

Computer Architecture

Lecture 22: Interconnects

Prof. Onur Mutlu

ETH Zürich

Fall 2020

27 December 2020

Summary of Last Week

- Multiprocessing Fundamentals
- Memory Ordering (Consistency)
- Cache Coherence

Interconnection Networks

Readings

■ Required

- Moscibroda and Mutlu, “[A Case for Bufferless Routing in On-Chip Networks](#),” ISCA 2009.

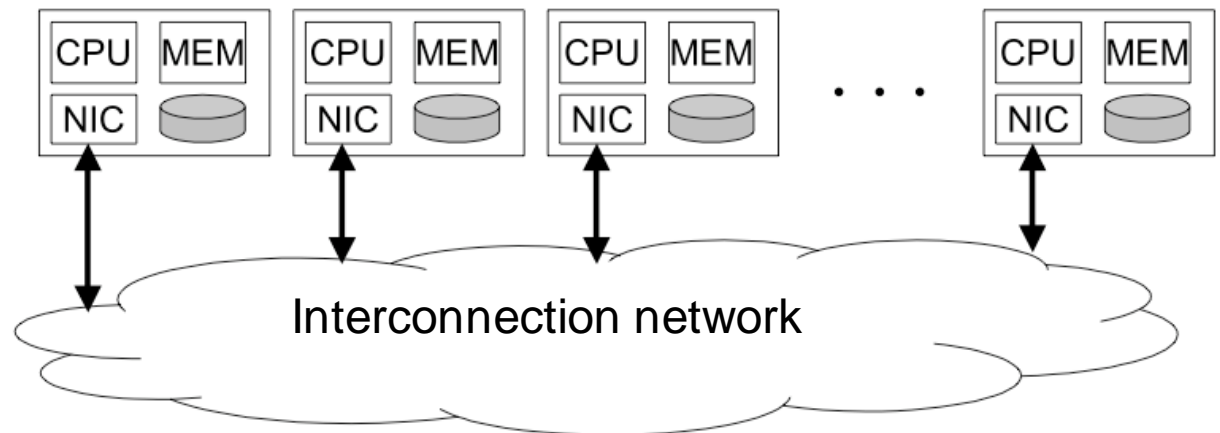
■ Recommended

- Das et al., “[Application-Aware Prioritization Mechanisms for On-Chip Networks](#),” MICRO 2009.
- Janak H. Patel, “[Processor-Memory Interconnections for Multiprocessors](#),” ISCA 1979.
- Gottlieb et al. “[The NYU Ultracomputer - Designing an MIMD Shared Memory Parallel Computer](#),” IEEE Trans. On Comp., 1983.
- Das et al., “[Aergia: Exploiting Packet Latency Slack in On-Chip Networks](#),” ISCA 2010.

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 and cost** 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?
- Affects **reliability and security**
 - Can you guarantee messages are delivered or your protocol works?

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

Terminology

■ Network interface

- ❑ Module that connects endpoints (e.g. processors) to network
- ❑ Decouples computation/communication

■ Link

- ❑ Bundle of wires that carry a signal

■ Switch/router

- ❑ Connects fixed number of input channels to fixed number of output channels

■ Channel

- ❑ A single logical connection between routers/switches

More Terminology

■ Node

- A router/switch within a network

■ Message

- Unit of transfer for network's clients (processors, memory)

■ Packet

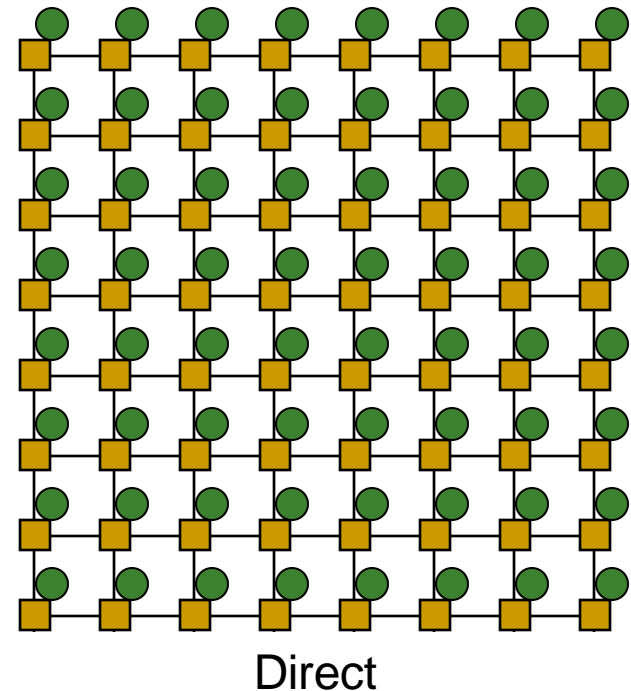
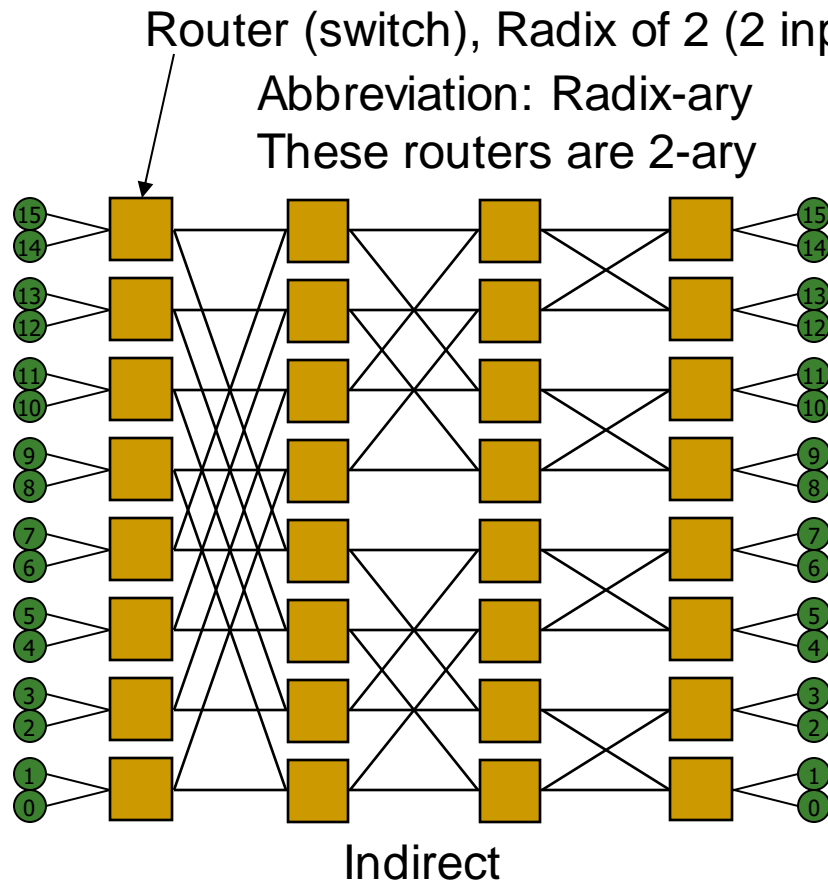
- Unit of transfer for network

■ Flit

- Flow control digit
- Unit of flow control within network

Some More Terminology

- Direct or Indirect Networks
- Endpoints sit “inside” (direct) or “outside” (indirect) the network
- E.g. mesh is direct; every node is both endpoint and switch



Interconnection Network Topology

Properties of a Topology/Network

- Regular or Irregular

- Regular if topology is regular graph (e.g. ring, mesh).

- Routing Distance

- number of links/hops along a route

- Diameter

- maximum routing distance within the network

- Average Distance

- Average number of hops across all valid routes

Properties of a Topology/Network

■ Bisection Bandwidth

- ❑ Often used to describe network performance
- ❑ **Cut network in half and sum bandwidth of links severed**
 - $(\text{Min \# channels spanning two halves}) * (\text{BW of each channel})$
- ❑ Meaningful only for recursive topologies
- ❑ Can be misleading, because does not account for switch and routing efficiency (and certainly not execution time)

■ Blocking vs. Non-Blocking

- ❑ If connecting any permutation of sources & destinations is possible, network is non-blocking; otherwise network is blocking.
- ❑ Rearrangeable non-blocking: Same as non-blocking but might require rearranging connections when switching from one permutation to another.

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

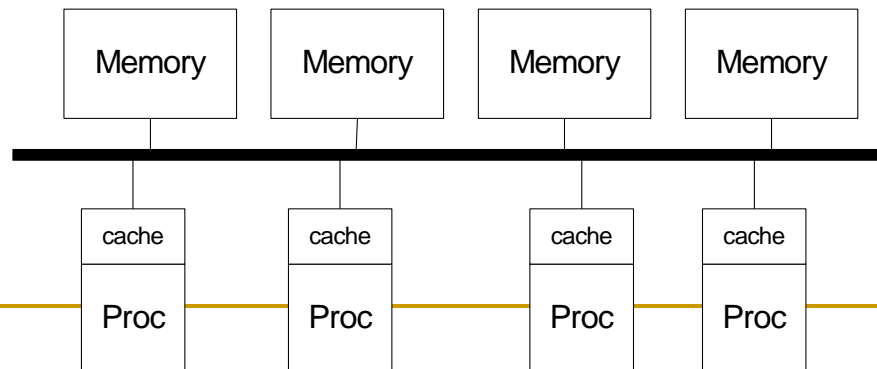
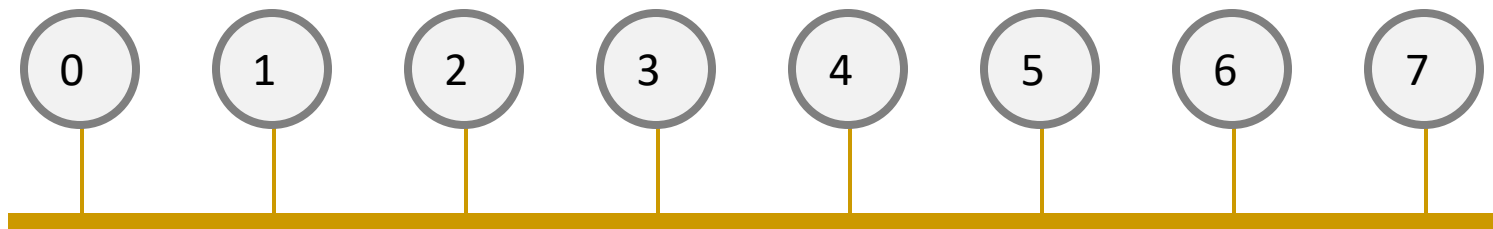
- Cost
- Latency (in hops, in nanoseconds)
- Contention

- 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

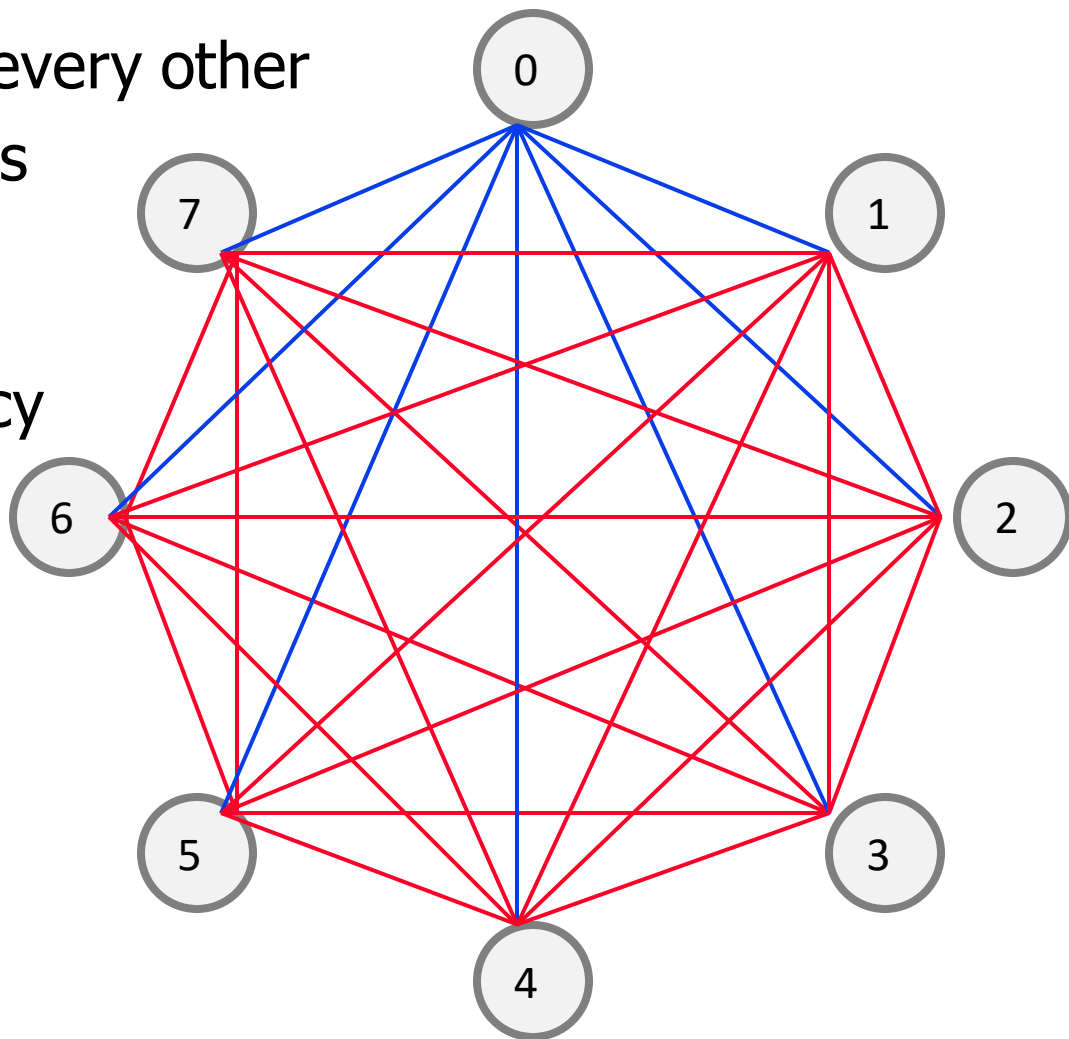
-- Highest cost

$O(N)$ connections/ports
per node

$O(N^2)$ links

-- Not scalable

-- How to lay out on chip?



Crossbar

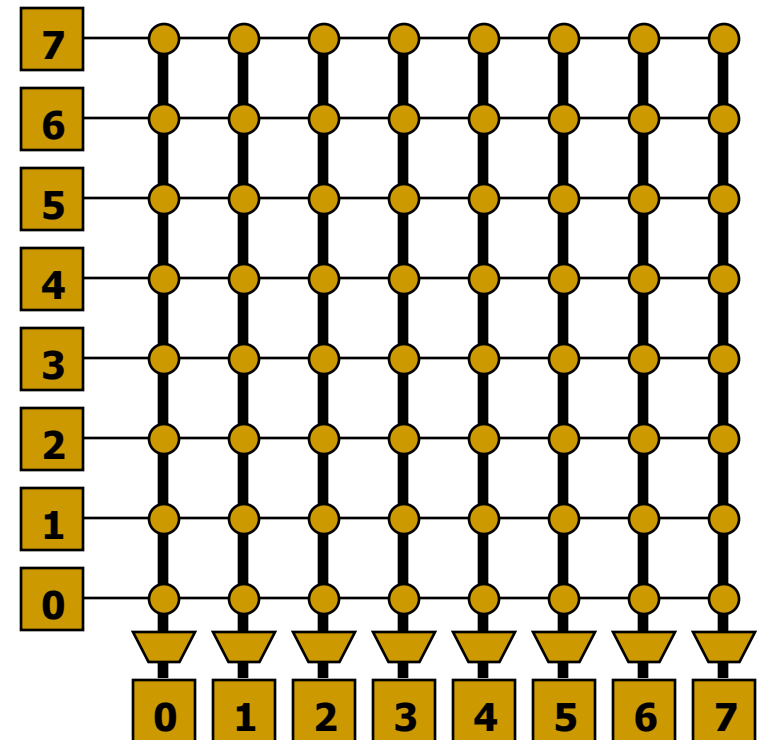
- Every node connected to every other with a shared link for each destination
- Enables concurrent transfers to non-conflicting destinations
- Could be cost-effective for small number of nodes

+ Low latency and high throughput

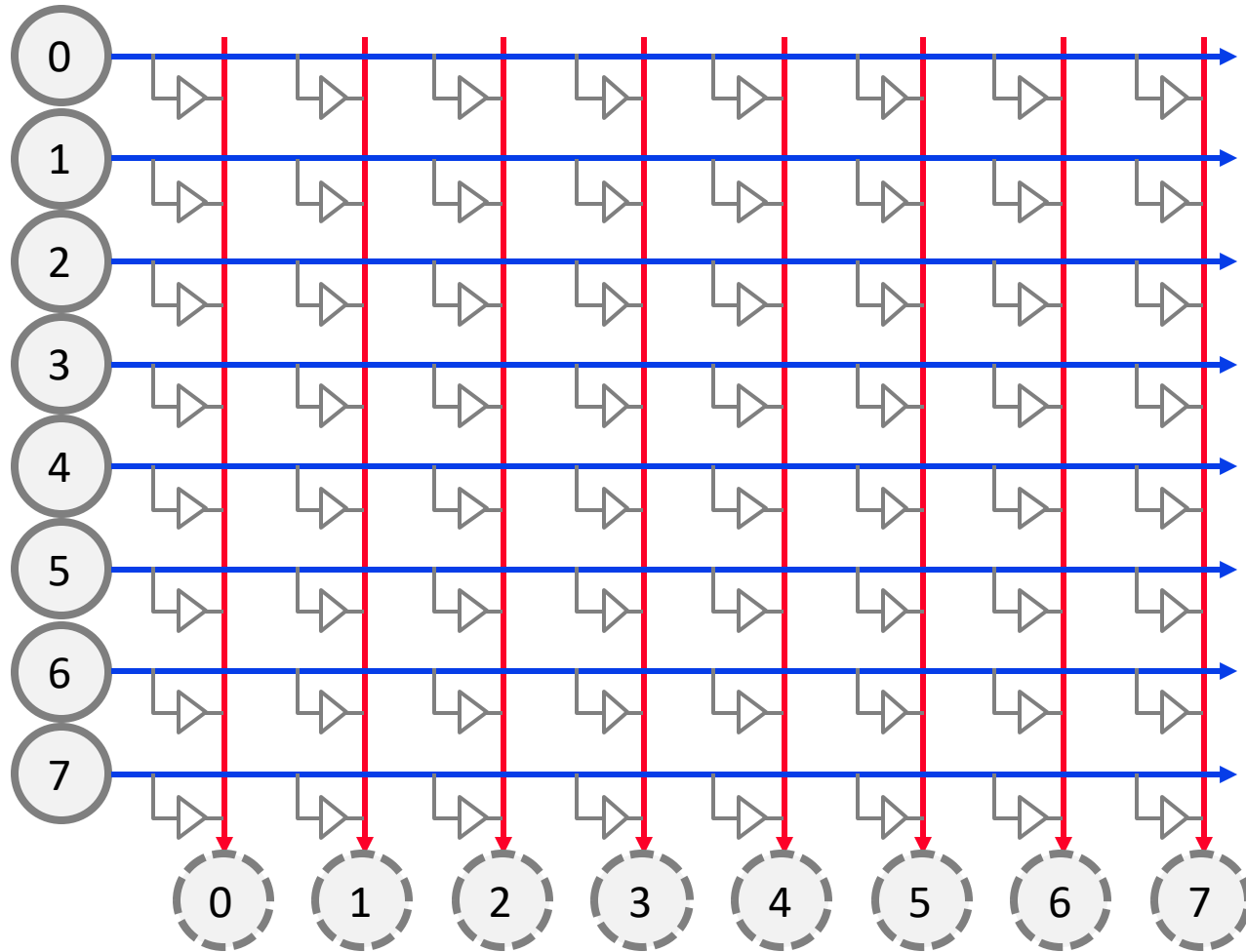
- Expensive
- Not scalable $\rightarrow O(N^2)$ cost
- Difficult to arbitrate as N increases

Used in core-to-cache-bank
networks in

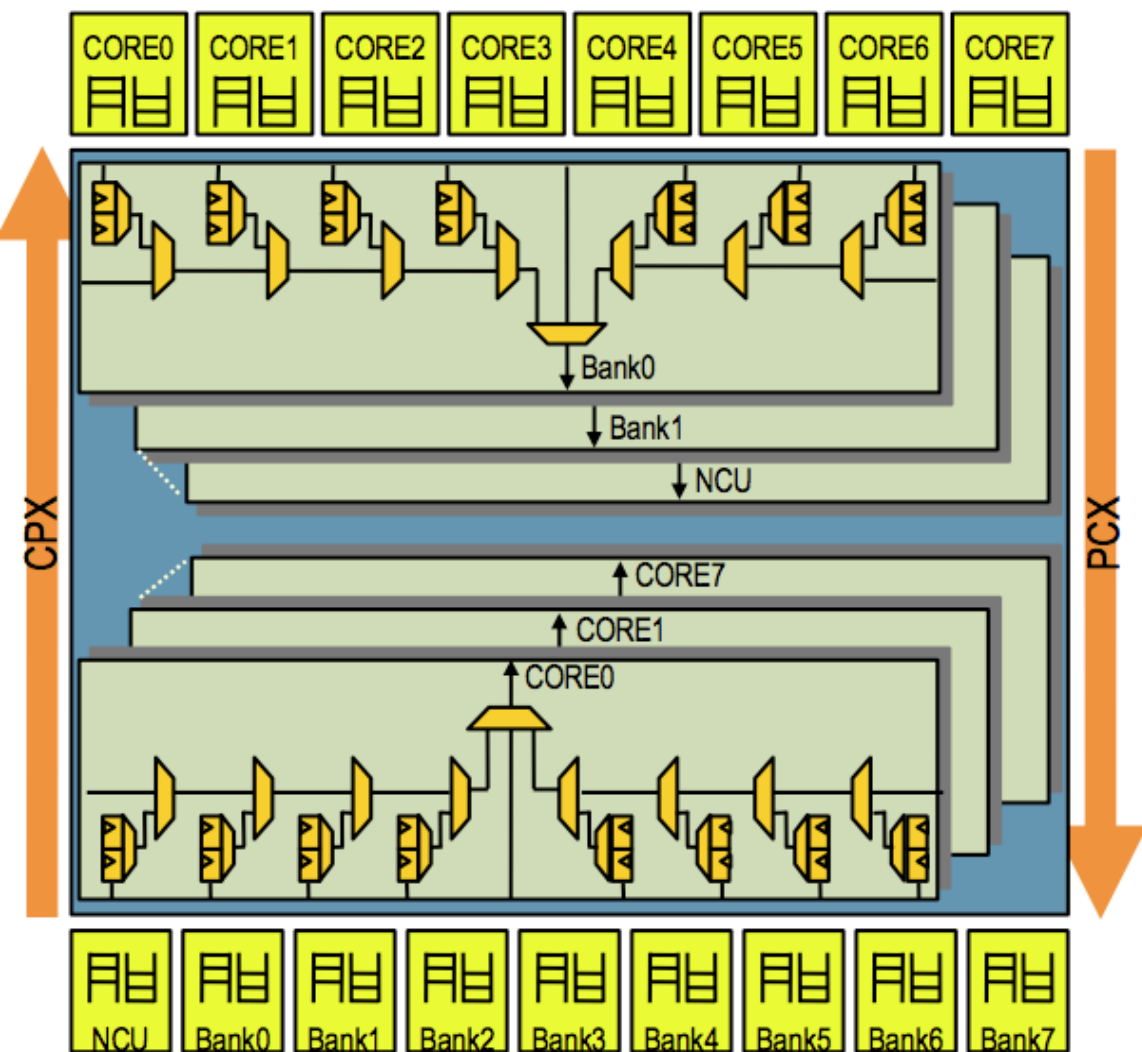
- IBM POWER5
- Sun Niagara I/II



Another Crossbar Design

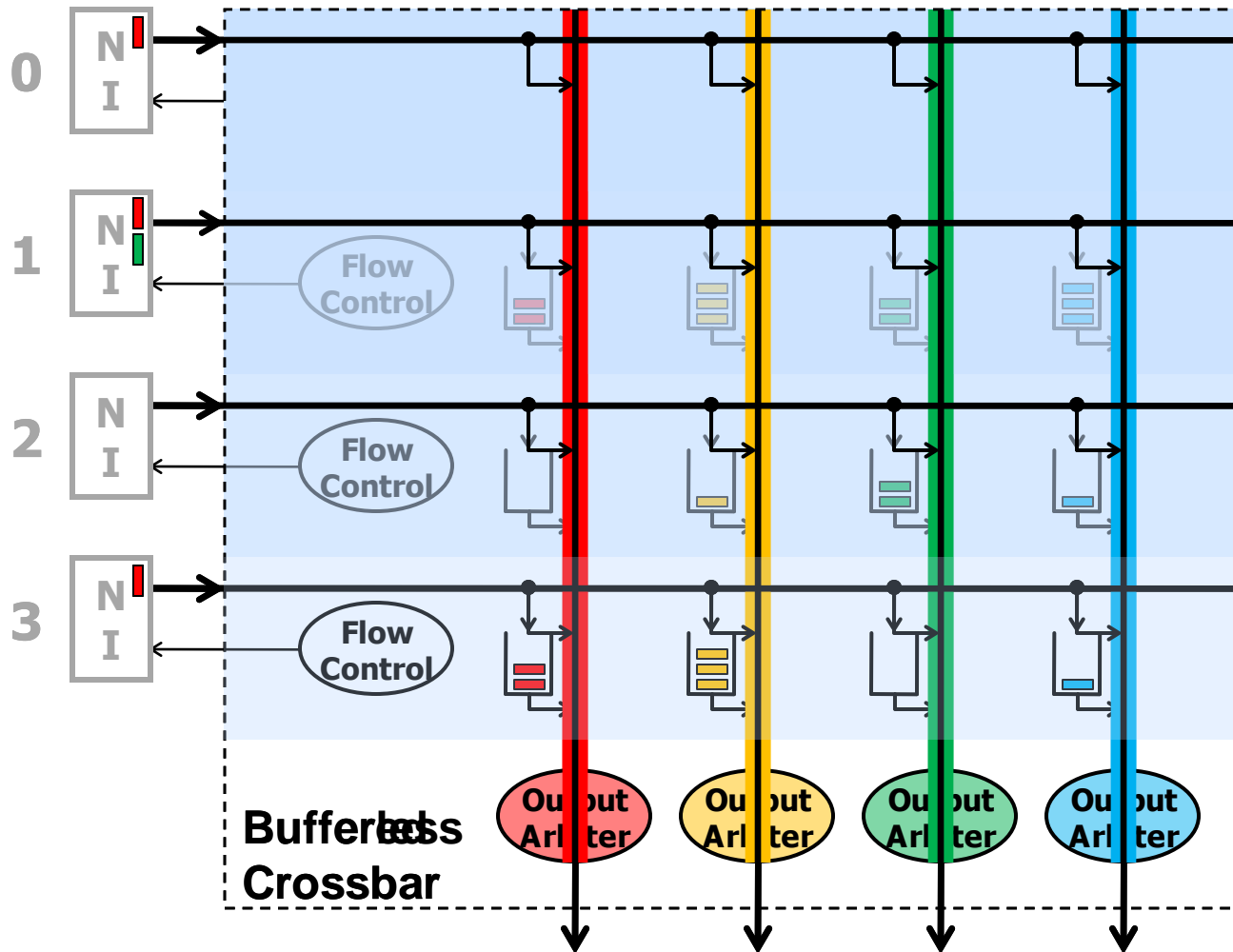


Sun UltraSPARC T2 Core-to-Cache Crossbar



- 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



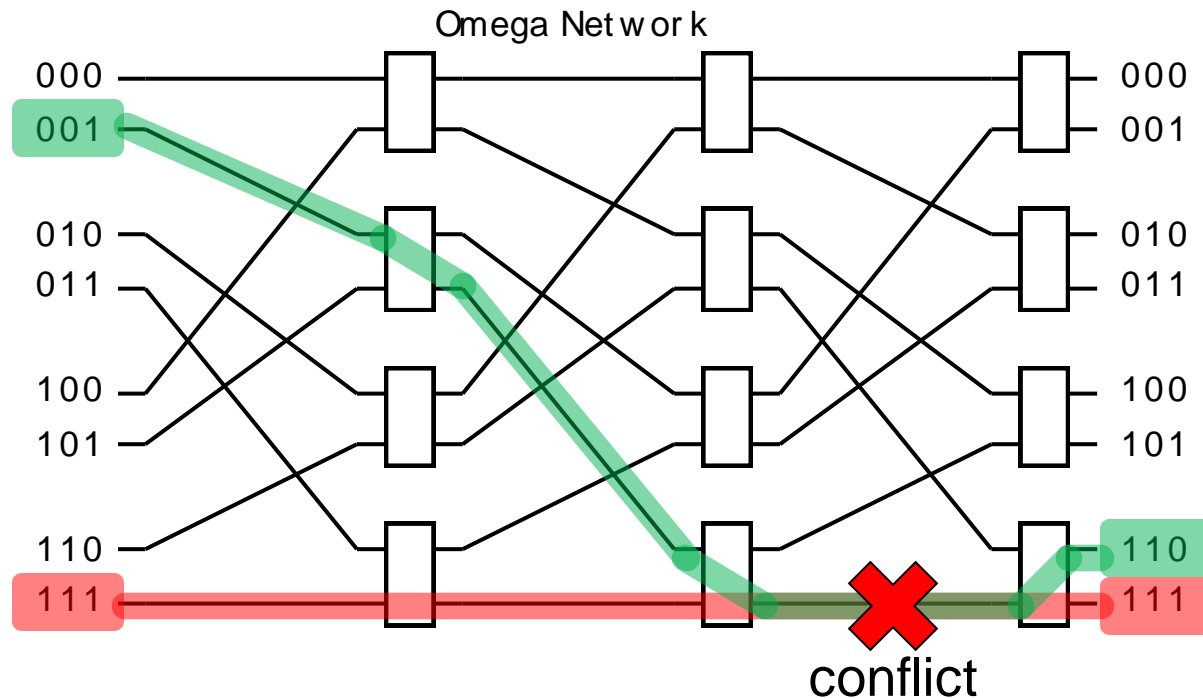
- + Simpler arbitration/scheduling
- + Efficient support for variable-size packets
- Requires N^2 buffers

Can We Get Lower Cost than A Crossbar?

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

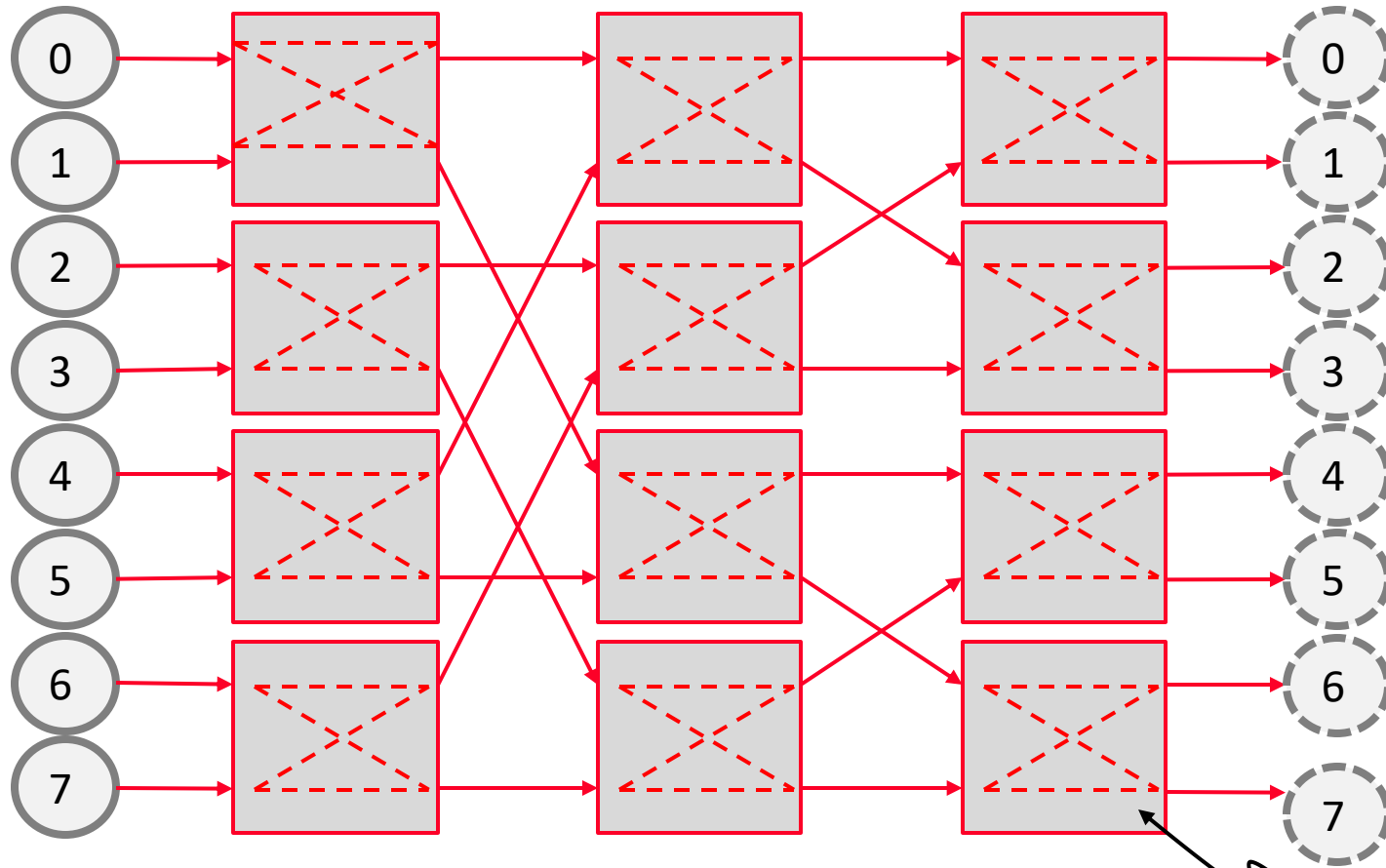
Multistage Logarithmic Networks

- Idea: Indirect networks with multiple layers of switches between terminals/nodes
- Cost: $O(N \log N)$, Latency: $O(\log N)$
- Many variations (Omega, Butterfly, Benes, Banyan, ...)
- Omega Network:



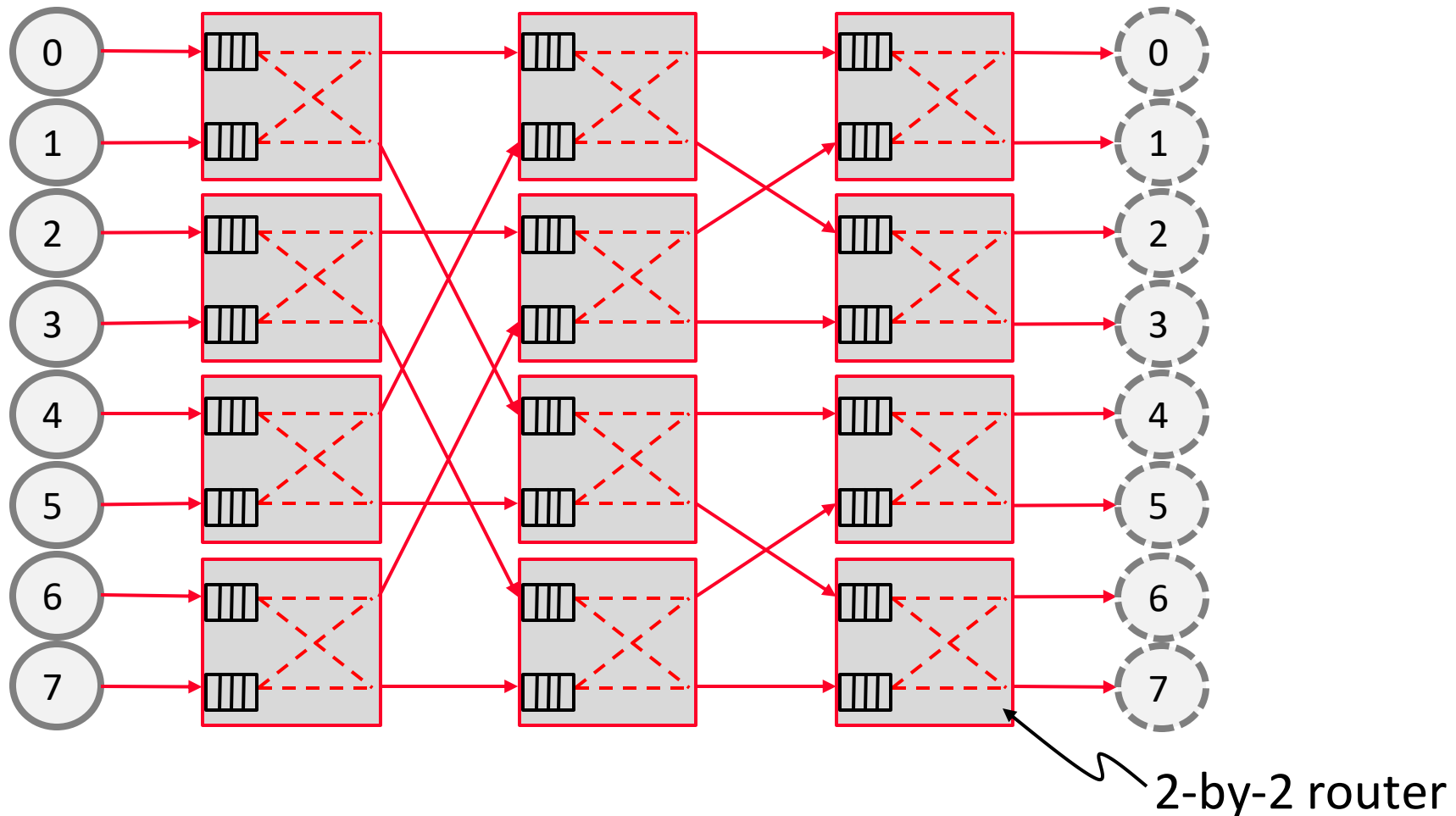
Blocking
or
Non-blocking?

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., $O(N \log N)$ 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
 - + no buffering
 - setting up and bringing down “path” takes time

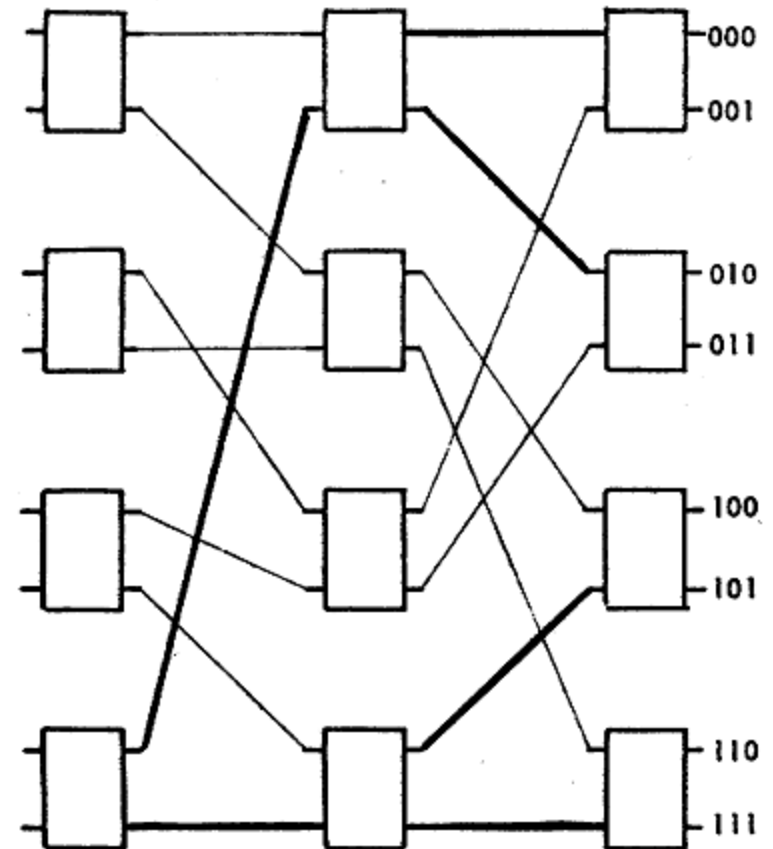
- **Packet switching** routes per packet in each router
 - Route each packet individually (possibly via different paths)
 - If link is free, any packet can use it
 - potentially slower --- must dynamically switch
 - need buffering
 - + no setup, bring down time
 - + more flexible, does not underutilize 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.

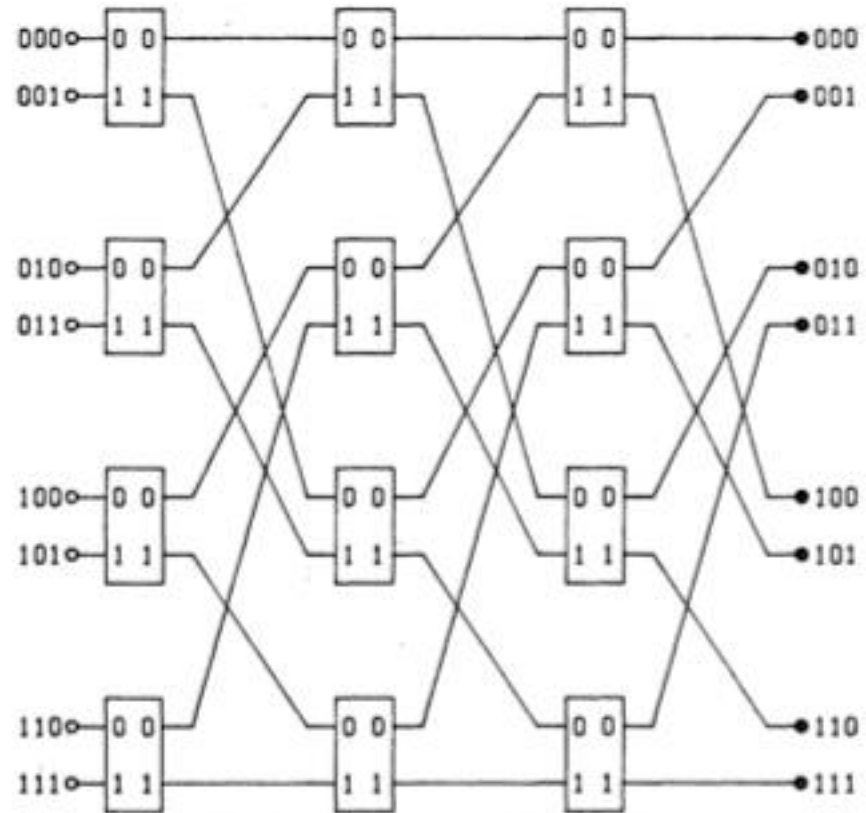


Fig. 2. Omega-network ($N = 8$).

Combining Operations in the Network

- Idea: Combine multiple operations on a shared memory location
- Example: Omega network switches combine fetch-and-add operations in NYU Ultracomputer
- Fetch-and-add(M, I): return M , replace M with $M+I$
 - Common when parallel processors modify a shared variable, e.g. obtain a chunk of the array
- Combining reduces synchronization latency

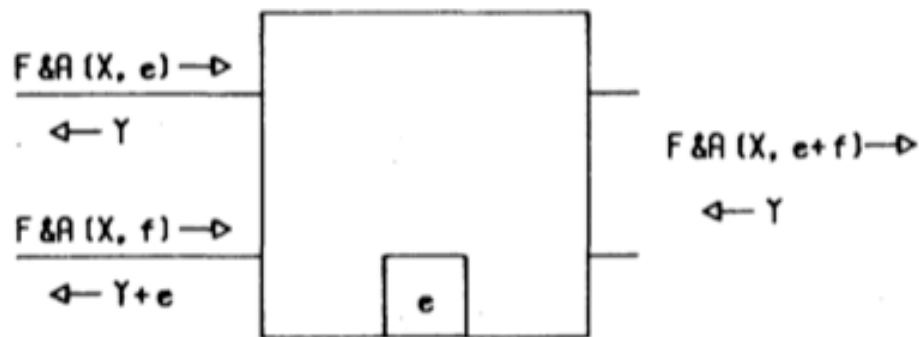


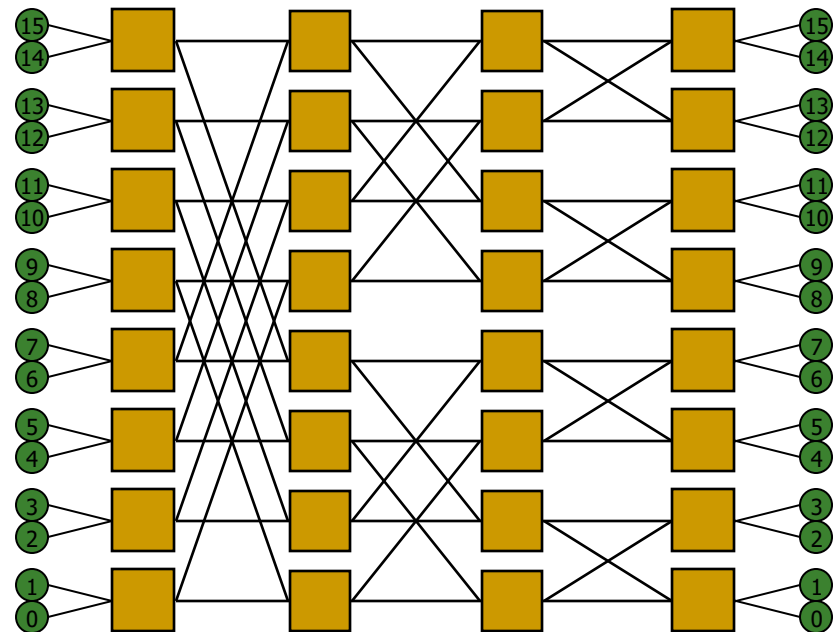
Fig. 3. Combining Fetch-and-Adds.

```
TestAndSet(V)
{Temp  $\leftarrow V$ 
  V  $\leftarrow$  TRUE}
RETURN Temp.
```

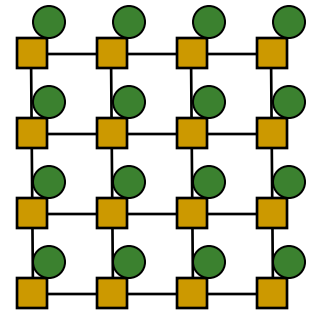
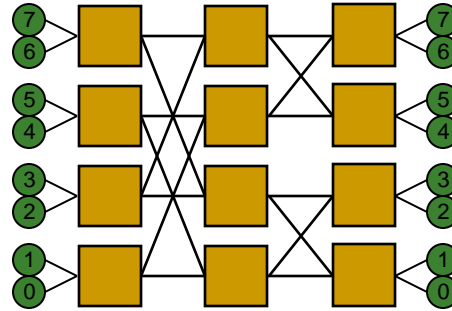
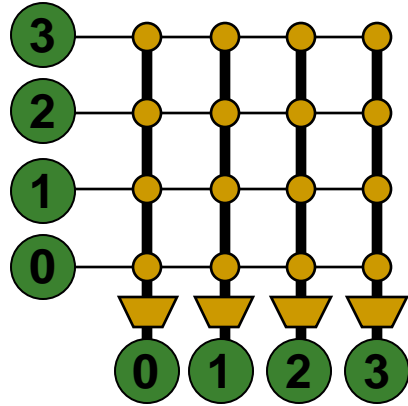
Fetch&OR(V, TRUE).

Butterfly

- Equivalent to Omega Network
- Indirect
- Used in BBN Butterfly
- Conflicts can cause "*tree saturation*"
 - Randomization of route selection helps



Review: Topologies



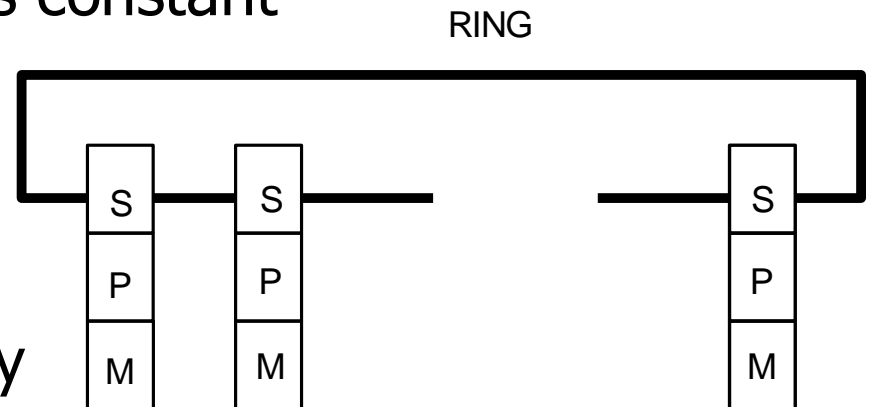
Topology	Crossbar	Multistage Logarithm.	Mesh
Direct/Indirect	Indirect	Indirect	Direct
Blocking/ Non-blocking	Non-blocking	Blocking	Blocking
Cost	$O(N^2)$	$O(N \log N)$	$O(N)$
Latency	$O(1)$	$O(\log N)$	$O(\sqrt{N})$

Ring

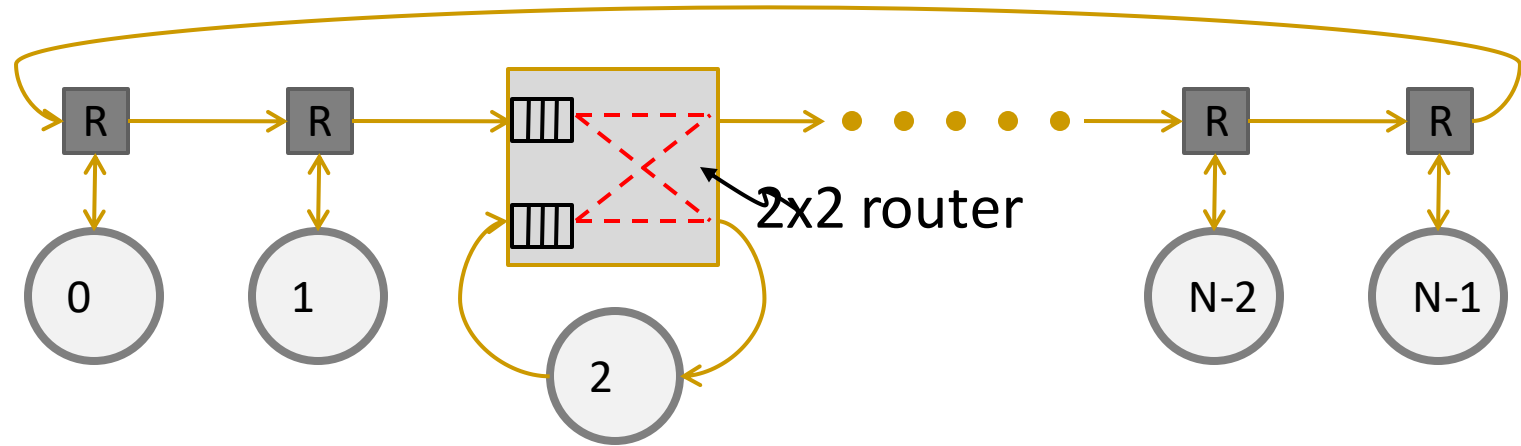
Each node connected to exactly two other nodes. Nodes form a continuous pathway such that packets can reach any node.

- + 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



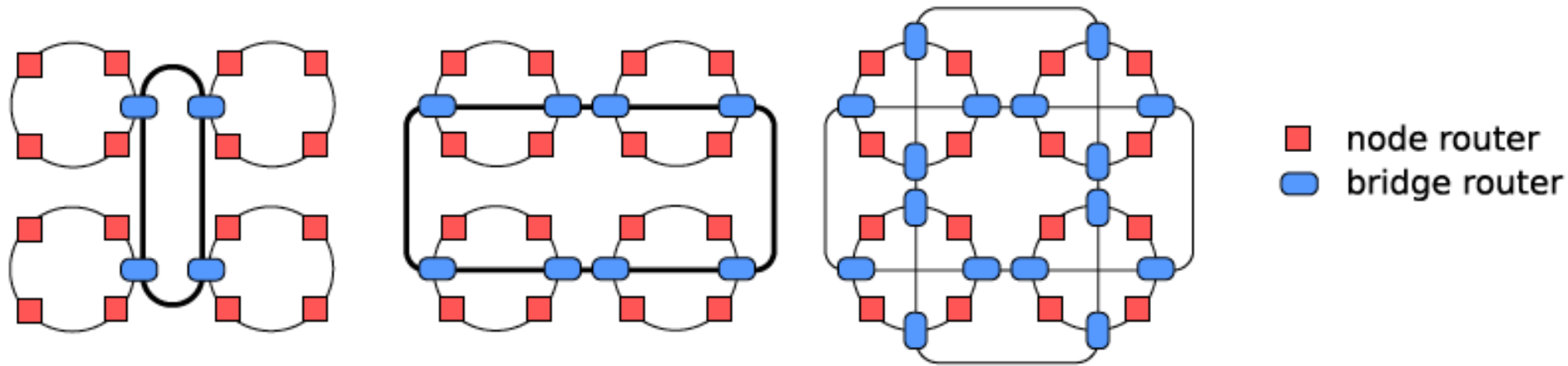
- Single directional pathway
- Simple topology and implementation
 - Reasonable performance if N and performance needs (bandwidth & latency) still moderately low
 - $O(N)$ cost
 - $N/2$ average hops; latency depends on utilization

Bidirectional Rings

Multi-directional pathways, or multiple rings

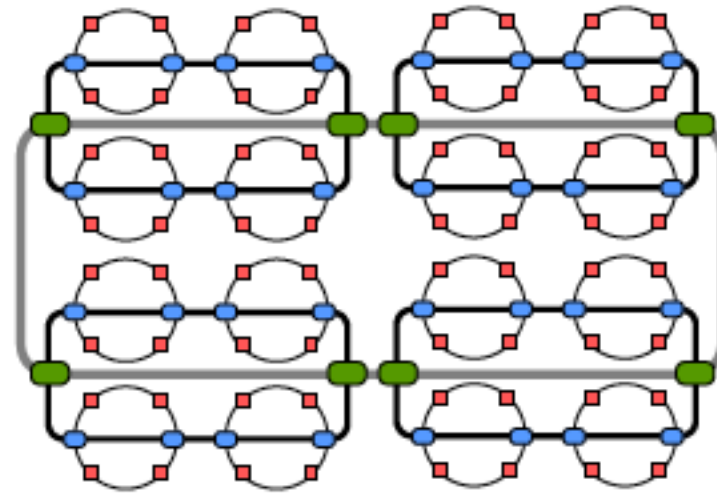
- + Reduces latency
- + Improves scalability
- Slightly more complex injection policy (need to select which ring to inject a packet into)

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 Ausavarungnirun, Chris Fallin, Xiangyao Yu, Kevin Chang, Greg Nazario, Reetuparna Das, Gabriel Loh, and Onur Mutlu,
"Design and Evaluation of Hierarchical Rings with Deflection Routing"
Proceedings of the 26th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), Paris, France, October 2014. [[Slides \(pptx\)](#)] [[pdf](#)] [[Source Code](#)]
- Describes the design and implementation of a mostly-bufferless hierarchical ring

Design and Evaluation of Hierarchical Rings with Deflection Routing

Rachata Ausavarungnirun Chris Fallin Xiangyao Yu† Kevin Kai-Wei Chang
Greg Nazario Reetuparna Das§ Gabriel H. Loh‡ Onur Mutlu

Carnegie Mellon University §University of Michigan †MIT ‡Advanced Micro Devices, Inc.

More on Hierarchical Rings (II)

- Rachata Ausavarungnirun, Chris Fallin, Xiangyao Yu, Kevin Chang, Greg Nazario, Reetuparna Das, Gabriel Loh, and Onur Mutlu,
"A Case for Hierarchical Rings with Deflection Routing: An Energy-Efficient On-Chip Communication Substrate"
Parallel Computing (PARCO), to appear in 2016. [arXiv.org version](#), February 2016.

A case for hierarchical rings with deflection routing:
An energy-efficient on-chip communication substrate

Rachata Ausavarungnirun^{a,*}, Chris Fallin^a, Xiangyao Yu^b, Kevin Kai-Wei Chang^a,
Greg Nazario^a, Reetuparna Das^c, Gabriel H. Loh^d, Onur Mutlu^a

^a Carnegie Mellon University, United States

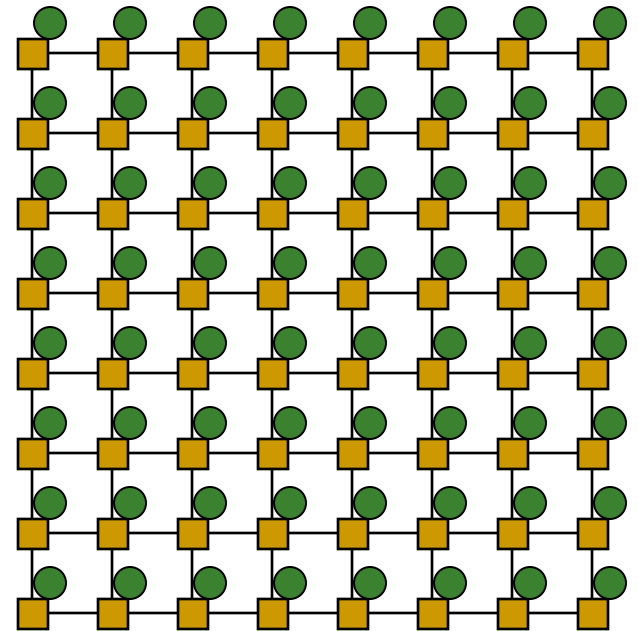
^b University of Michigan, United States

^c Massachusetts Institute of Technology, United States

^d Advanced Micro Devices, United States

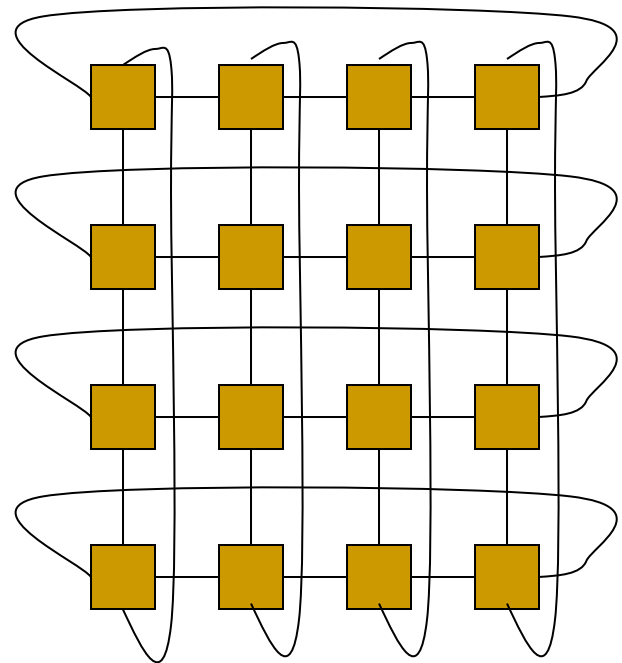
Mesh

- 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



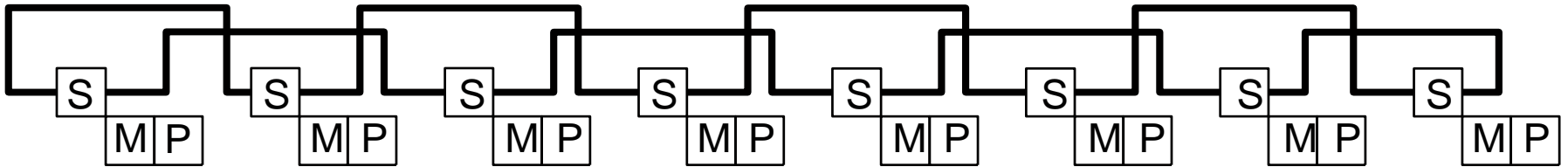
Torus

- 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 \sim constant



Trees

Planar, hierarchical topology

Latency: $O(\log N)$

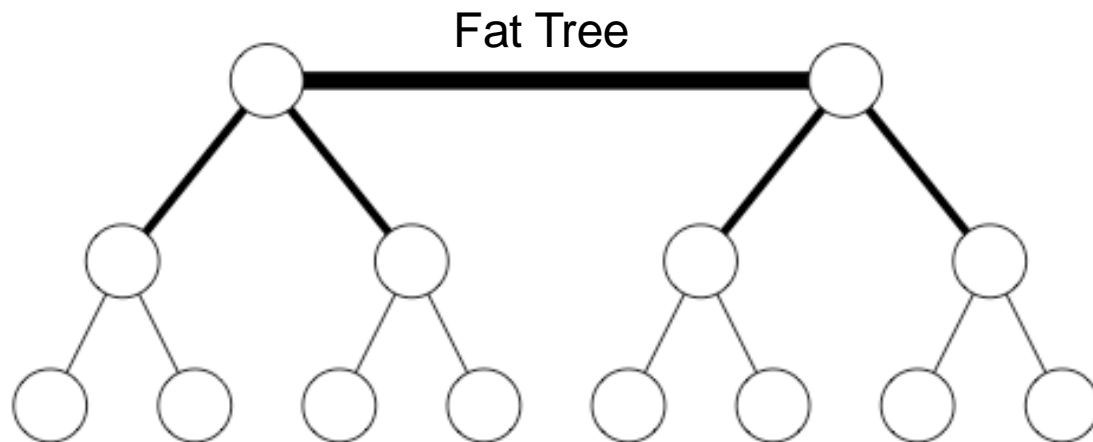
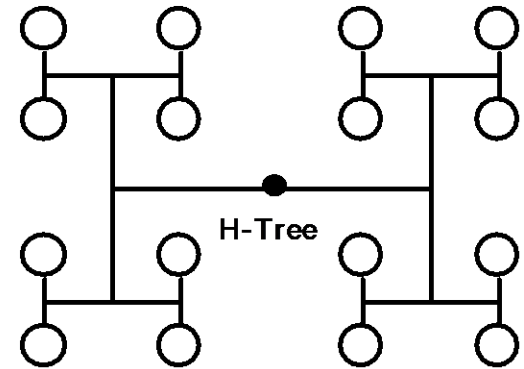
Good for local traffic

+ Cheap: $O(N)$ cost

+ Easy to Layout

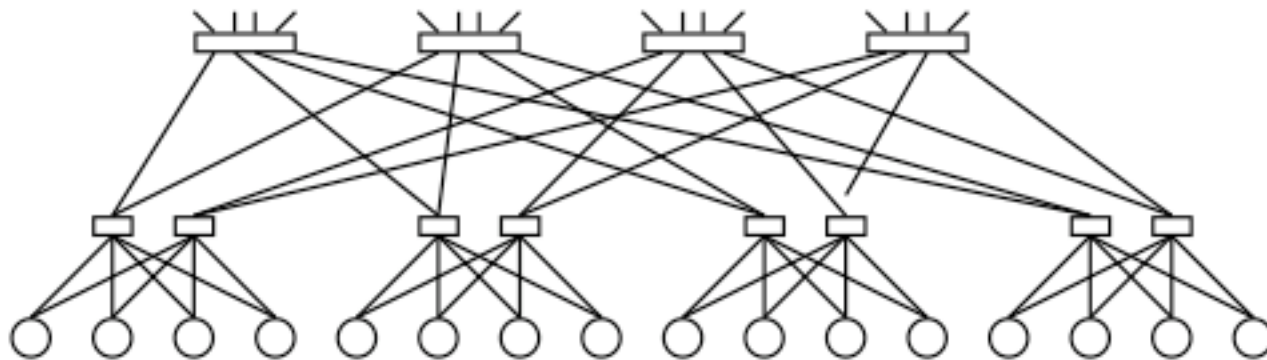
- Root can become a bottleneck

Fat trees avoid this problem (CM-5)



CM-5 Fat Tree

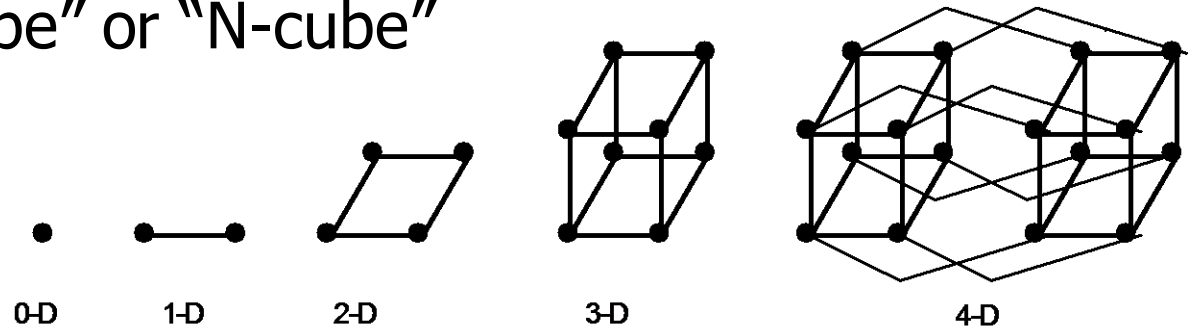
- Fat tree based on 4x2 switches
- Randomized routing on the way up
- Combining, multicast, reduction operators supported in hardware
 - Thinking Machines Corp., “[The Connection Machine CM-5 Technical Summary](#),” Jan. 1992.



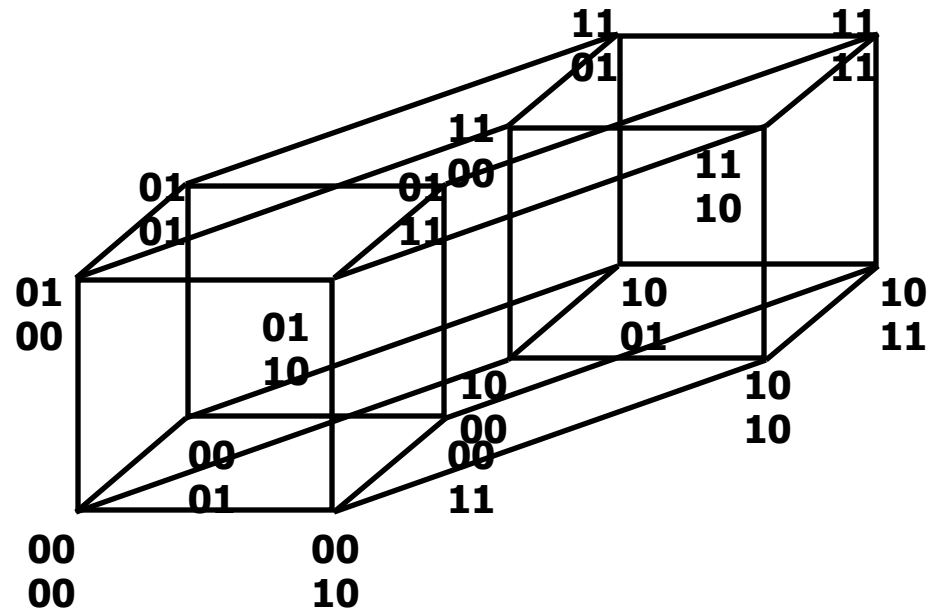
CM-5 Thinned Fat Tree

Hypercube

- “N-dimensional cube” or “N-cube”

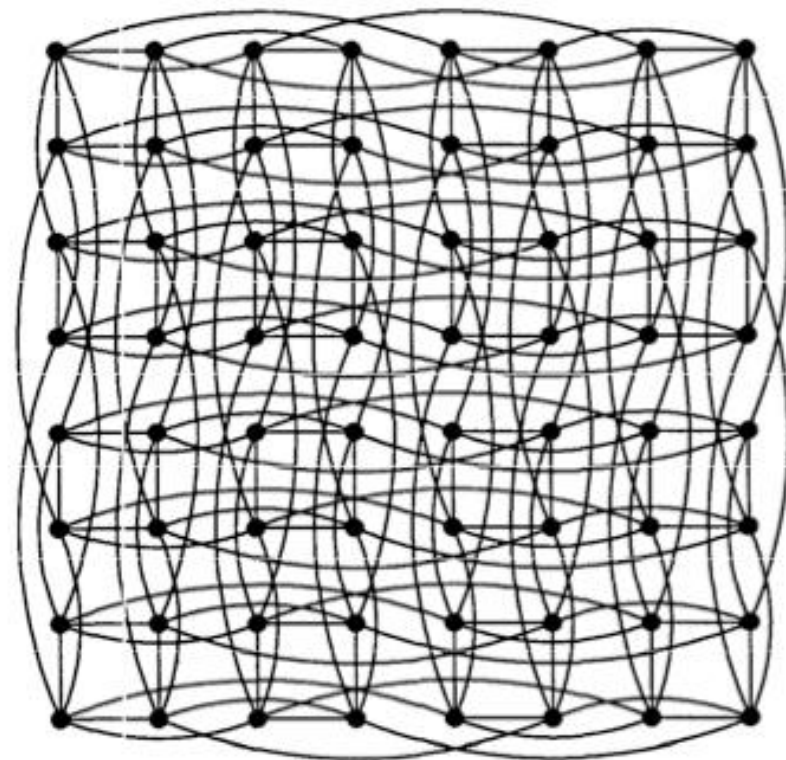
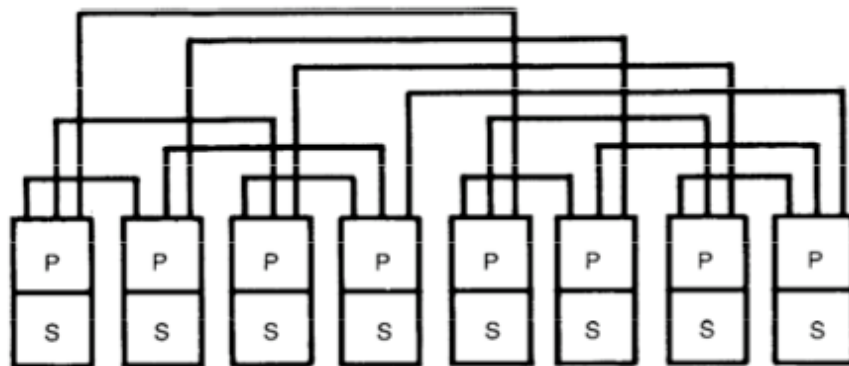


- Latency: $O(\log N)$
 - Radix: $O(\log N)$
 - #links: $O(N \log N)$
- + Low latency
- Hard to lay out in 2D/3D



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)

Routing

Routing Mechanism

■ Arithmetic

- Simple arithmetic to determine route in regular topologies
- Dimension order routing in meshes/tori

■ Source Based

- Source specifies output port for each switch in route
- + Simple switches
 - no control state: strip output port off header
- Large header

■ Table Lookup Based

- Index into table for output port
- + Small header
- More complex switches

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

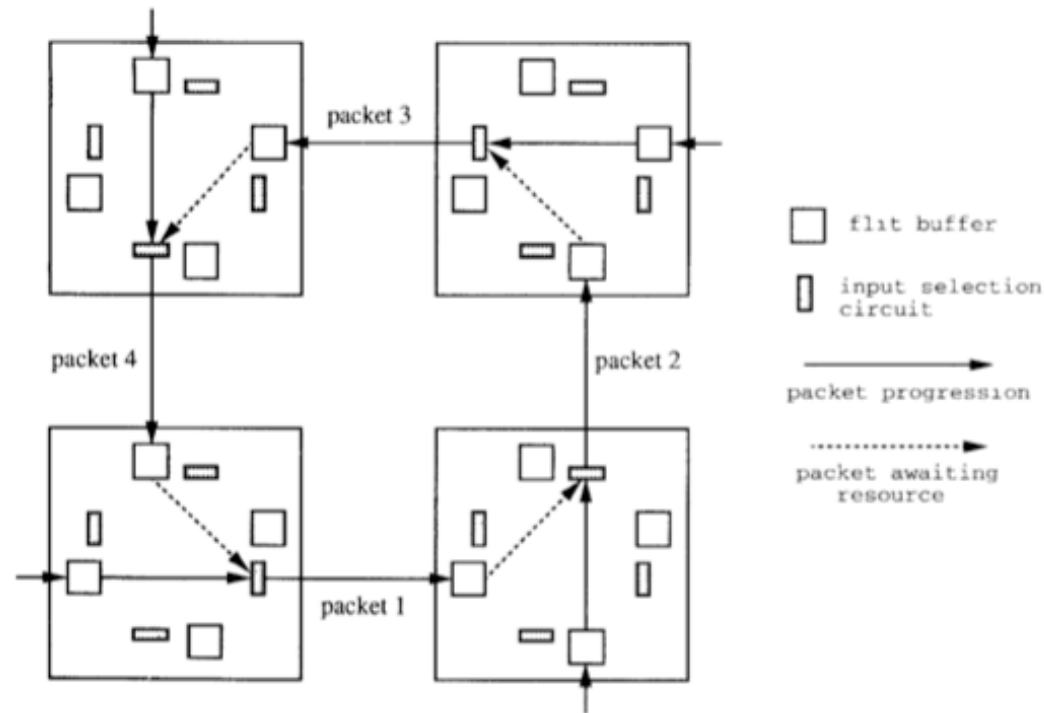
- ❑ Local/global feedback
- ❑ Minimal or non-minimal paths

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.

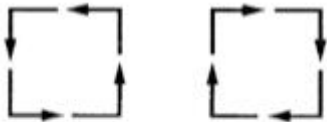


FIG. 2. The possible turns and simple cycles in a two-dimensional mesh.



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

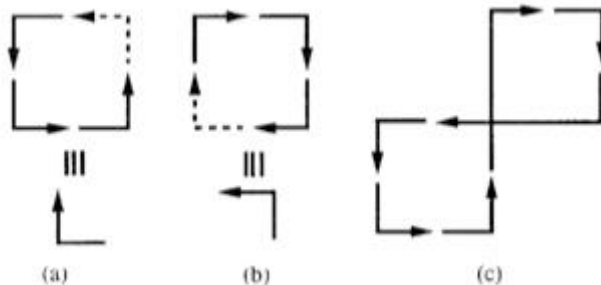


FIG. 4. Six turns that complete the cycles and allow deadlock.

Oblivious Routing: Valiant's Algorithm

- 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)

More on Valiant's Algorithm

- Valiant and Brebner, “[Universal Schemes for Parallel Communication](#),” STOC 1981.
- Valiant, “[A Scheme for Fast Parallel Communication](#),” SIAM Journal of Computing, 1982.

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

More on Adaptive Routing

- Can avoid faulty links/routers
 - Idea: **Route around faults**
- + Deterministic routing cannot handle faulty components
- Need to change the routing table to disable faulty routes
 - Assuming the faulty link/router is detected

One recent example:

Fattah et al., **"A Low-Overhead, Fully-Distributed, Guaranteed-Delivery Routing Algorithm for Faulty Network-on-Chips"**, NOCS 2015.

Recent Example: Fault Tolerance

- Mohammad Fattah, Antti Airola, Rachata Ausavarungnirun, Nima Mirzaei, Pasi Liljeberg, Juha Plosila, Siamak Mohammadi, Tapio Pahikkala, Onur Mutlu, and Hannu Tenhunen, **"A Low-Overhead, Fully-Distributed, Guaranteed-Delivery Routing Algorithm for Faulty Network-on-Chips"**

Proceedings of the 9th ACM/IEEE International Symposium on Networks on Chip (NOCS), Vancouver, BC, Canada, September 2015.

[[Slides \(pptx\)](#) ([pdf](#))]

[[Source Code](#)]

One of the three papers nominated for the Best Paper Award by the Program Committee.

A Low-Overhead, Fully-Distributed, Guaranteed-Delivery Routing Algorithm for Faulty Network-on-Chips

Mohammad Fattah¹, Antti Airola¹, Rachata Ausavarungnirun², Nima Mirzaei³,
Pasi Liljeberg¹, Juha Plosila¹, Siamak Mohammadi³, Tapio Pahikkala¹,
Onur Mutlu² and Hannu Tenhunen^{1,4}

Buffering and Flow Control

Recall: Circuit vs. Packet Switching

- **Circuit switching** sets up full path before transmission
 - Establish route then send data
 - No one else can use those links while “circuit” is set
 - + faster arbitration
 - setting up and bringing down “path” takes time

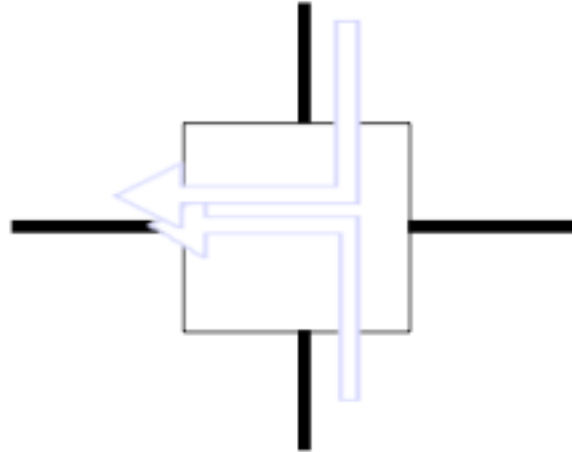
- **Packet switching** routes per packet in each router
 - Route each packet individually (possibly via different paths)
 - If link is free, any packet can use it
 - potentially slower --- must dynamically switch
 - + no setup, bring down time
 - + more flexible, does not underutilize links

Packet Switched Networks: Packet Format

- **Header**
 - routing and control information
- **Payload**
 - carries data (non HW specific information)
 - can be further divided (framing, protocol stacks...)
- **Error Code**
 - generally at tail of packet so it can be generated on the way out



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?

Flow Control Methods

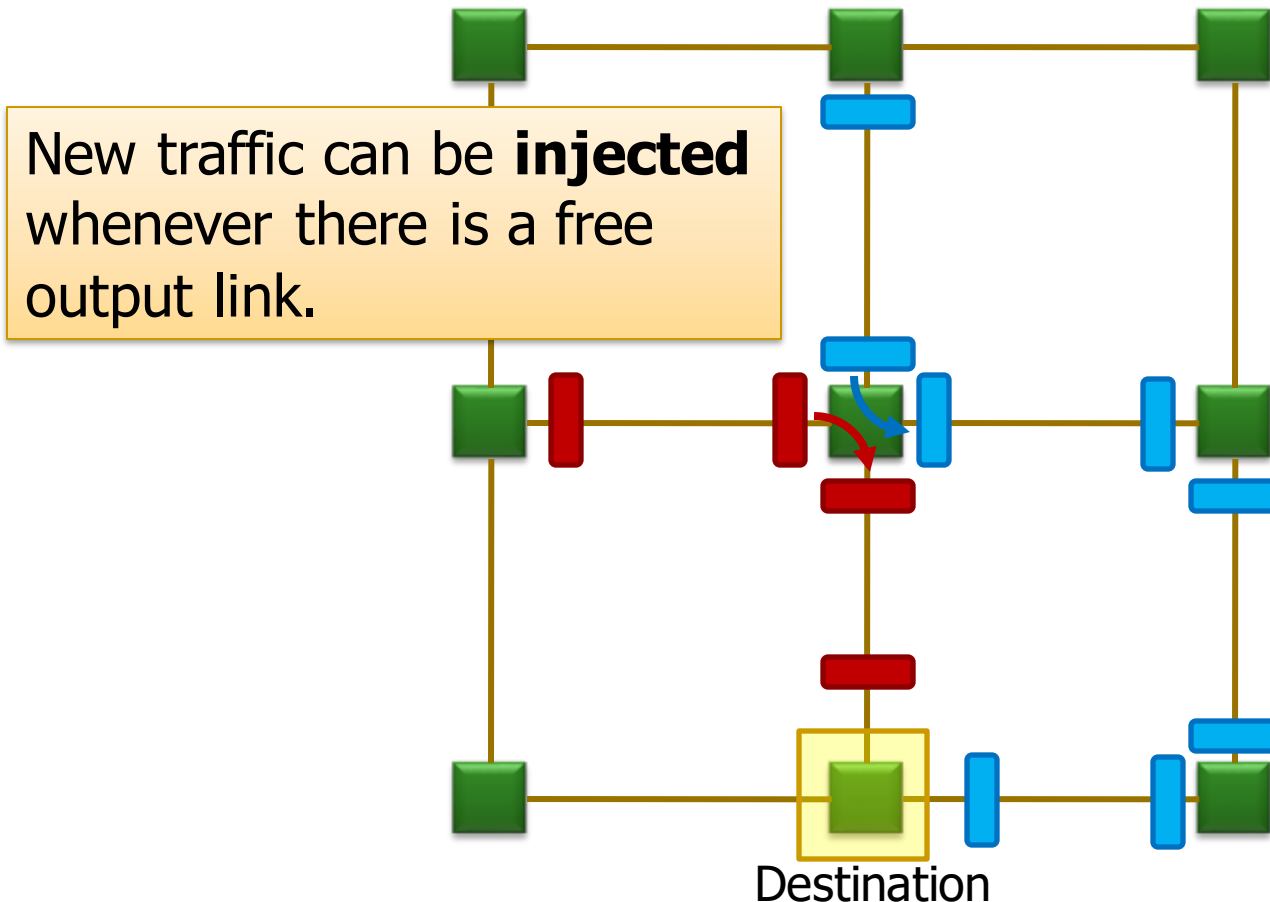
- Circuit switching
- Bufferless (Packet/flit based)
- Store and forward (Packet based)
- Virtual cut through (Packet based)
- Wormhole (Flit based)

Circuit Switching Revisited

- Resource allocation granularity is high
 - Idea: Pre-allocate resources across multiple switches for a given “flow”
 - Need to send a probe to set up the path for pre-allocation
-
- + No need for buffering
 - + No contention (flow’s performance is isolated)
 - + Can handle arbitrary message sizes
 - Lower link utilization: two flows cannot use the same link
 - Handshake overhead to set up a “circuit”

Bufferless Deflection Routing

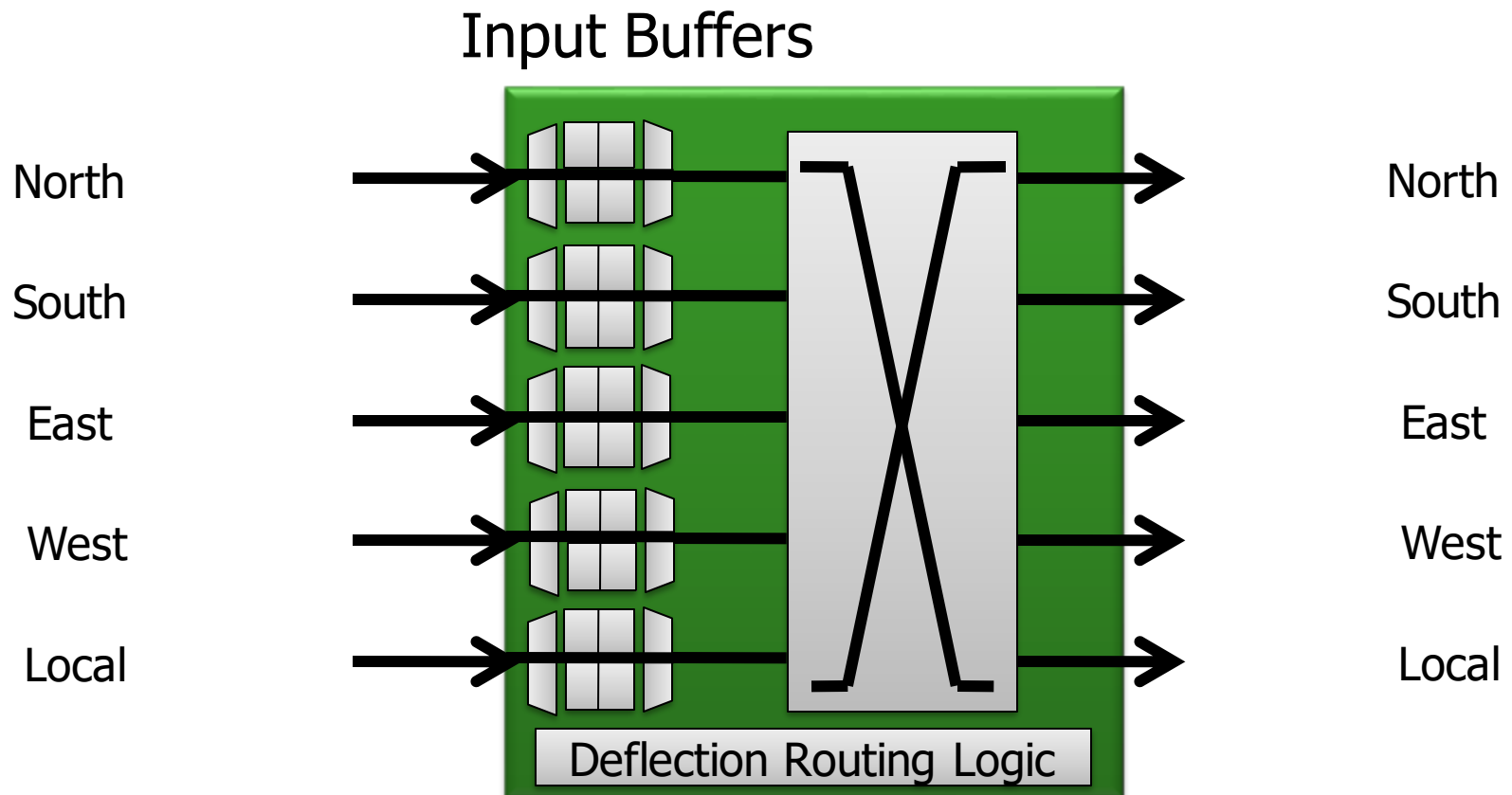
- **Key idea:** Packets are never buffered in the network. When two packets contend for the same link, one is **deflected**.¹



¹Baran, "On Distributed Communication Networks." RAND Tech. Report., 1962 / IEEE Trans.Comm., 1964.₆₅

Bufferless Deflection Routing

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



Issues In Bufferless Deflection Routing

- Livelock
- Resulting Router Complexity
- Performance & Congestion at High Loads
- Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin Chang, Rachata Ausavarungnirun, and Onur Mutlu,
"Bufferless and Minimally-Buffered Deflection Routing"
Invited Book Chapter in Routing Algorithms in Networks-on-Chip, pp. 241-275, Springer, 2014.

Bufferless Deflection Routing in NoCs

- Moscibroda and Mutlu, “A Case for Bufferless Routing in On-Chip Networks,” ISCA 2009.
 - https://users.ece.cmu.edu/~omutlu/pub/bless_isca09.pdf

A Case for Bufferless Routing in On-Chip Networks

Thomas Moscibroda
Microsoft Research
moscitho@microsoft.com

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu

Low-Complexity Bufferless Routing

- Chris Fallin, Chris Craik, and Onur Mutlu,
"CHIPPER: A Low-Complexity Bufferless Deflection Router"
Proceedings of the 17th International Symposium on High-Performance Computer Architecture (HPCA), pages 144-155, San Antonio, TX, February 2011. Slides (pptx)
An extended version as *SAFARI Technical Report*, TR-SAFARI-2010-001, Carnegie Mellon University, December 2010.

CHIPPER: A Low-complexity Bufferless Deflection Router

Chris Fallin Chris Craik Onur Mutlu
cfallin@cmu.edu craik@cmu.edu onur@cmu.edu

Computer Architecture Lab (CALCM)
Carnegie Mellon University

Minimally-Buffered Deflection Routing

- Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin Chang, Rachata Ausavarungnirun, and Onur Mutlu,
"MinBD: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect"
Proceedings of the 6th ACM/IEEE International Symposium on Networks on Chip (NOCS), Lyngby, Denmark, May 2012. [Slides \(pptx\)](#) [\(pdf\)](#)

MinBD: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect

Chris Fallin, Greg Nazario, Xiangyao Yu[†], Kevin Chang, Rachata Ausavarungnirun, Onur Mutlu

Carnegie Mellon University
{cfallin,gnazario,kevincha,rachata,onur}@cmu.edu

[†]Tsinghua University & Carnegie Mellon University
yxythu@gmail.com

“Bufferless” Hierarchical Rings

- Ausavarungnirun et al., “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

Design and Evaluation of Hierarchical Rings with Deflection Routing

Rachata Ausavarungnirun Chris Fallin Xiangyao Yu† Kevin Kai-Wei Chang
Greg Nazario Reetuparna Das§ Gabriel H. Loh‡ Onur Mutlu

Carnegie Mellon University §University of Michigan †MIT ‡Advanced Micro Devices, Inc.

“Bufferless” Hierarchical Rings (II)

- Rachata Ausavarungnirun, Chris Fallin, Xiangyao Yu, Kevin Chang, Greg Nazario, Reetuparna Das, Gabriel Loh, and Onur Mutlu,
"A Case for Hierarchical Rings with Deflection Routing: An Energy-Efficient On-Chip Communication Substrate"
Parallel Computing (PARCO), to appear in 2016.
 - [arXiv.org version](https://arxiv.org/abs/1602.04878), February 2016.

Achieving both High Energy Efficiency
and High Performance in On-Chip Communication
using Hierarchical Rings with Deflection Routing

Rachata Ausavarungnirun Chris Fallin Xiangyao Yu[†] Kevin Kai-Wei Chang
Greg Nazario Reetuparna Das[§] Gabriel H. Loh[‡] Onur Mutlu
Carnegie Mellon University [§]University of Michigan [†]MIT [‡]AMD

Summary of Six Years of Research

- Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin Chang, Rachata Ausavarungnirun, and Onur Mutlu,
"Bufferless and Minimally-Buffered Deflection Routing"
Invited Book Chapter in Routing Algorithms in Networks-on-Chip, pp. 241-275, Springer, 2014.

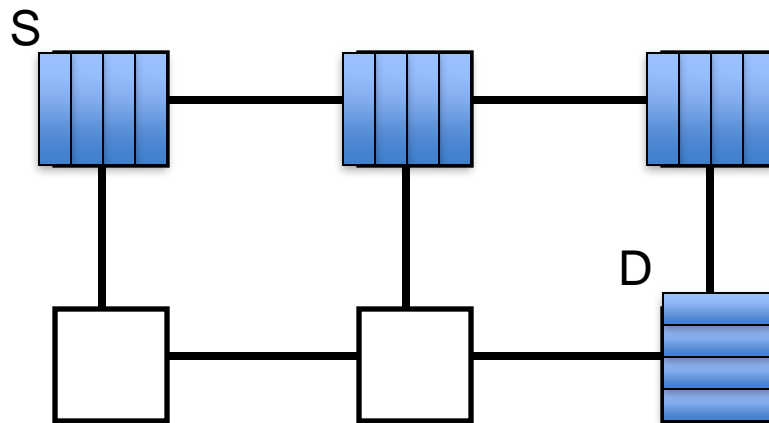
Chapter 1

Bufferless and Minimally-Buffered Deflection Routing

Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin Chang, Rachata Ausavarungnirun, Onur Mutlu

Store and Forward Flow Control

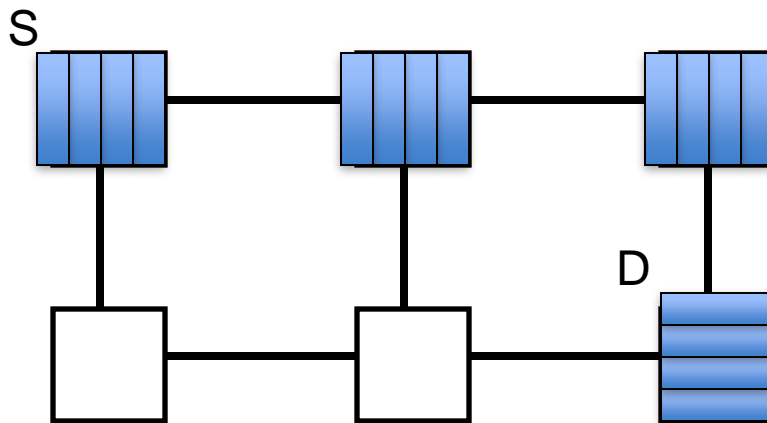
- Packet-based flow control
- Store and Forward
 - Packet copied entirely into network router before moving to the next node
 - Flow control unit is the entire packet
- Leads to high per-packet latency
- Requires buffering for entire packet in each node



Can we do better?

Cut through Flow Control

- Another form of packet-based flow control
- Start forwarding as soon as header is received and resources (buffer, channel, etc) allocated
 - Dramatic reduction in latency
- Still allocate buffers and channel bandwidth for full packets

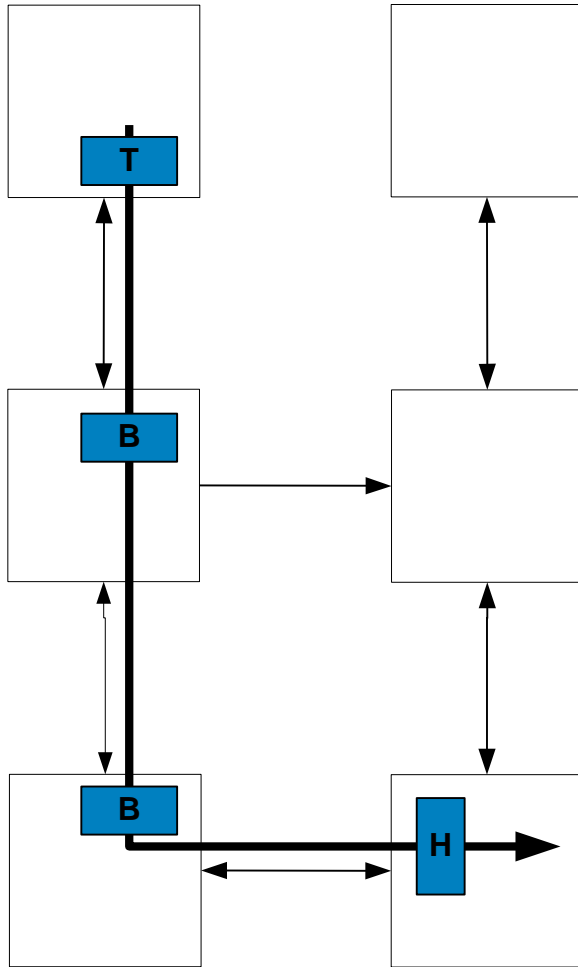


- What if packets are large?

Cut through Flow Control

- What to do if output port is blocked?
- Lets the tail continue when the head is blocked, absorbing the whole message into a single switch.
 - Requires a buffer large enough to hold the largest packet.
- Degenerates to store-and-forward with high contention
- **Can we do better?**

Wormhole Flow Control



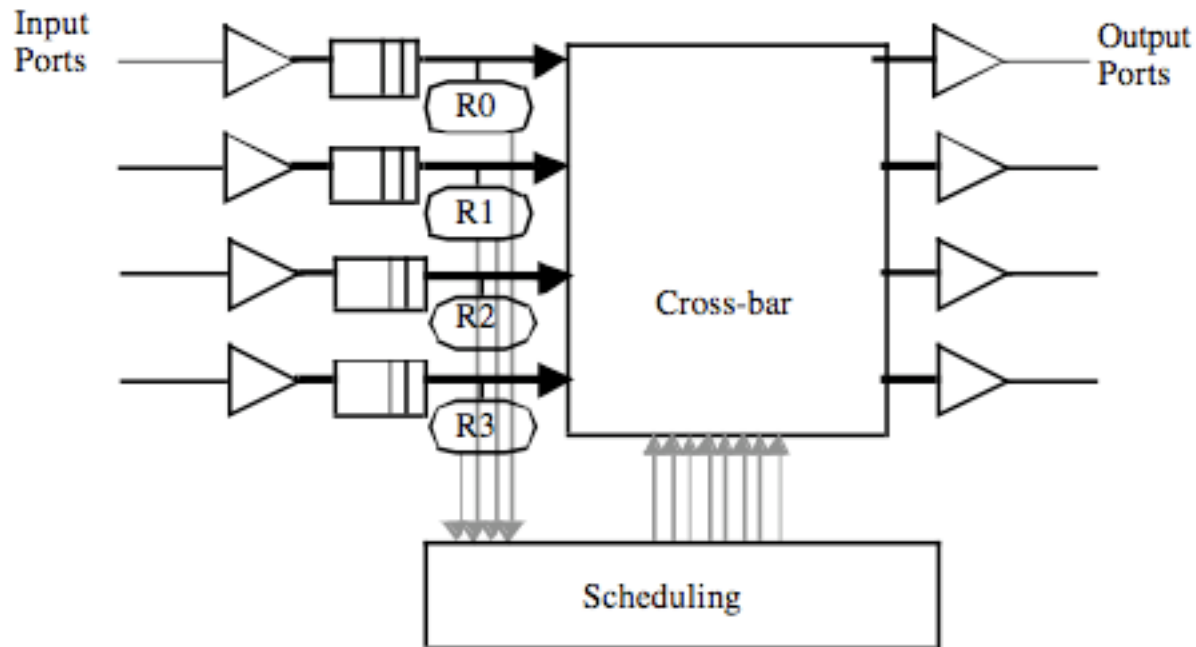
- Packets broken into (potentially) smaller flits (buffer/bw allocation unit)
- Flits are sent across the fabric in a *wormhole fashion*
 - Body follows head, tail follows body
 - Pipelined
 - If head blocked, rest of packet stops
 - Routing (src/dest) information only in head
- How does body/tail know where to go?
- Latency almost independent of distance for long messages

Wormhole Flow Control

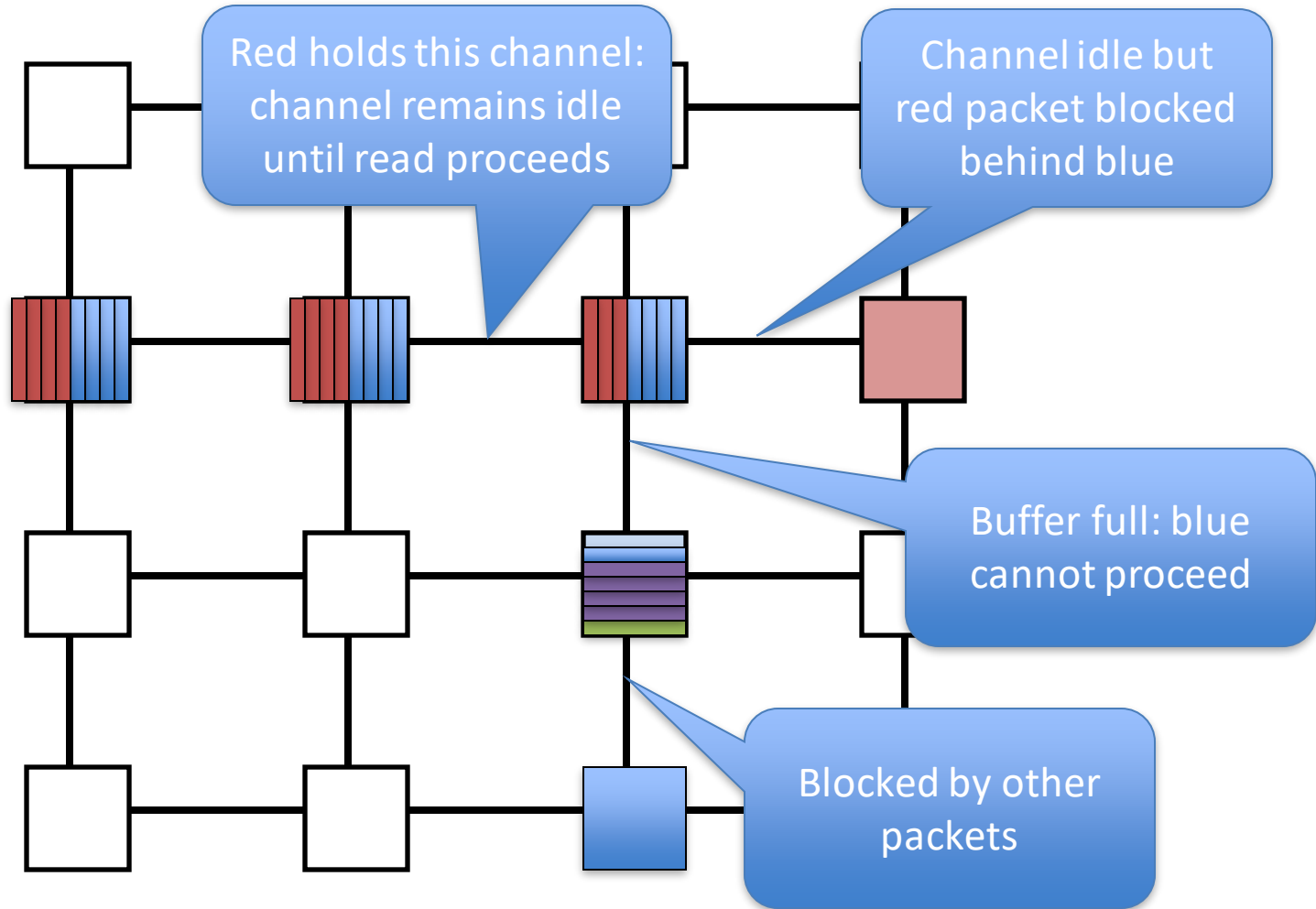
- Advantages over “store and forward” flow control
 - + Lower latency
 - + More efficient buffer utilization
- Limitations
 - Suffers from **head of line blocking**
 - If head flit cannot move due to contention, another worm cannot proceed even though links may be idle

Head of Line Blocking

- A worm can be before another in the router input buffer
- Due to FIFO nature, the second worm cannot be scheduled even though it may need to access another output port

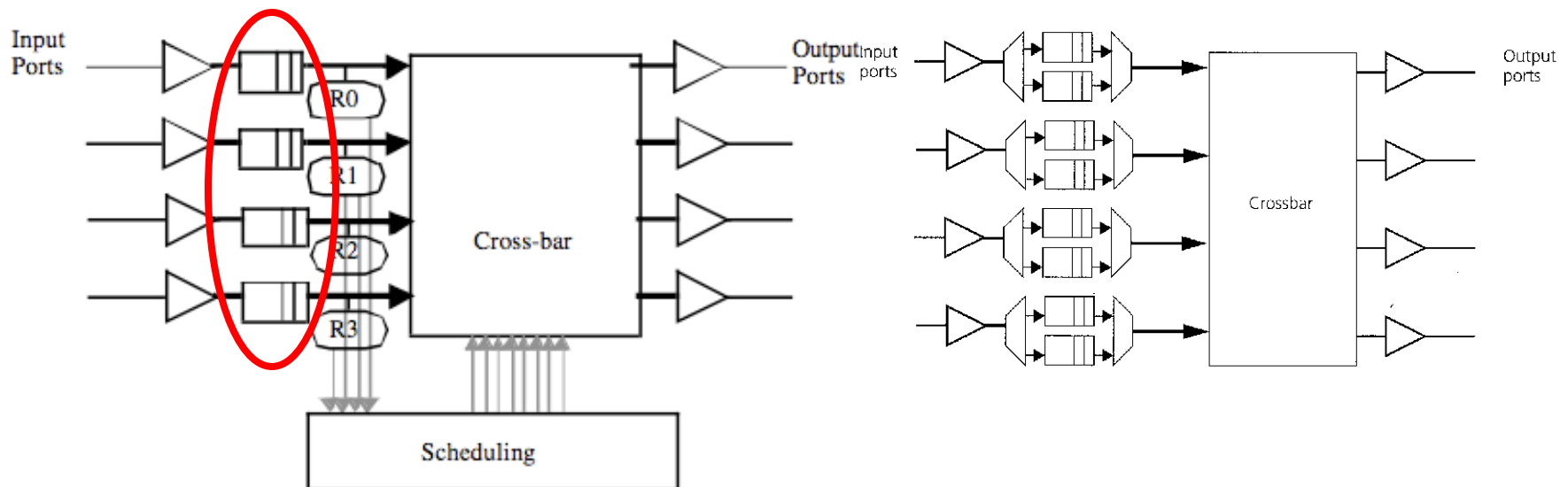


Head of Line Blocking



Virtual Channel Flow Control

- Idea: Multiplex multiple channels over one physical channel
- Divide up the input buffer into multiple buffers sharing a single physical channel
- Dally, “Virtual Channel Flow Control,” ISCA 1990.



Virtual Channel Flow Control

- Idea: Multiplex multiple channels over one physical channel
- Divide up the input buffer into multiple buffers sharing a single physical channel
- Dally, “Virtual Channel Flow Control,” ISCA 1990.

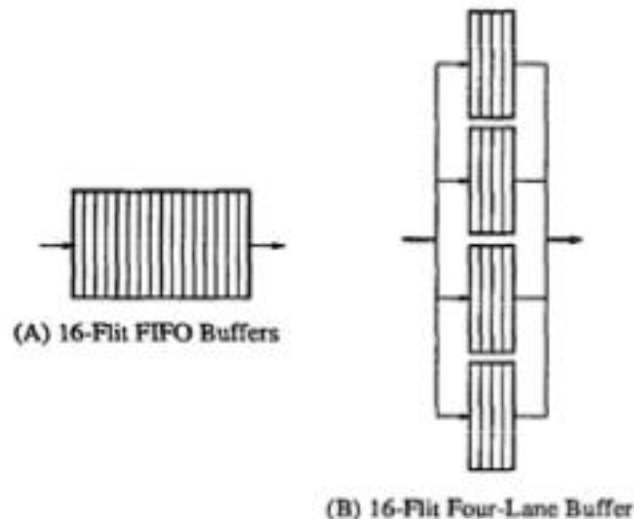
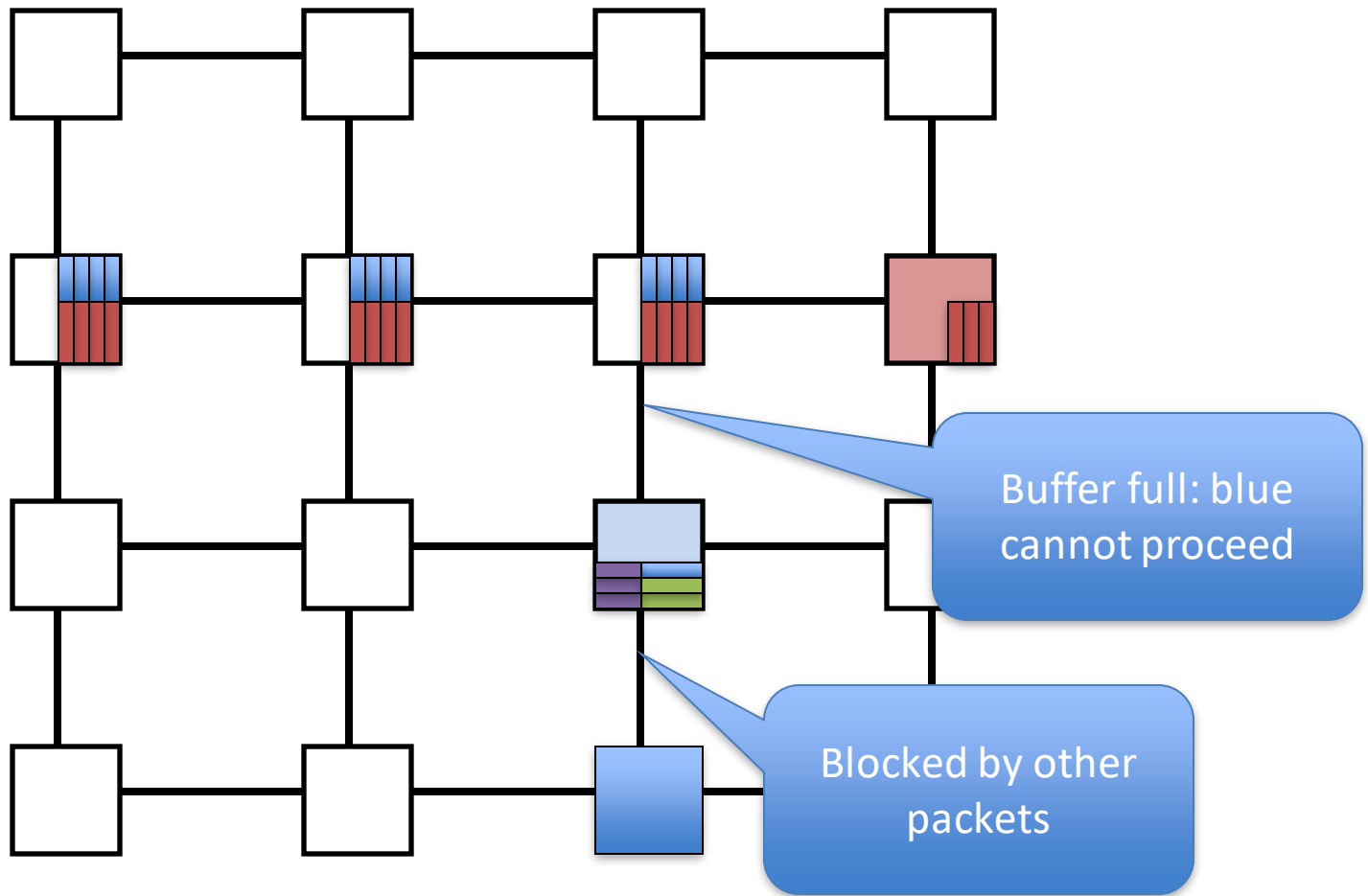
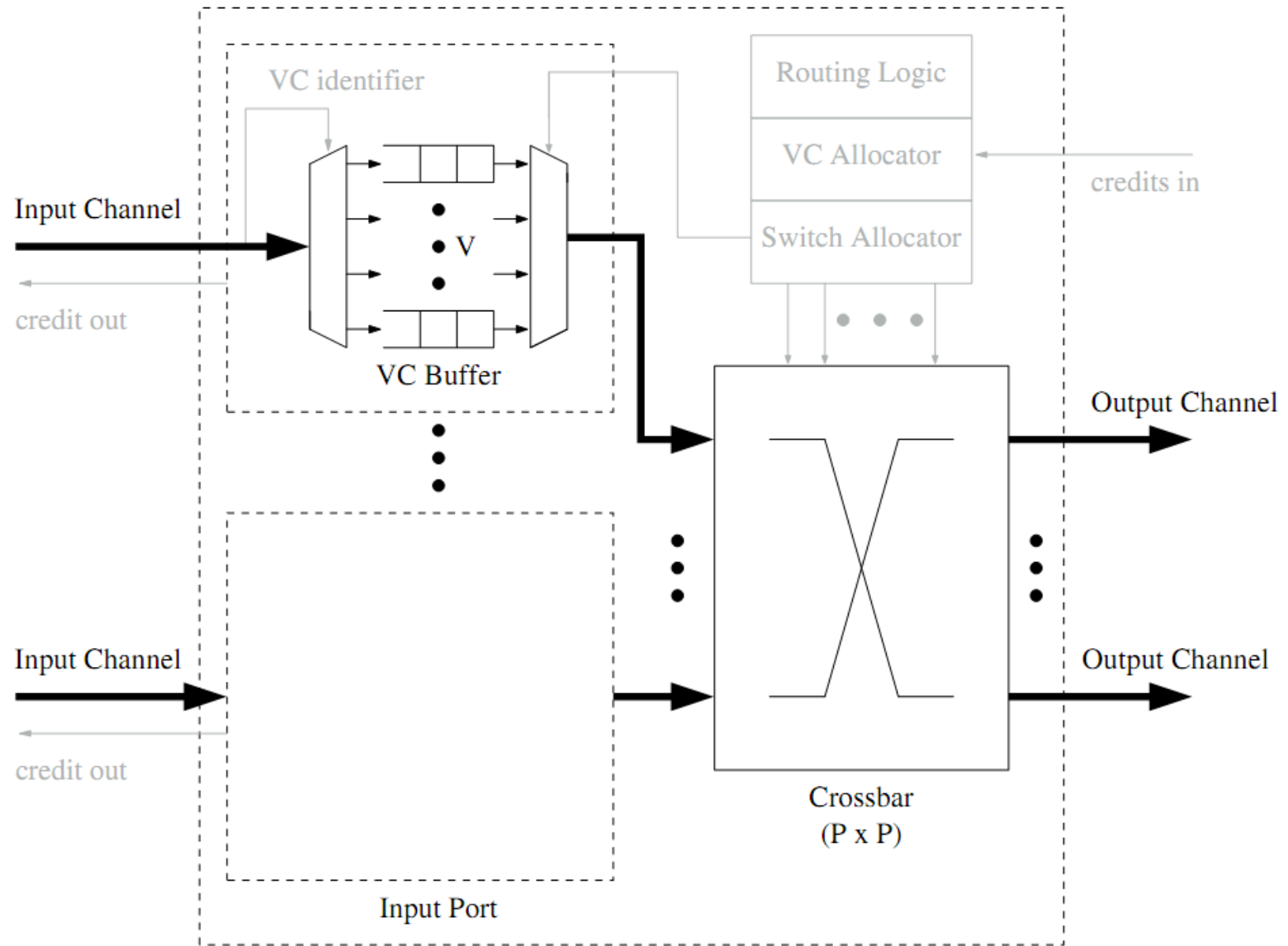


Figure 5: (A) Conventional nodes organize their buffers into FIFO queues restricting routing. (B) A network using virtual-channel flow control organizes its buffers into several independent lanes.

Virtual Channel Flow Control



A Modern Virtual Channel Based Router



Other Uses of Virtual Channels

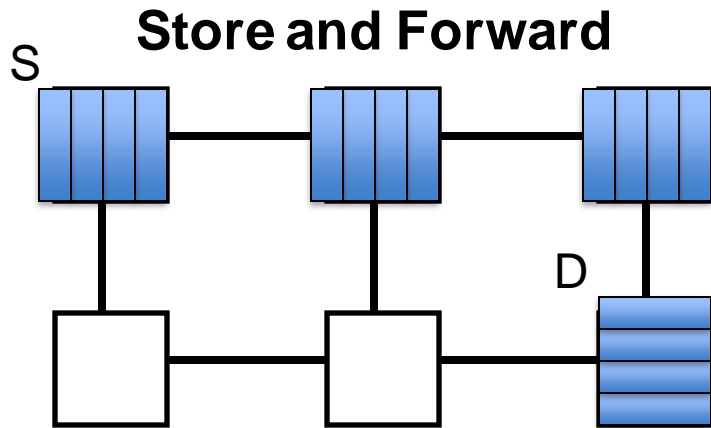
■ Deadlock avoidance

- ❑ Enforcing switching to a different set of virtual channels on some “turns” can break the cyclic dependency of resources
 - Enforce order on VCs
- ❑ **Escape VCs:** Have at least one VC that uses deadlock-free routing. Ensure each flit has fair access to that VC.
- ❑ **Protocol level deadlock:** Ensure address and data packets use different VCs → prevent cycles due to intermixing of different packet classes

■ Prioritization of traffic classes

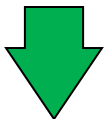
- ❑ Some virtual channels can have higher priority than others

Review: Flow Control



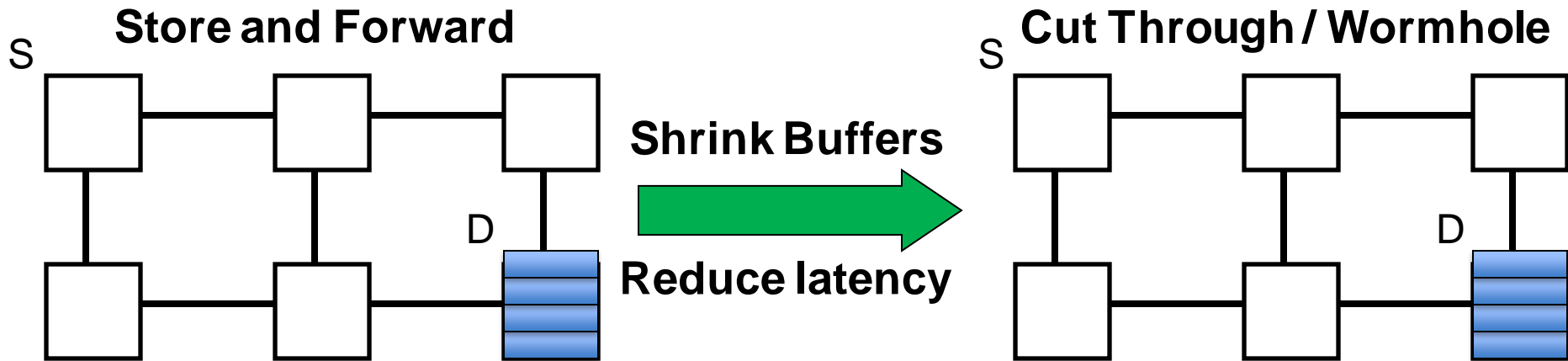
Any other
issues?

**Head-of-Line
Blocking**



**Use Virtual
Channels**

Review: Flow Control



Any other
issues?

**Head-of-Line
Blocking**

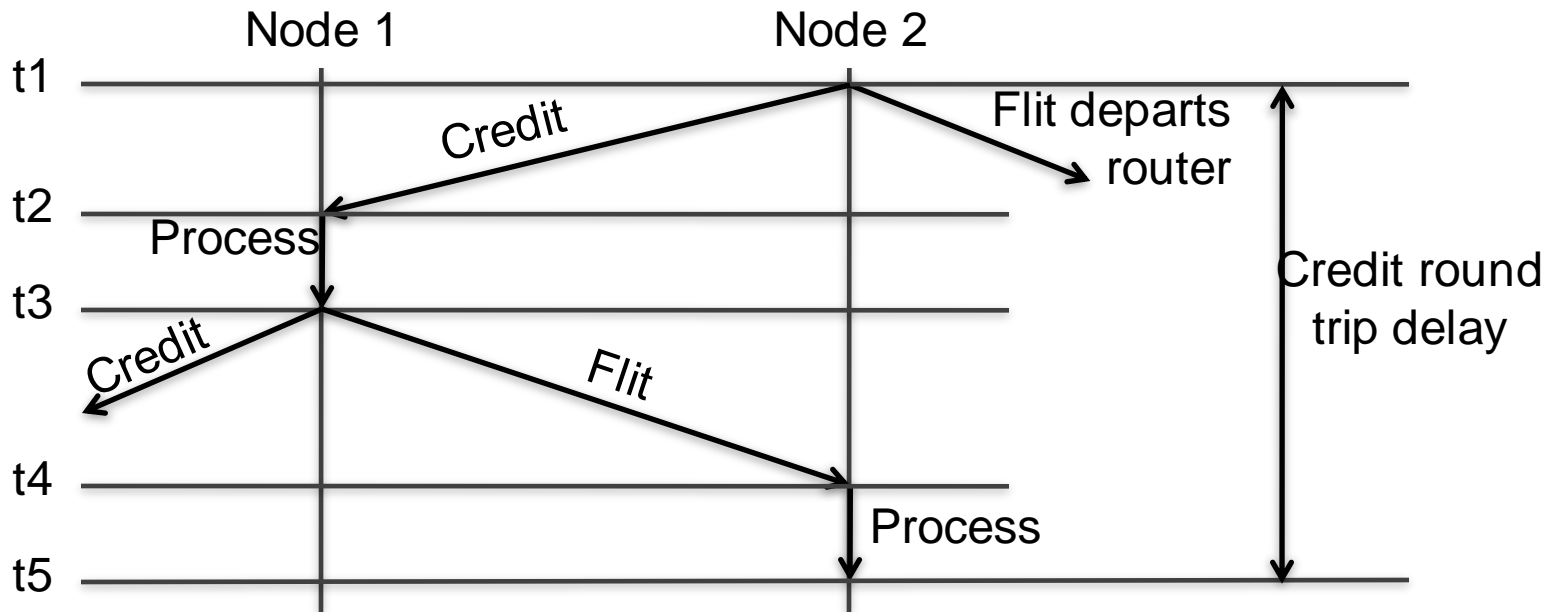
Use Virtual
Channels

This section addresses a problem with the Cut Through / Wormhole method. It shows a 3x3 grid of nodes. The top row of nodes has buffers containing packets of different colors (blue, red, and purple). The middle and bottom rows of nodes have empty buffers. A blue callout bubble points to the top-right node, stating 'Buffer full: blue cannot proceed'. Another blue callout bubble points to the bottom-middle node, stating 'Blocked by other packets'. A large green arrow points from the text 'Use Virtual Channels' to the diagram, indicating this as a solution to the 'Head-of-Line Blocking' problem.

Communicating Buffer Availability

- Credit-based flow control
 - Upstream knows how many buffers are downstream
 - Downstream passes back credits to upstream
 - Significant upstream signaling (esp. for small flits)
- On/Off (XON/XOFF) flow control
 - Downstream has on/off signal to upstream
- Ack/Nack flow control
 - Upstream optimistically sends downstream
 - Buffer cannot be deallocated until ACK/NACK received
 - Inefficiently utilizes buffer space

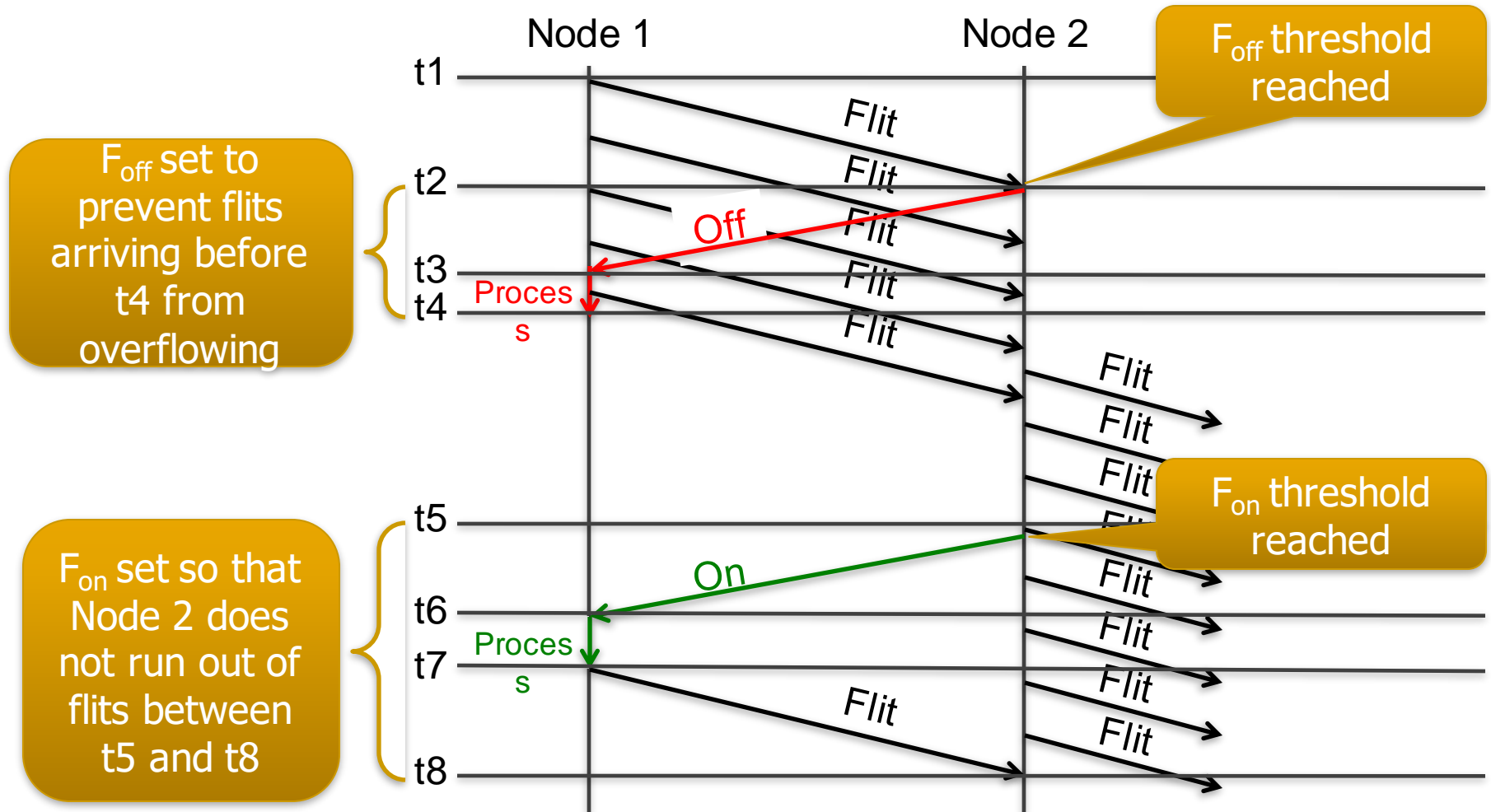
Credit-based Flow Control



- Round-trip credit delay:
 - Time between when buffer empties and when next flit can be processed from that buffer entry
- Significant throughput degradation if there are few buffers
- Important to size buffers to tolerate credit turn-around

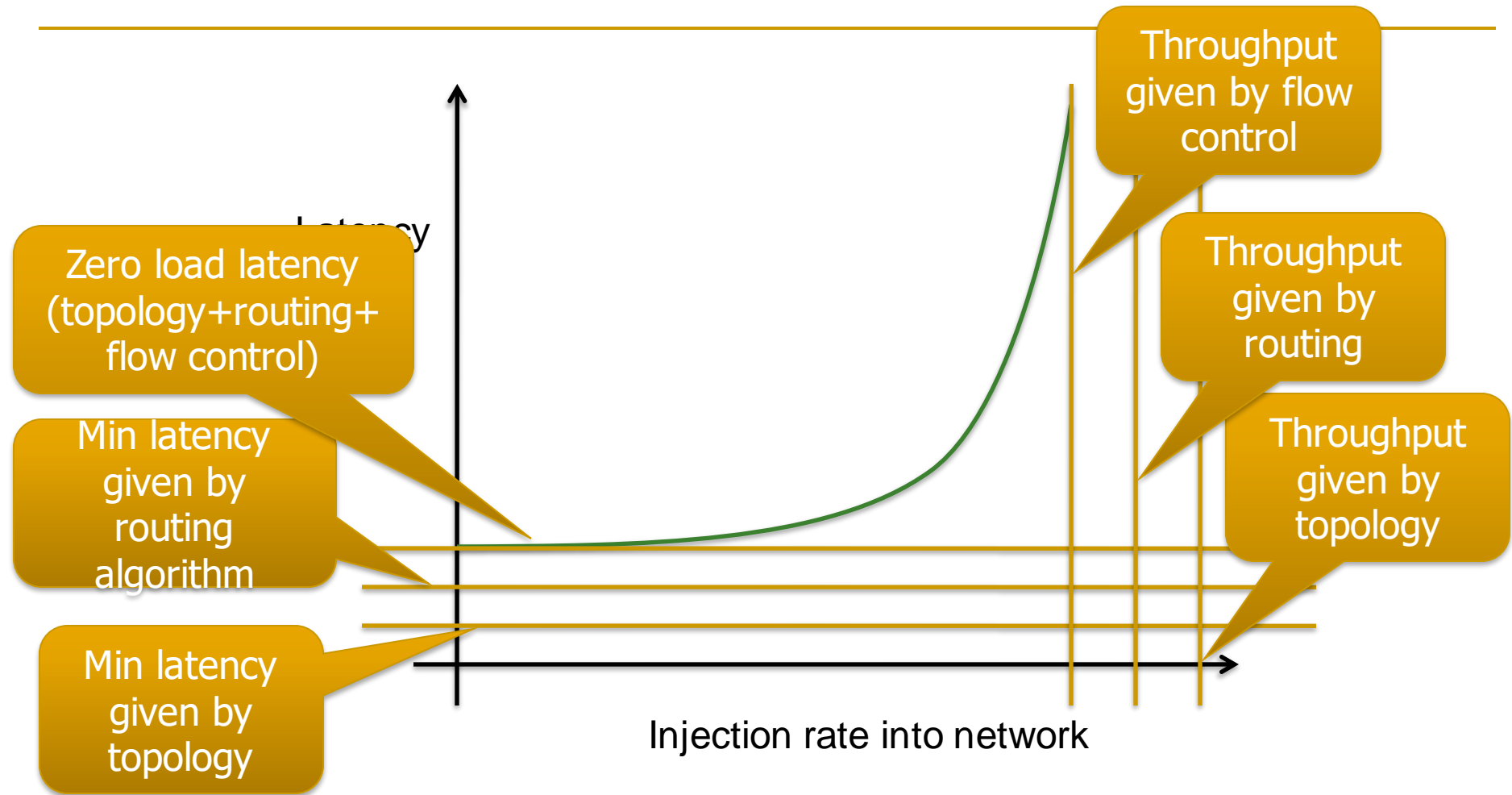
On/Off (XON/XOFF) Flow Control

- Downstream has on/off signal to upstream



Interconnection Network Performance

Interconnection Network Performance



Ideal Latency

- Ideal latency
 - Solely due to wire delay between source and destination

$$T_{ideal} = \frac{D}{v} + \frac{L}{b}$$

- D = Manhattan distance
 - The distance between two points measured along axes at right angles.
- v = propagation velocity
- L = packet size
- b = channel bandwidth

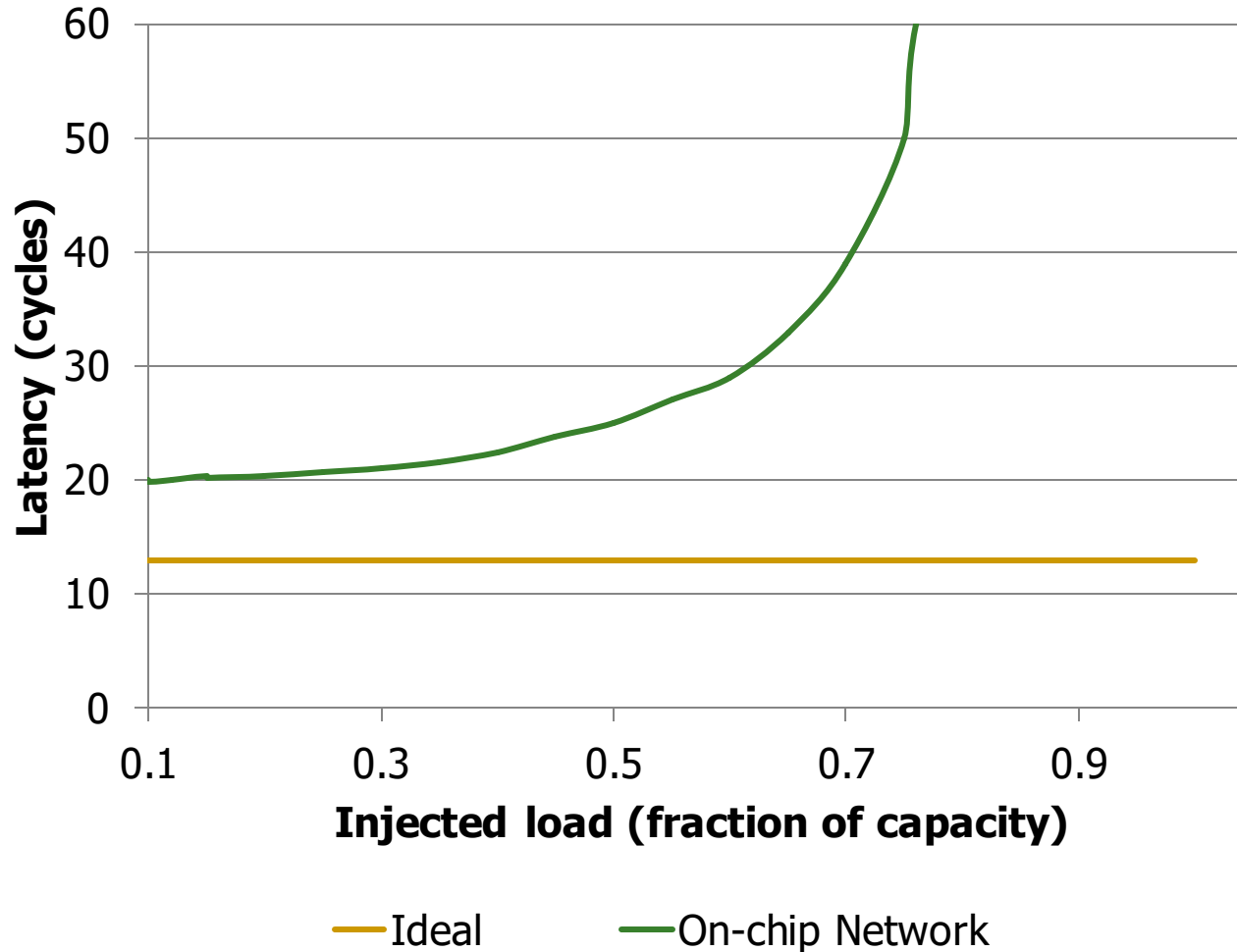
Actual Latency

- Dedicated wiring impractical
 - Long wires segmented with insertion of routers

$$T_{actual} = \frac{D}{v} + \frac{L}{b} + H \cdot T_{router} + T_c$$

- D = Manhattan distance
- v = propagation velocity
- L = packet size
- b = channel bandwidth
- H = hops
- T_{router} = router latency
- T_c = latency due to contention

Latency and Throughput Curve



Network Performance Metrics

- Packet latency
- Round trip latency
- Saturation throughput
- Application-level performance: system performance
 - Affected by interference among threads/applications

Computer Architecture

Lecture 22: Interconnects

Prof. Onur Mutlu

ETH Zürich

Fall 2020

27 December 2020