

# On-Chip Networks I: Topology/Flow Control

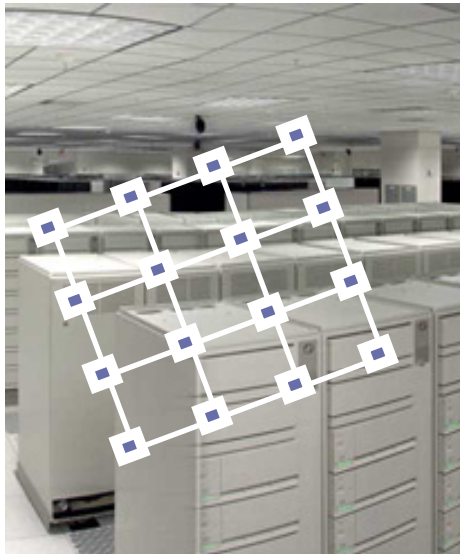
*Ryan Lee*

Computer Science & Artificial Intelligence Lab  
M.I.T.

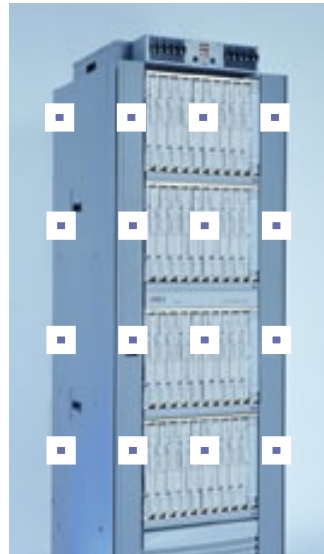
# History: From interconnection networks to on-chip networks

---

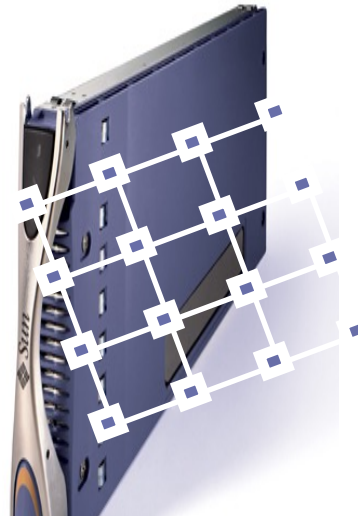
Box-to-box networks



Board-to-board networks



Chip-to-chip networks



On-chip networks

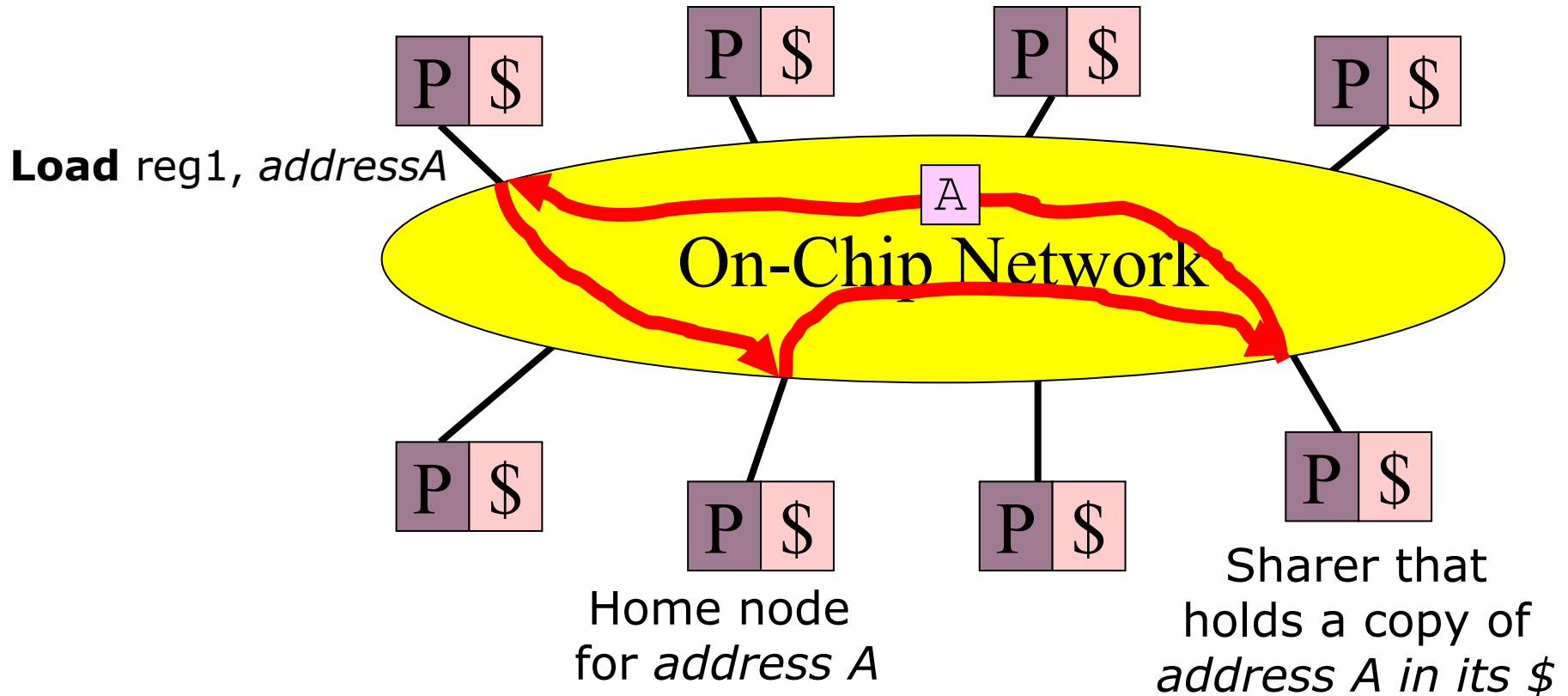


**Focus on on-chip networks connecting caches in shared-memory processors**

Multi-Chip: Supercomputers, Data Centers, Internet Routers, Servers  
On-Chip: Servers, Laptops, Phones, HDTVs, Access routers

# What's an on-chip network?

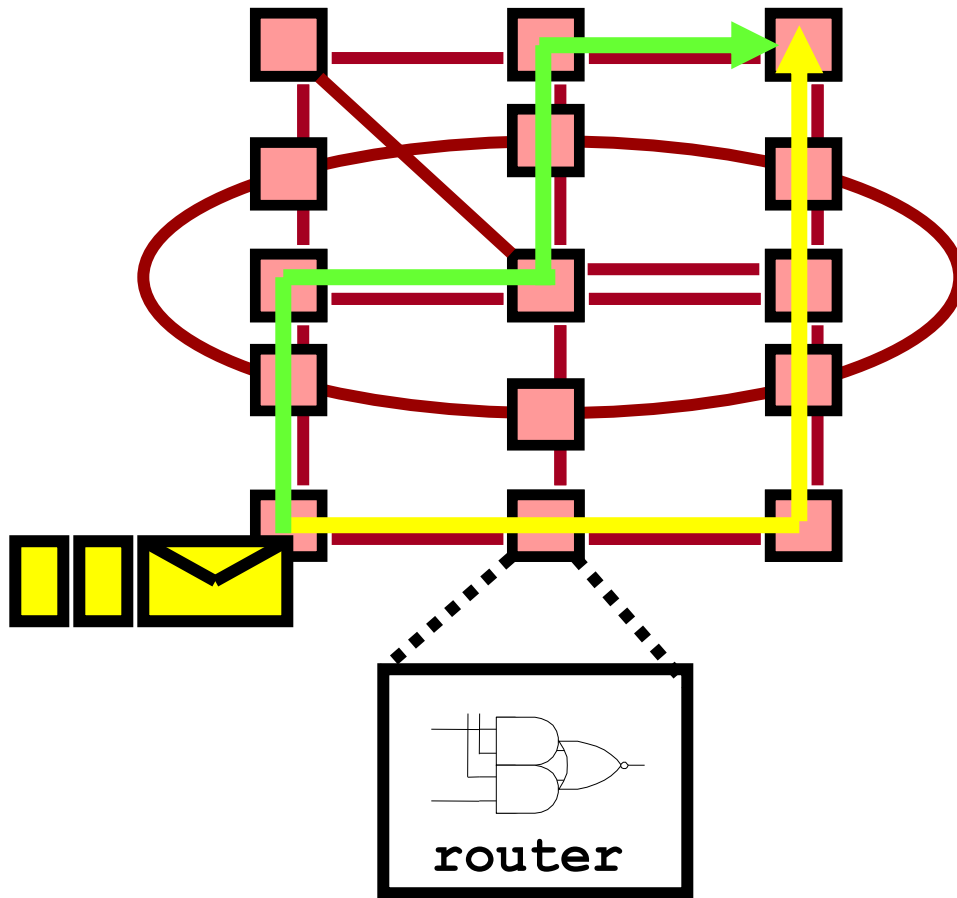
E.g. Cache-coherent chip multiprocessor



**Network transports cache coherence messages and cache lines between processor cores**

# Designing an on-chip network

---



- Topology
- Flow control
- Router microarchitecture
- Routing

# Interconnection Network Architecture

---

- *Topology*: How to connect the nodes up?  
(processors, memories, router line cards, ...)
- *Routing*: Which path should a message take?
- *Flow control*: How is the message actually forwarded from source to destination?
- *Router microarchitecture*: How to build the routers?
- *Link microarchitecture*: How to build the links?

# Topology

# Topological Properties

---

- *Diameter*
- *Average Distance*
- *Bisection Bandwidth*

# Topological Properties

---

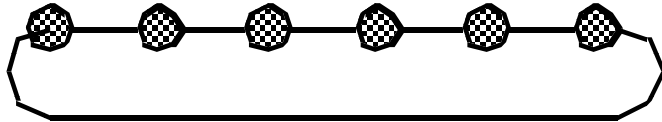
- *Routing Distance* - number of links on route
- *Diameter* - maximum routing distance
- *Average Distance*
- A network is *partitioned* by a set of links if their removal disconnects the graph
- *Bisection Bandwidth* is the bandwidth crossing a minimal cut that divides the network in half



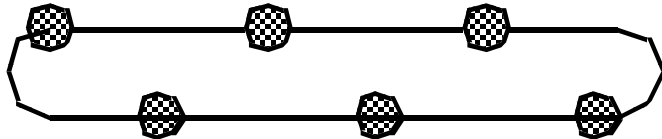
# Linear Arrays and Rings



Linear Array



Torus



Torus arranged to use short wires

Route A  $\rightarrow$  B given by relative address  $R = B - A$

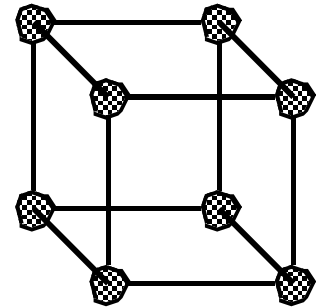
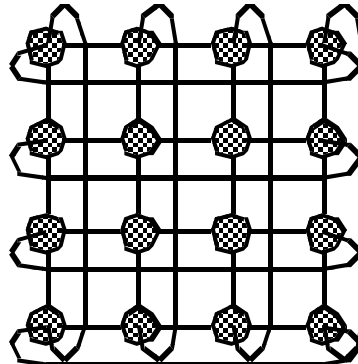
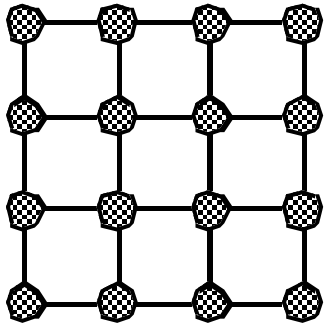
	Linear Array	Ring (1-D Torus)
Diameter?	$N-1$	$N/2$ (if even $N$ )
Average distance?	$N/3 - 1/(3N)$	$N/4$ (if even $N$ )
Bisection bandwidth?	1	2

- Torus Examples:

- FDDI, SCI, FiberChannel Arbitrated Loop, Intel Xeon

# Multidimensional Meshes and Tori

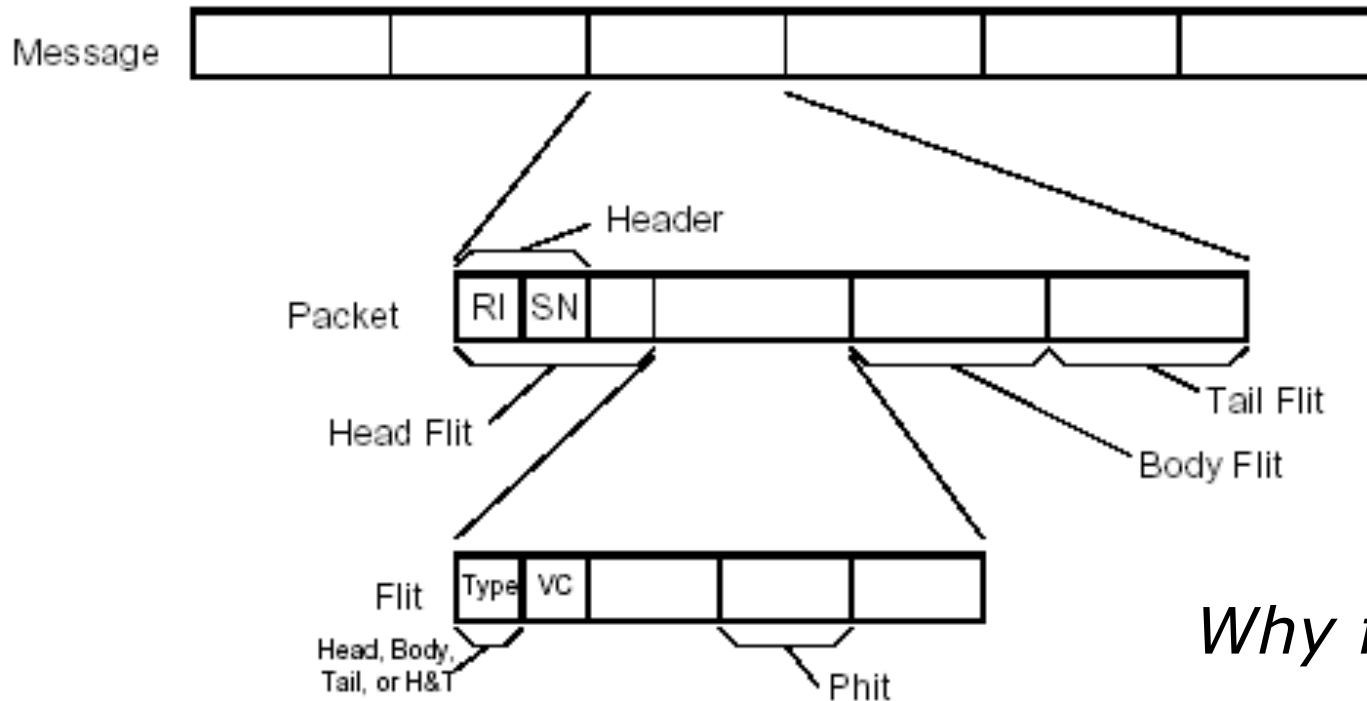
---



- $d$ -dimensional array
  - $n = k_{d-1} \times \dots \times k_0$  nodes
  - described by  $d$ -vector of coordinates  $(i_{d-1}, \dots, i_0)$
- $d$ -dimensional  $k$ -ary mesh:  $N = k^d$ 
  - $k = \sqrt[d]{N}$
  - described by  $d$ -vector of radix  $k$  coordinate
- $d$ -dimensional  $k$ -ary torus (or  $k$ -ary  $d$ -cube)

# Routing & Flow Control Overview

# Messages, Packets, Flits, Phits



*Why flits?*

For variable  
packet sizes

*Packet*: Basic unit of routing and sequencing

- Limited size (e.g. 64 bits – 64 KB)

*Flit* (flow control digit): Basic unit of bandwidth/storage allocation

- All flits in packet follow the same path

*Phit* (physical transfer digit): data transferred in single clock

# Routing vs Flow Control

---

- Routing algorithm chooses path that packets should follow to get from source to destination
- Flow control schemes allocate resources (buffers, links, control state) to packets traversing the network
- Our approach: Bottom-up
  - Today: Flow control, assuming routes are set
  - Next lecture: Routing algorithms

# Properties of Routing Algorithms

---

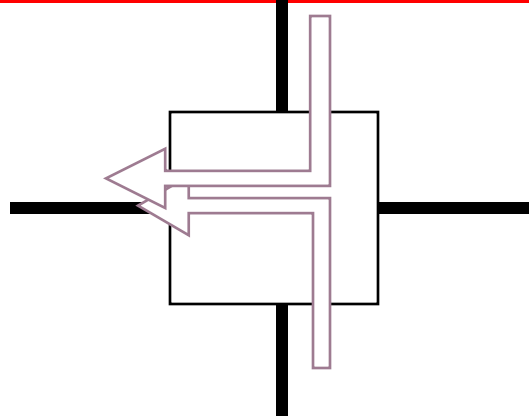
- **Deterministic/Oblivious**
  - Route determined by (source, dest), not intermediate state (i.e. traffic)
- **Adaptive**
  - Route influenced by traffic along the way
- **Minimal**
  - Only selects shortest paths
- **Deadlock-free**
  - No traffic pattern can lead to a situation where no packets move forward

*(more in next lecture)*

# Flow Control

# Contention

---



- Two packets trying to use the same link at the same time
  - Limited or no buffering
- Problem arises because we are sharing resources
  - Sharing bandwidth and buffers



# Flow Control Protocols

---

- **Bufferless**
  - Circuit switching
  - Dropping
  - Misrouting
- **Buffered**
  - Store-and-forward
  - Virtual cut-through
  - Wormhole
  - Virtual-channel

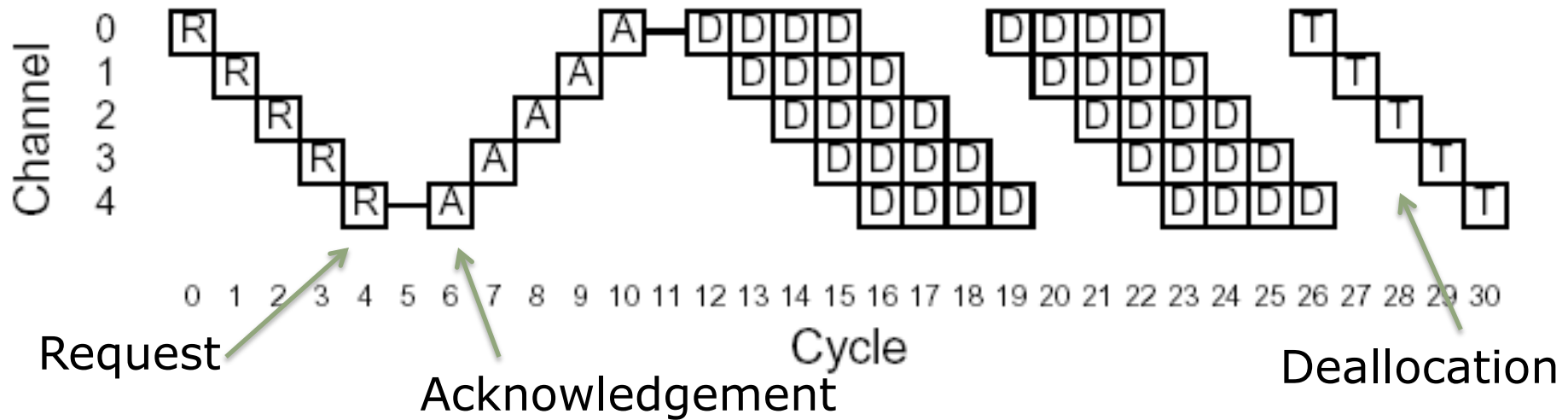


# Circuit Switching

---

- Form a circuit from source to dest
- Probe to set up path through network
- Reserve all links
- Data sent through links
  
- Bufferless

# Time-space View: Circuit Switching

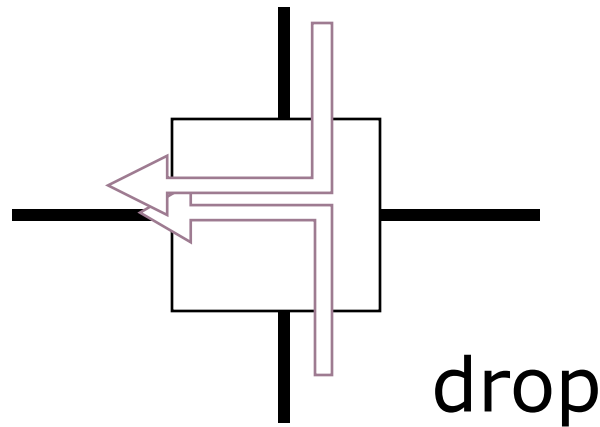


- *Why is this good?* Simple to implement
- *Why is it not?* Wasteful, 3x latency for short packets

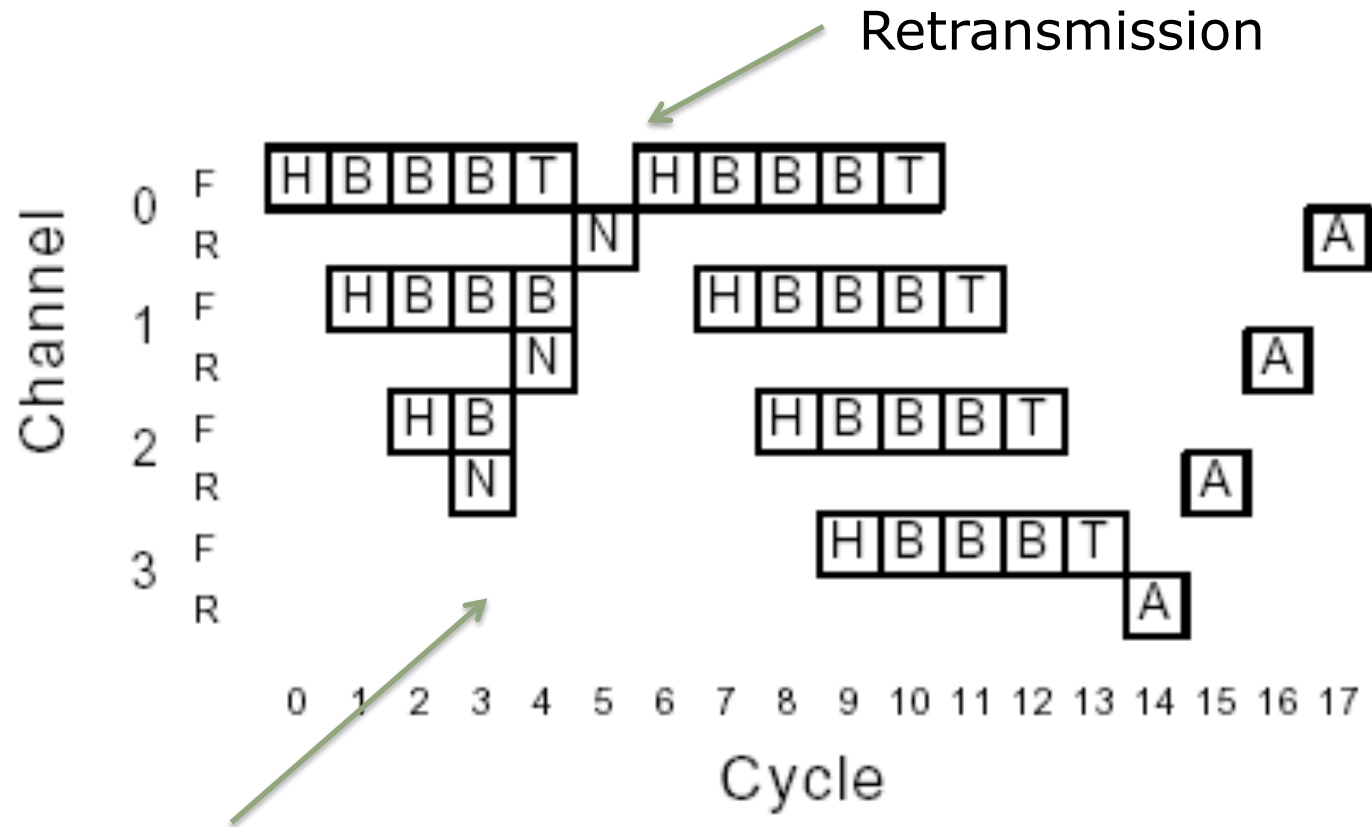
# Speculative Flow Control: Dropping

---

- If two things arrive and I don't have resources, drop one of them
- Flow control protocol on the Internet



# Time-space Diagram: Dropping



Unable to allocate channel 3

*Disadvantages?*

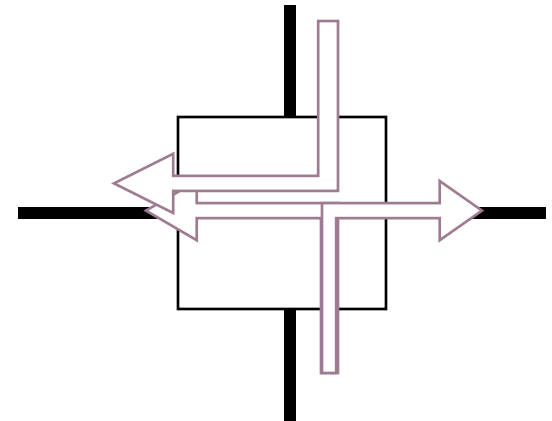
Poor tradeoff of traffic  
and buffering

# Less Simple Flow Control: Misrouting

---

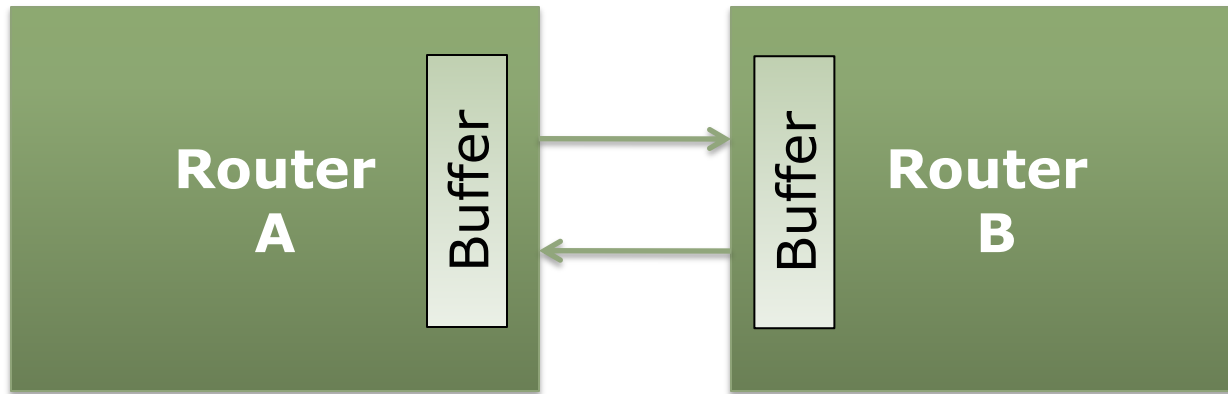
- If only one message can enter the network at each node, and one message can exit the network at each node, the network can never be congested. Right?  
**Wrong! Multiple hops cause congestion**

- Philosophy behind misrouting: intentionally route away from congestion
- No need for buffering
- Problems? **Livelock: need to guarantee that progress is made**



# Buffered Routing

---



- Link-level flow control:
  - Given that you can't drop packets, how to manage the buffers?  
When can you send stuff forward, when not?
- Metrics of interest:
  - Throughput/Latency
  - Buffer utilization (turnaround time)

# Techniques for link backpressure

---

- Naïve stall-based (on/off):
  - Can source send or not?
- Sophisticated stall-based (credit-based):
  - How many flits can be sent to the next node?
- Speculative (ack/nack):
  - Guess can always send, but keep copy
  - Resolve if send was successful (ack/nack)
    - On ack – drop copy
    - On nack - resend



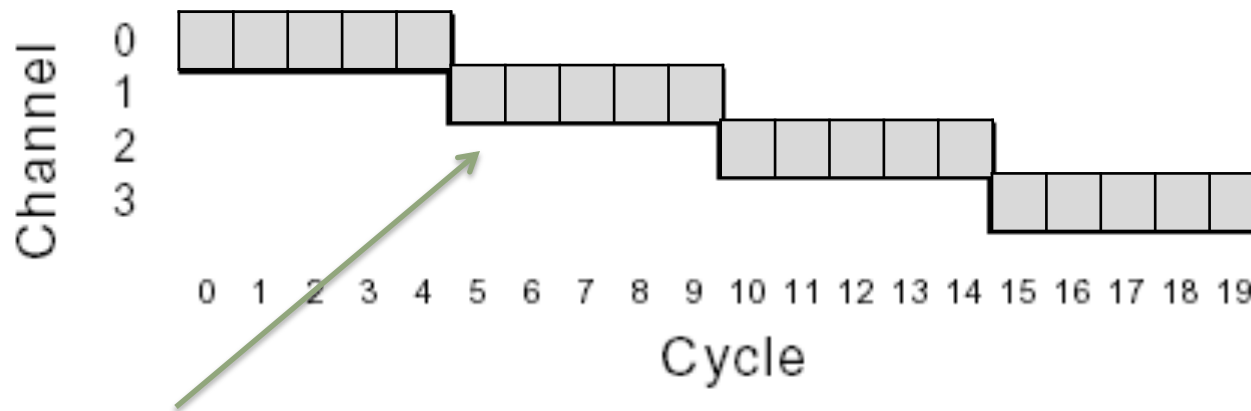
# Store-and-Forward (packet-based, no flits)

---

- **Strategy:**
  - Make intermediate stops and wait until the entire packet has arrived before you move on
- **Advantage:**
  - Other packets can use intermediate links

# Time-space View: Store-and-Forward

---



Could be allocated at a much later time without packet dropping

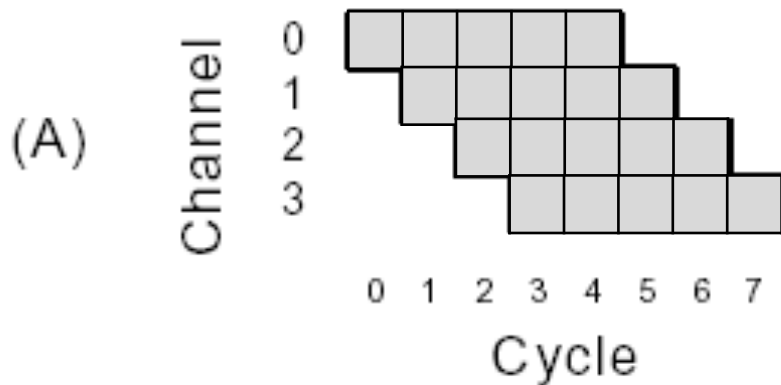
- Buffering allows packet to wait for channel
- *Drawback?* **Serialization latency experienced at each hop/channel**

# Virtual Cut-through (packet-based)

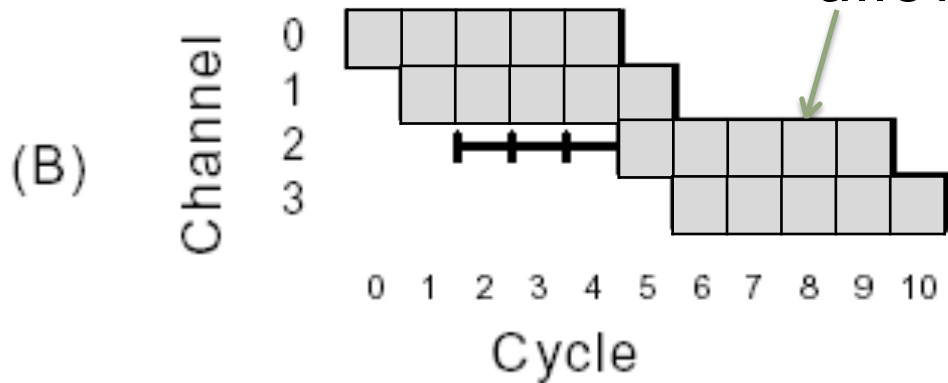
---

- Why wait till entire message has arrived at each intermediate stop?
- The head flit of the packet can dash off first
- When the head gets blocked, whole packet gets blocked at one intermediate node
- Used in Alpha 21364

# Time-space View: Virtual Cut-through



- *Advantages?*  
Lower latency



- *Disadvantages?*  
Buffers allocated in packets  
→ large buffers & low utilization  
Channels allocated in packets  
→ unfairness & low utilization

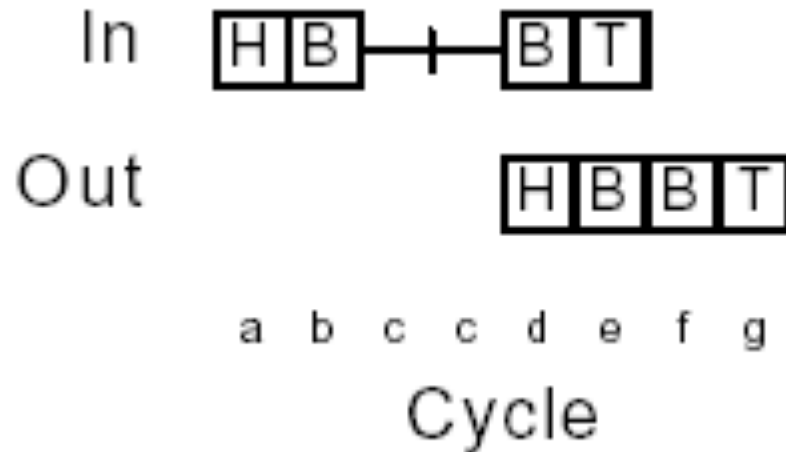
# Flit-Buffer Flow Control: Wormhole

---

- When a packet blocks, just block wherever the pieces (flits) of the message are at that time.
- Operates like cut-through but with channel and buffers allocated to flits rather than packets
  - Channel state (virtual channel) allocated to **packet** so body flits can follow head flit

# Time-space View: Wormhole

---



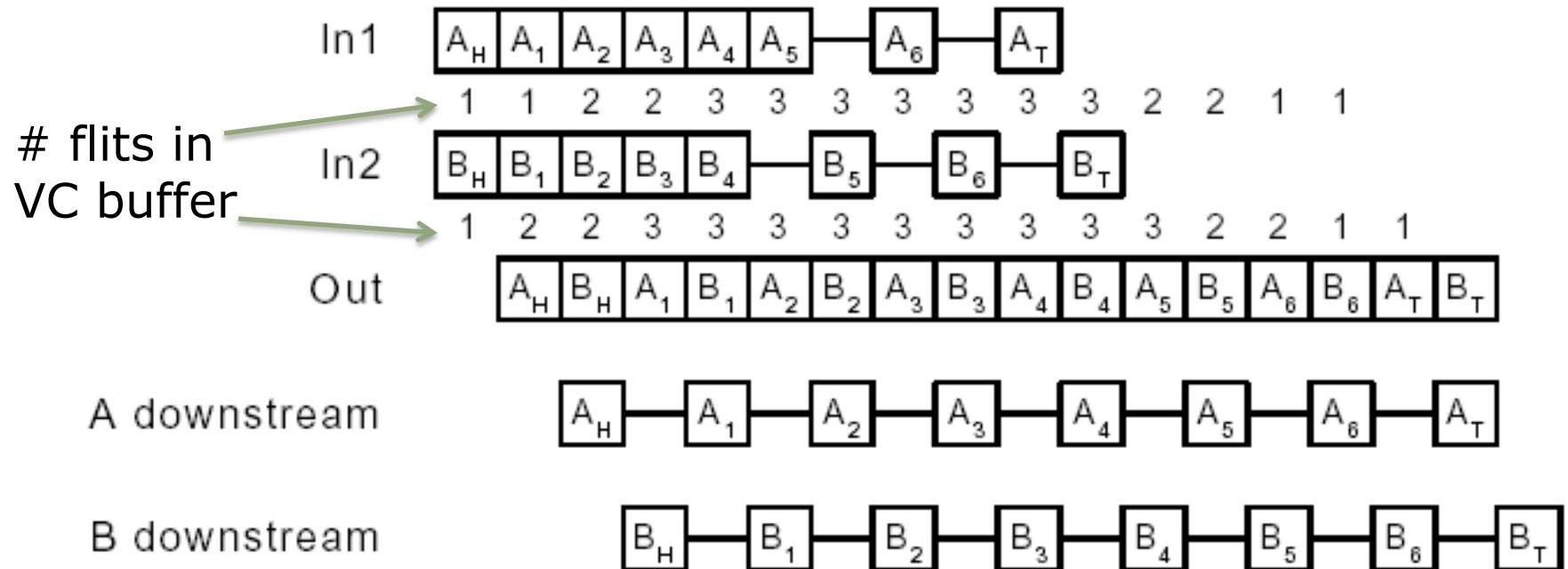
- *Advantages?* Smaller amount of buffer space required
- *Disadvantages?* May block a channel mid-packet, another packet cannot use bandwidth

# Virtual-Channel (VC) Flow Control

---

- When a message blocks, instead of holding on to links so others can't use them, hold on to **virtual** links
- Multiple queues in buffer storage
  - Like lanes on the highway
- Virtual channel can be thought of as channel state and flit buffers

# Time-space View: Virtual-Channel



- *Advantages?*
- *Disadvantages?*

Significantly reduces blocking

More complex router,  
fair VC allocation required



*Thank you!*

*Next Lecture:  
Router (Switch) Microarchitecture  
Routing Algorithms*