18-742
Parallel Computer Architecture
Lecture 17: Interconnection Networks I

Chris Fallin

Carnegie Mellon University

Material based on Michael Papamichael's 18-742 lecture slides from Spring 2011, in turn based on Onur Mutlu's 18-742 lecture slides from Spring 2010.

Readings: Interconnection Networks

- Required
  - Dally, "Virtual-Channel Flow Control," ISCA 1990.
  - Mullins et al., "Low-Latency Virtual-Channel Routers for On-Chip Networks," ISCA 2004.
  - Moscibroda and Mutlu, "A Case for Bufferless Routing in On-Chip Networks," ISCA 2009.
  - Wentzlaff et al., "On-Chip Interconnection Architecture of the Tile Processor," IEEE Micro 2007.
  - Patel et al., "Processor-Memory Interconnections for Multiprocessors," ISCA 1979.

- Recommended
  - Fallin et al., "CHIPPER: A Low-complexity Bufferless Deflection Router," HPCA 2011.
  - Fallin et al., "MinBD: Minimally-Buffered Deflection Routing for On-Chip Interconnect," NOCS 2012.
  - Tobias Bjerregaard, Shankar Mahadevan, "A Survey of Research and Practices of Network-on-Chip", ACM Computing Surveys (CSUR) 2006.

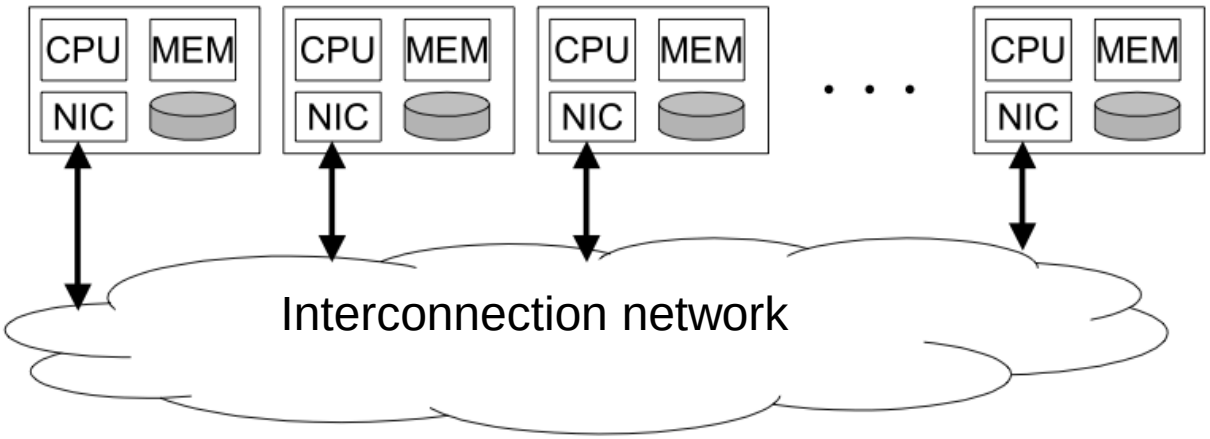
Agenda

- **Interconnection networks**
  - **Introduction and Terminology**
  - Topology
  - Buffering and Flow control

Where is a Network Used?

- To connect components

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



Interconnection Network Basics

- Topology
  - Specifies 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

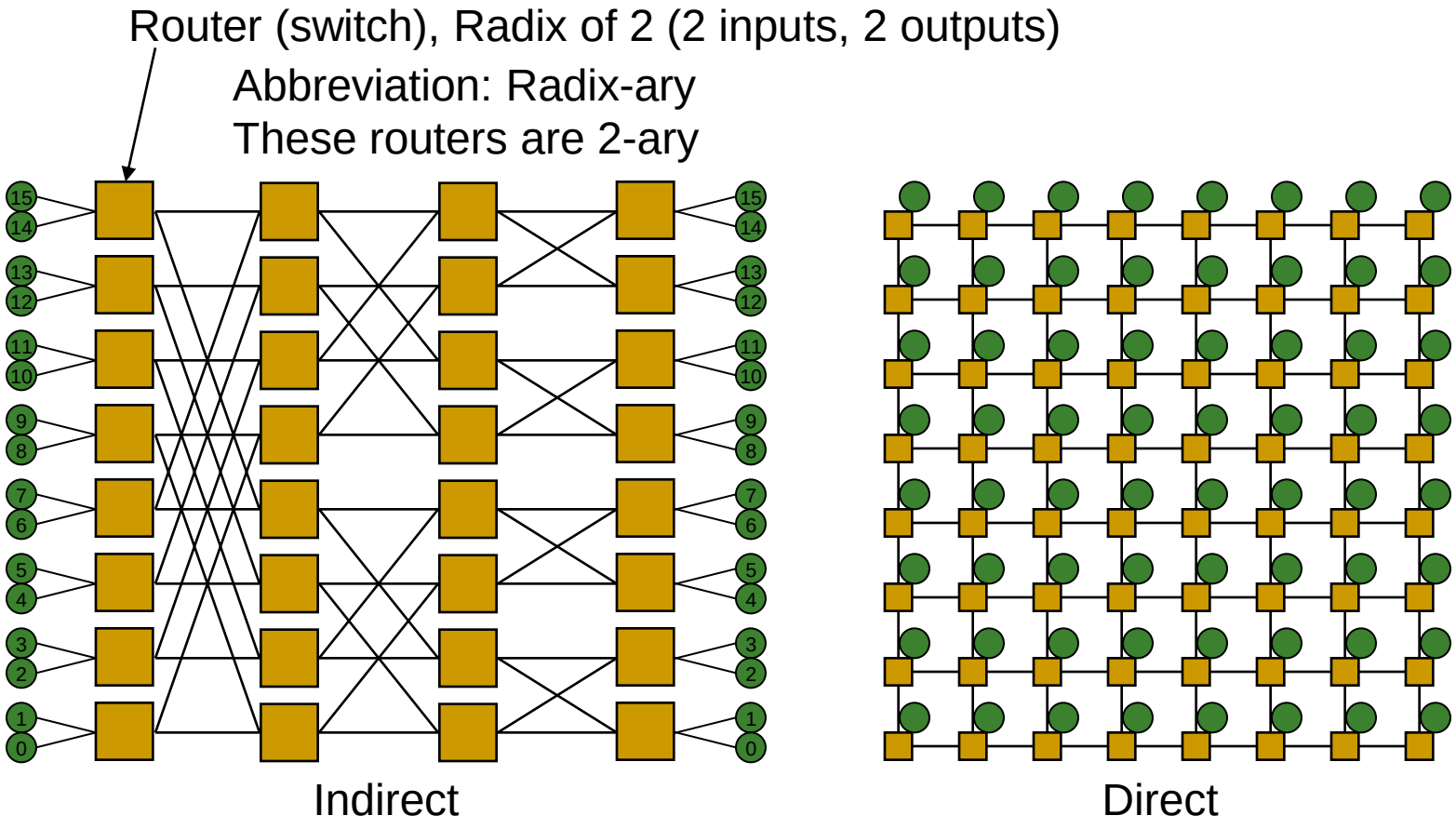
- Network interface
  - Module that connects endpoints (e.g. processors) to network.
  - Decouples computation/communication
- Links
  - Bundle of wires that carries a signal
- Switch/router
  - Connects fixed number of input channels to fixed number of output channels
- Channel
  - A single logical connection between routers/switches

Some 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



Agenda

- **Interconnection networks**
  - Introduction and Terminology
  - **Topology**
  - Buffering and Flow control

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 route
- Diameter
  - maximum routing distance
- 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
    - (Min # channels spanning two halves) * (BW of each channel)
  - Meaningful only for recursive topologies
  - Can be misleading, because does not account for switch and routing efficiency

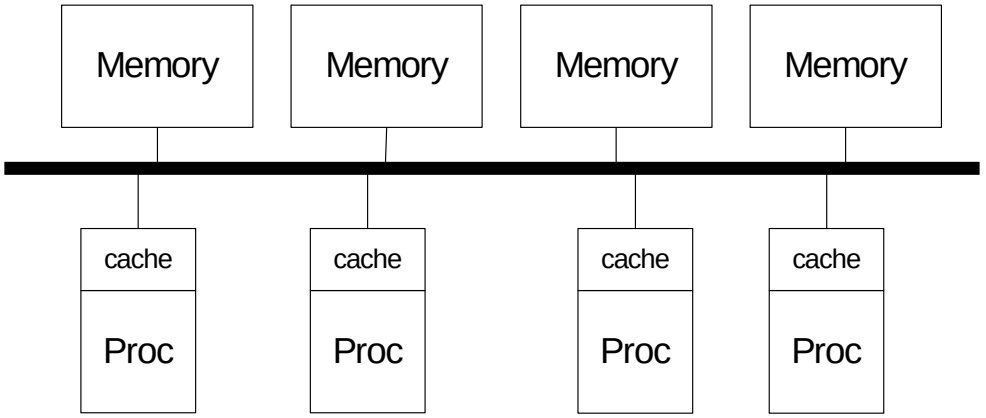
- Blocking vs. Non-Blocking
  - If connecting any permutation of sources & destinations is possible, network is non-blocking; otherwise network is blocking.
  - Rearrangeable non-blocking: Same as non-blocking but might require rearranging connections when switching from one permutation to another.

Topology

- Bus
- Crossbar
- Ring
- Tree
- Omega
- Hypercube
- Mesh
- Torus
- Butterfly
- …

Bus

+ Simple
+ Cost effective for a small number of nodes
+ Easy to implement coherence (snooping)
- Not scalable to large number of nodes (limited bandwidth, electrical loading → reduced frequency)
- High contention



Crossbar

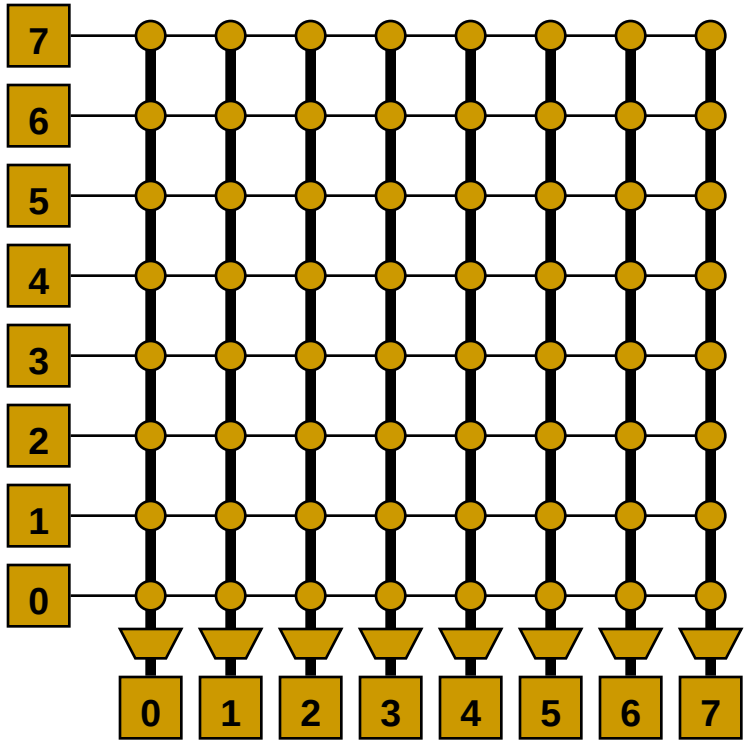
- Every node connected to every other (non-blocking)
- Good for small number of nodes

+ Low latency and high throughput

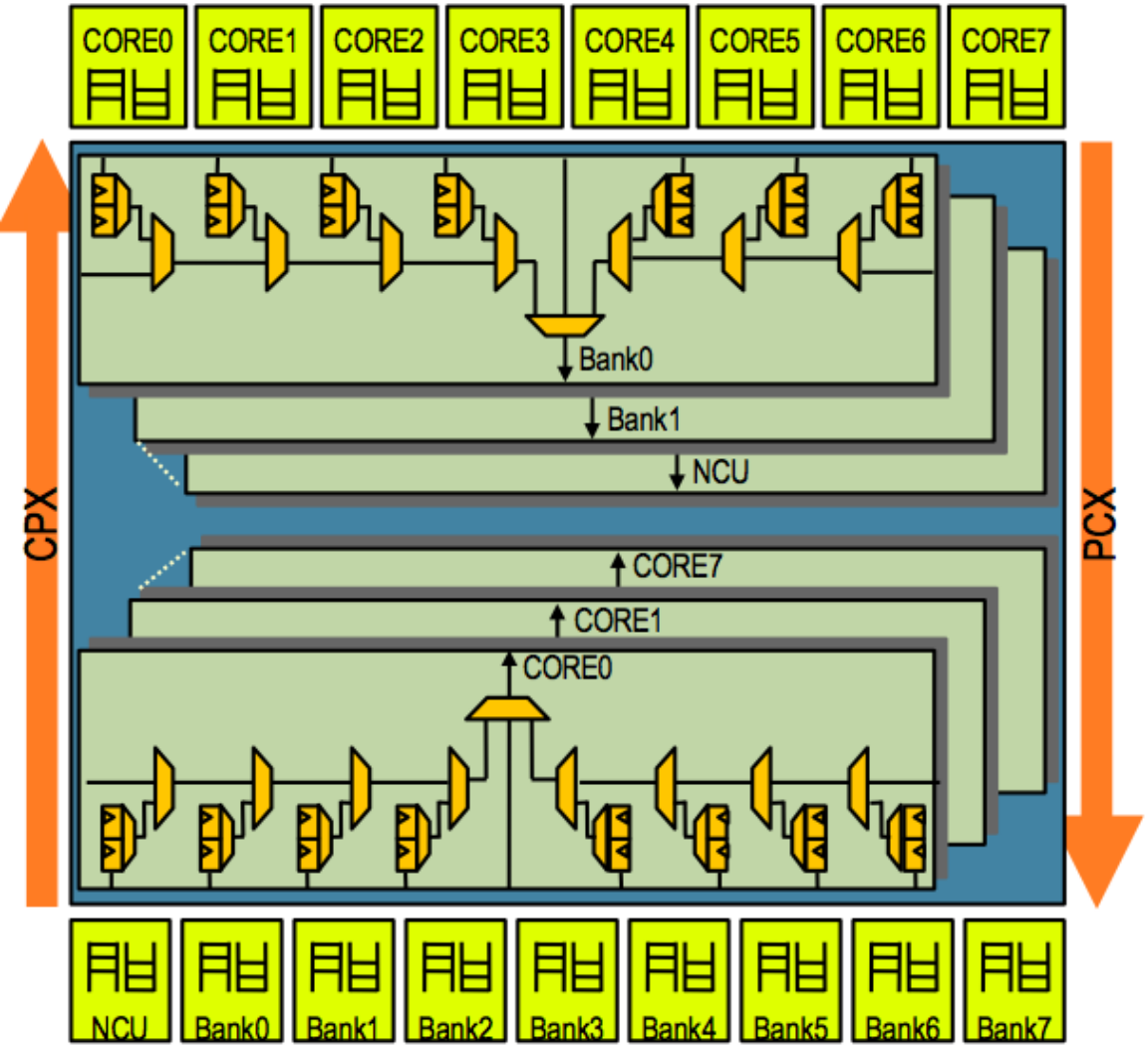
- Expensive
- Not scalable → $O(N^2)$ cost
- Difficult to arbitrate

Used in core-to-cache-bank networks in

- IBM POWER5
- Sun Niagara I/II

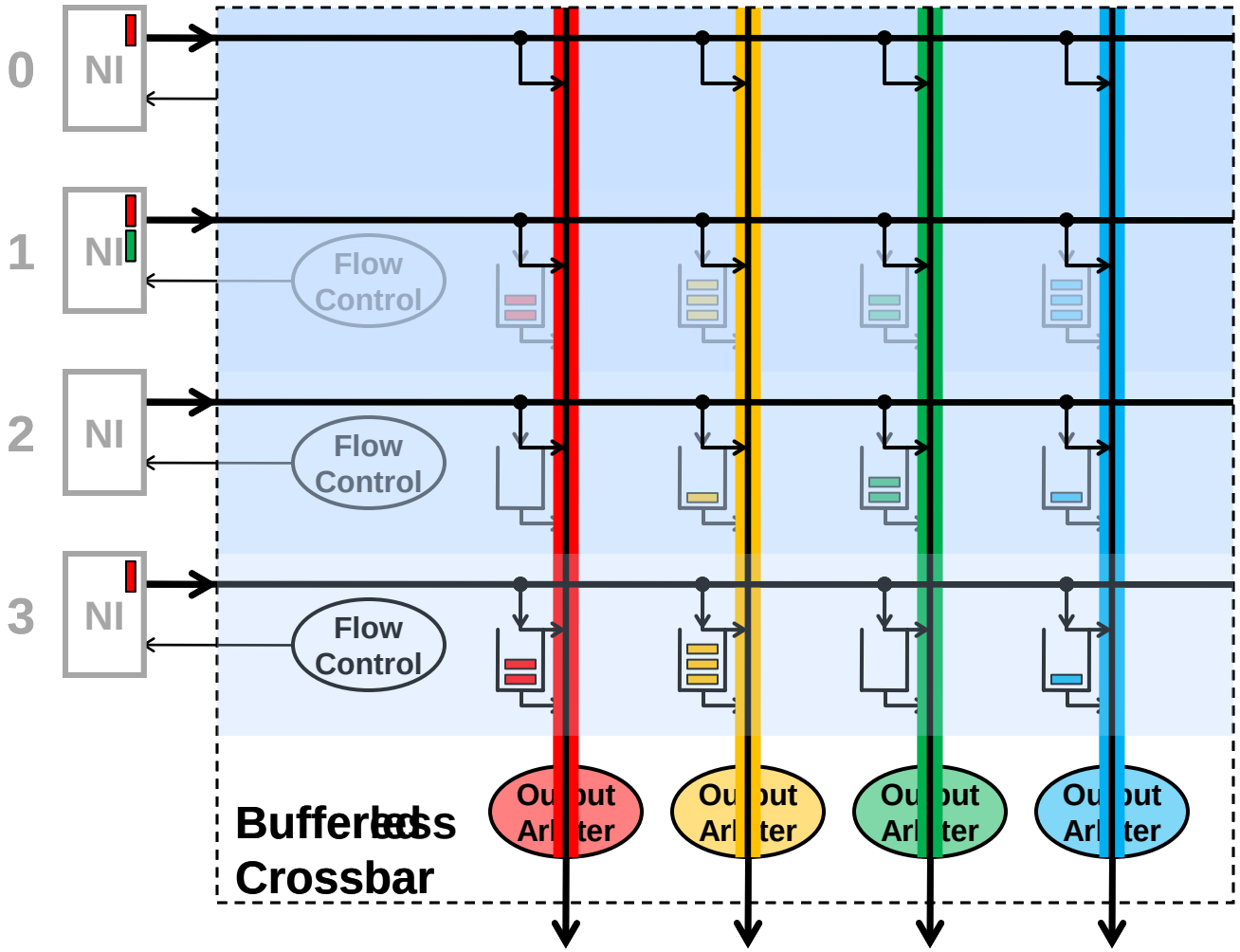


Sun UltraSPARC T2 Core-to-Cache Crossbar



- 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

Buffered Crossbar



+ Simpler arbitration/ scheduling

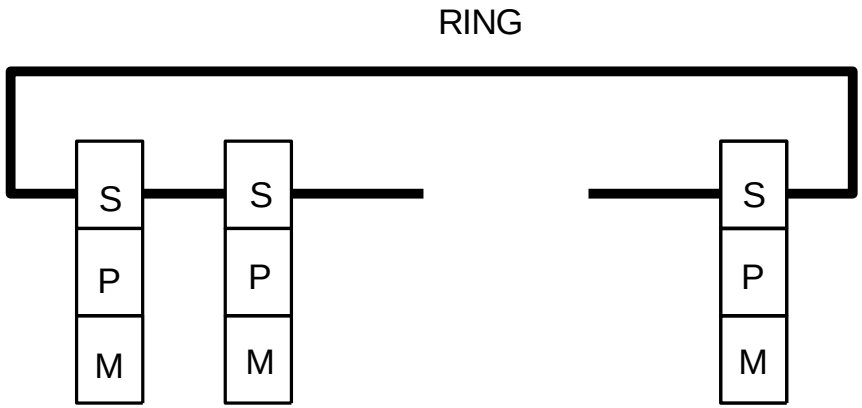
+ Efficient support for variable-size packets

- Requires N^2 buffers

Ring

+ Cheap: O(N) cost
- High latency: O(N)
- Not easy to scale
  - Bisection bandwidth remains constant
  - But, hierarchy/multi-ring designs can address scalability

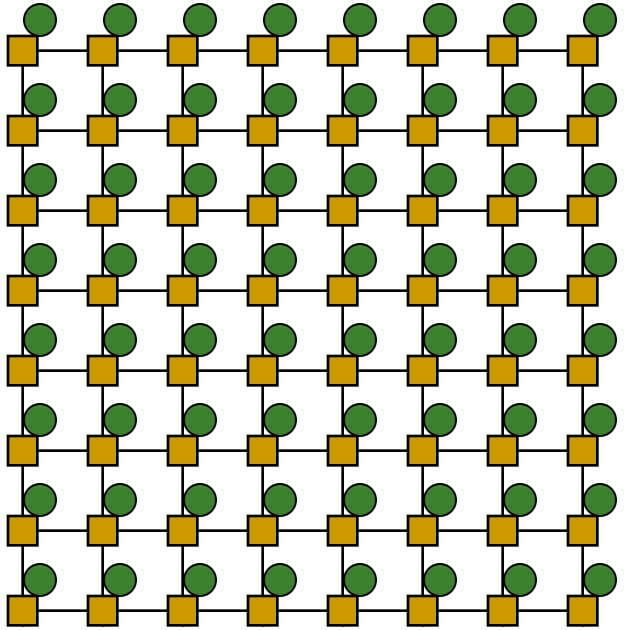
Used in Intel Larrabee, IBM Cell



Mesh

- 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

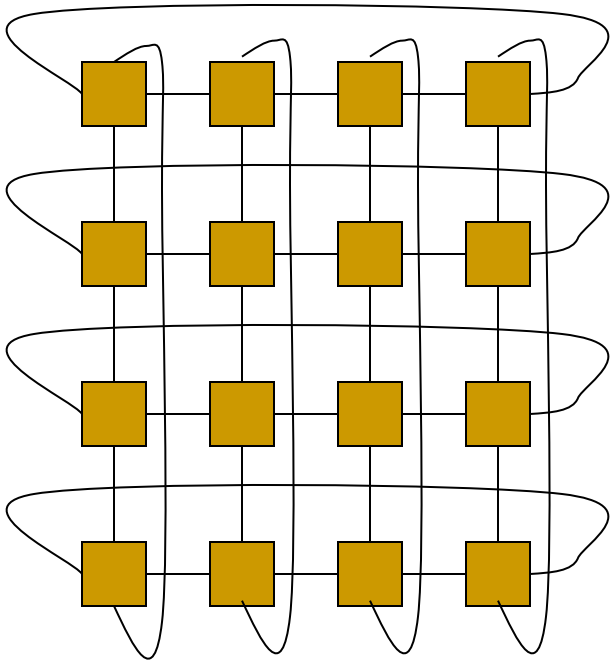
- Mesh is not symmetric on edges: performance very sensitive to placement of task on edge vs. middle
- Torus avoids this problem

+ Higher path diversity (and bisection bandwidth) than mesh

- Higher cost

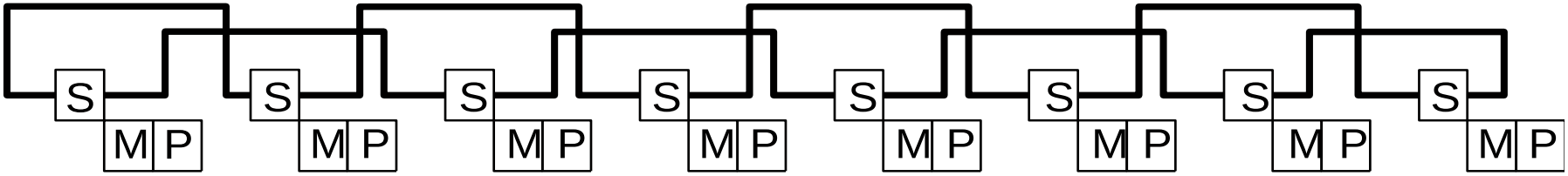
- Harder to lay out on-chip

  - Unequal link lengths



Torus, continued

- Weave nodes to make inter-node latencies ~constant



Trees

Planar, hierarchical topology

Latency: O(logN)

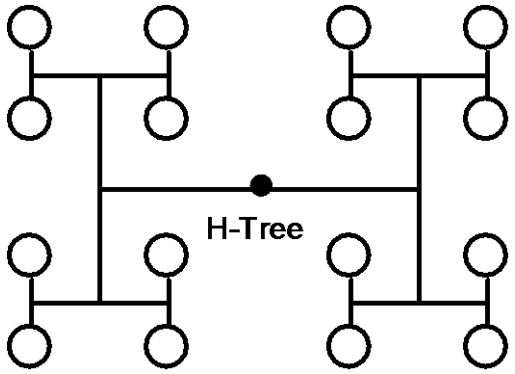
Good for local traffic

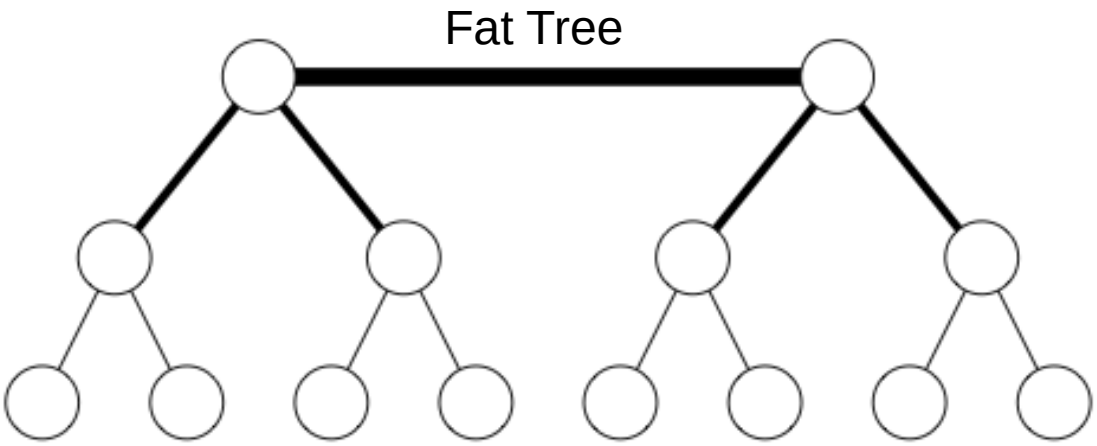
+ Cheap: O(N) cost

+Easy to Layout

- Root can become a bottleneck

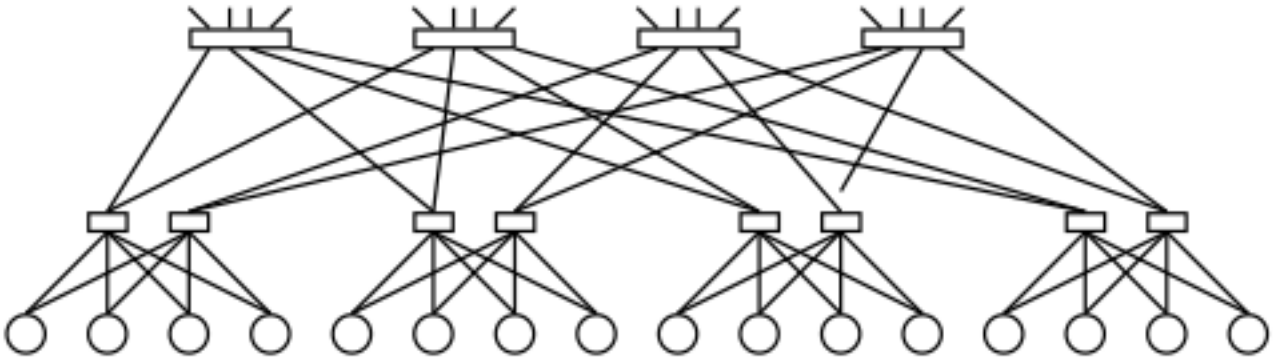
  Fat trees avoid this problem (CM-5)





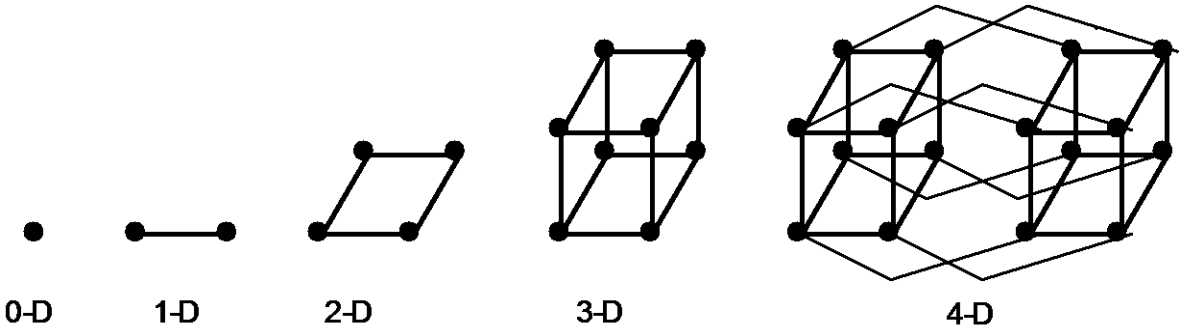
CM-5 Fat Tree

- 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

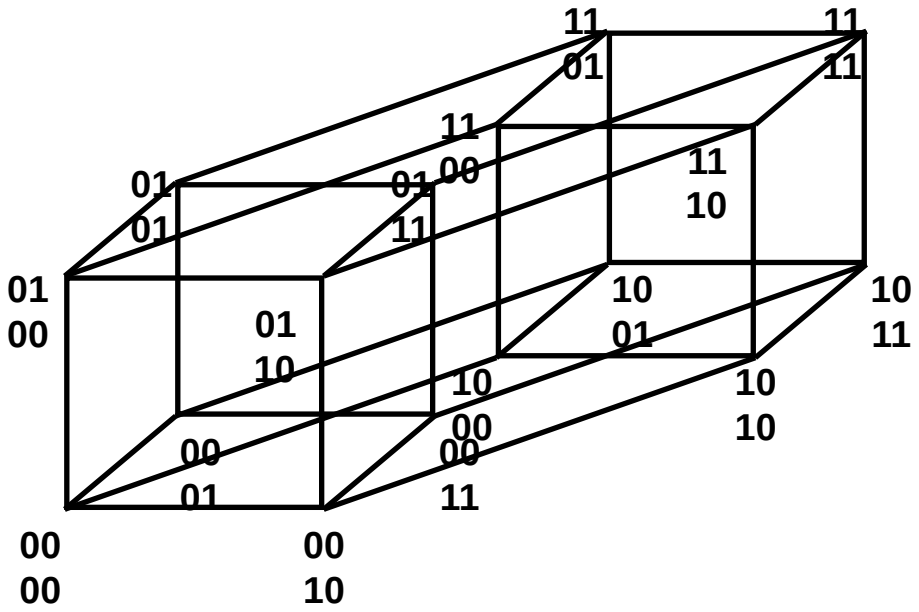
Hypercube



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

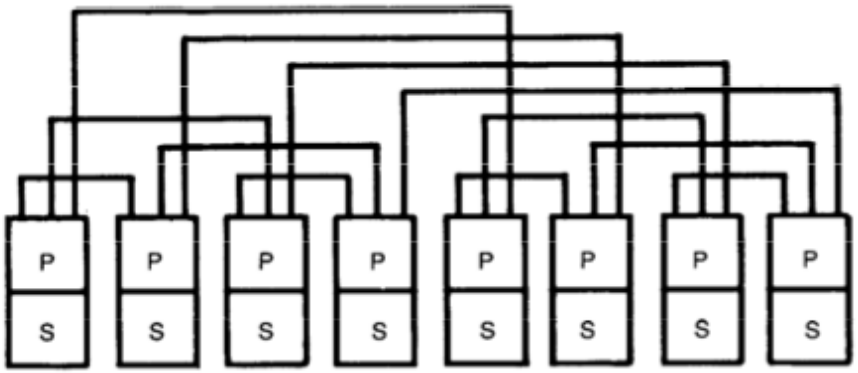
+ Low latency

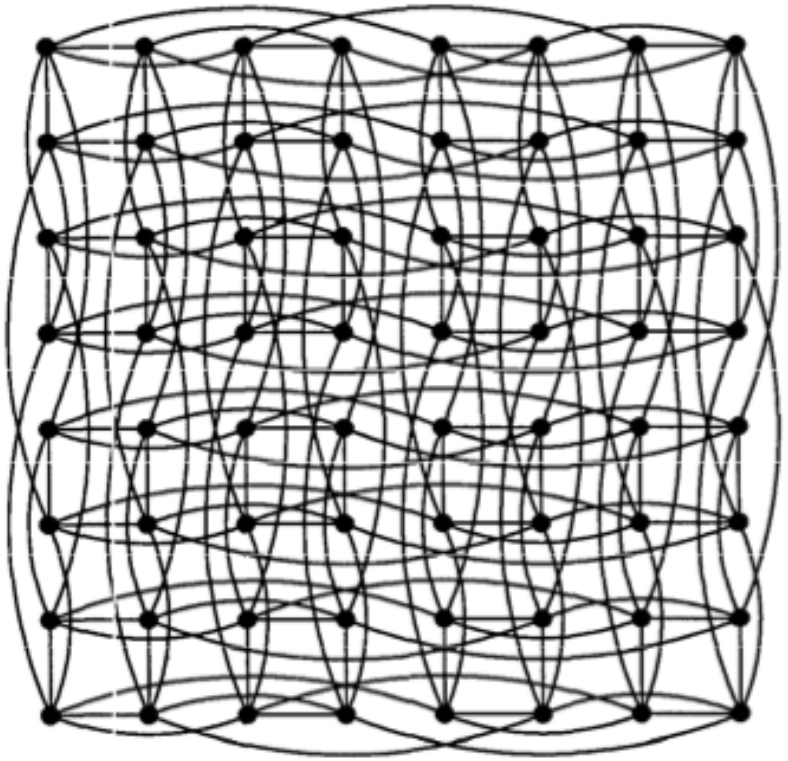
- Hard to lay out in 2D/3D



Caltech Cosmic Cube

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



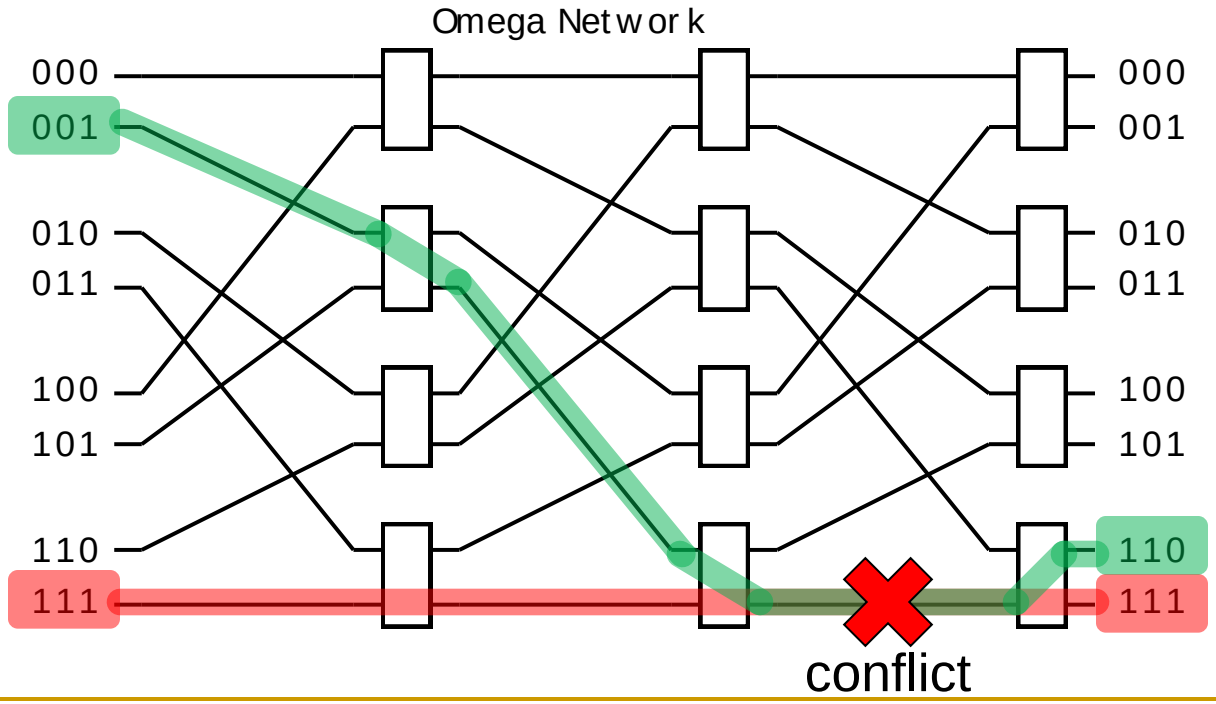


A hypercube connects $N = 2^n$ small computers, called nodes, through point-to-point communication channels in the Cosmic Cube. Shown here is a two-dimensional projection of a six-dimensional hypercube, or binary 6-cube, which corresponds to a 64-node machine.

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

Multistage Logarithmic Networks

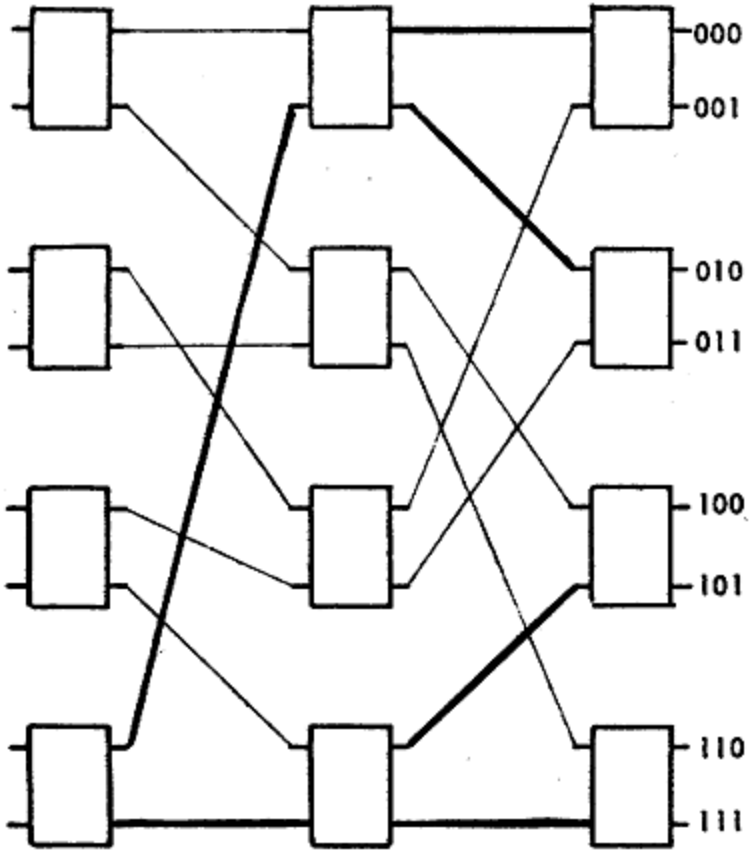
- Idea: Indirect networks with multiple layers of switches between terminals
- Cost: O(NlogN), Latency: O(logN)
- Many variations (Omega, Butterfly, Benes, Banyan, …)
- Omega Network:



Q: Blocking or non-blocking?

Delta Network

- Single path from source to destination
- Does not support all possible permutations
- Proposed to replace costly crossbars as processor-memory interconnect
- Janak H. Patel ,"Processor-Memory Interconnections for Multiprocessors," ISCA 1979.



8x8 Delta network

Omega Network

- Single path from source to destination
- All stages are the same
- Used in NYU Ultracomputer
- Gottlieb et al. "The NYU Ultracomputer-designing MIMD, shared-memory parallel machine," ISCA 1982.

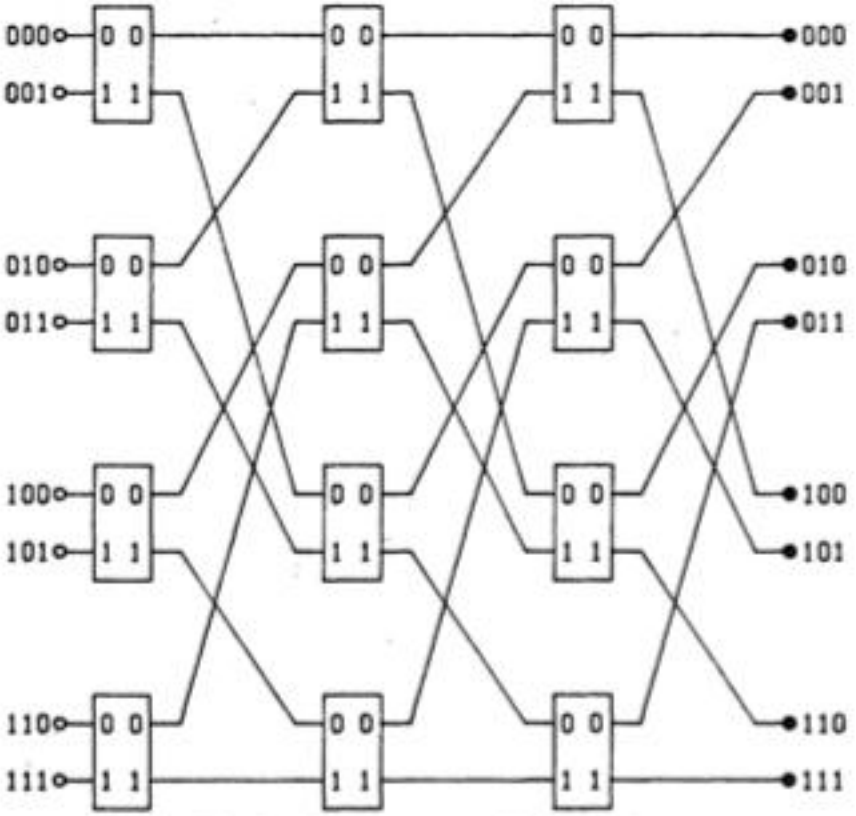


Fig. 2. Omega-network ($N = 8$).

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

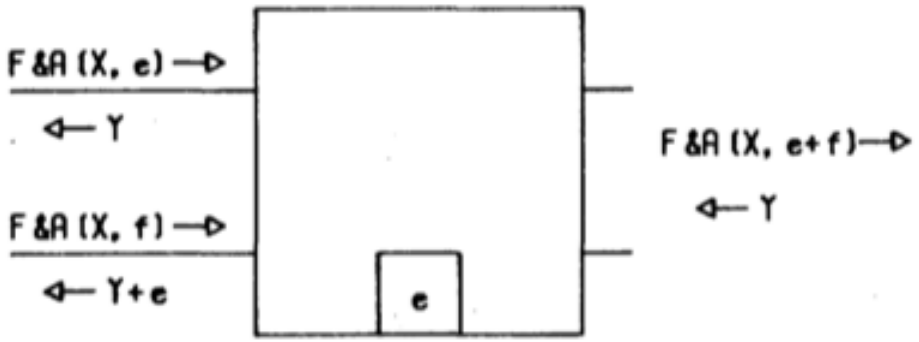


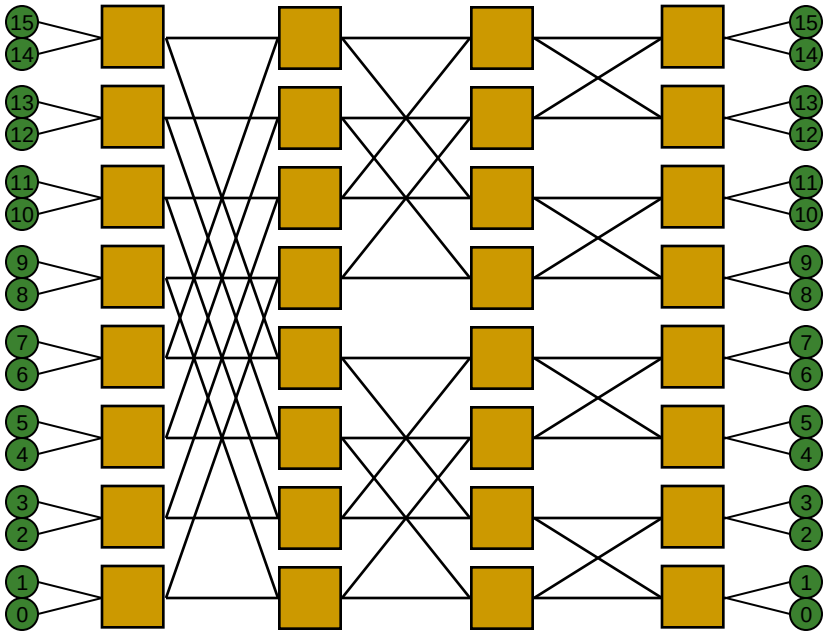
Fig. 3. Combining Fetch-and-Adds.

```
TestAndSet(V)
    {Temp ← V
     V ← 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



Agenda

- **Interconnection networks**
  - Introduction and Terminology
  - Topology
  - **Buffering and Flow control**

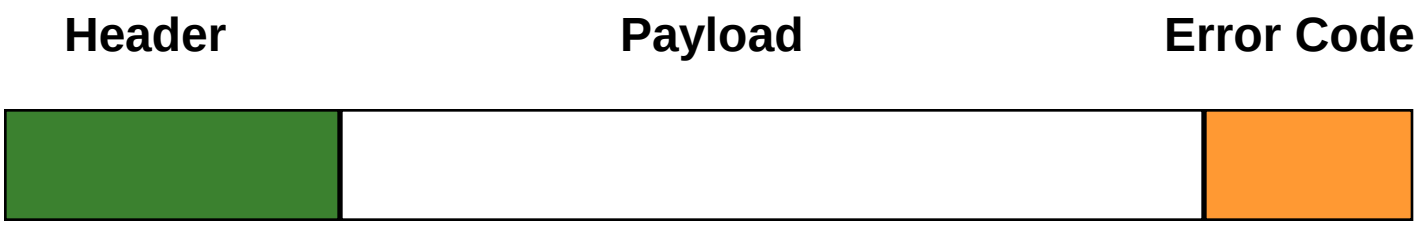
Circuit vs. Packet Switching

- Circuit switch sets up full path
  - Establish route then send data
  - (no one else can use those links)
  - faster and higher bandwidth
  - setting up and bringing down links slow

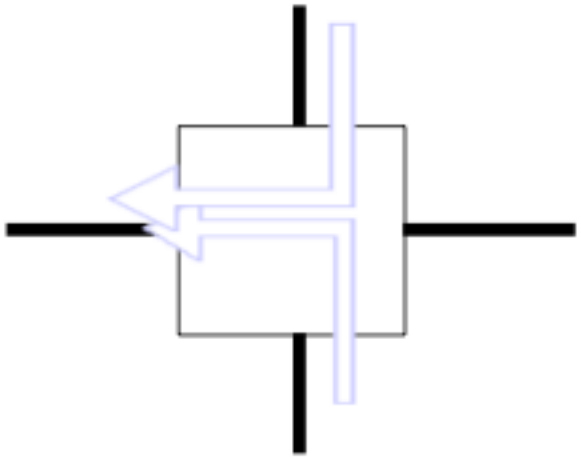
- Packet switching routes per packet
  - Route each packet individually (possibly via different paths)
  - if link is free can use
  - potentially slower --- must dynamically switch
  - no setup, bring down time

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)
- Assume buffering for now

Flow Control Methods

- Circuit switching
- Store and forward (Packet based)
- Virtual Cut Through (Packet based)
- Wormhole (Flit based)

Circuit Switching Revisited

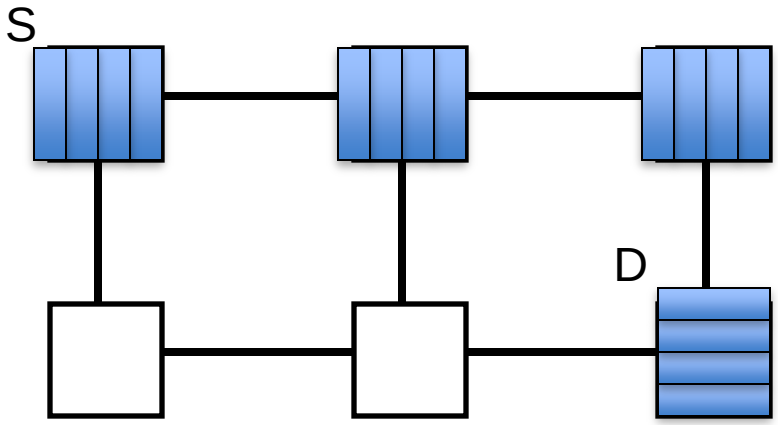
- Resource allocation granularity is high

- 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"

Store and Forward Flow Control

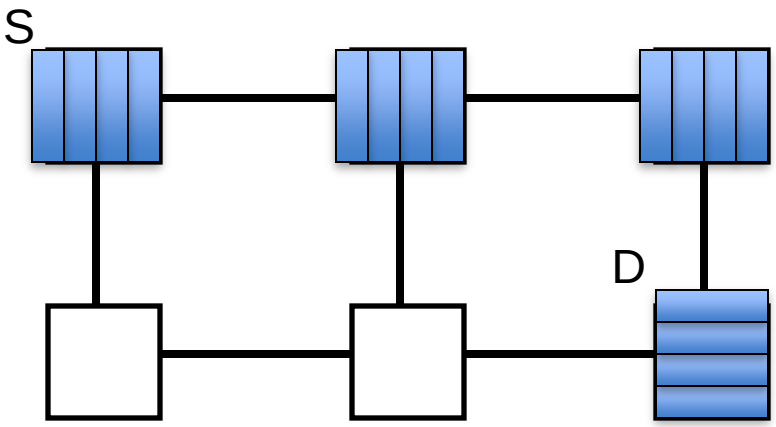
- Packet based flow control
- Store and Forward
  - Packet copied entirely into network router before moving to the next node
  - Flow control unit is the entire packet
- Leads to high per-packet latency
- Requires buffering for entire packet in each node



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



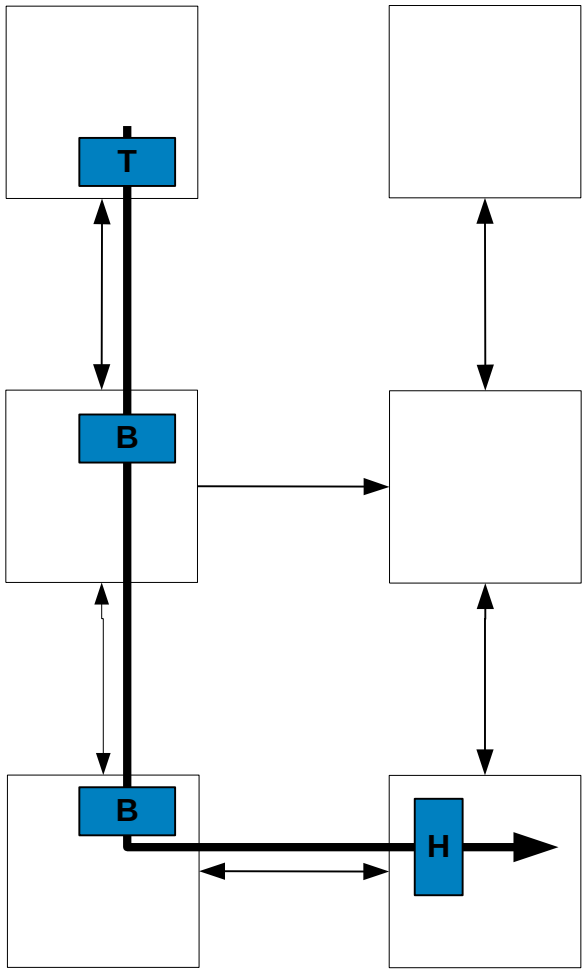
- What if packets are large?

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



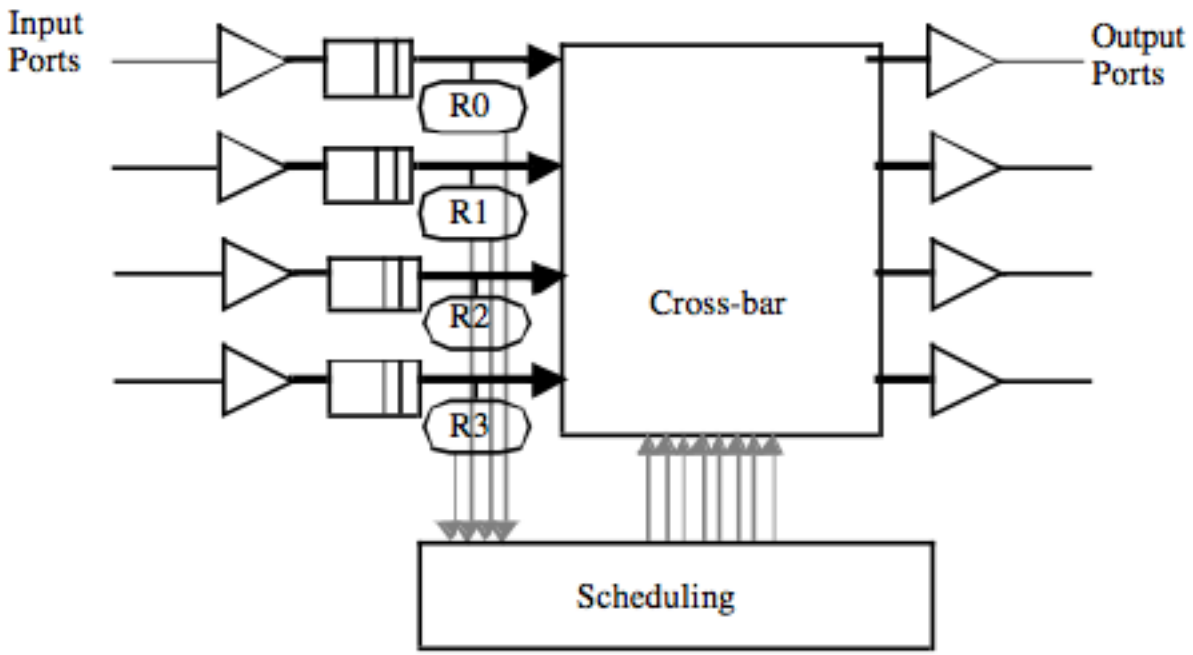
- Packets broken into (potentially) smaller flits (buffer/bw allocation unit)
- Flits are sent across the fabric in a *wormhole fashion*
  - Body follows head, tail follows body
  - Pipelined
  - If head blocked, rest of packet stops
  - Routing (src/dest) information only in head
- How does body/tail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

Wormhole Flow Control

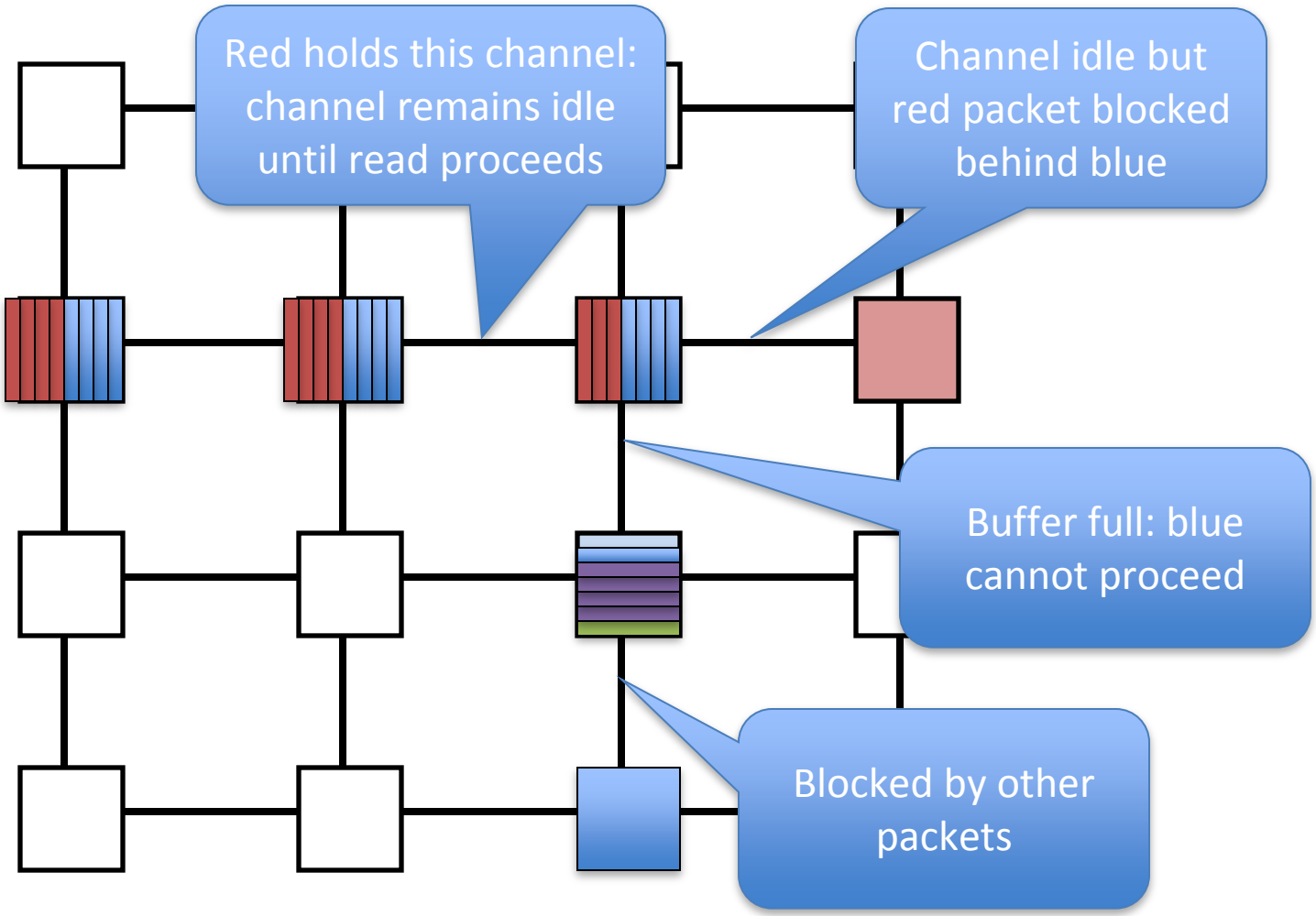
- Advantages over "store and forward" flow control
  + Lower latency
  + More efficient buffer utilization
- Limitations
  - Suffers from **head of line blocking**
    - If head flit cannot move due to contention, another worm cannot proceed even though links may be idle

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

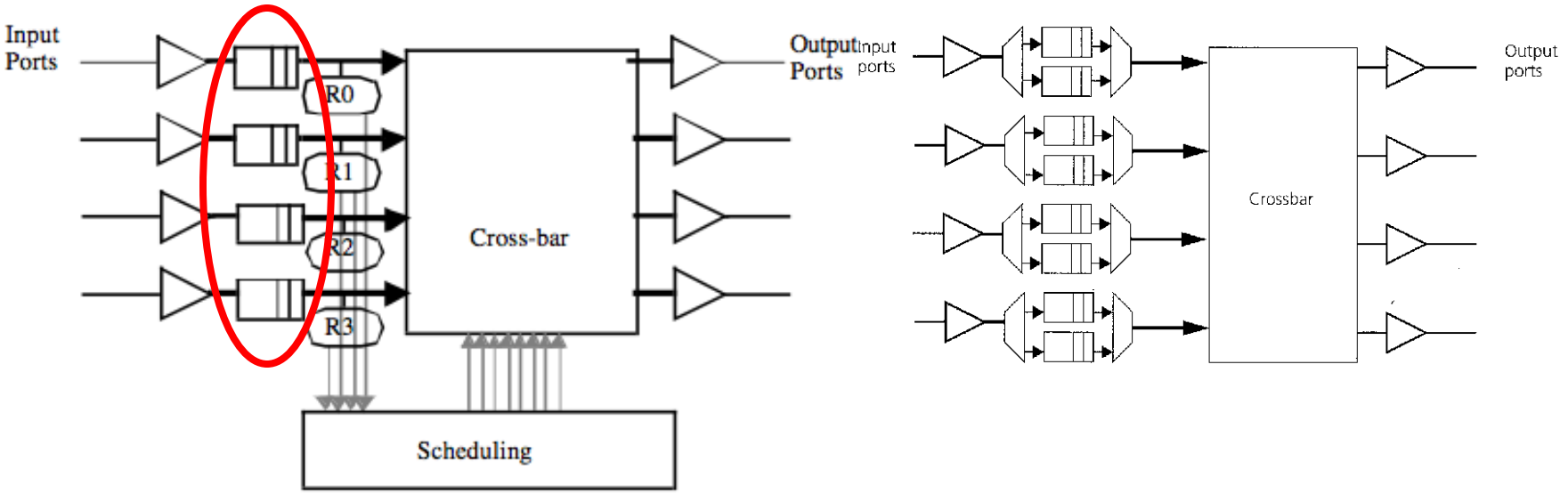


Head of Line Blocking



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.

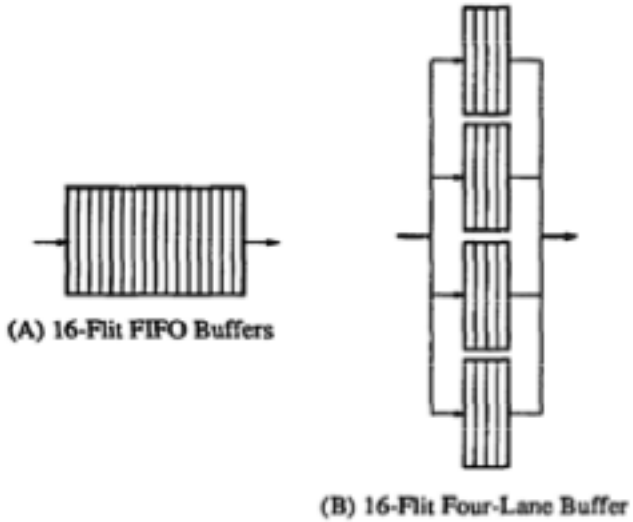
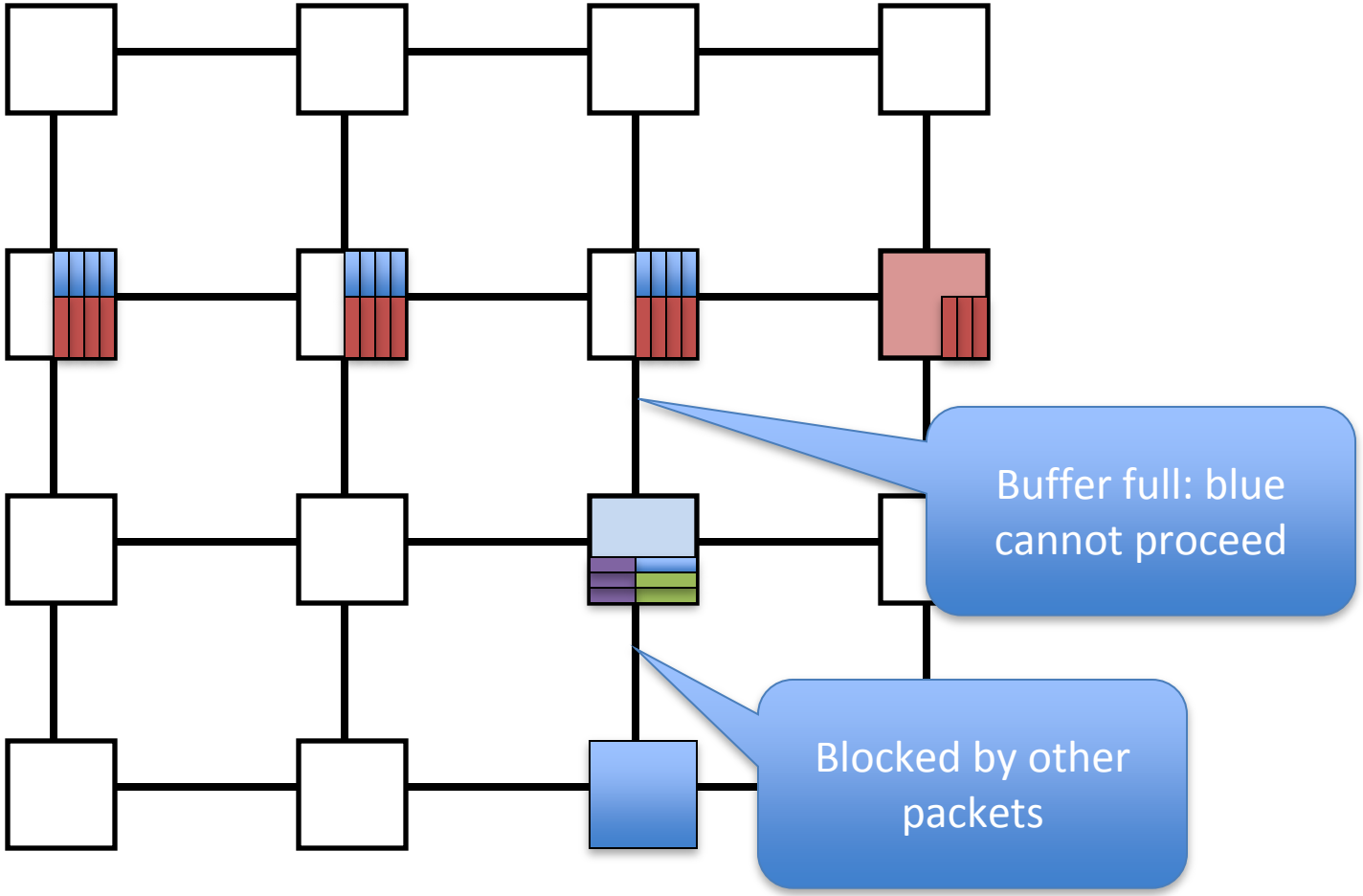
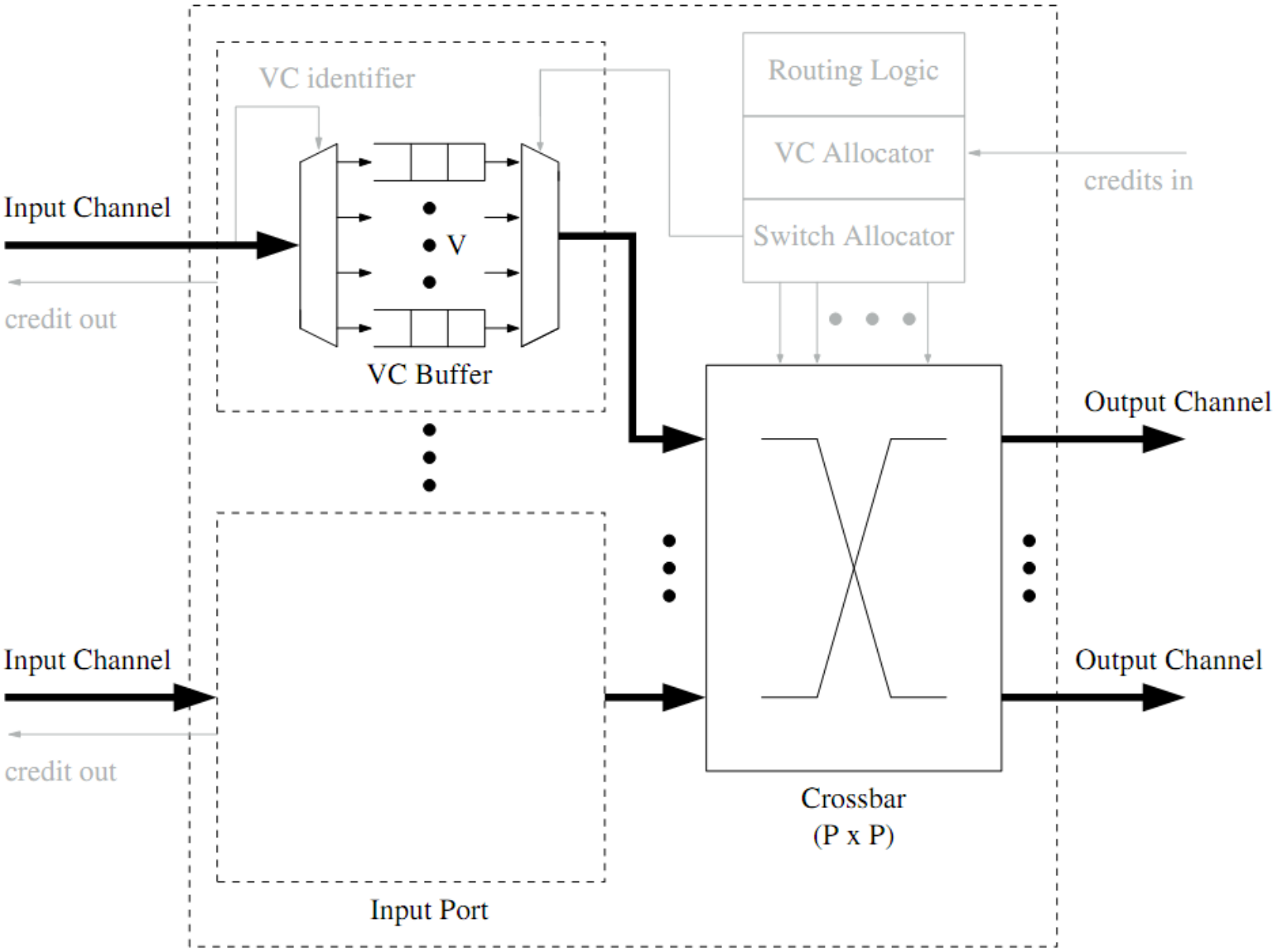


Figure 5: (A) Conventional nodes organize their buffers into FIFO queues restricting routing. (B) A network using virtual-channel flow control organizes its buffers into several independent lanes.

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

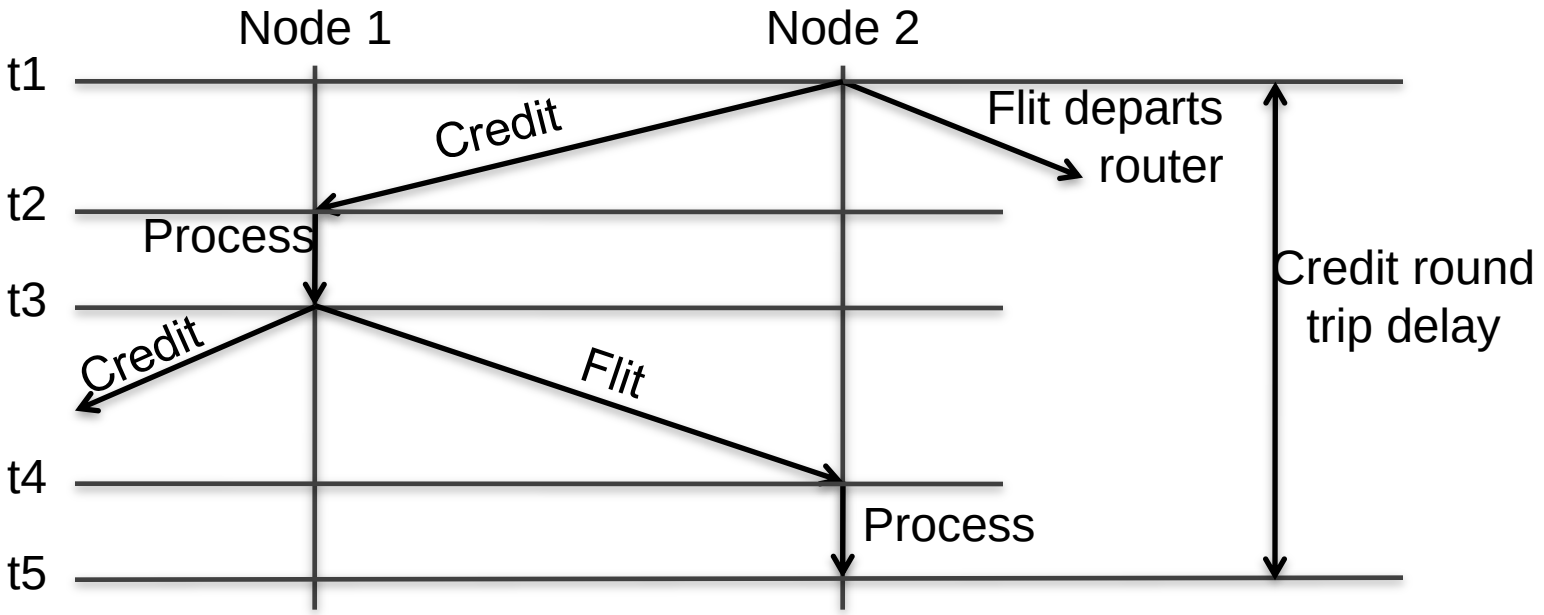
Communicating Buffer Availability

- 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