Interconnection Networks

Dr. Mohammad Sadrosadati

Prof. Onur Mutlu

ETH Zürich

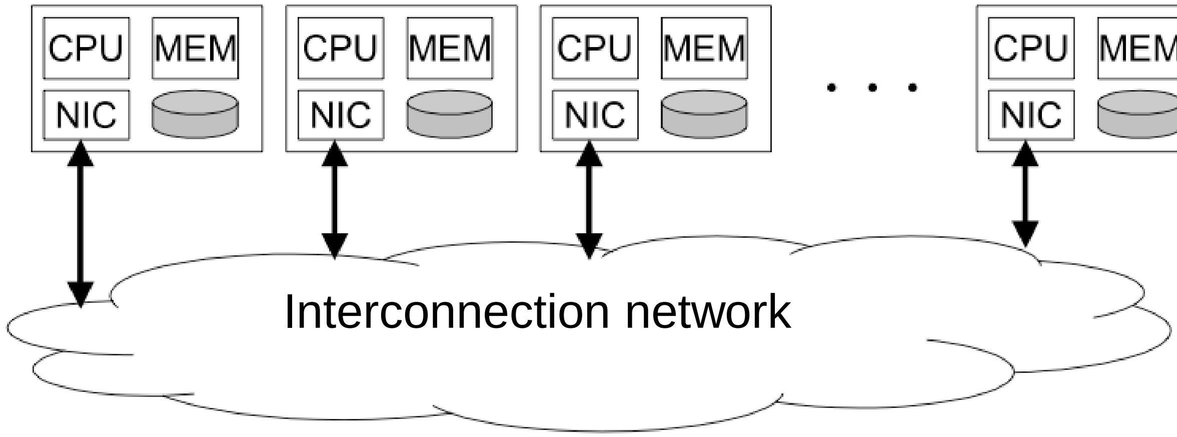
Fall 2023

7 December 2023

Where Is Interconnect Used?

- To connect & communicate between 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?

Many Parameters & Goals in a Network

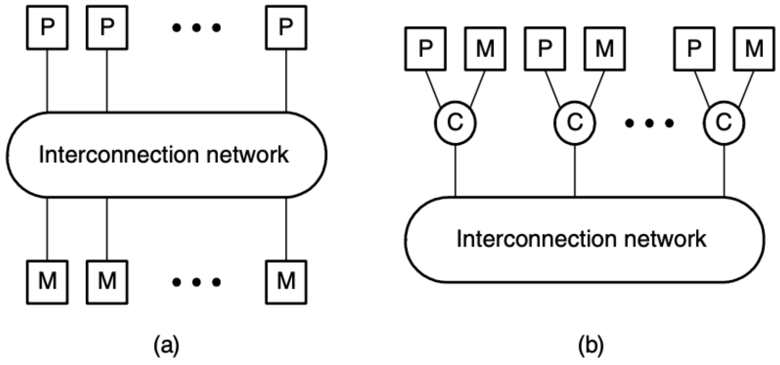


Figure 1.2 Use of an interconnection network to connect processor and memory. (a) *Dance-hall* architecture with separate processor (P) and memory (M) ports. (b) Integrated-node architecture with combined processor and memory ports and local access to one memory bank.

Table 1.1 Parameters of processor-memory interconnection networks.

Parameter	*Value*
Processor ports	1–2,048
Memory ports	0–4,096
Peak bandwidth	8 Gbytes/s
Average bandwidth	400 Mbytes/s
Message latency	100 ns
Message size	64 or 576 bits
Traffic patterns	arbitrary
Quality of service	none
Reliability	no message loss
Availability	0.999 to 0.99999

Dally & Towles, “Principles and Practices of Interconnection Networks,” 2004.

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 routers & links?
    - Entire packets, parts of packets, etc?
  - 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

Interconnection Network Topology

Properties of a Topology/Network

- Regular or Irregular
  - Regular if topology is a 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

Topology

- Bus (simplest)
- Point-to-point connections (most costly)
- Crossbar
- Ring
- Tree
- Omega
- Hypercube
- Mesh
- Torus
- Butterfly
- ...

Metrics to Evaluate Interconnect Topology

- Cost
- Latency (in hops, in nanoseconds)
- Contention
- Energy
- Bandwidth
- Overall system performance

Bus

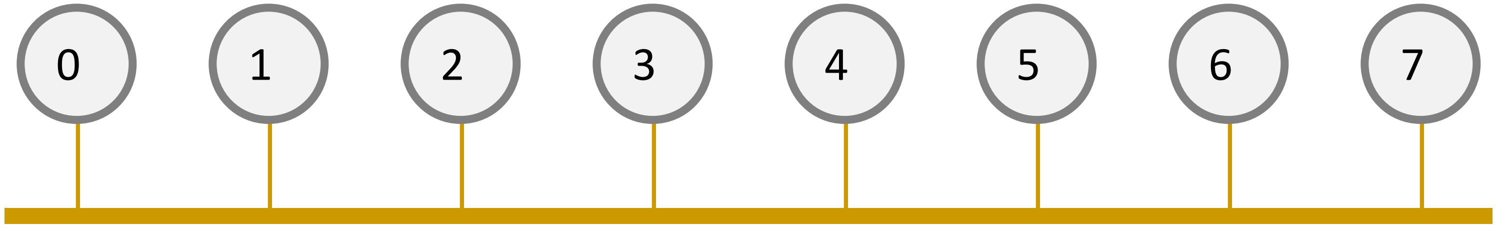
All nodes connected to a single link

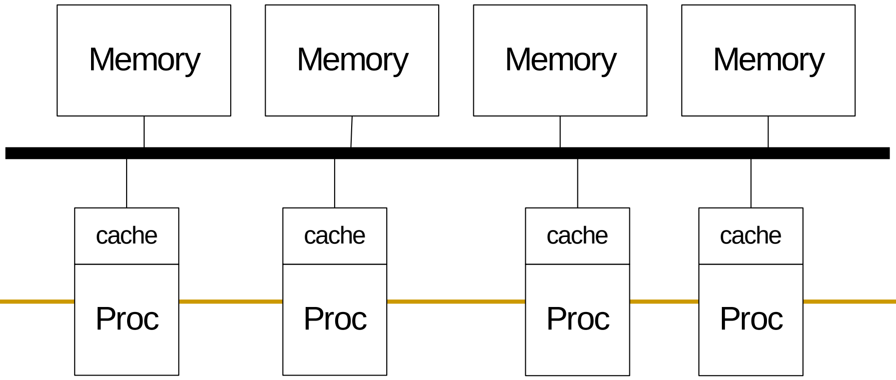
+ Simple + Cost effective for a small number of nodes

+ Easy to implement coherence (snooping and serialization)

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

- High contention → fast saturation



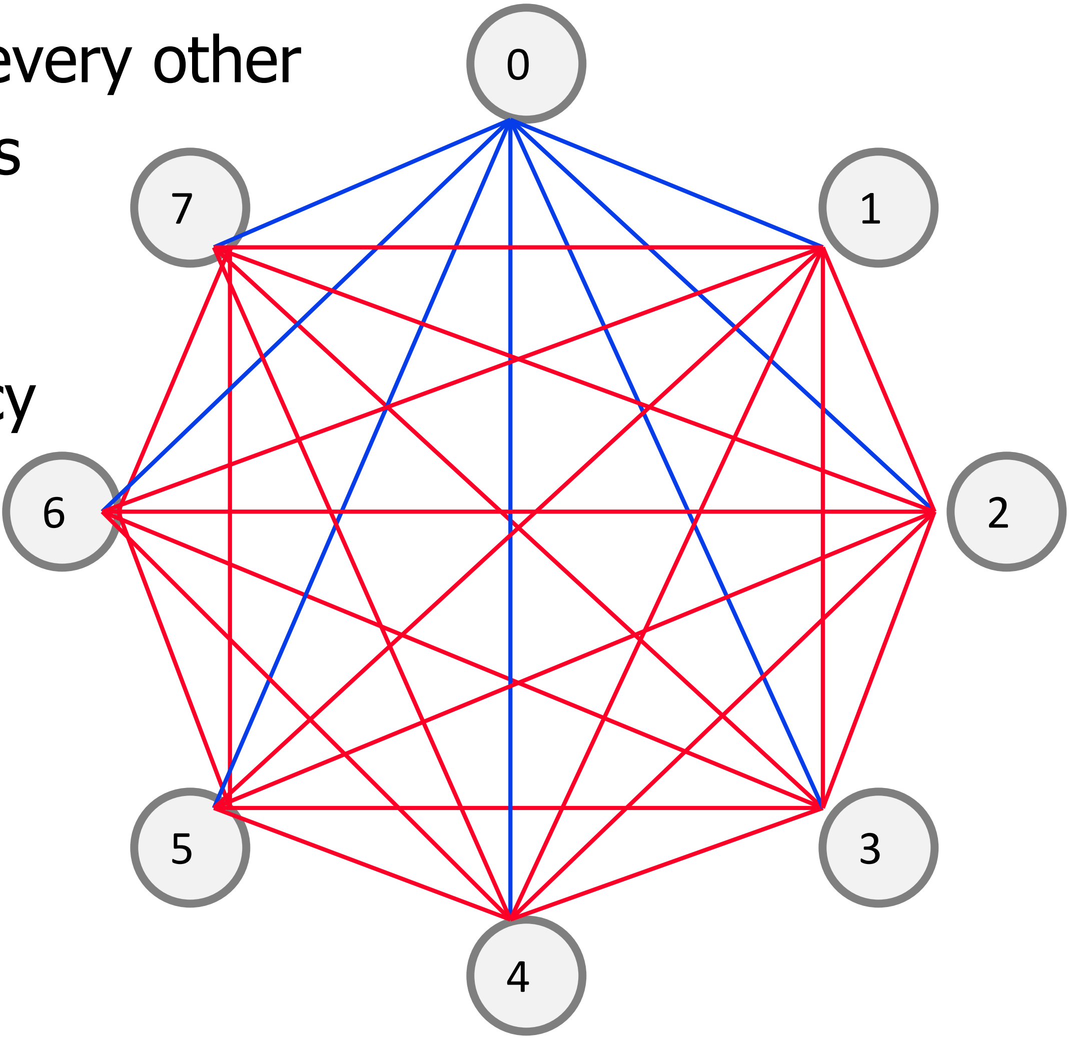


Point-to-Point

Every node connected to every other
with direct/isolated links

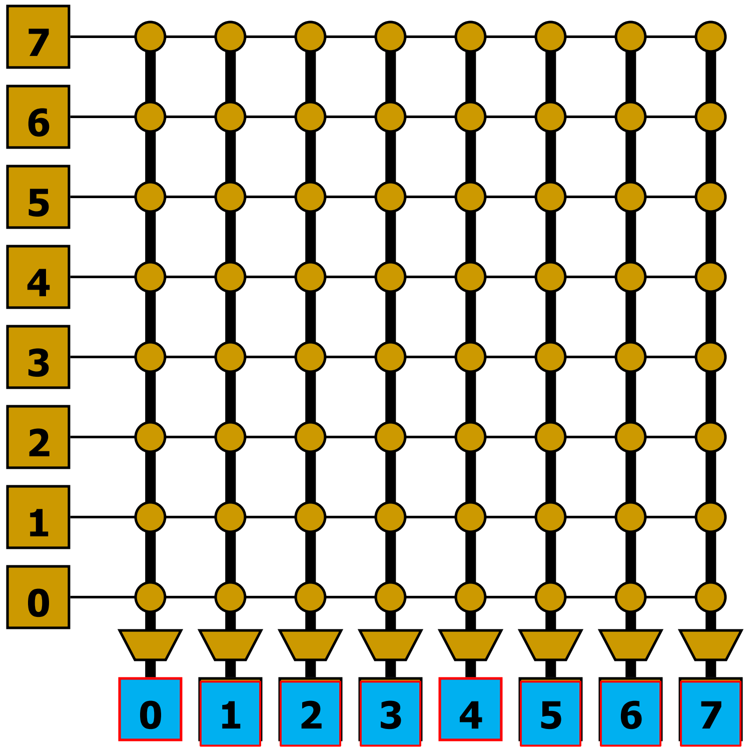
+ Lowest contention
+ Potentially lowest latency
+ Ideal, if cost is no issue

-- 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

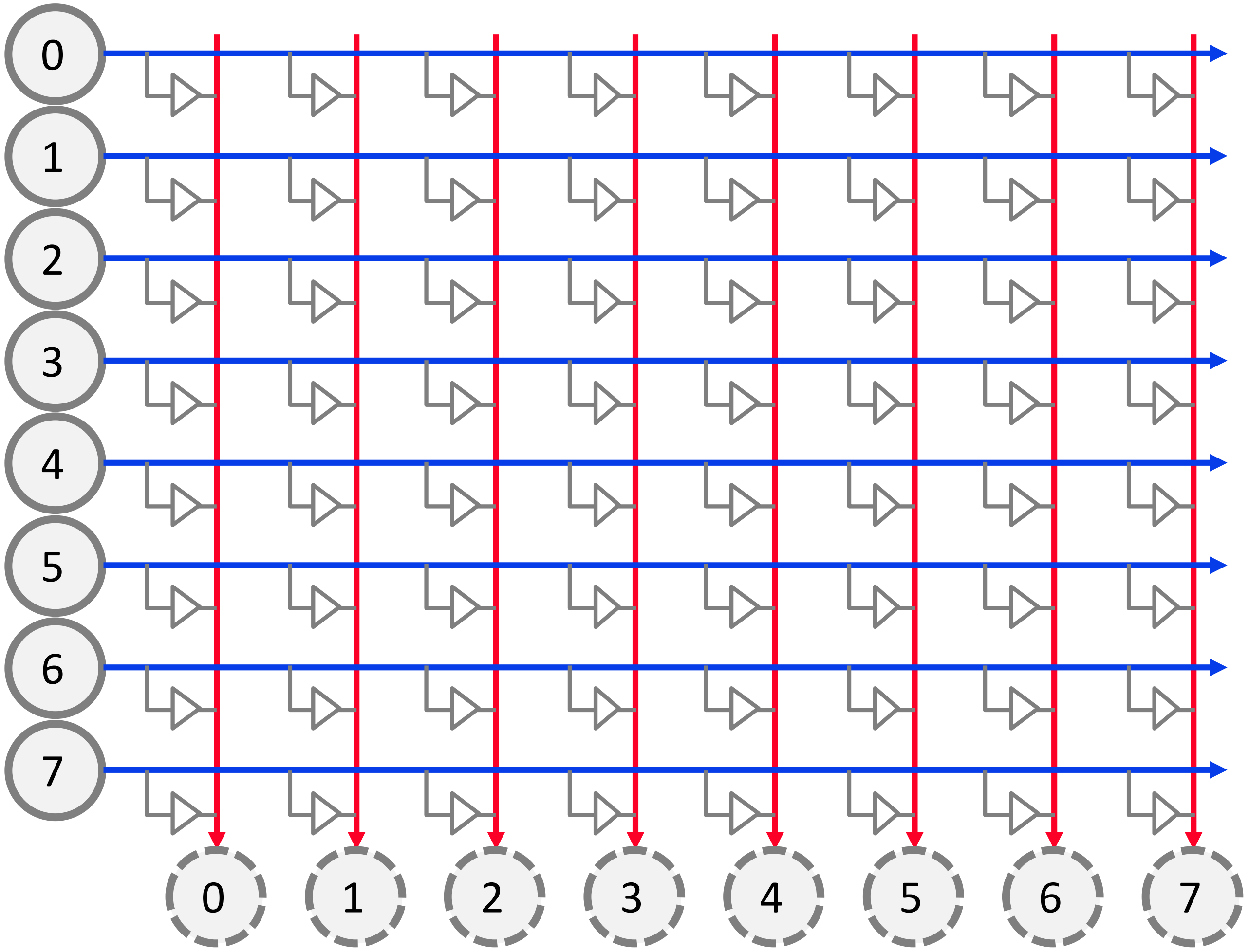
- Difficult to arbitrate as N increases

Used in core-to-cache-bank networks in

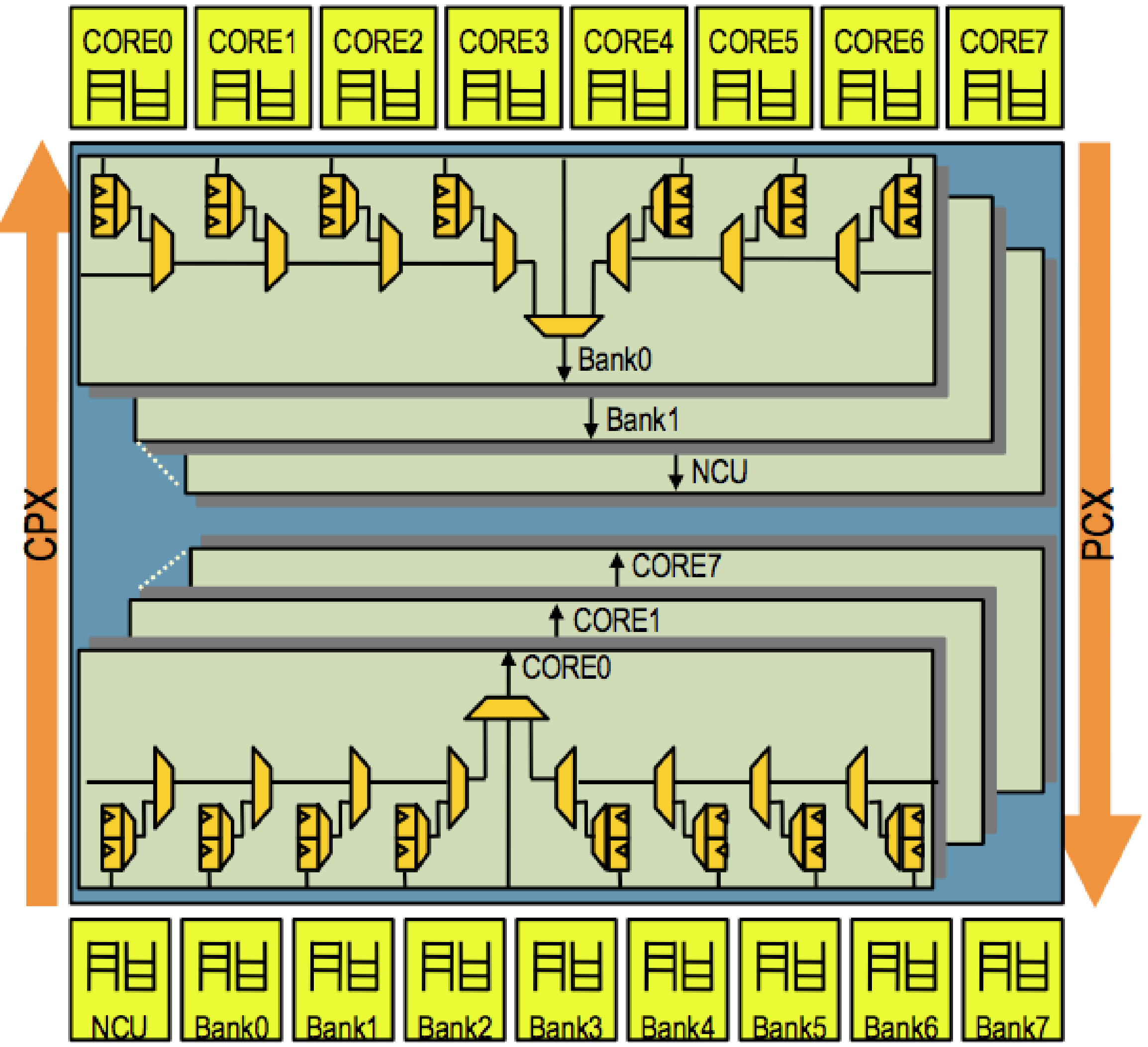
- IBM POWER5

- Sun Niagara I/II

Another Crossbar Design

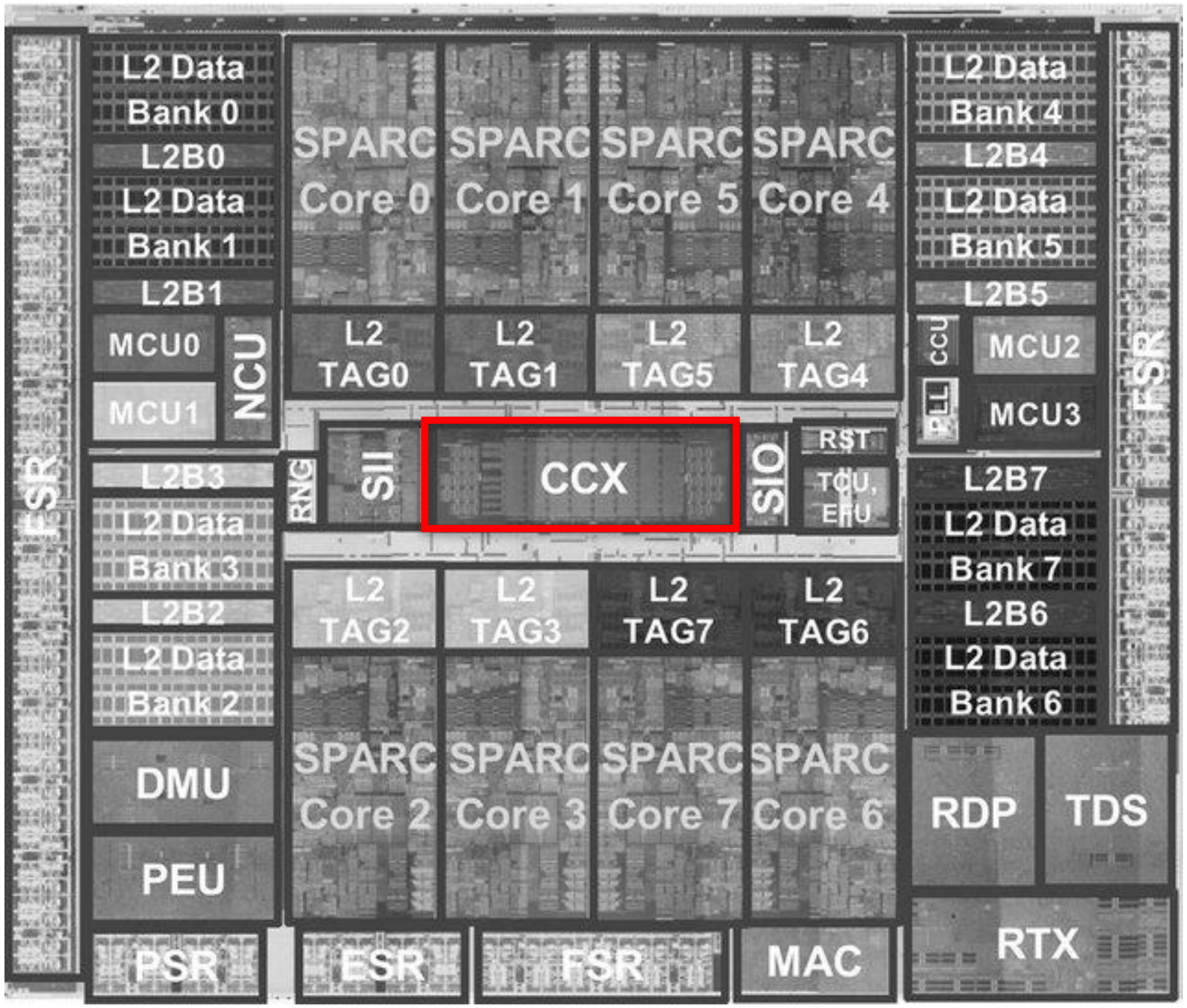


Sun UltraSPARC T2 Core-to-Cache Crossbar



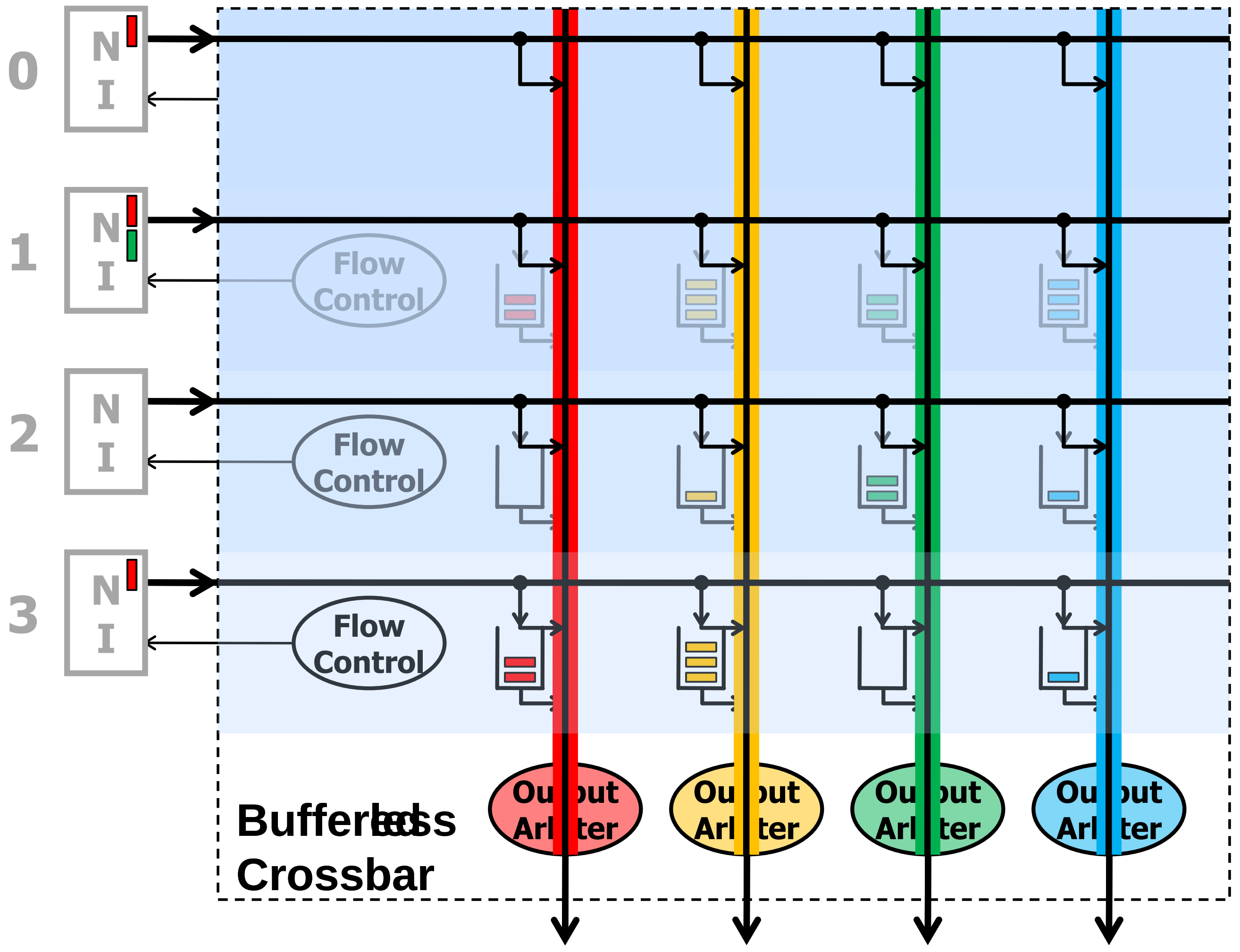
- High bandwidth interface between 8 cores and 8 L2 banks & NCU
- 4-stage pipeline: req, arbitration, selection, transmission
- 2-deep queue for each requestor to hold data transfer request

Sun UltraSPARC T2 Core-to-Cache Crossbar



Nawathe+, "Implementation of an 8-Core, 64-Thread, Power-Efficient SPARC Server on a Chip," JSSC 2008.

Bufferless and Buffered Crossbars



+ Simpler arbitration/ scheduling

+ Efficient support for variable-size packets

- Requires N^2 buffers

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

  + no buffering

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

  -- path cannot be used by multiple flows concurrently
- Packet switching performs routing 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 handling contention (e.g., via buffering)

  + no setup, bring down time

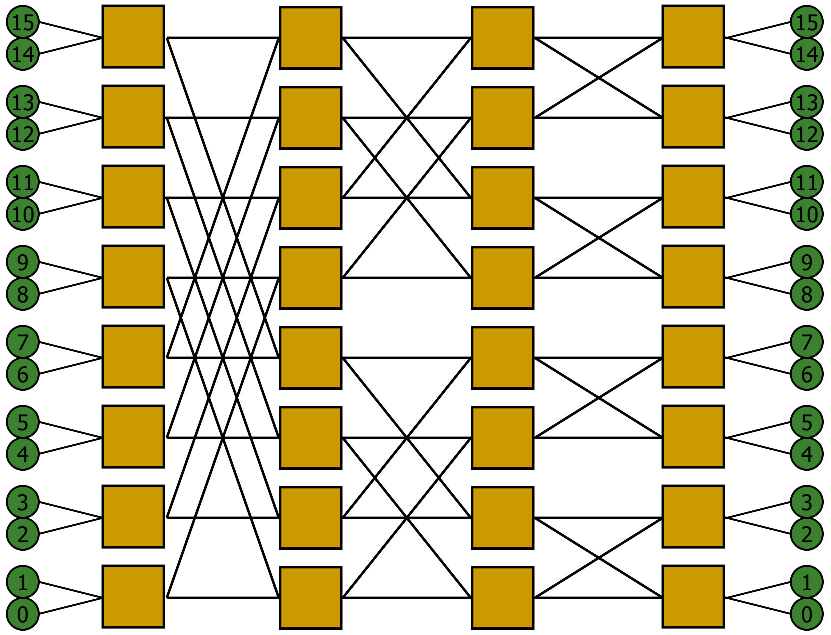
  + more flexible, does not underutilize links

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

Butterfly



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)

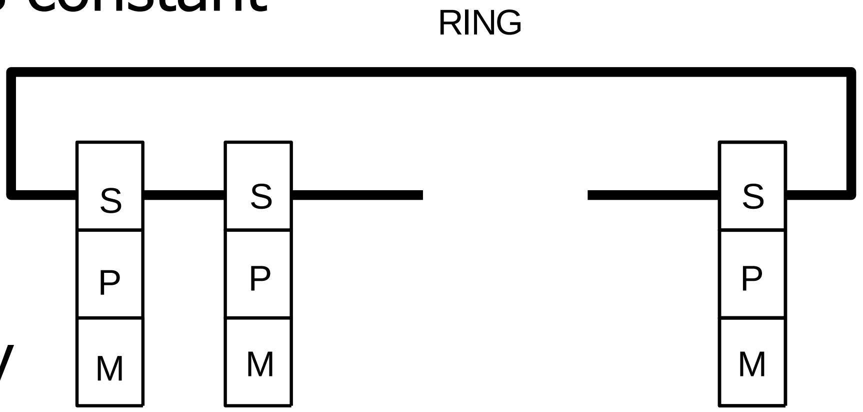
- Not easy to scale

  - Bisection bandwidth remains constant

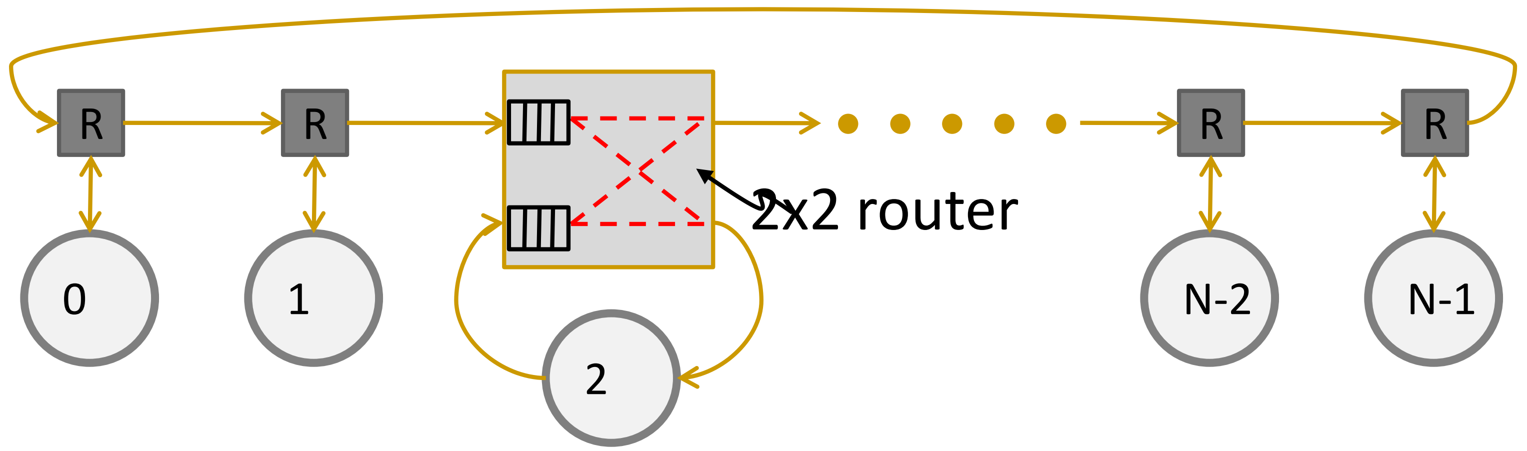
Used in Intel Haswell,

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

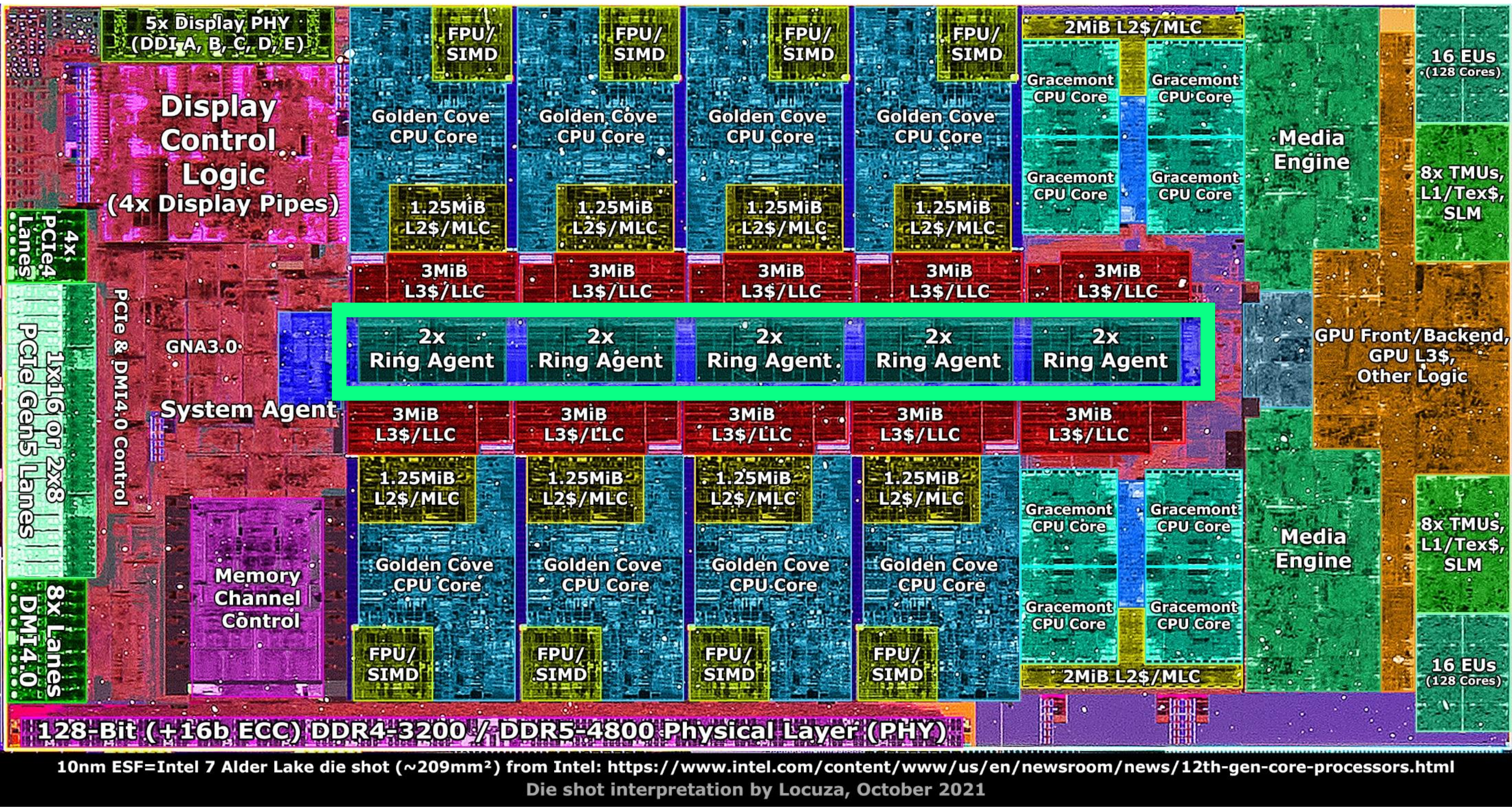
Bidirectional Rings

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)

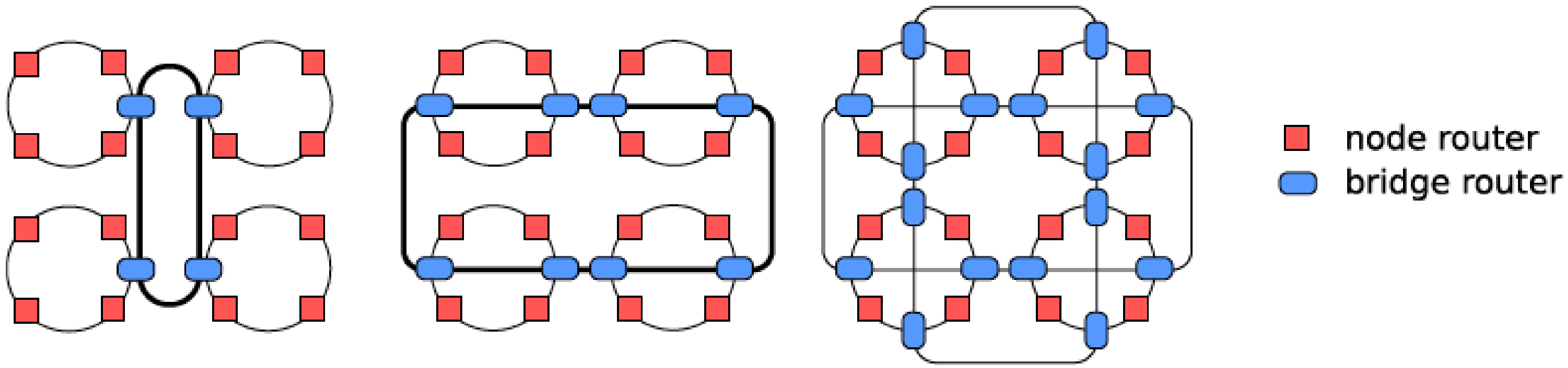
Rings in Existing Systems


5x Display PHY (DDI A, B, C, D, E)
Display Control Logic (4x Display Pipes)
4x PCIe4 Lanes
1x16 or 2x8 PCIe Gen5 Lanes
PCIe & DMI4.0 Control
GNA3.0
System Agent
Memory Channel Control
8x Lanes DMI4.0
FPU/ SIMD
Golden Cove CPU Core
1.25MiB L2$/MLC
3MiB L3$/LLC
2x Ring Agent
2MiB L2$/MLC
Gracemont CPU Core
Media Engine
16 EUs (128 Cores)
8x TMUs, L1/Tex$, SLM
GPU Front/Backend, GPU L3$, Other Logic
128-Bit (+16b ECC) DDR4-3200 / DDR5-4800 Physical Layer (PHY)
10nm ESF=Intel 7 Alder Lake die shot (~209mm²) from Intel: https://www.intel.com/content/www/us/en/newsroom/news/12th-gen-core-processors.html
Die shot interpretation by Locuza, October 2021

Intel Alder Lake, 2021

Source: https://twitter.com/Locuza_/status/1454152714930331652

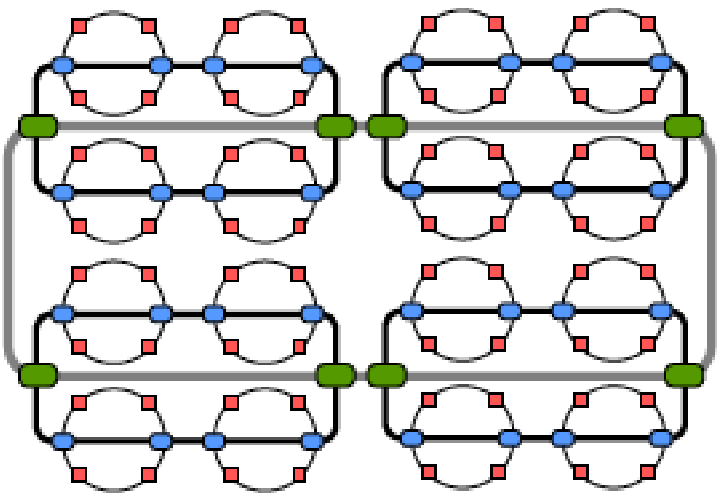
Hierarchical Rings



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

+ More scalable
+ Lower latency

- More complex

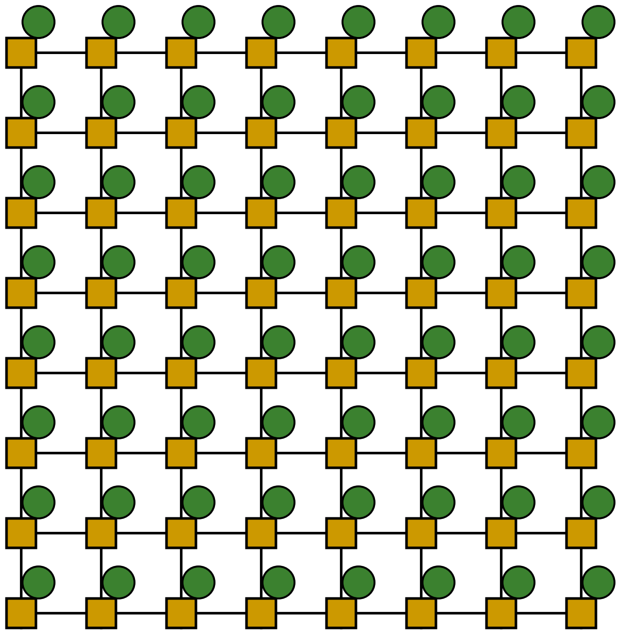


(b) Three-level hierarchy (8x8).

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

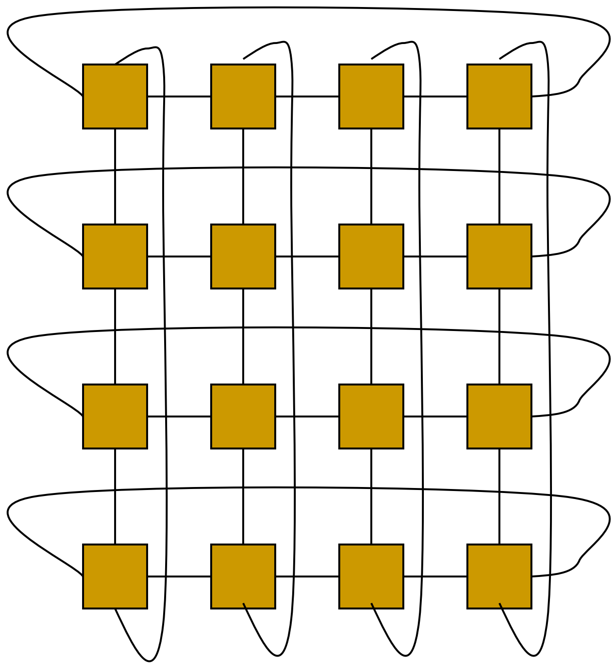
- Mesh is not symmetric on edges: performance very sensitive to placement of task on edge vs. middle
- Torus avoids this problem

+ Higher path diversity (and bisection bandwidth) than mesh

- Higher cost

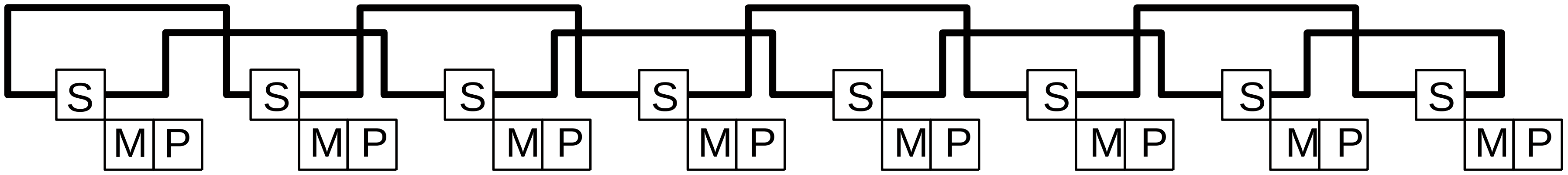
- Harder to lay out on-chip

  - Unequal link lengths



Torus, continued

- Weave nodes to make inter-node latencies ~constant



Trees

Planar, hierarchical topology

Latency: O(logN)

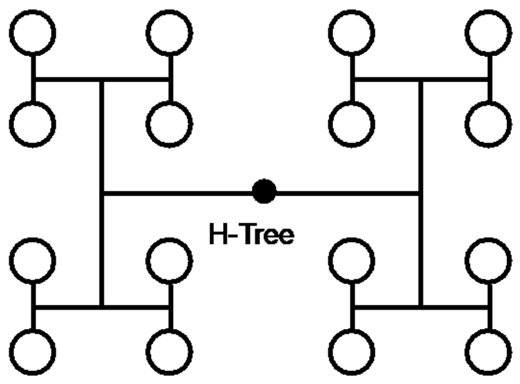
Good for local traffic

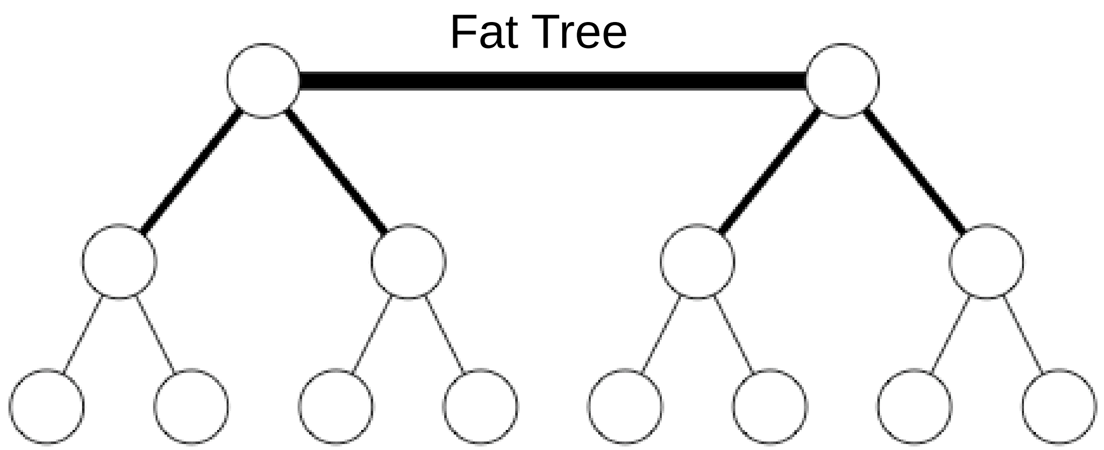
+ Cheap: O(N) cost

+ Easy to Layout

- Root can become a bottleneck

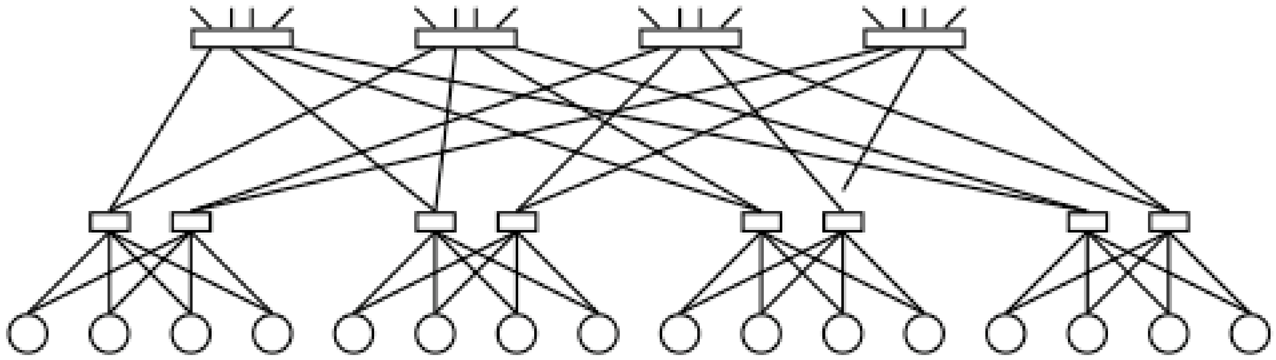
  Fat trees avoid this problem (CM-5)





CM-5 Fat Tree

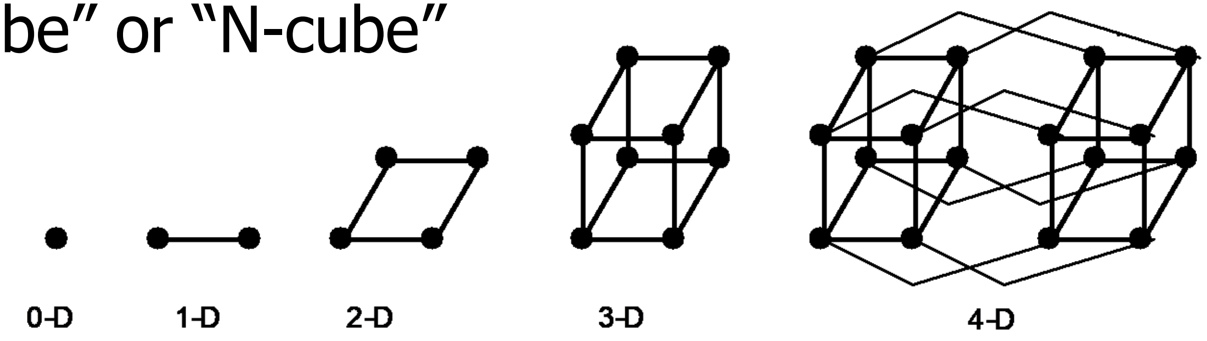
- Fat tree based on 4x2 switches, packet switched
- Randomized routing on the way up
- Combining, multicast, reduction operators supported in hardware



CM-5 Thinned Fat Tree

Hypercube

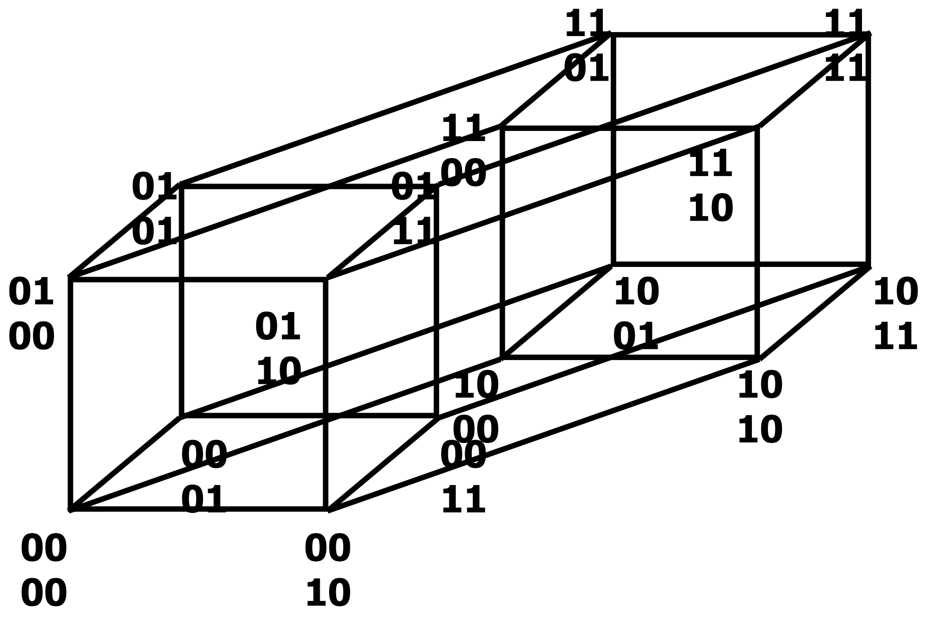
- "N-dimensional cube" or "N-cube"



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

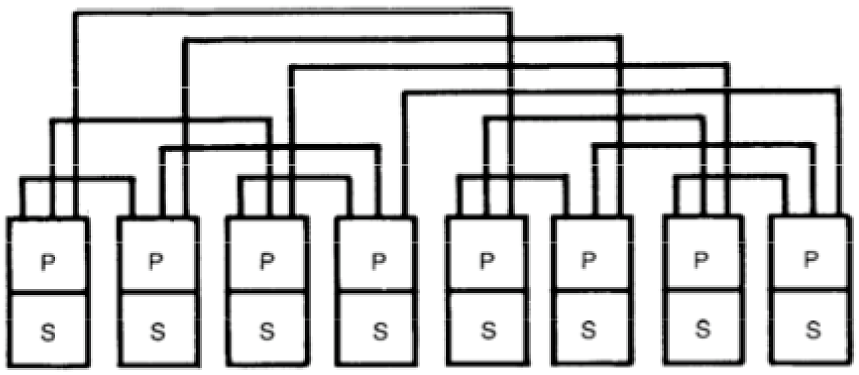
+ Low latency

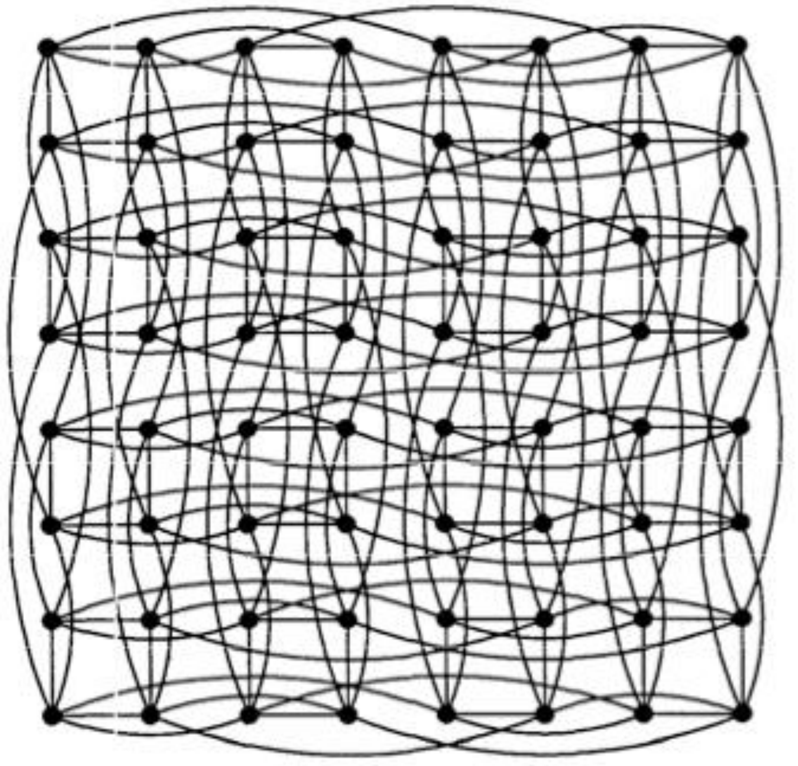
- Hard to lay out in 2D/3D



Caltech Cosmic Cube

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

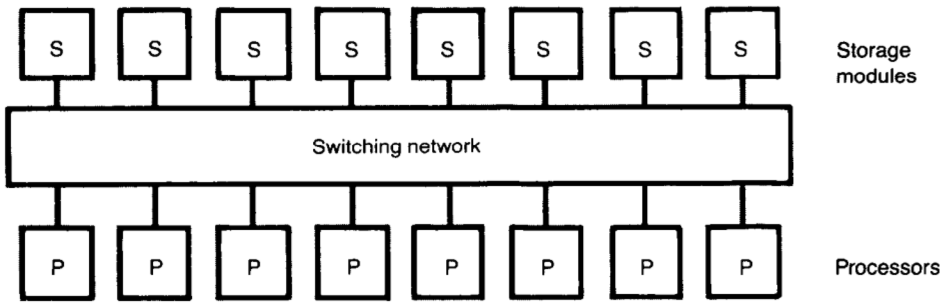




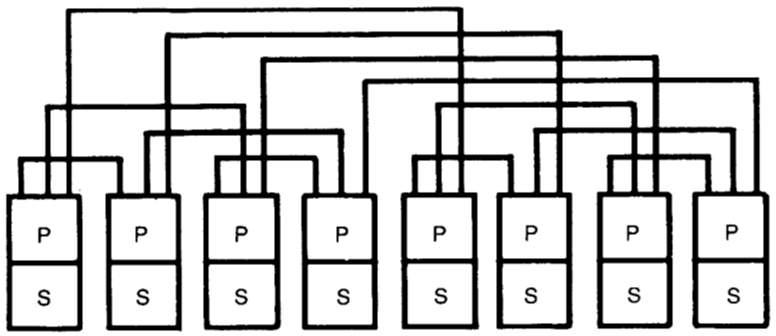
A hypercube connects $N = 2^n$ small computers, called nodes, through point-to-point communication channels in the Cosmic Cube. Shown here is a two-dimensional projection of a six-dimensional hypercube, or binary 6-cube, which corresponds to a 64-node machine.

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

Caltech Cosmic Cube Motivation



(a) Most multiprocessors are structured with a switching network, either a crossbar connection of buses or a multi-stage routing network, between the processors and storage. The switching network introduces a latency in the communication between processors and storage, and does not scale well to large sizes. Communication between processes running concurrently in different processors occurs through shared variables and common access to one large address space.



(b) Message-passing multicomputer systems retain a physically close and fast connection between processors and their associated storage. The concurrent computers (nodes) can send messages through a network of communication channels. The network shown here is a three-dimensional cube, which is a small version of the communication plan used in six dimensions in the 64-node Cosmic Cube.

NOTE: Actual machines need not follow one model or the other absolutely: Various hybrids are possible.

FIGURE 2. A Comparison of Shared-Storage Multiprocessors and Message-Passing Machines

Seitz, "The Cosmic Cube," CACM 1985.

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

+ Simple

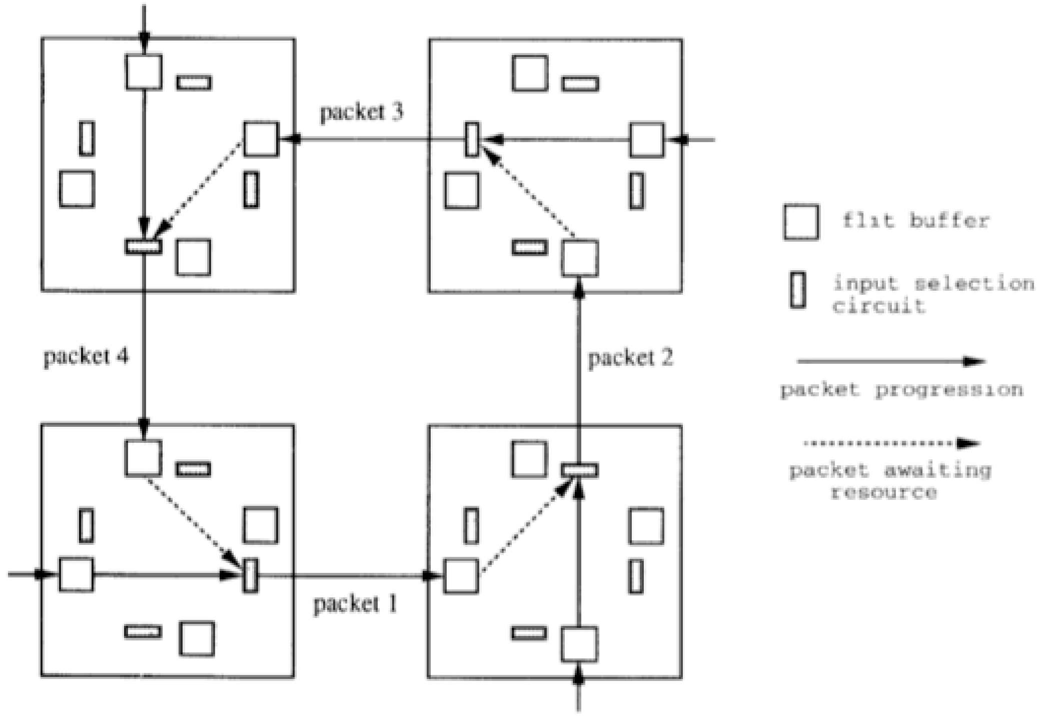
+ Deadlock freedom (no cycles in resource allocation)

- 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

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)

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

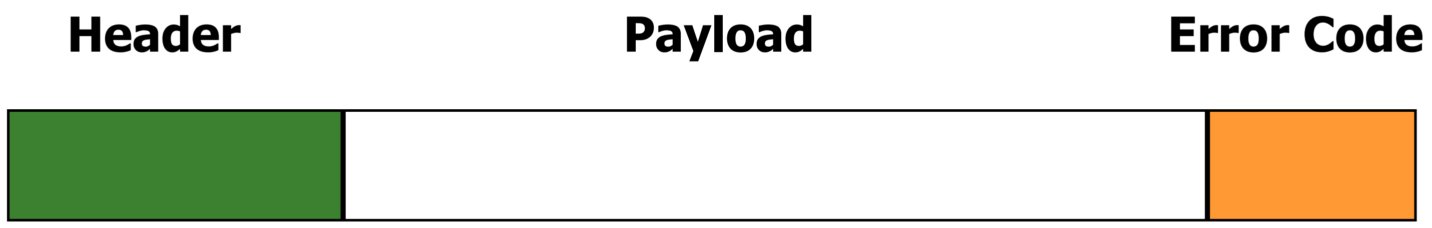
- Need to change the routing table to disable faulty routes

  - Assuming the faulty link/router is detected

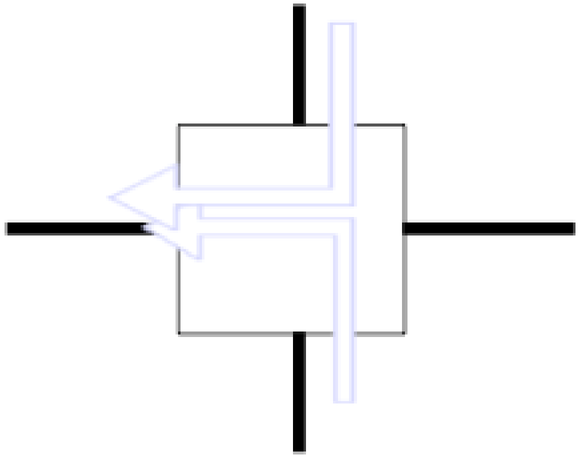
Buffering and Flow Control

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 large

- 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)

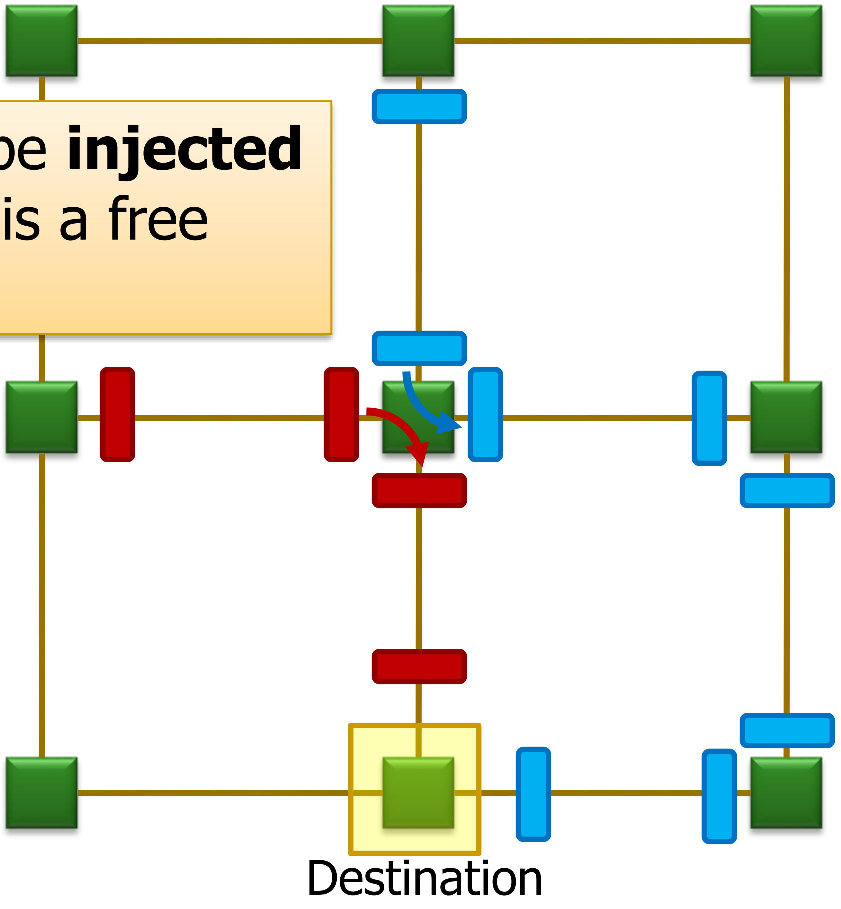
+ Can handle arbitrary message sizes

- 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]

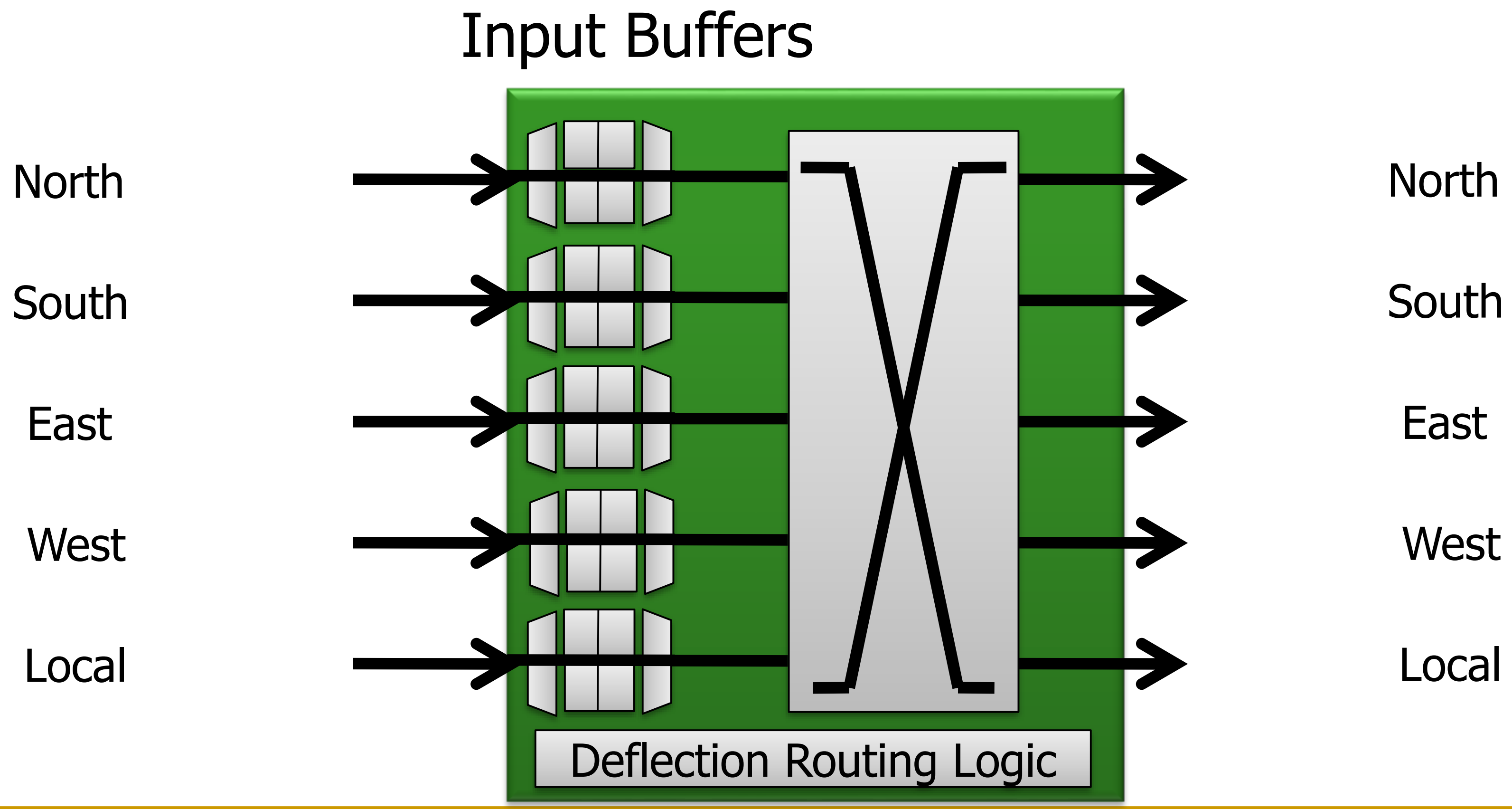


New traffic can be **injected** whenever there is a free output link.

[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**



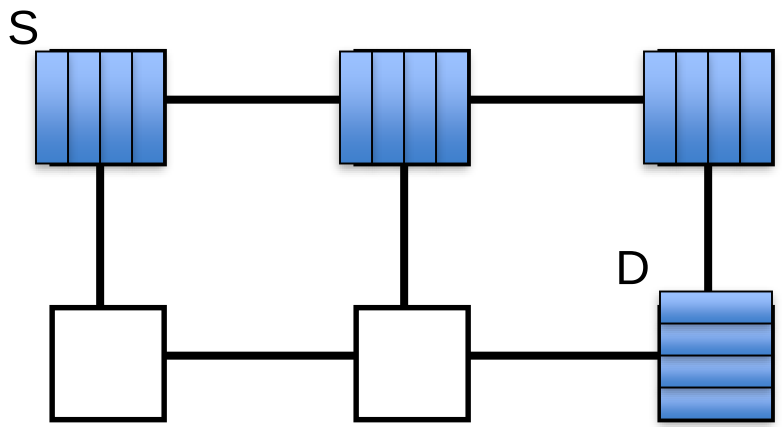
Moscibroda and Mutlu, "A Case for Bufferless Routing in On-Chip Networks," ISCA 2009.

Issues In Bufferless Deflection Routing

- Livelock
- Resulting Router Complexity
- Performance & Congestion at High Loads

Store and Forward Flow Control

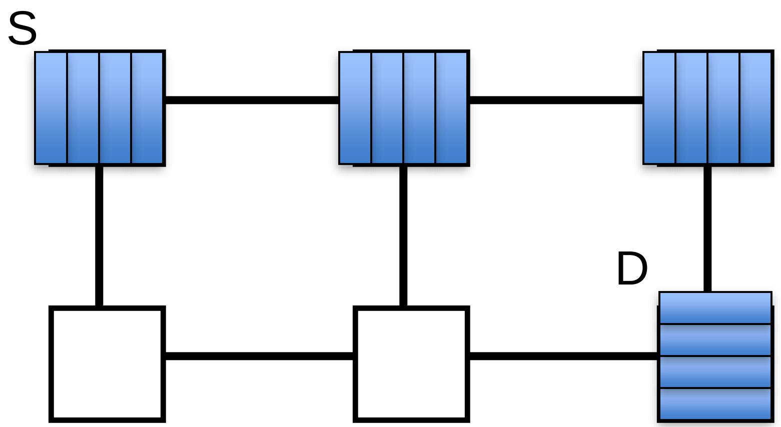
- Packet-based flow control
- Store and Forward
  - Packet copied entirely into network router before moving to the next node
  - Flow control unit is the entire packet
- Leads to high per-packet latency
- Requires buffering for entire packet in each node



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
  - Large reduction in latency
- Still allocates buffers and channel bandwidth for full packets

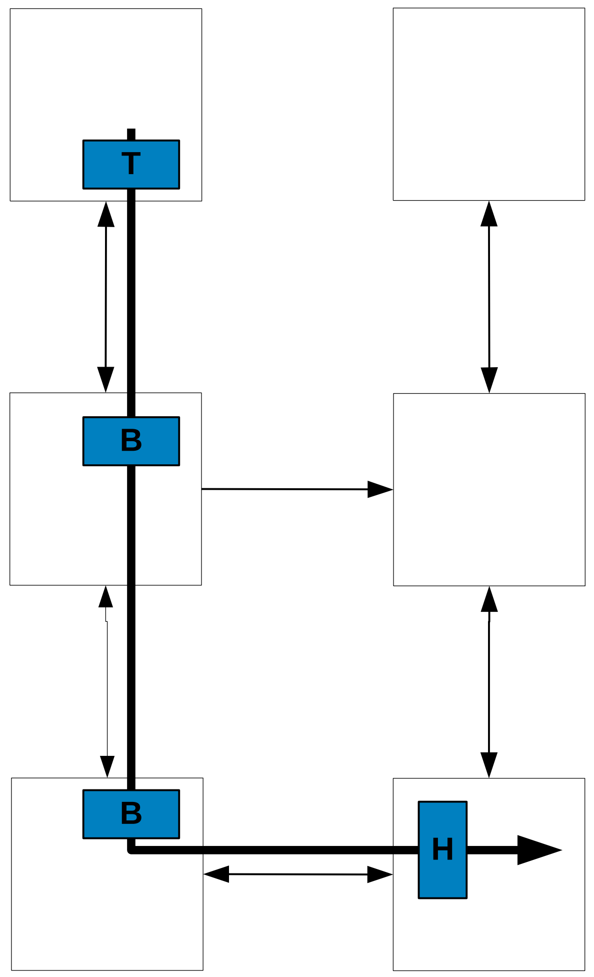


- What if packets are large?

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

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?
  - Follow the head (need state in router)
- Latency almost independent of distance for long messages

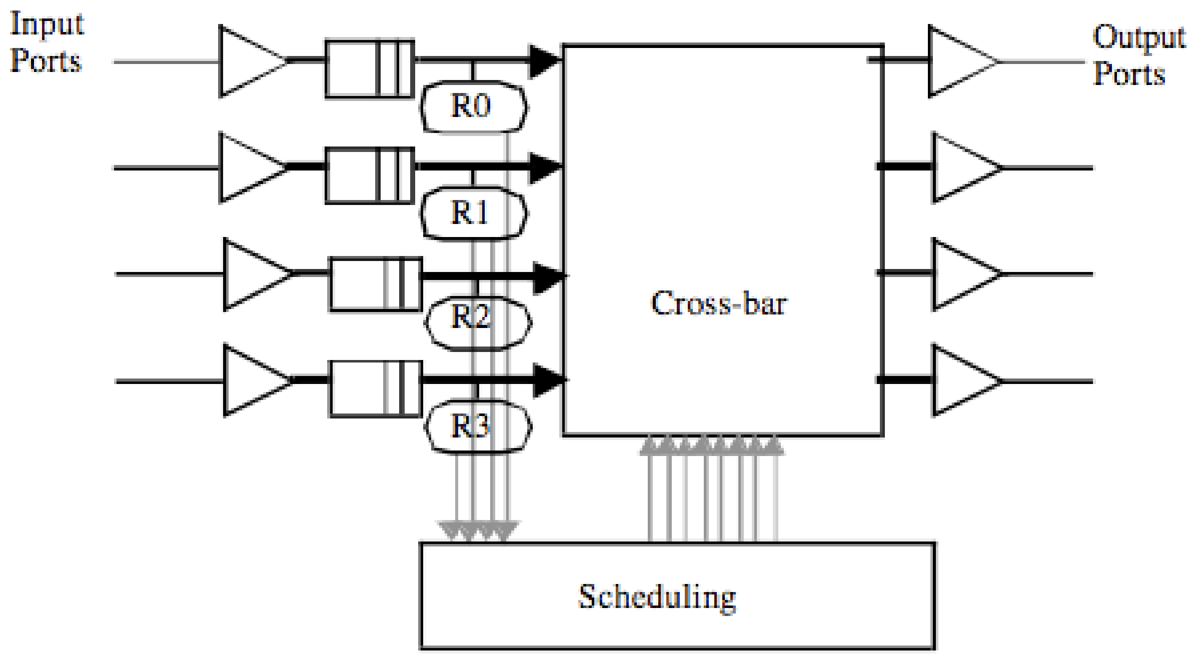
Wormhole Flow Control

- Advantages over "store and forward" flow control
  + Lower latency
  + More efficient buffer utilization

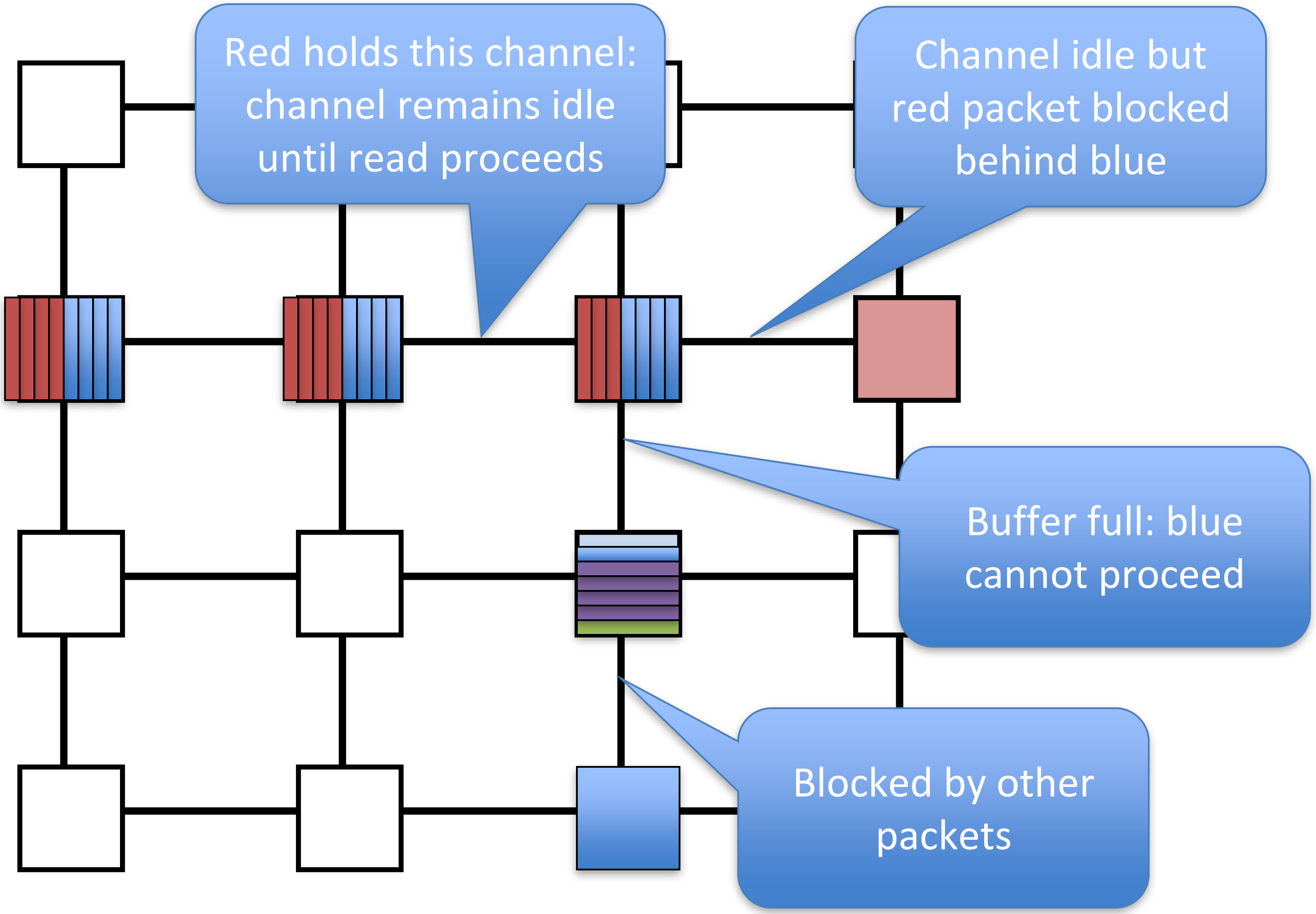
- Limitations
  - Suffers from **head of line blocking**
    - If head flit cannot move due to contention, another worm cannot proceed even though links may be idle

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

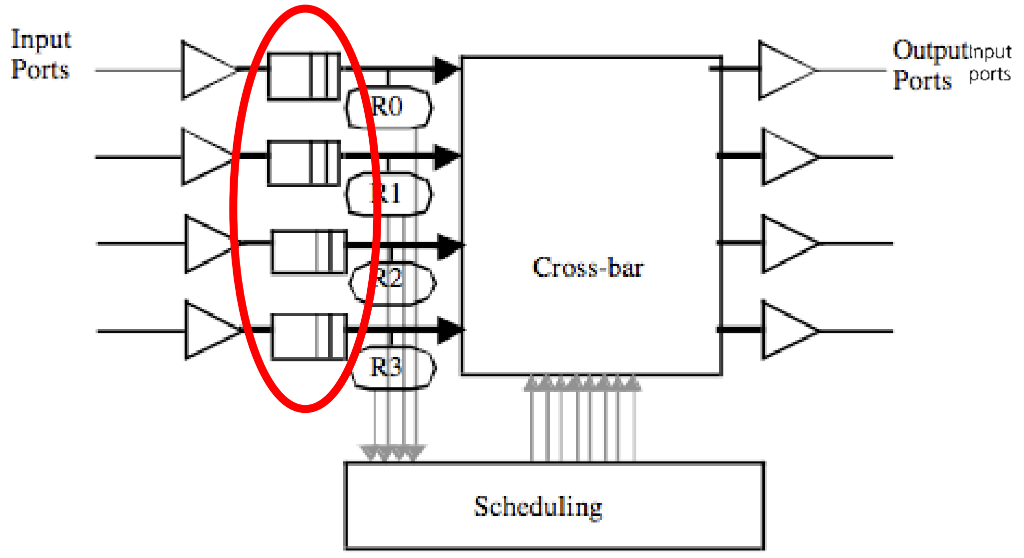


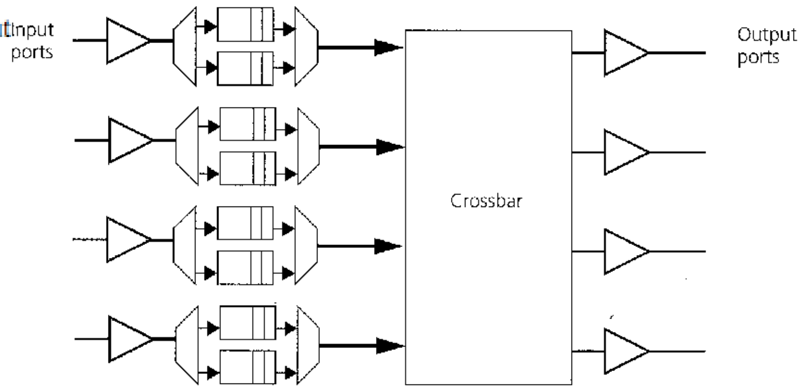
Head of Line Blocking


Red holds this channel: channel remains idle until read proceeds
Channel idle but red packet blocked behind blue
Buffer full: blue cannot proceed
Blocked by other packets

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.

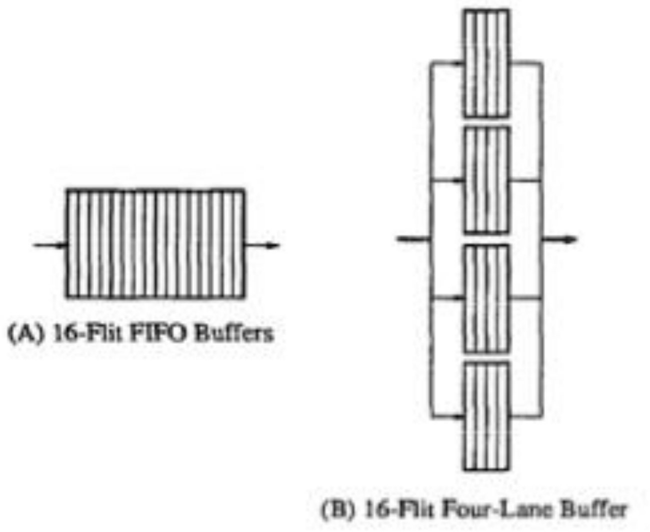
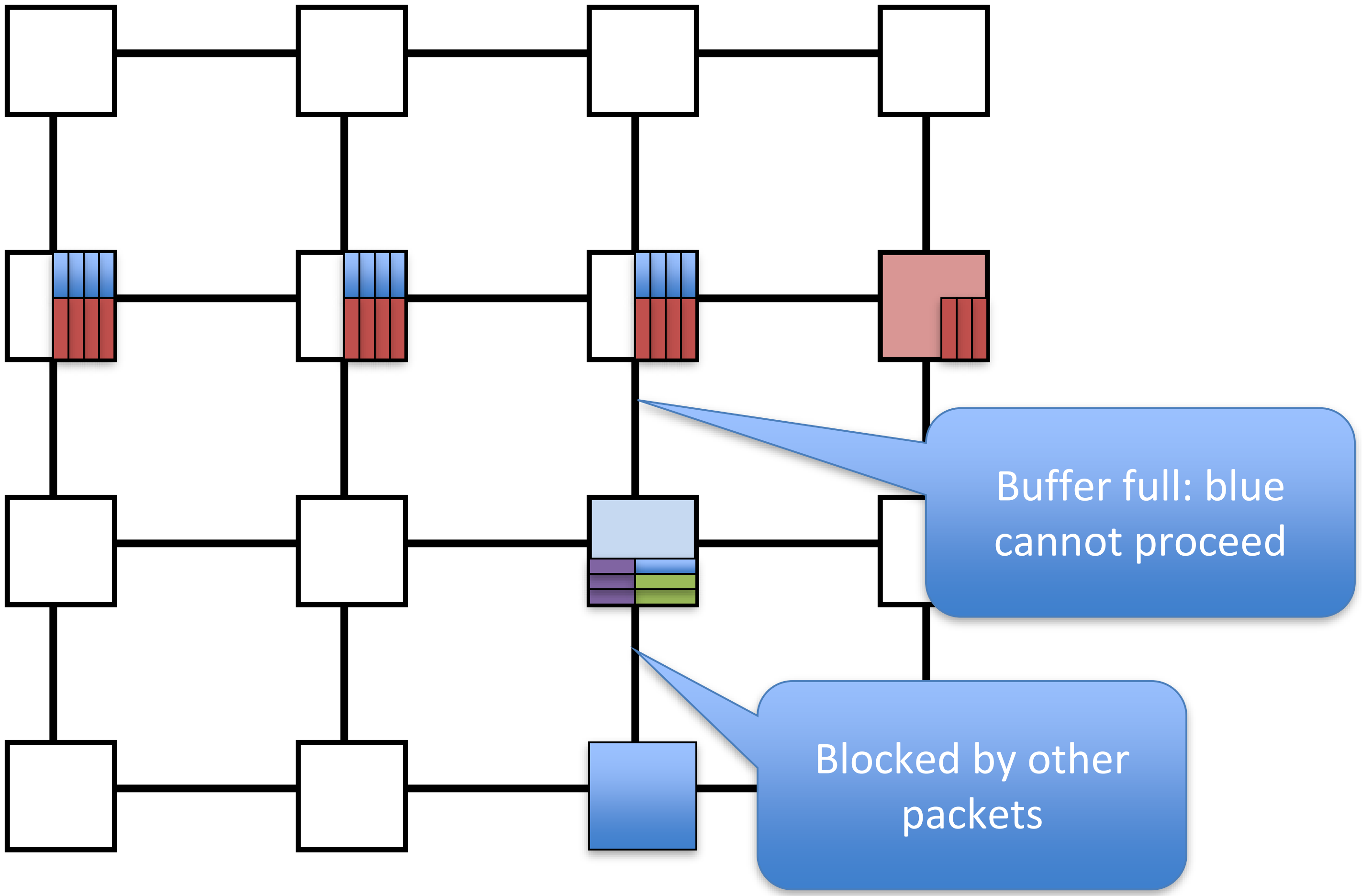


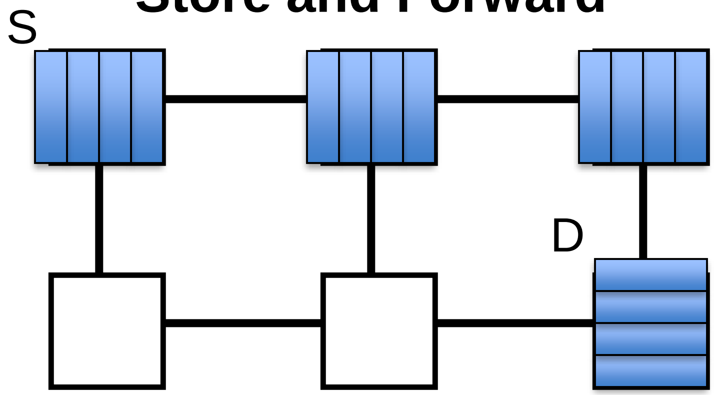
Figure 5: (A) Conventional nodes organize their buffers into FIFO queues restricting routing. (B) A network using virtual-channel flow control organizes its buffers into several independent lanes.

Virtual Channel Flow Control



Review: Flow Control

Store and Forward

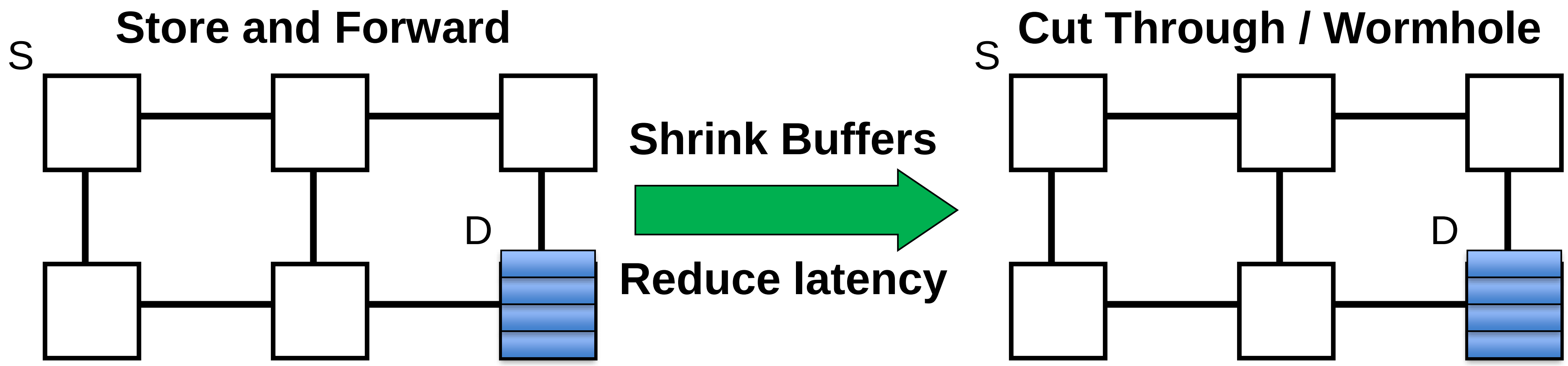


Any other issues?

Head-of-Line Blocking

Use Virtual Channels

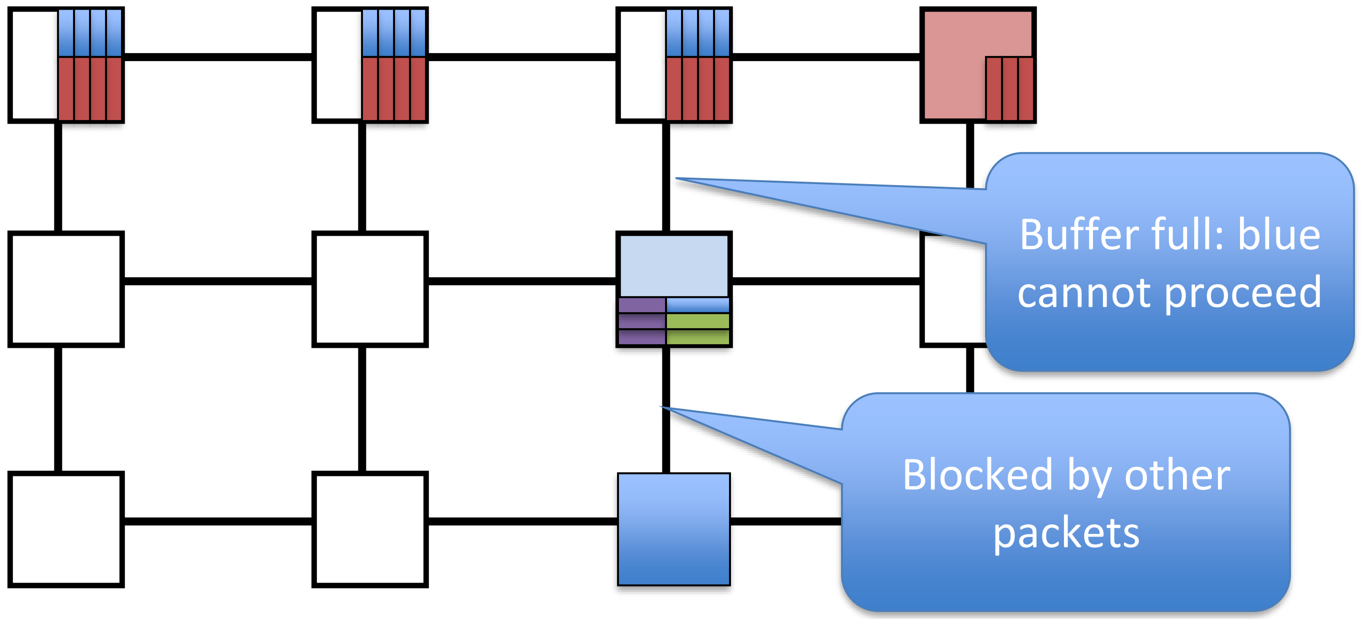
Review: Flow Control



Any other issues?

Head-of-Line Blocking

Use Virtual Channels



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

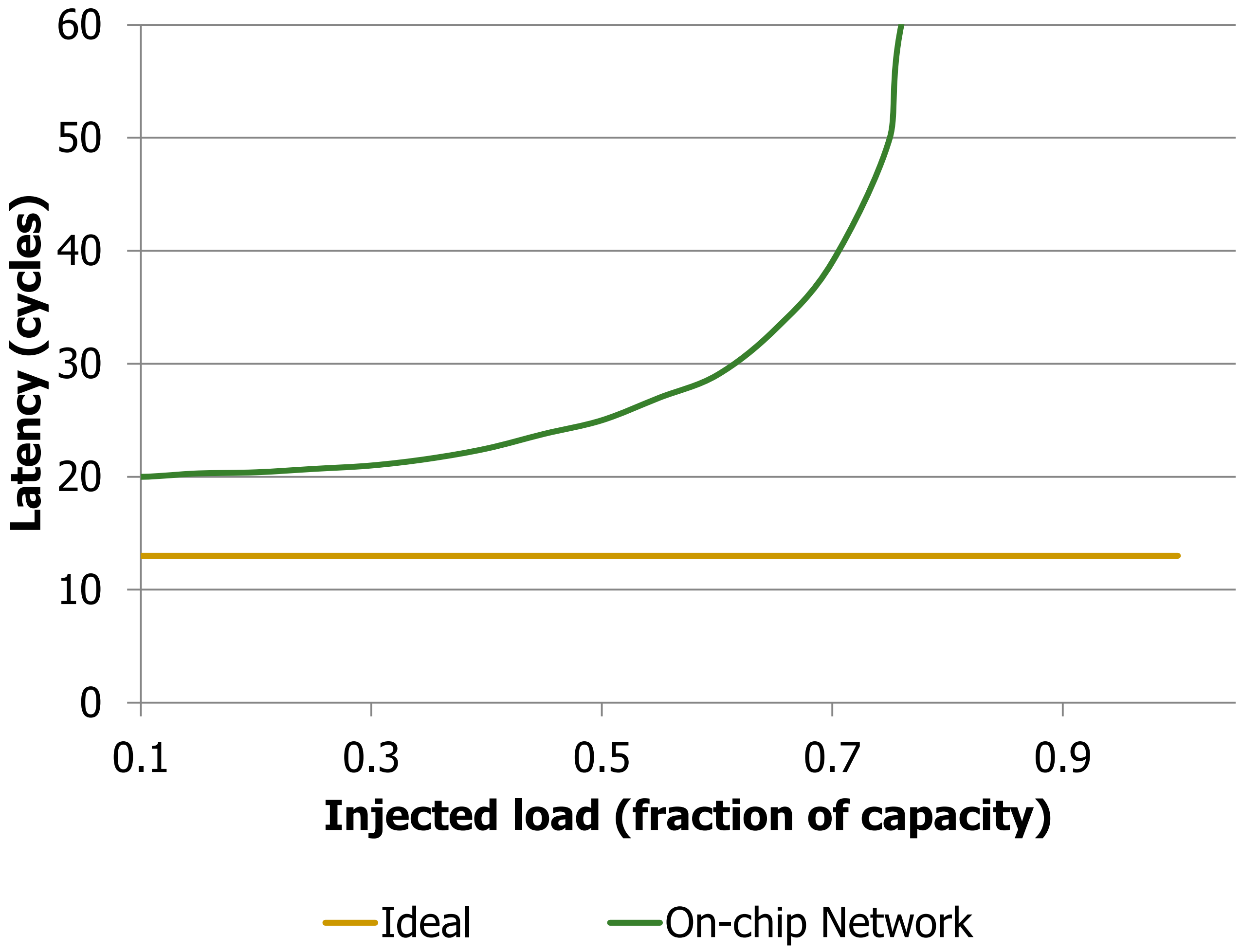
Actual Latency

- 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

Load-Latency Curve



Network Performance Metrics

- Packet latency (avg/max)
- Round trip latency (avg/max)
- Saturation throughput
- Application-level performance: execution time
- System performance: job throughput
  - Affected by interference among threads/applications