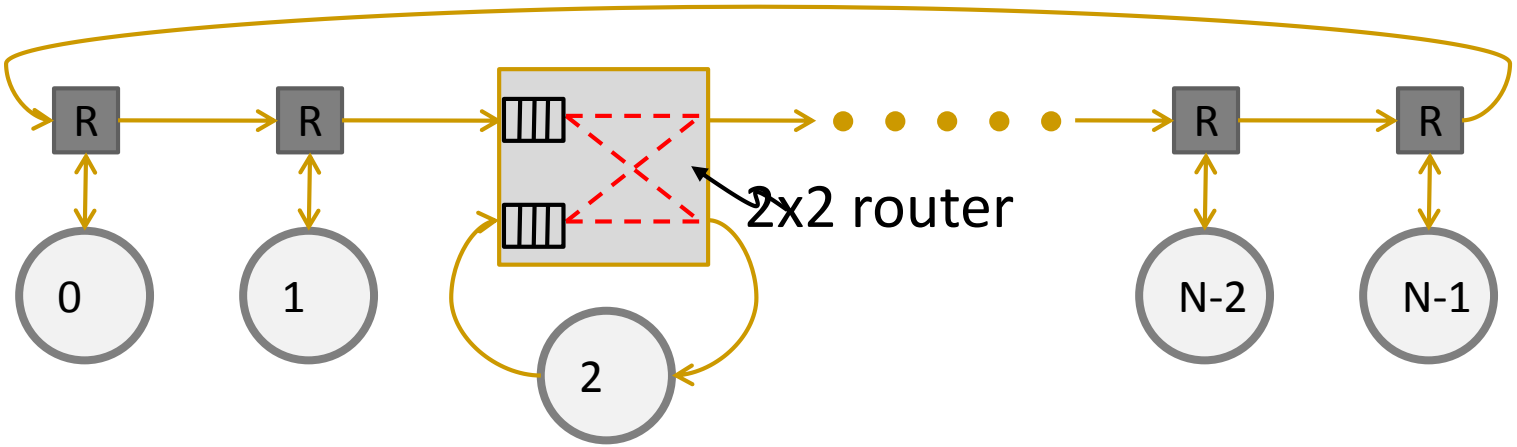
Each node connected to exactly two other nodes. Nodes form a continuous pathway such that packets can reach any node.

+ Cheap: O(N) cost
- High latency: O(N)
- Not easy to scale
  - Bisection bandwidth remains constant

Used in Intel Haswell,
Intel Larrabee, IBM Cell,
many commercial systems today



Unidirectional Ring



- Single directional pathway
- Simple topology and implementation
  - Reasonable performance if N and performance needs (bandwidth & latency) still moderately low
  - O(N) cost
  - N/2 average hops; latency depends on utilization

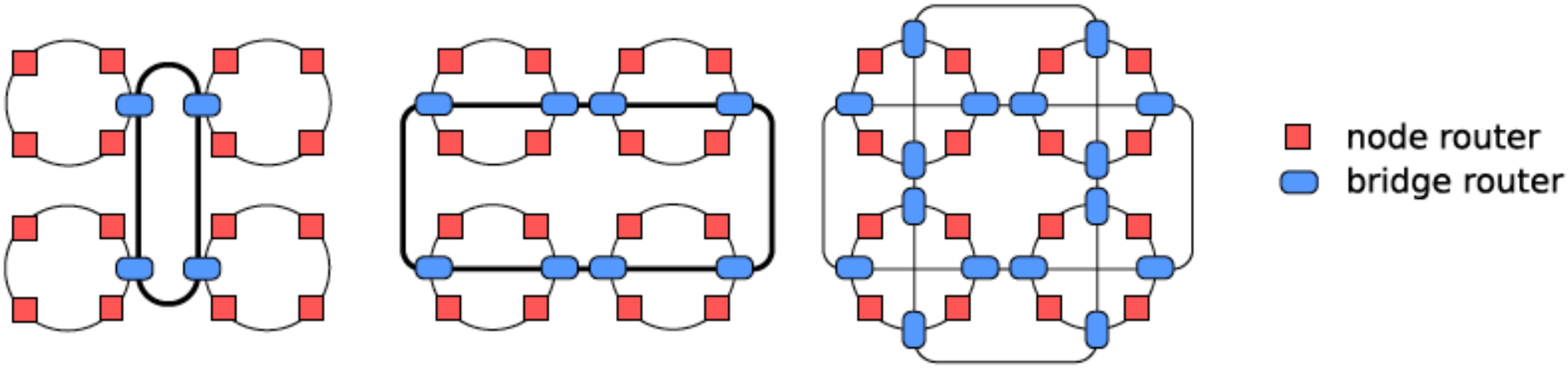
Bidirectional Rings

Multi-directional pathways, or multiple rings

+ Reduces latency
+ Improves scalability

- Slightly more complex injection policy (need to select which ring to inject a packet into)

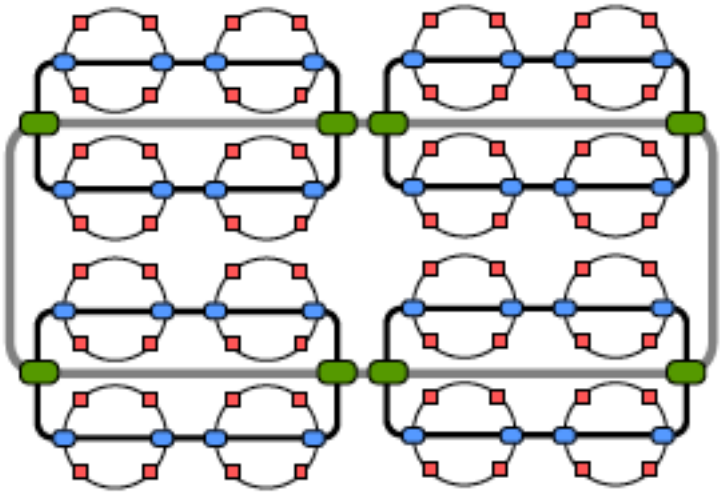
Hierarchical Rings



(a) 4-, 8-, and 16-bridge hierarchical ring topologies.

+ More scalable
+ Lower latency

- More complex



(b) Three-level hierarchy (8x8).

More on Hierarchical Rings

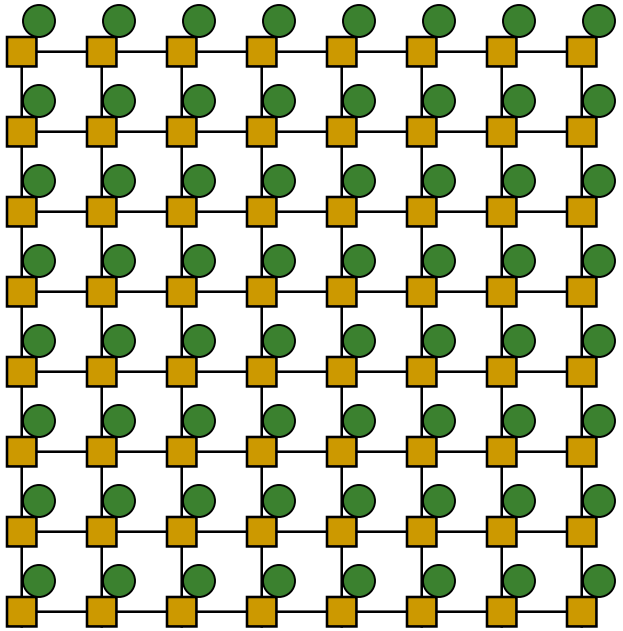
- Rachata+, "Design and Evaluation of Hierarchical Rings with Deflection Routing," SBAC-PAD 2014.
  - http://users.ece.cmu.edu/~omutlu/pub/hierarchical-rings-with-deflection_sbacpad14.pdf

- Discusses the design and implementation of a mostly-bufferless hierarchical ring

Mesh

- Each node connected to 4 neighbors (N, E, S, W)
- O(N) cost
- Average latency: O(sqrt(N))
- Easy to layout on-chip: regular and equal-length links
- Path diversity: many ways to get from one node to another

- Used in Tilera 100-core
- And many on-chip network prototypes



Torus

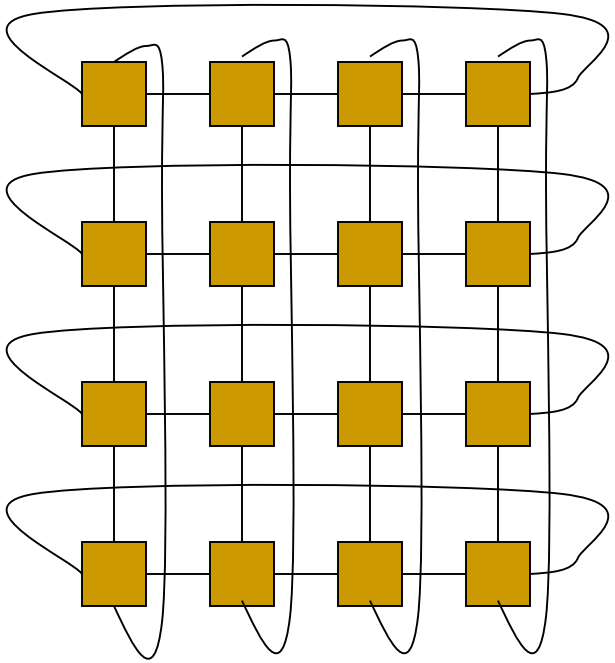
- Mesh is not symmetric on edges: performance very sensitive to placement of task on edge vs. middle
- Torus avoids this problem

+ Higher path diversity (and bisection bandwidth) than mesh

- Higher cost

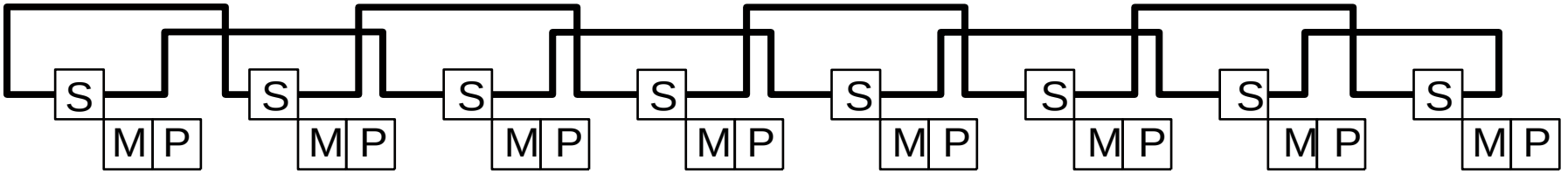
- Harder to lay out on-chip

  - Unequal link lengths



Torus, continued

- Weave nodes to make inter-node latencies ~constant



Trees

Planar, hierarchical topology

Latency: O(logN)

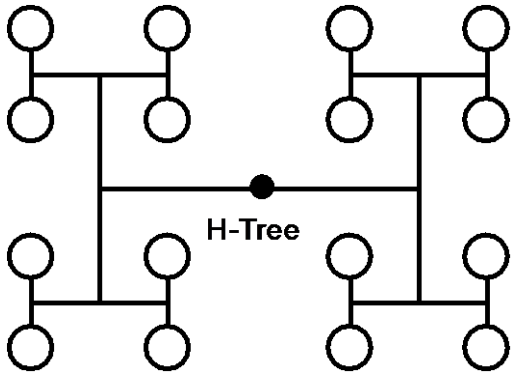
Good for local traffic

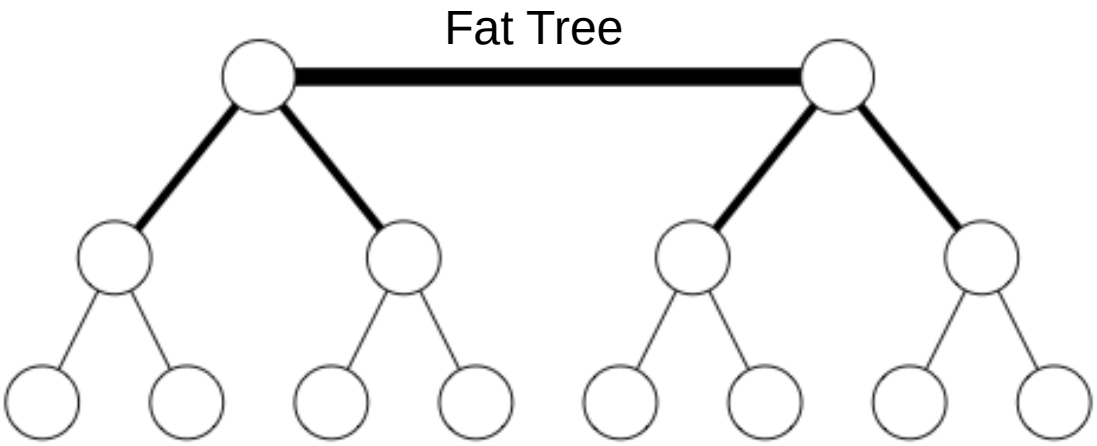
+ Cheap: O(N) cost

+ Easy to Layout

- Root can become a bottleneck

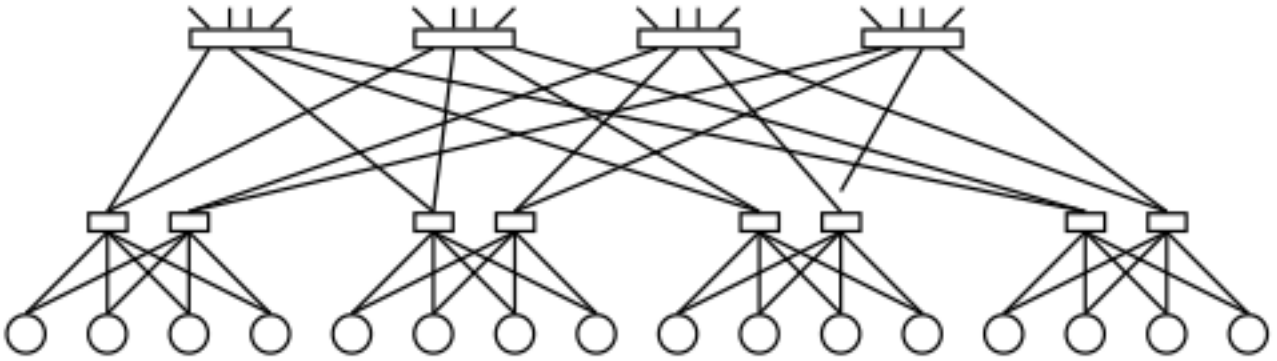
  Fat trees avoid this problem (CM-5)





CM-5 Fat Tree

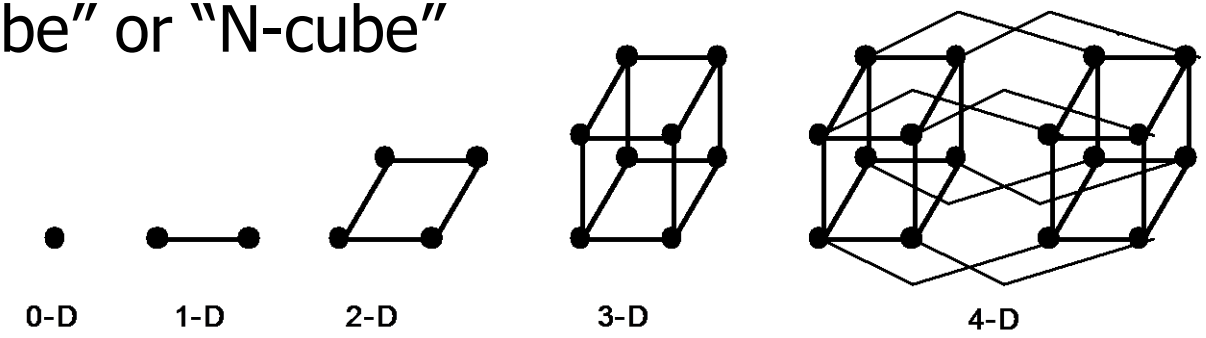
- Fat tree based on 4x2 switches
- Randomized routing on the way up
- Combining, multicast, reduction operators supported in hardware
  - Thinking Machines Corp., "The Connection Machine CM-5 Technical Summary," Jan. 1992.



CM-5 Thinned Fat Tree

Hypercube

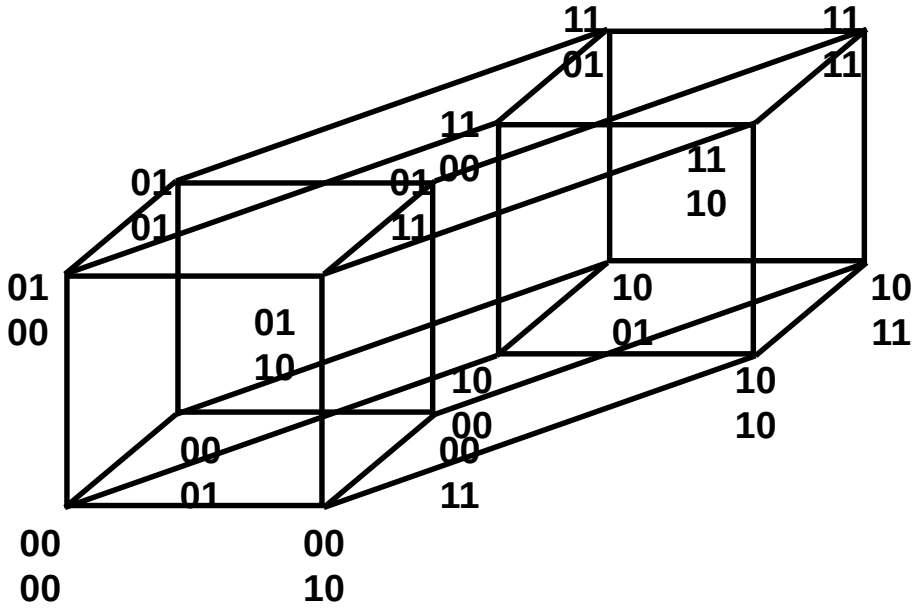
- "N-dimensional cube" or "N-cube"



- Latency: O(logN)
- Radix: O(logN)
- #links: O(NlogN)

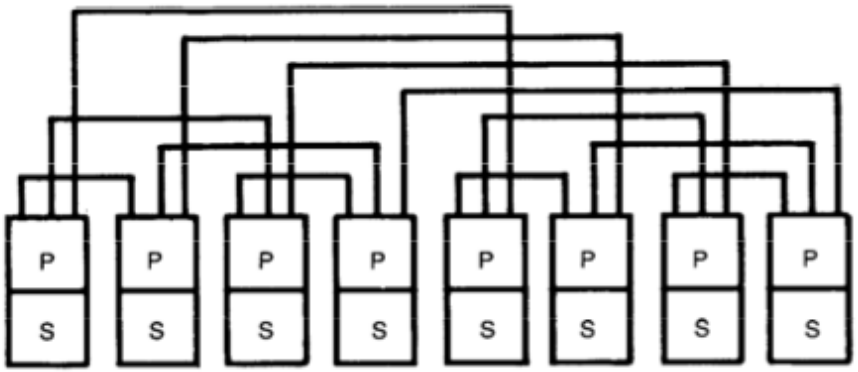
+ Low latency

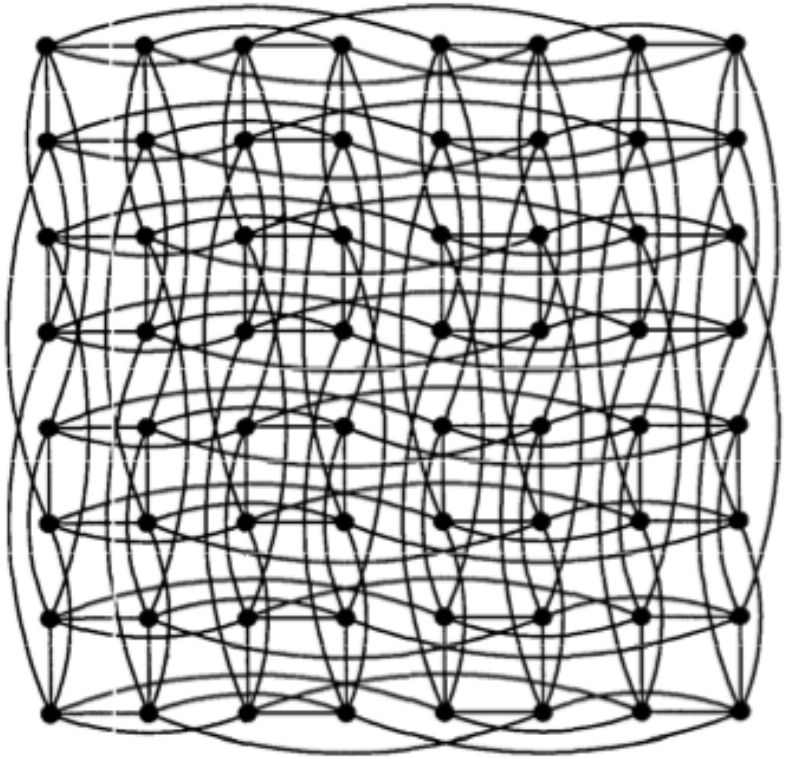
- Hard to lay out in 2D/3D



Caltech Cosmic Cube

- 64-node message passing machine
- Seitz, "The Cosmic Cube," CACM 1985.





A hypercube connects $N = 2^n$ small computers, called nodes, through point-to-point communication channels in the Cosmic Cube. Shown here is a two-dimensional projection of a six-dimensional hypercube, or binary 6-cube, which corresponds to a 64-node machine.

FIGURE 1. A Hypercube (also known as a binary cube or a Boolean *n*-cube)

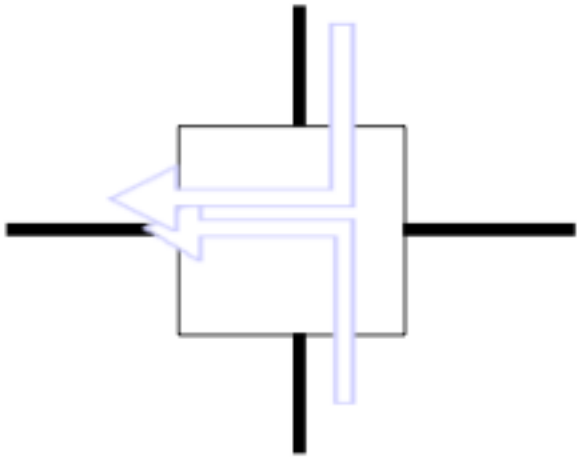
Interconnection Network Basics

- Topology
  - Specifies the way switches are wired
  - Affects routing, reliability, throughput, latency, building ease

- Routing (algorithm)
  - How does a message get from source to destination
  - Static or adaptive

- Buffering and Flow Control
  - What do we store within the network?
    - Entire packets, parts of packets, etc?
  - How do we throttle during oversubscription?
  - Tightly coupled with routing strategy

Handling Contention

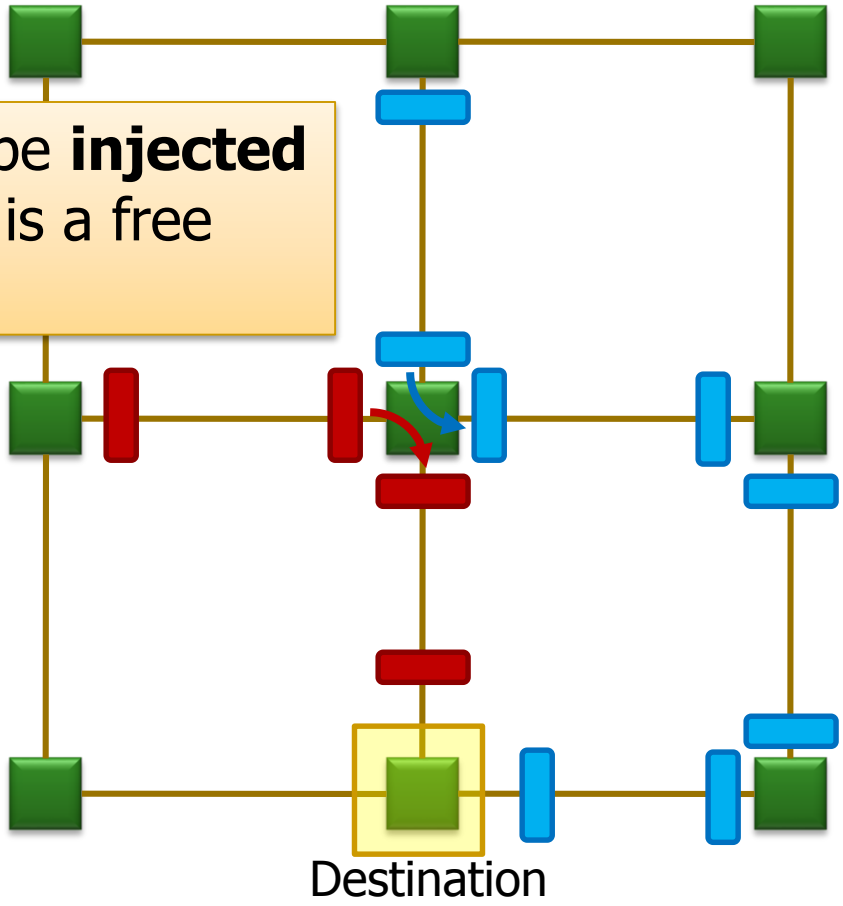


- Two packets trying to use the same link at the same time
- What do you do?
  - Buffer one
  - Drop one
  - Misroute one (deflection)
- Tradeoffs?

Bufferless Deflection Routing

- **Key idea**: Packets are never buffered in the network. When two packets contend for the same link, one is deflected.[1]

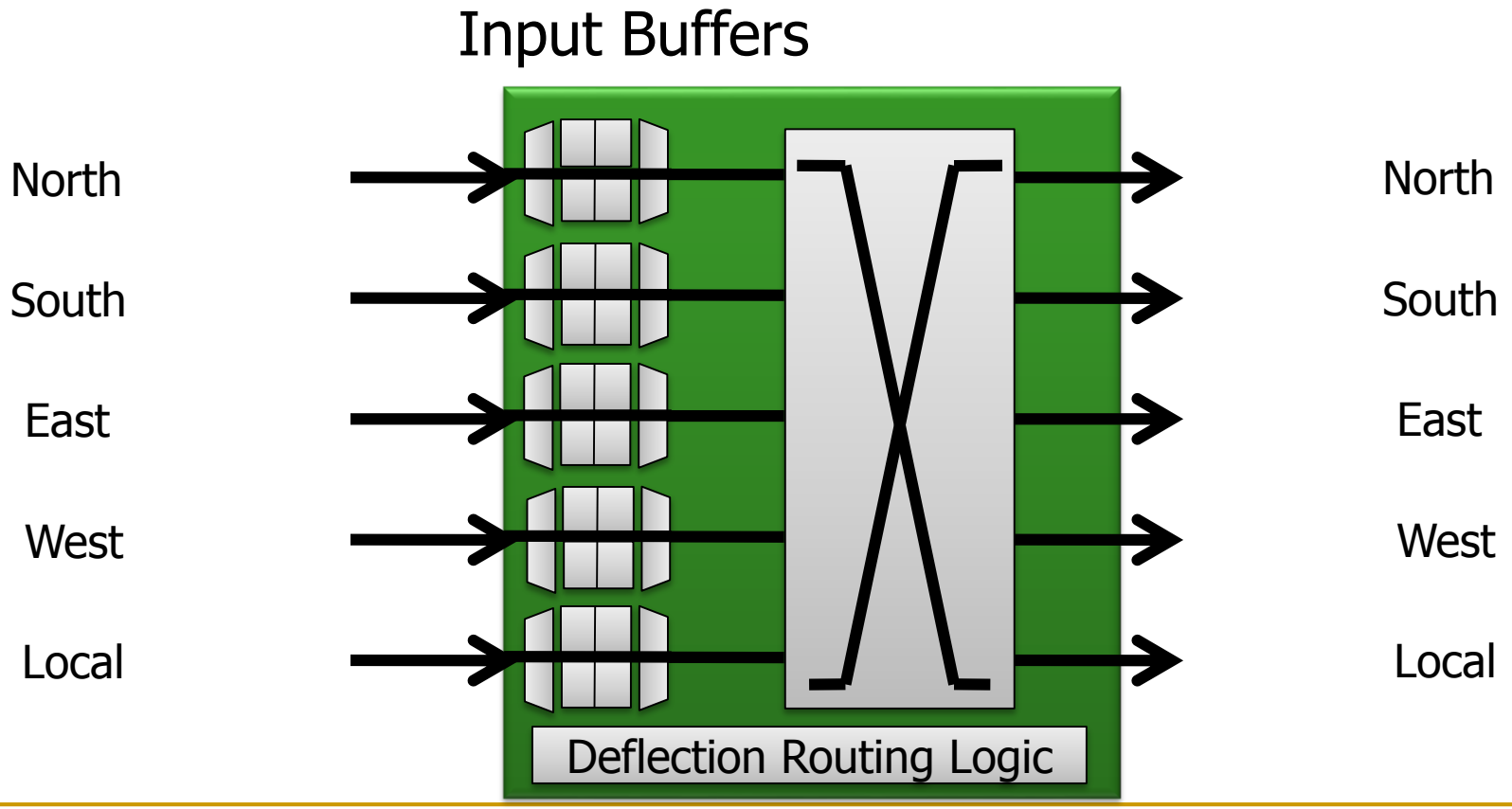
New traffic can be **injected** whenever there is a free output link.



[1]Baran, "On Distributed Communication Networks." RAND Tech. Report., 1962 / IEEE Trans.Comm., 1964.

Bufferless Deflection Routing

- Input buffers are eliminated: packets are buffered in **pipeline latches** and on **network links**



Routing Algorithm

- Three Types
  - Deterministic: always chooses the same path for a communicating source-destination pair
  - Oblivious: chooses different paths, without considering network state
  - Adaptive: can choose different paths, adapting to the state of the network

- How to adapt
  - Local/global feedback
  - Minimal or non-minimal paths

Deterministic Routing

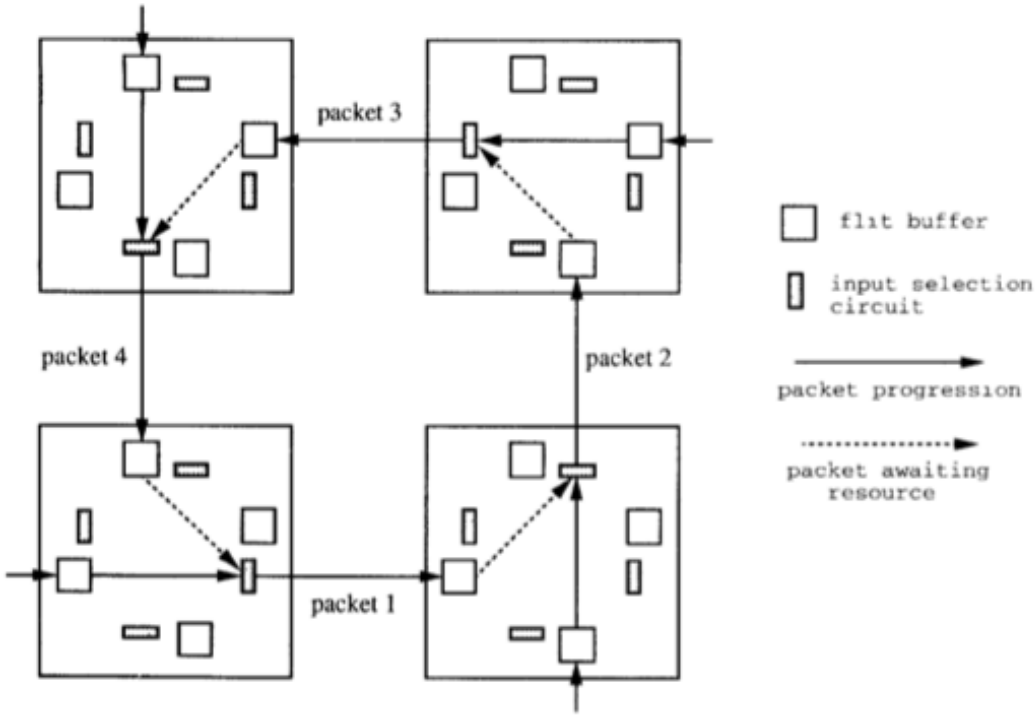
- All packets between the same (source, dest) pair take the same path

- Dimension-order routing
  - First traverse dimension X, then traverse dimension Y
  - E.g., XY routing (used in Cray T3D, and many on-chip networks)

+ Simple
+ Deadlock freedom (no cycles in resource allocation)
- Could lead to high contention
- Does not exploit path diversity

Deadlock

- No forward progress
- Caused by circular dependencies on resources
- Each packet waits for a buffer occupied by another packet downstream



Handling Deadlock

- Avoid cycles in routing
  - Dimension order routing
    - Cannot build a circular dependency
  - Restrict the “turns” each packet can take
- Avoid deadlock by adding more buffering (escape paths)
- Detect and break deadlock
  - Preemption of buffers

Turn Model to Avoid Deadlock

- Idea
  - Analyze directions in which packets can turn in the network
  - Determine the cycles that such turns can form
  - Prohibit just enough turns to break possible cycles
- Glass and Ni, "The Turn Model for Adaptive Routing," ISCA 1992.

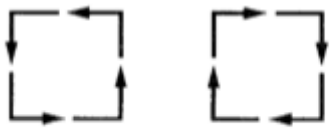


FIG. 2. The possible turns and simple cycles in a two-dimensional mesh.

FIG. 3. The four turns allowed by the *xy* routing algorithm.

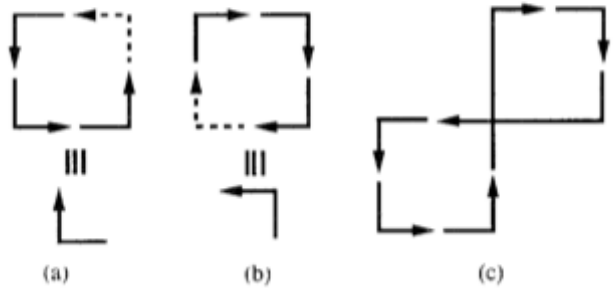


FIG. 4. Six turns that complete the cycles and allow deadlock.

Oblivious Routing: Valiant's Algorithm

- An example of oblivious algorithm
- Goal: Balance network load
- Idea: Randomly choose an intermediate destination, route to it first, then route from there to destination
  - Between source-intermediate and intermediate-dest, can use dimension order routing

+ Randomizes/balances network load

- Non minimal (packet latency can increase)

- Optimizations:
  - Do this on high load
  - Restrict the intermediate node to be close (in the same quadrant)

Adaptive Routing

- Minimal adaptive
  - Router uses network state (e.g., downstream buffer occupancy) to pick which "productive" output port to send a packet to
  - Productive output port: port that gets the packet closer to its destination

  + Aware of local congestion

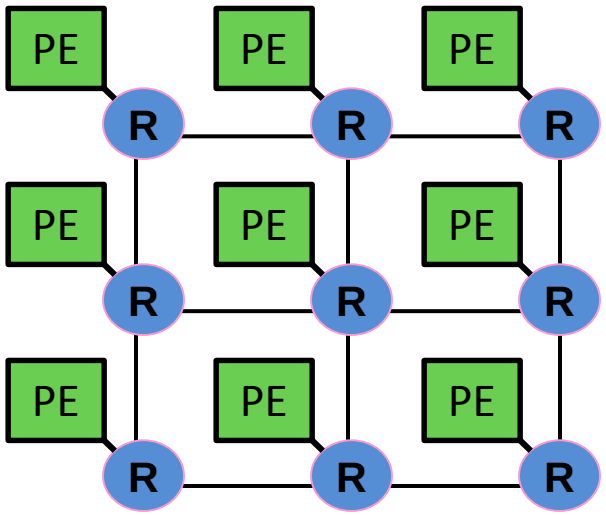
  - Minimality restricts achievable link utilization (load balance)

- Non-minimal (fully) adaptive
  - "Misroute" packets to non-productive output ports based on network state

  + Can achieve better network utilization and load balance

  - Need to guarantee livelock freedom

On-Chip Networks



- Connect **cores, caches, memory controllers, etc**
  - Buses and crossbars are not scalable
- **Packet switched**
- **2D mesh:** Most commonly used topology
- Primarily serve **cache misses** and **memory requests**

R Router

PE Processing Element
(Cores, L2 Banks, Memory Controllers, etc)