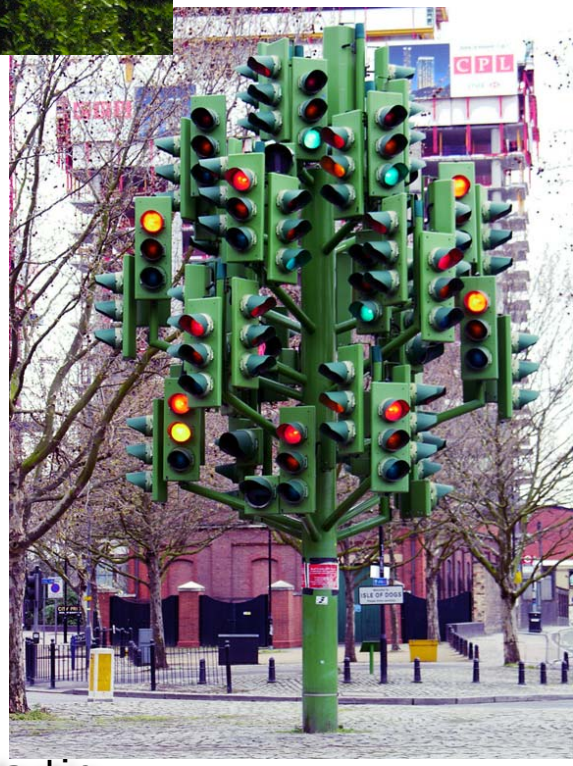
EECS 570

Interconnects: Routing II & Flow Control



Fall 2025

Prof. Satish Narayanasamy

http://www.eecs.umich.edu/courses/eecs570/

Slides developed in part by Profs. Adve, Falsafi, Hill, Lebeck, Martin, Narayanasamy, Nowatzyk, Reinhardt, Singh, Smith, Torrellas and Wenisch. Special acknowledgement to Prof. Jerger of U. Toronto.

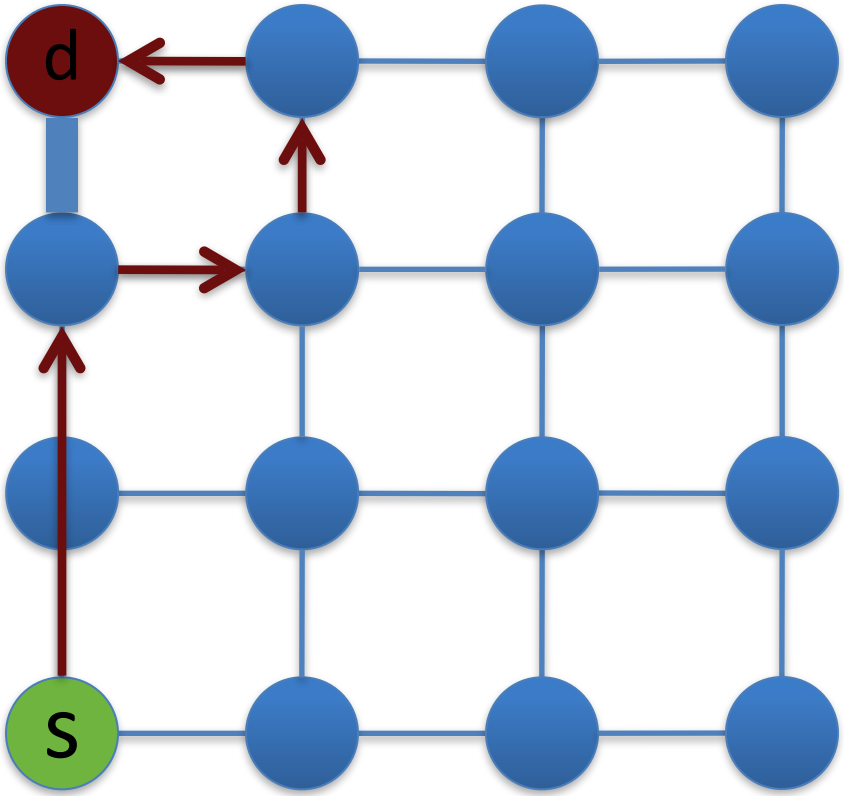
Non-minimal adaptive

- Fully adaptive
- Not restricted to take shortest path

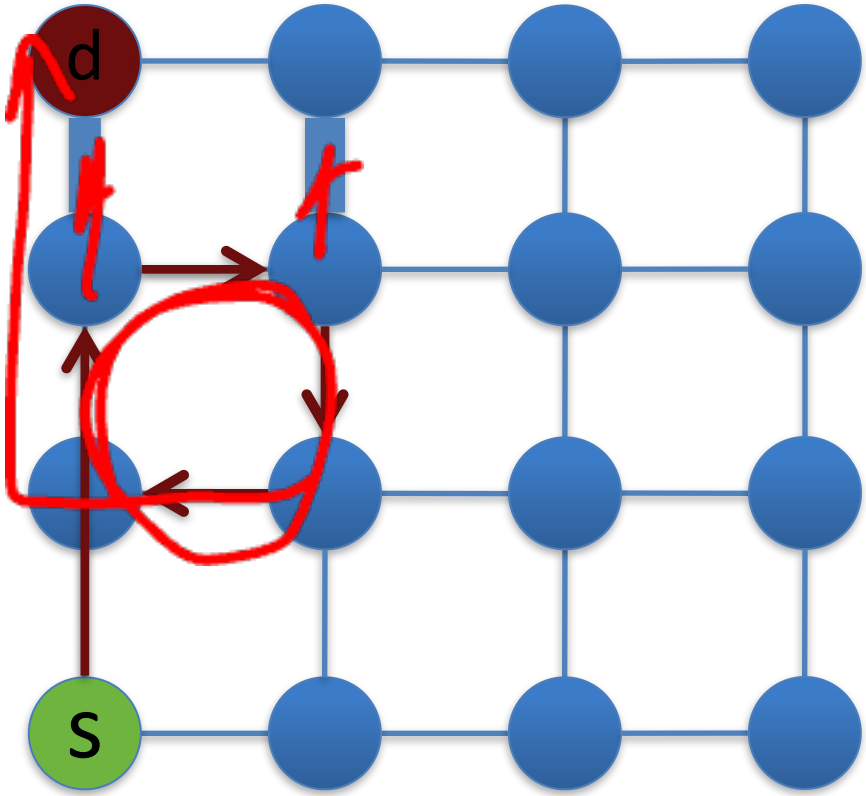
- Misrouting: directing packet along non-productive channel
  - Priority given to productive output
  - Some algorithms forbid U-turns

- Livelock potential: traversing network without ever reaching destination
  - Mechanism to guarantee forward progress
    - Limit number of misroutings

Non-minimal routing example

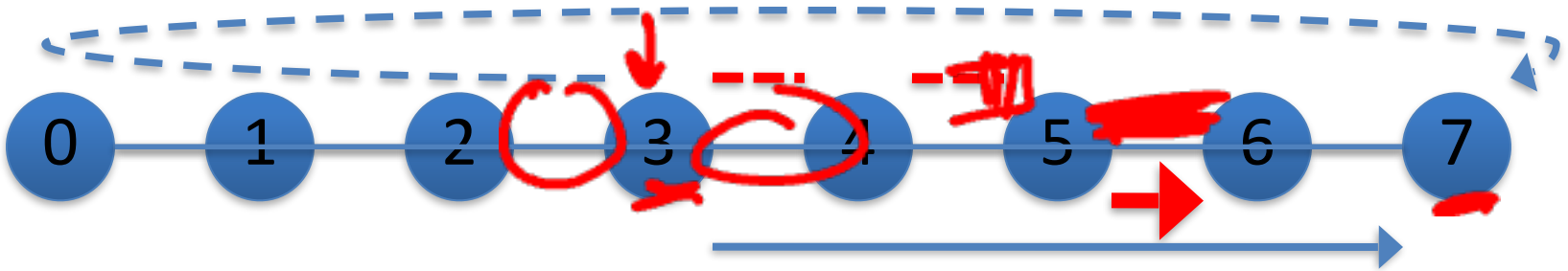


- Longer path with potentially lower latency



- Livelock: continue routing in cycle

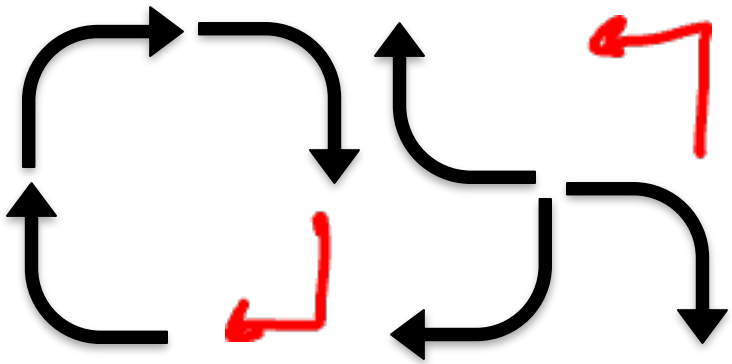
Adaptive Routing Example



- Should 3 route clockwise or counterclockwise to 7?
  - If 5 is using all the capacity of link 5 → 6…
  - …queue at node 5 will sense contention but not at node 3
- Backpressure: allows nodes to indirectly sense congestion
  - Queue in one node fills up, it will stop receiving flits
  - Previous queue will fill up
- If each queue holds 4 packets
  - 3 will send 8 packets before sensing congestion

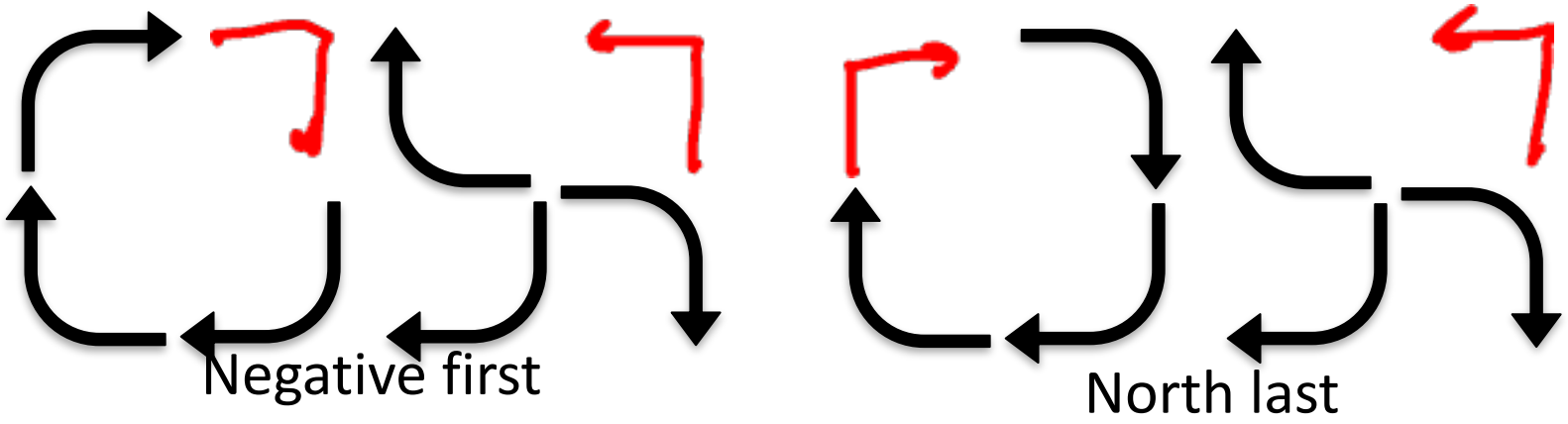
Adaptive Routing: Turn Model

- DOR eliminates 4 turns
  - N to E, N to W, S to E, S to W
  - No adaptivity
- Some adaptivity by removing 2 of 8 turns
  - Remains deadlock free (like DOR)
- West first
  - Eliminates S to W and N to W



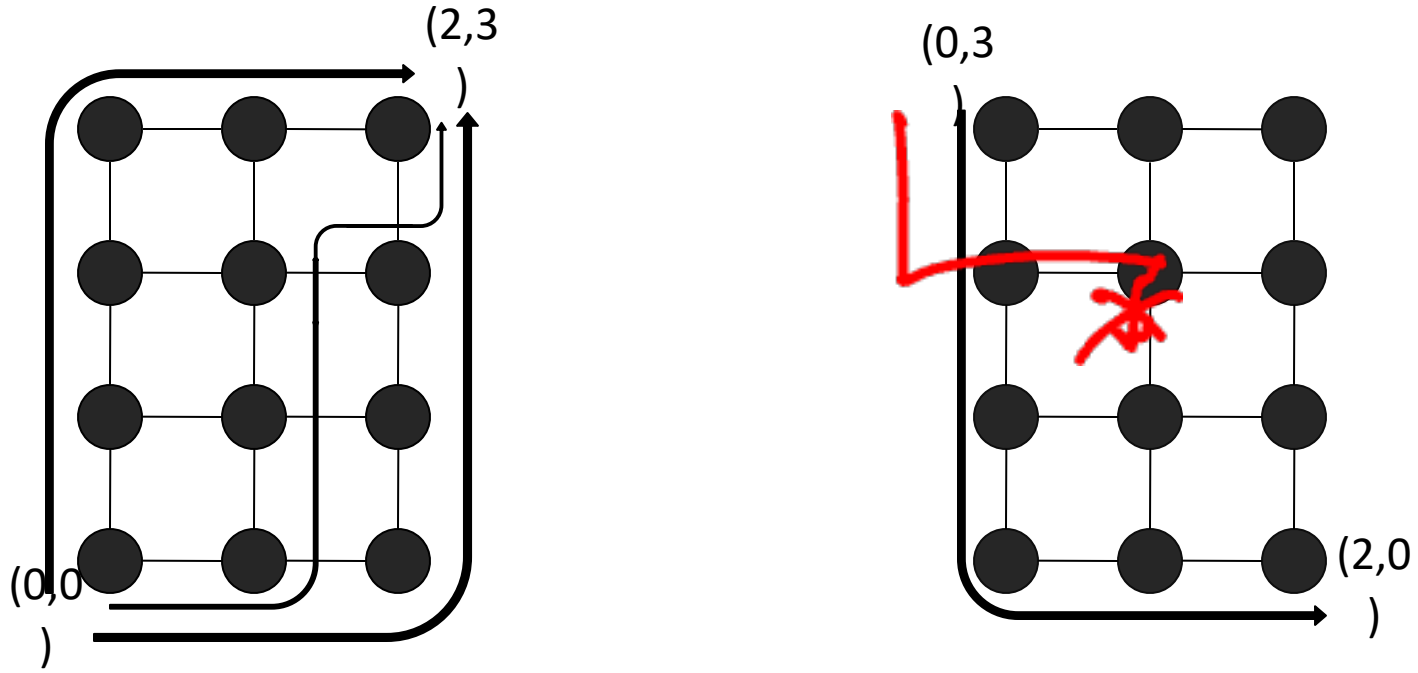
West first

Turn Model Routing



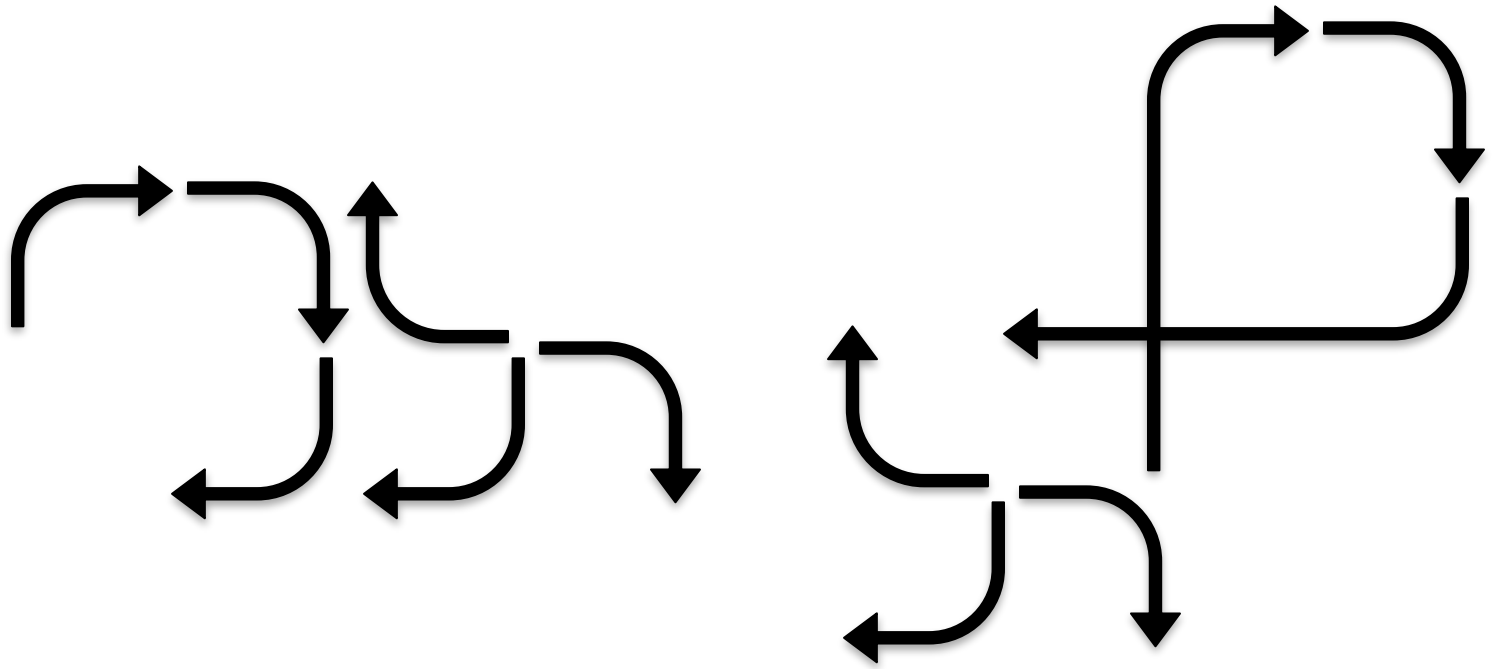
- Negative first
  - Eliminates E to S and N to W
- North last
  - Eliminates N to E and N to W
- Odd-Even
  - Eliminates 2 turns depending on if current node is in odd or even col.
    - Even column: E to N and N to W
    - Odd column: E to S and S to W
  - Deadlock free if 180 turns are disallowed
  - Better adaptivity

Negative-First Routing Example



- Limited or no adaptivity for certain source-destination pairs

Turn Model Routing Deadlock



- What about eliminating turns NW and WN?
- Not a valid turn elimination
  - Resource cycle results

Adaptive Routing and Deadlock

- Option 1: Eliminate turns that lead to deadlock
  - Limits flexibility

- Option 2: Allow all turns
  - Give more flexibility
  - Must use other mechanism to prevent deadlock
  - Rely on flow control (later)
    - Escape virtual channels

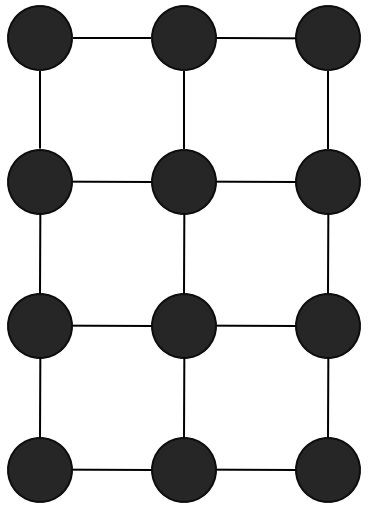
Routing Algorithm Implementation

Routing Implementation

- Source tables
  - Entire route specified at source
  - Avoids per-hop routing latency
  - Unable to adapt dynamically to network conditions
  - Can specify multiple routes per destination
    - Give fault tolerance and load balance
  - Support reconfiguration

Source Table Routing

Destination	Route 1	Route 2
00	X	X
10	EX	EX
20	EEX	EEX
01	NX	NX
11	NEX	ENX
21	NEEX	ENEX
02	NNX	NNX
12	ENNX	NNEX
22	EENNX	NNEEX
03	NNNX	NNNX
13	NENNX	ENNNX
23	EENNNX	NNNEEX



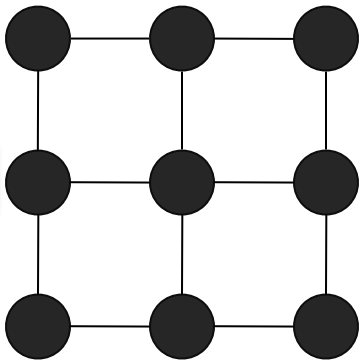
- Arbitrary length paths: storage overhead and packet overhead

Node Tables

- Store only next direction at each node
- Smaller tables than source routing
- Adds per-hop routing latency
- Can adapt to network conditions
  - Specify multiple possible outputs per destination
  - Select randomly to improve load balancing

Node Table Routing

	To								
From	00	01	02	10	11	12	20	21	22
00	X | -	N | -	N | -	E | -	E | N	E | N	E | -	E | N	E | N
01	S | -	X | -	N | -	E | S	E | -	E | N	E | S	E | -	E | N
02	S | -	S| -	X | -	E | S	E | S	E | -	E | S	E | S	E | -
10	W|-	W|-	W|-	X | -	N | -	N | -	E | -	E | N	E | N
11	W|-	W|-	W|-	S | -	X | -	N | -	E | S	E | -	E | N
12	W|-	W|-	W|-	S | -	S | -	X | -	E | S	E | S	E | -
20	W|-	W|-	W|-	W|-	W|-	W|-	X | -	N | -	N | -
21	W|-	W|-	W|-	W|-	W|-	W|-	S | -	X | -	N | -
22	W|-	W|-	W|-	W|-	W|-	W|-	S | -	S | -	X | -



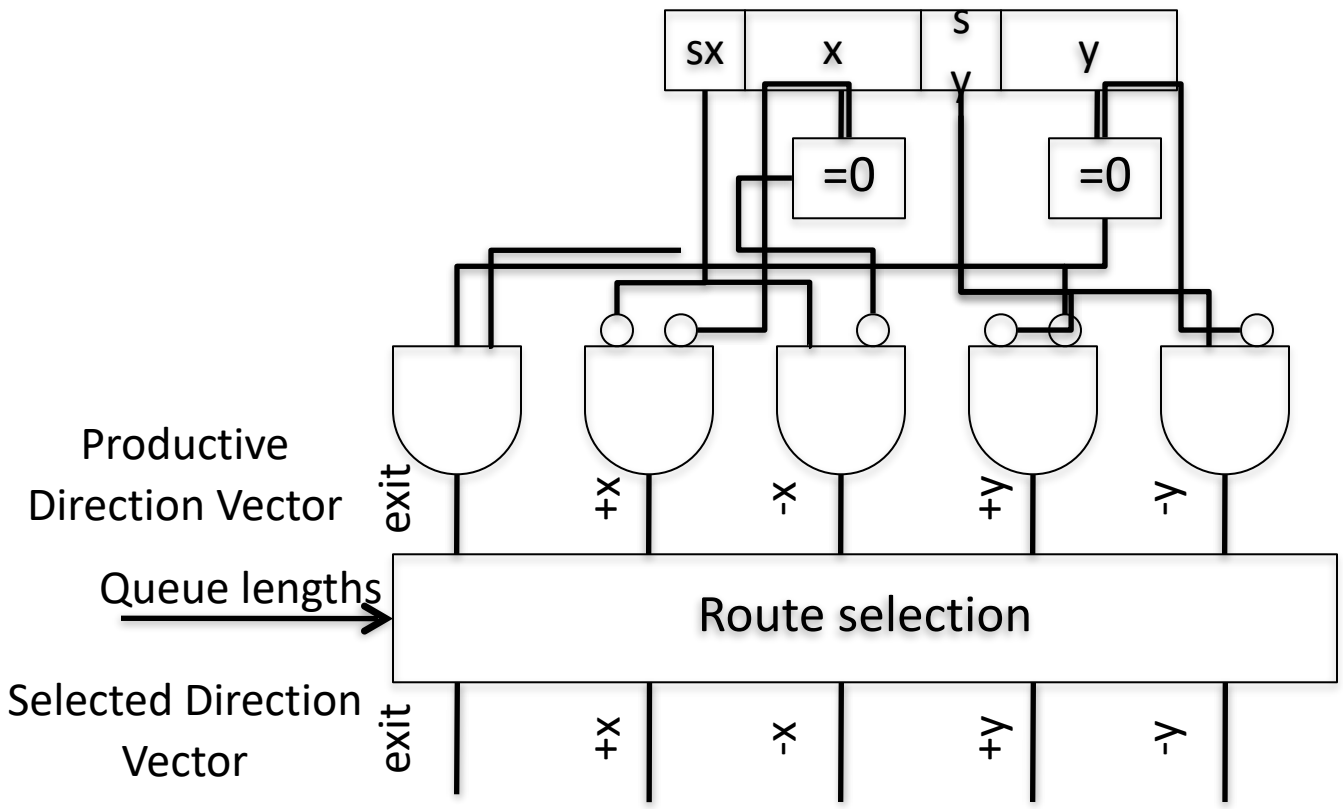
- Implements West-First Routing
- Each node would have 1 row of table
  - Max two possible output ports

Implementation

- Combinational circuits can be used
  - Simple (e.g. DOR): low router overhead
  - Specific to one topology and one routing algorithm
    - Limits fault tolerance

- OTOH, tables can be updated to reflect new configuration, network faults, etc

Circuit Based



- Next hop based on buffer occupancies in a 2D mesh
- Or could implement simple DOR
- Fixed w.r.t. topology

Routing Algorithms: Implementation

Routing Algorithm	Source Routing	Combinational	Node Table
Deterministic			
DOR	Yes	Yes	Yes
Oblivious			
Valiant's	Yes	Yes	Yes
Minimal	Yes	Yes	Yes
Adaptive	No	Yes	Yes

Routing: Irregular Topologies

- MPSoCs
  - Power and performance benefits from irregular/custom topologies
- Common routing implementations
  - Rely on source or node table routing
- Maintain deadlock freedom
  - Turn model may not be feasible
    - Limited connectivity

Routing Summary

- Latency paramount concern
  - Minimal routing most common for NoC
  - Non-minimal can avoid congestion and deliver low latency
- To date: NoC research favors DOR for simplicity and deadlock freedom
  - On-chip networks often lightly loaded
- Only covered unicast routing
  - Recent work on extending on-chip routing to support multicast

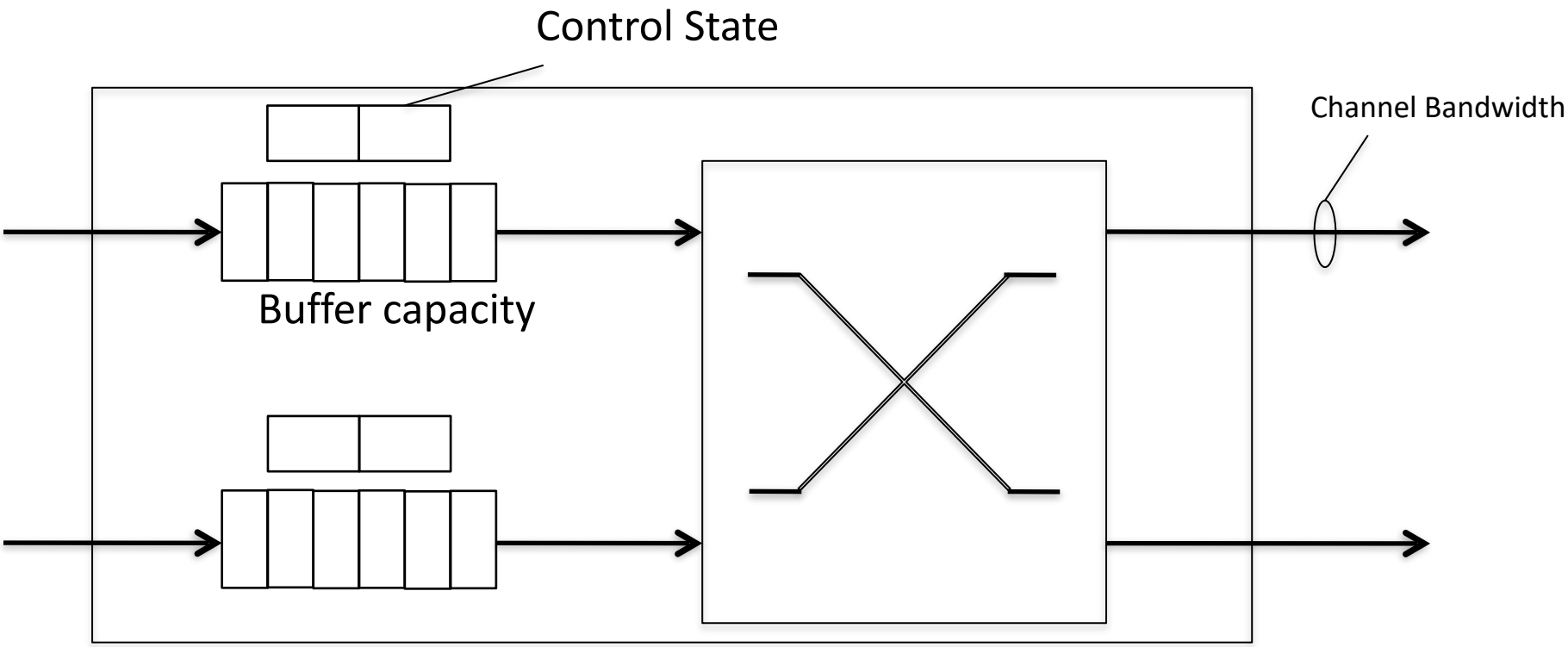
Topics to be covered

- Interfaces
- Topology
- Routing
- Flow Control
- Router Microarchitecture

Switching/Flow Control Overview

- Topology: determines **connectivity** of network
- Routing: determines **paths** through network
- Flow Control: determine **allocation** of resources to messages as they traverse network
  - Buffers and links
  - Significant impact on throughput and latency of network

Flow Control

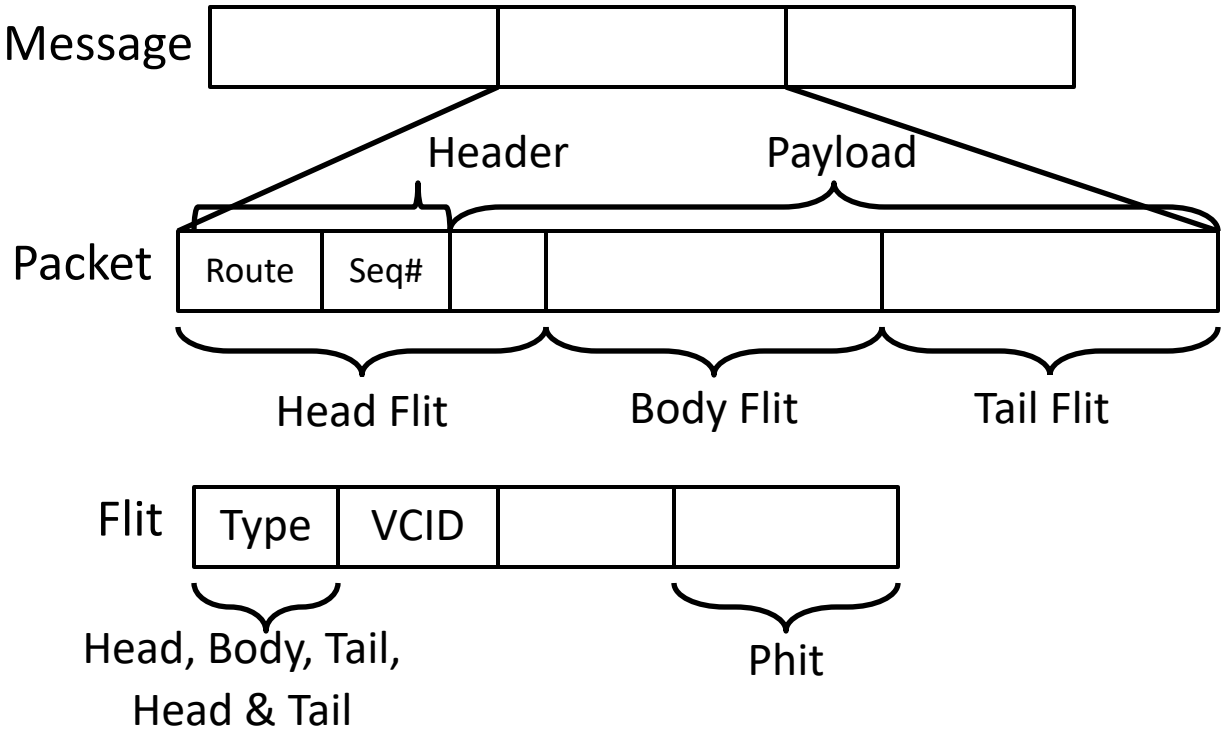


- Control state records:
  - allocation of channels and buffers to packets
  - current state of packet traversing node
- Channel bandwidth advances flits from this node to next
- Buffers hold flits waiting for channel bandwidth

Packets

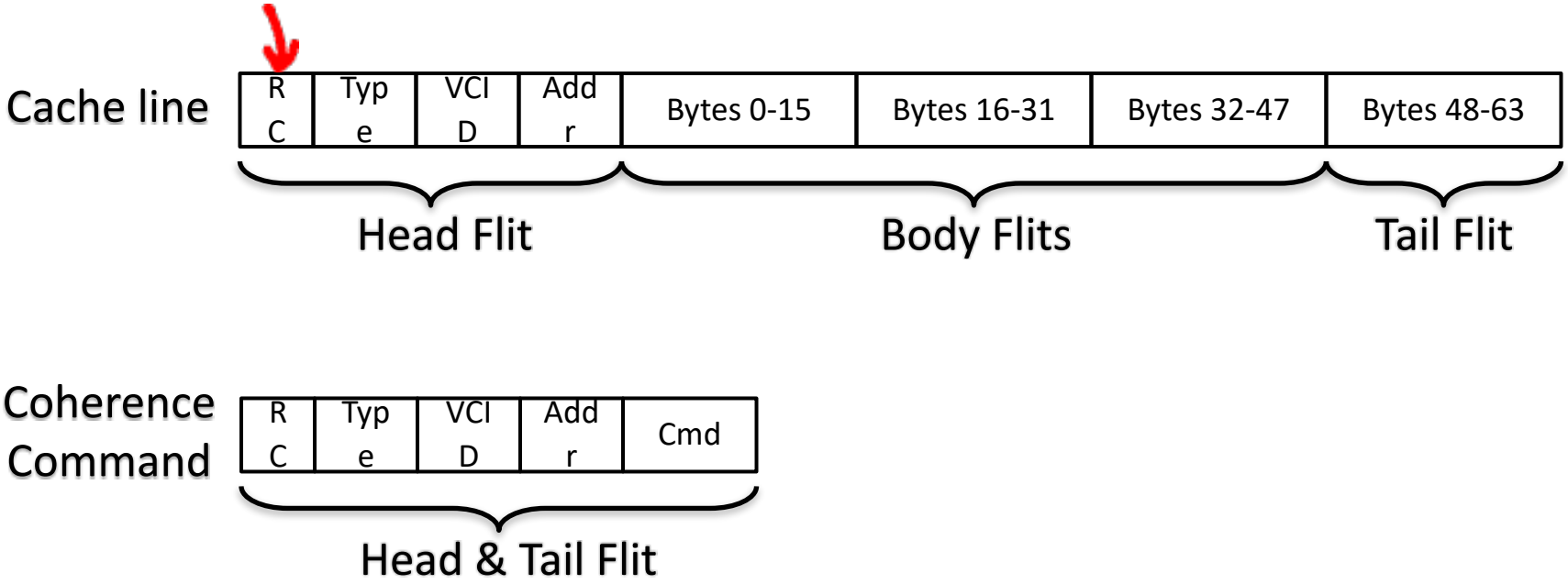
- Messages: composed of one or more packets
  - If message size is <= maximum packet size only one packet created
- Packets: composed of one or more flits
- Flit: flow control digit
- Phit: physical digit
  - Subdivides flit into chunks = to link width

Packets (2)



- Off-chip: channel width limited by **pins**
  - Requires phits
- On-chip: **abundant** wiring means phit size == flit size

Packets(3)



- Packet contains destination/route information
  - Flits may not → all flits of a packet must take same route

Types of Switching

Switching

- Different flow control techniques based on granularity
  - **Message-based**: allocation made at message granularity (circuit-switching)
  - **Packet-based**: allocation made to whole packets
  - **Flit-based**: allocation made on a flit-by-flit basis

Message-Based Flow Control

- Coarsest granularity

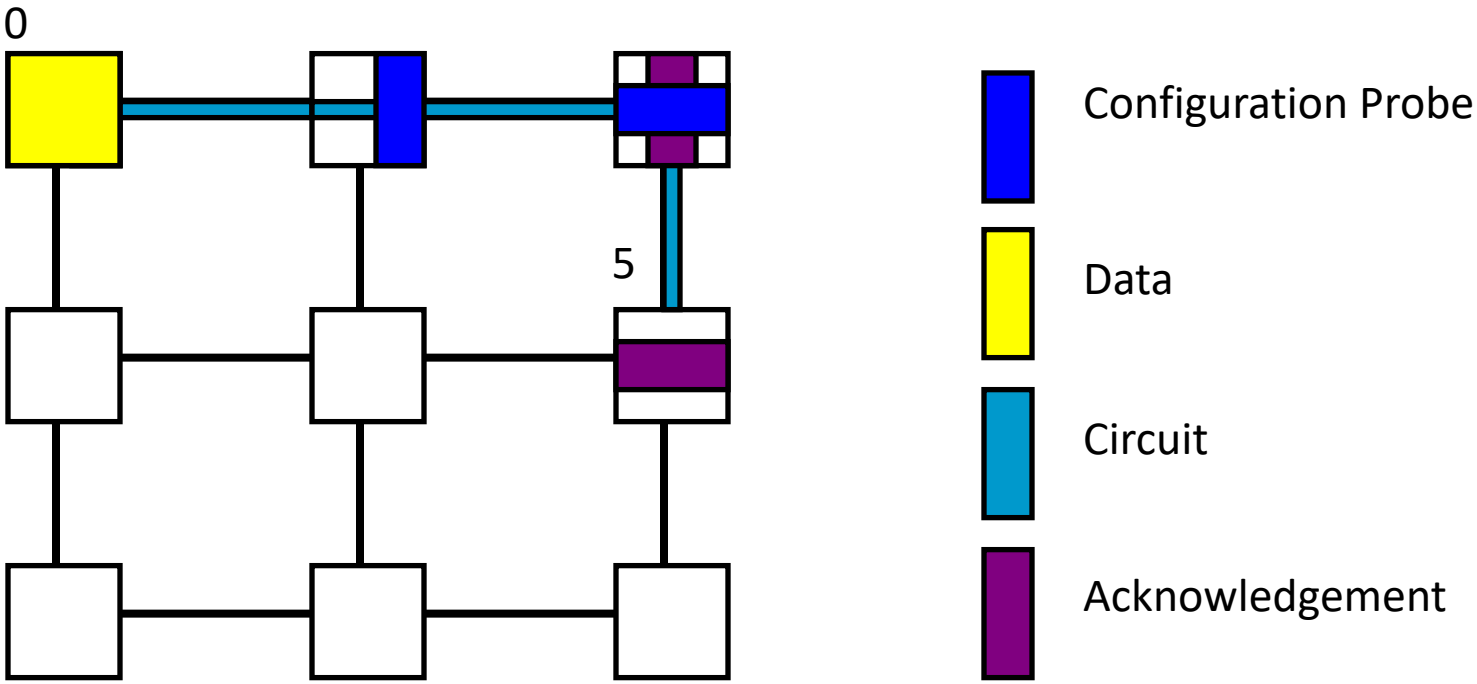
- Circuit-switching
  - **Pre-allocates** resources across **multiple** hops
    - Source to destination
    - Resources = links
    - Buffers are not necessary
  - Probe sent into network to reserve resources

Circuit Switching

- Once probe sets up circuit
  - Message does not need to perform any routing or allocation at each network hop
  - Good for transferring large amounts of data
    - Can amortize circuit setup cost by sending data with very low per-hop overheads

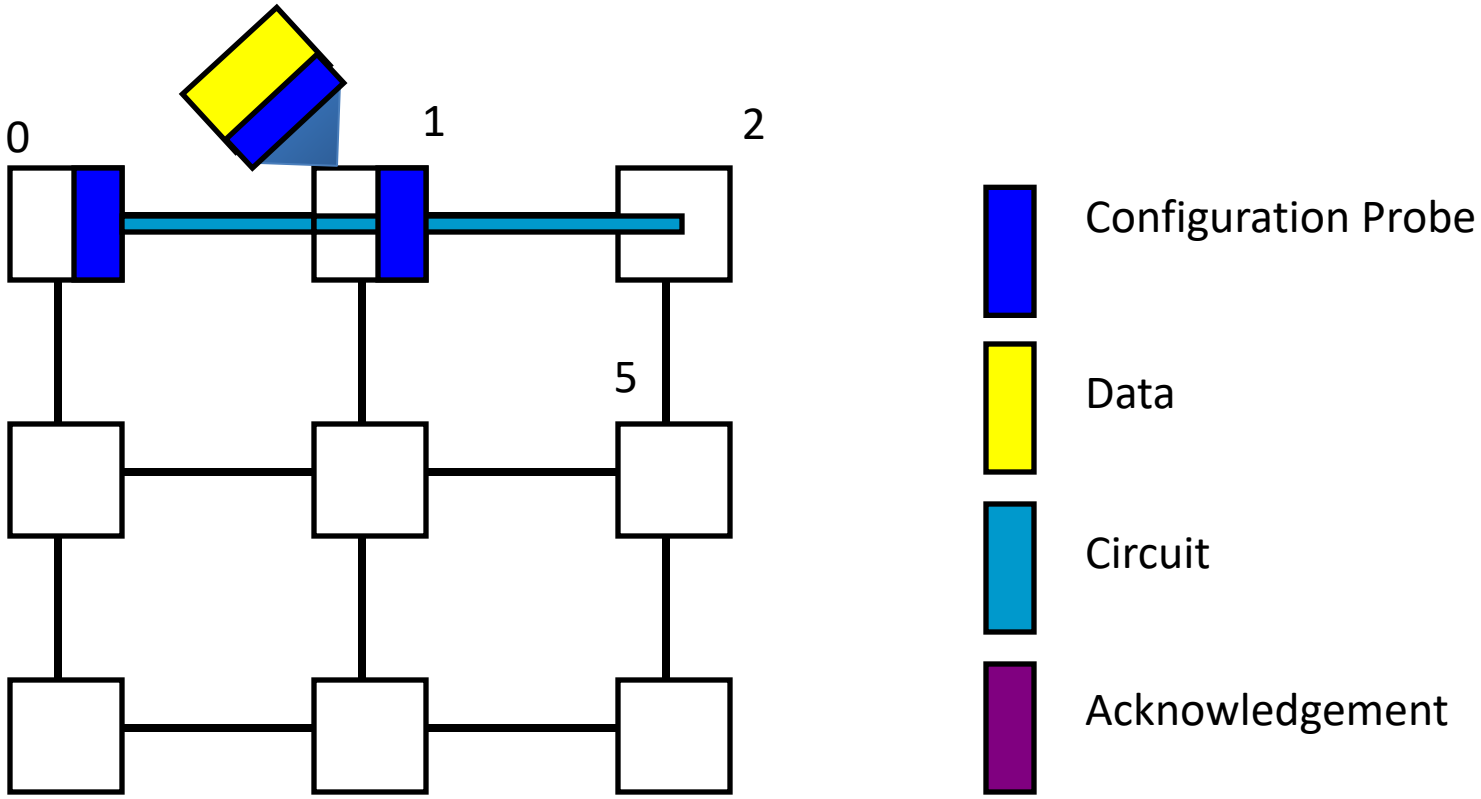
- No other message can use those resources until transfer is complete
  - Throughput can suffer due setup and hold time for network circuits
  - Links are idle until setup is complete

Circuit Switching Example



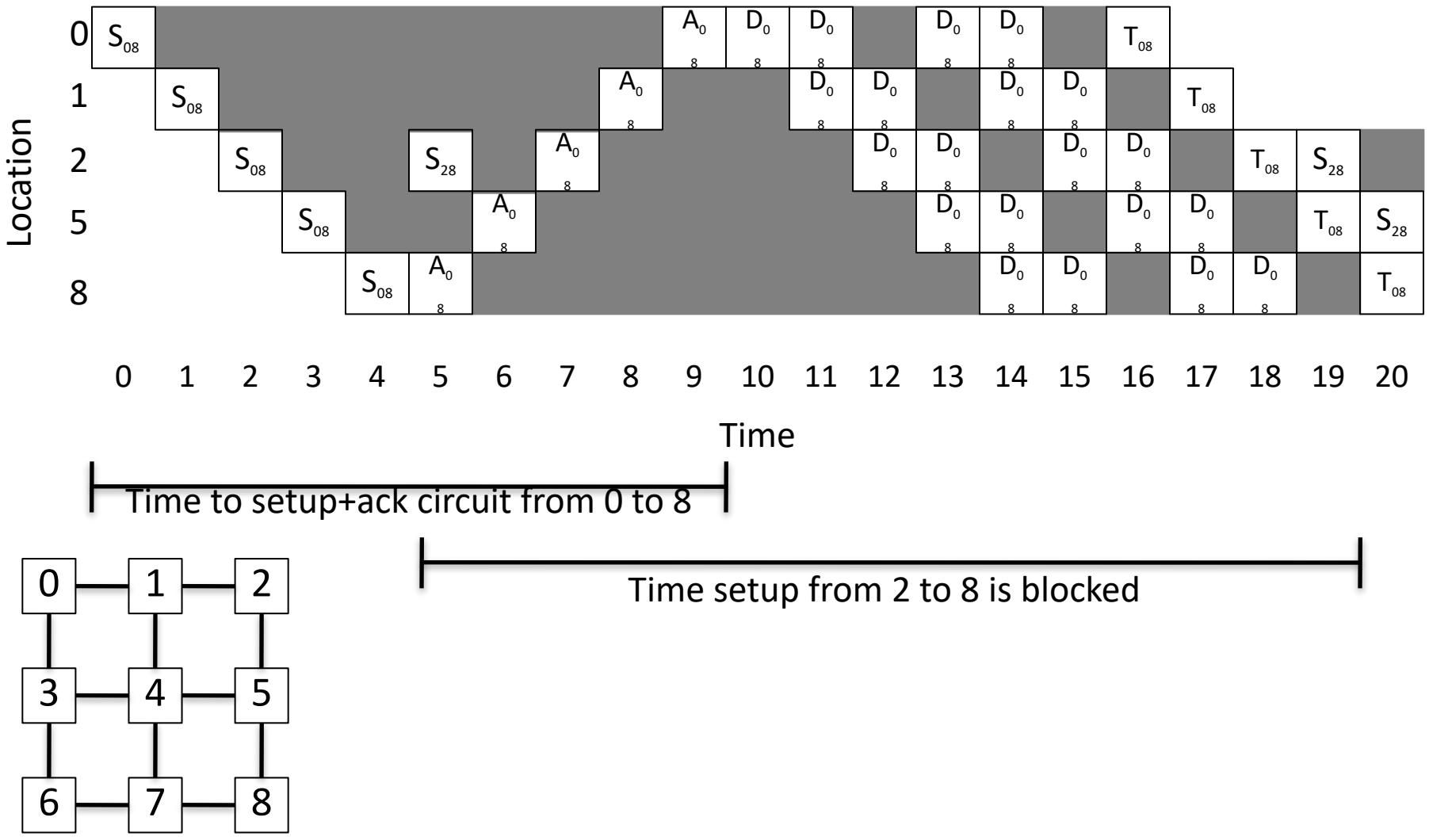
- Significant latency overhead prior to data transfer
  - Data transfer does not pay per-hop overhead for routing and allocation

Circuit Switching Example (2)



- When there is contention
  - Significant wait time
  - Message from 1 → 2 must wait

Time-Space Diagram: Circuit-Switching



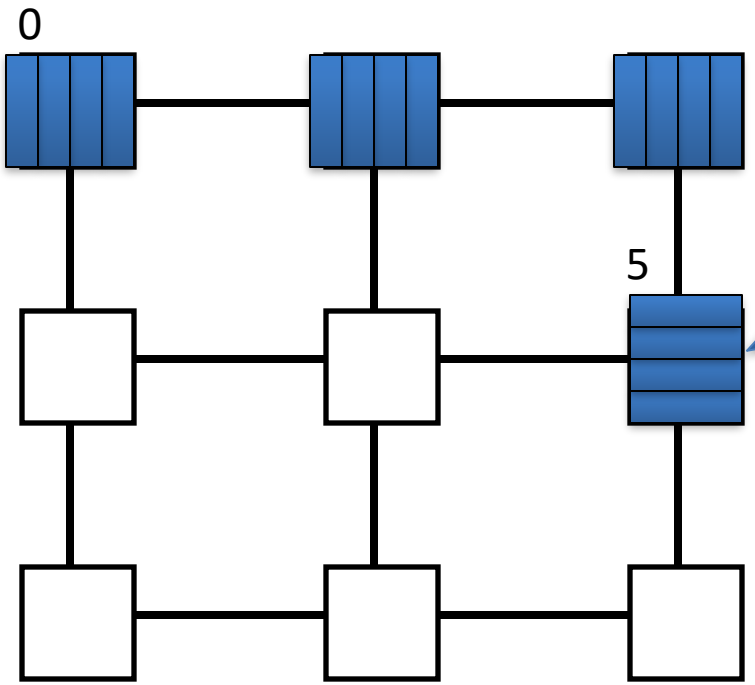
Packet-based Flow Control

- Break messages into packets
- **Interleave** packets on links
  - Better utilization
- Requires per-node **buffering** to store in-flight packets
- Two types of packet-based techniques

Store and Forward

- Links and buffers are allocated to **entire** packet
- Head flit **waits** at router until entire packet is received before being forwarded to the next hop
- Not suitable for on-chip
  - Requires buffering at each router to hold entire packet
    - Packet cannot traverse link until buffering allocated to entire packet
  - Incurs high latencies (pays **serialization** latency at each hop)
    - On-chip networks are usually delay-critical

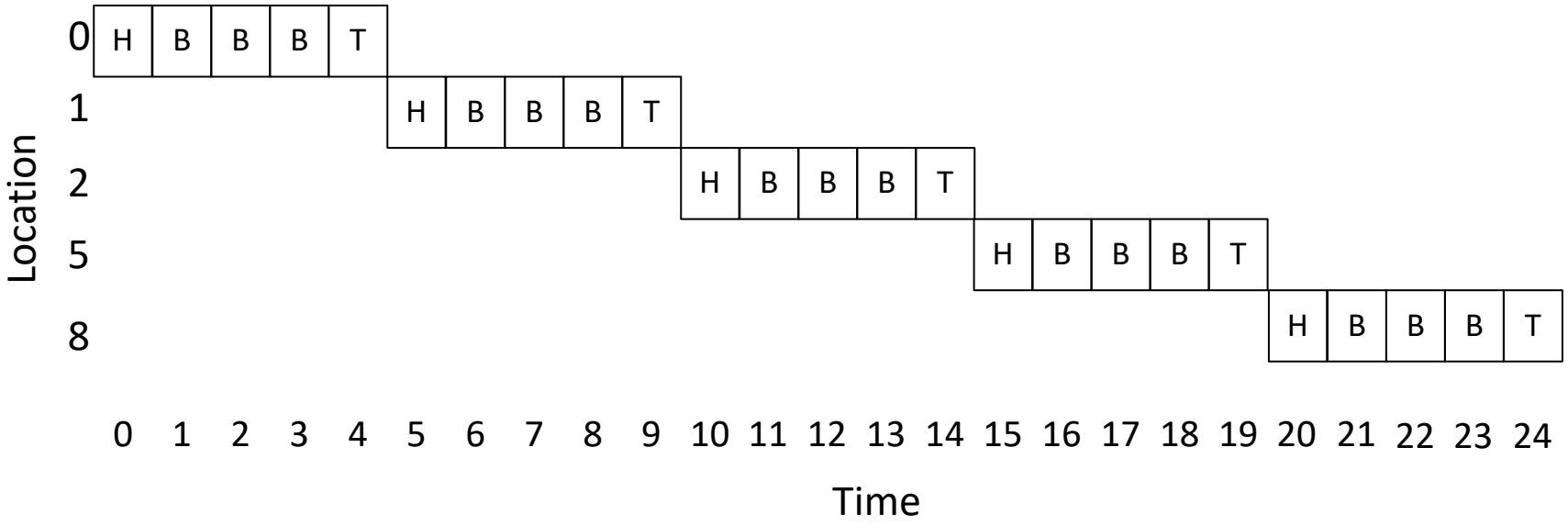
Store and Forward Example



Total delay = 4 cycles per hop x 3 hops = 12 cycles

- High per-hop latency
  - Serialization delay paid at each hop
- Larger buffering required

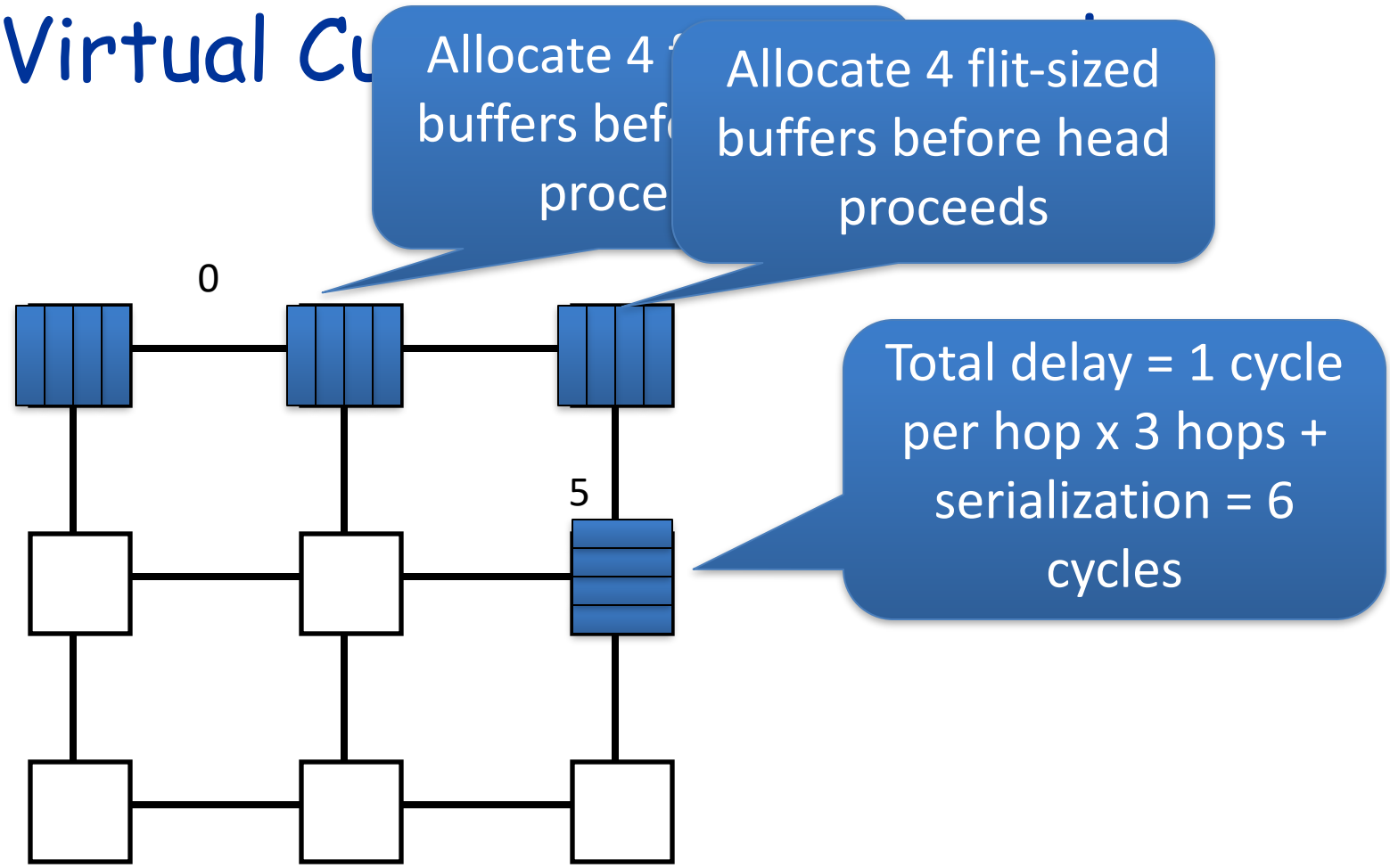
Time-Space Diagram: Store and Forward



Packet-based: Virtual Cut Through

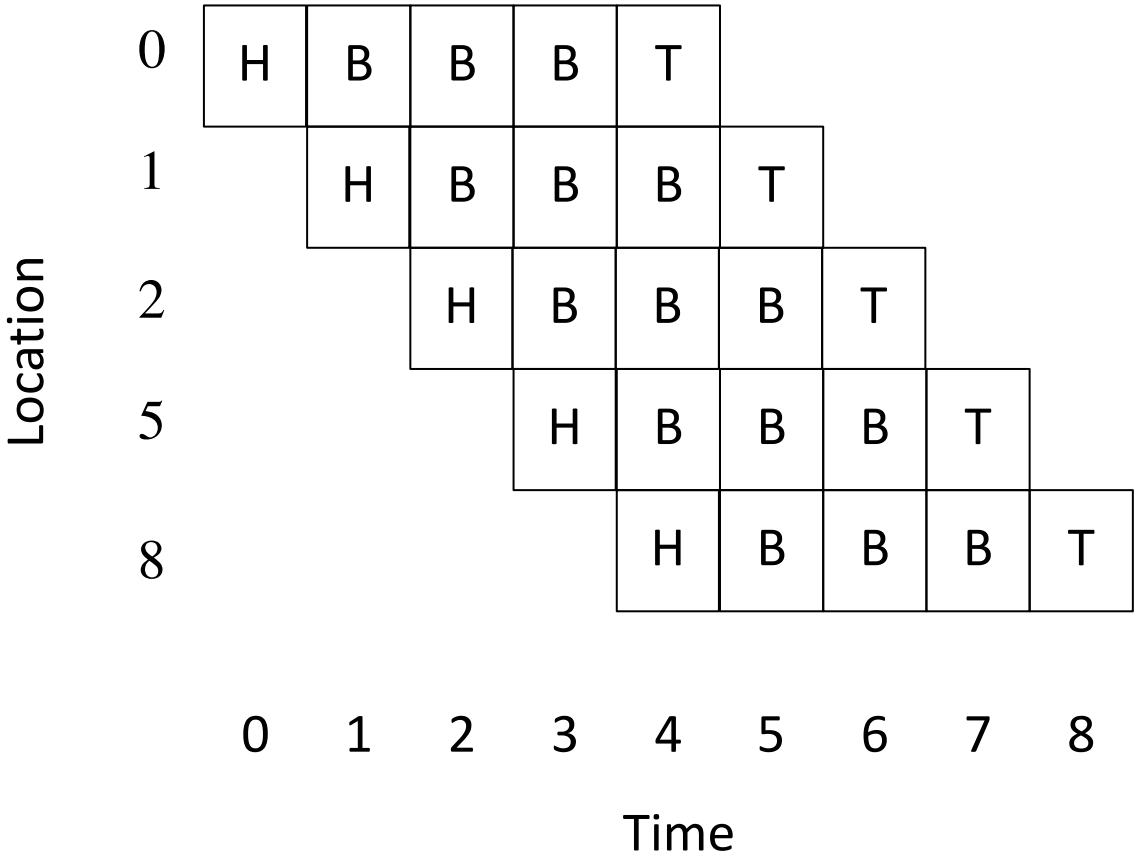
- Links and Buffers allocated to **entire** packets
- Flits can proceed to next hop before tail flit has been received by current router
  - But only if next router has enough buffer space for **entire** packet
- Reduces the latency significantly compared to SAF
- But still requires **large** buffers
  - Unsuitable for on-chip

Virtual Cu

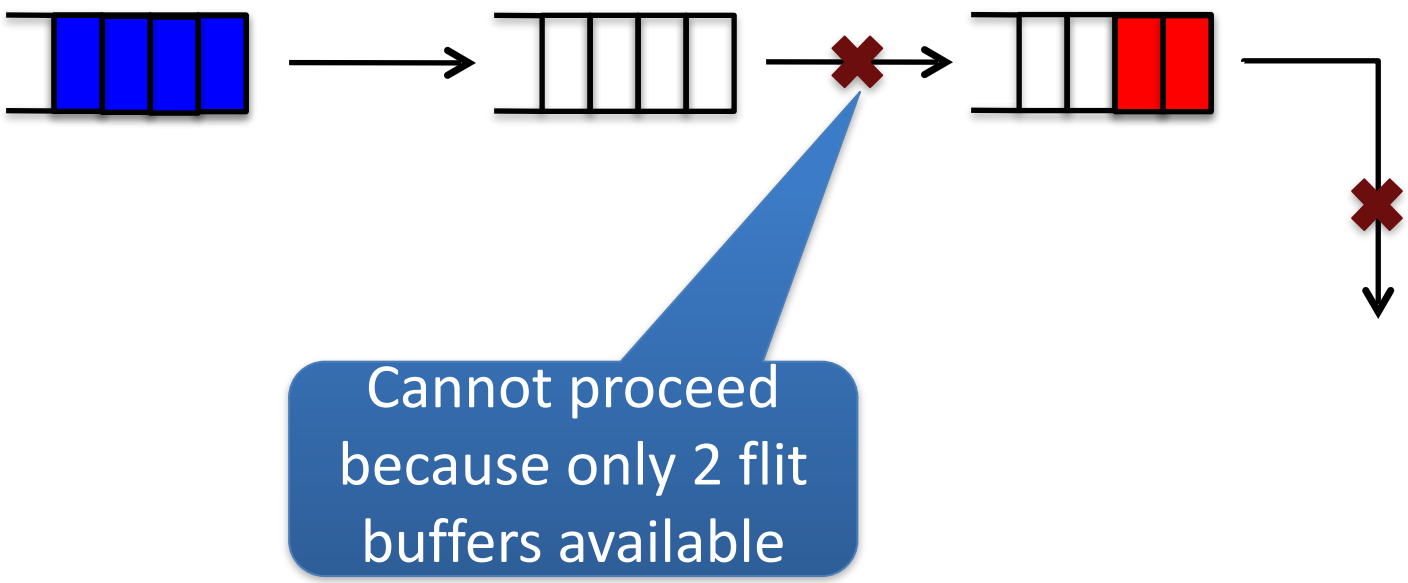


- Lower per-hop latency
- Large buffering required

Time-Space Diagram: VCT

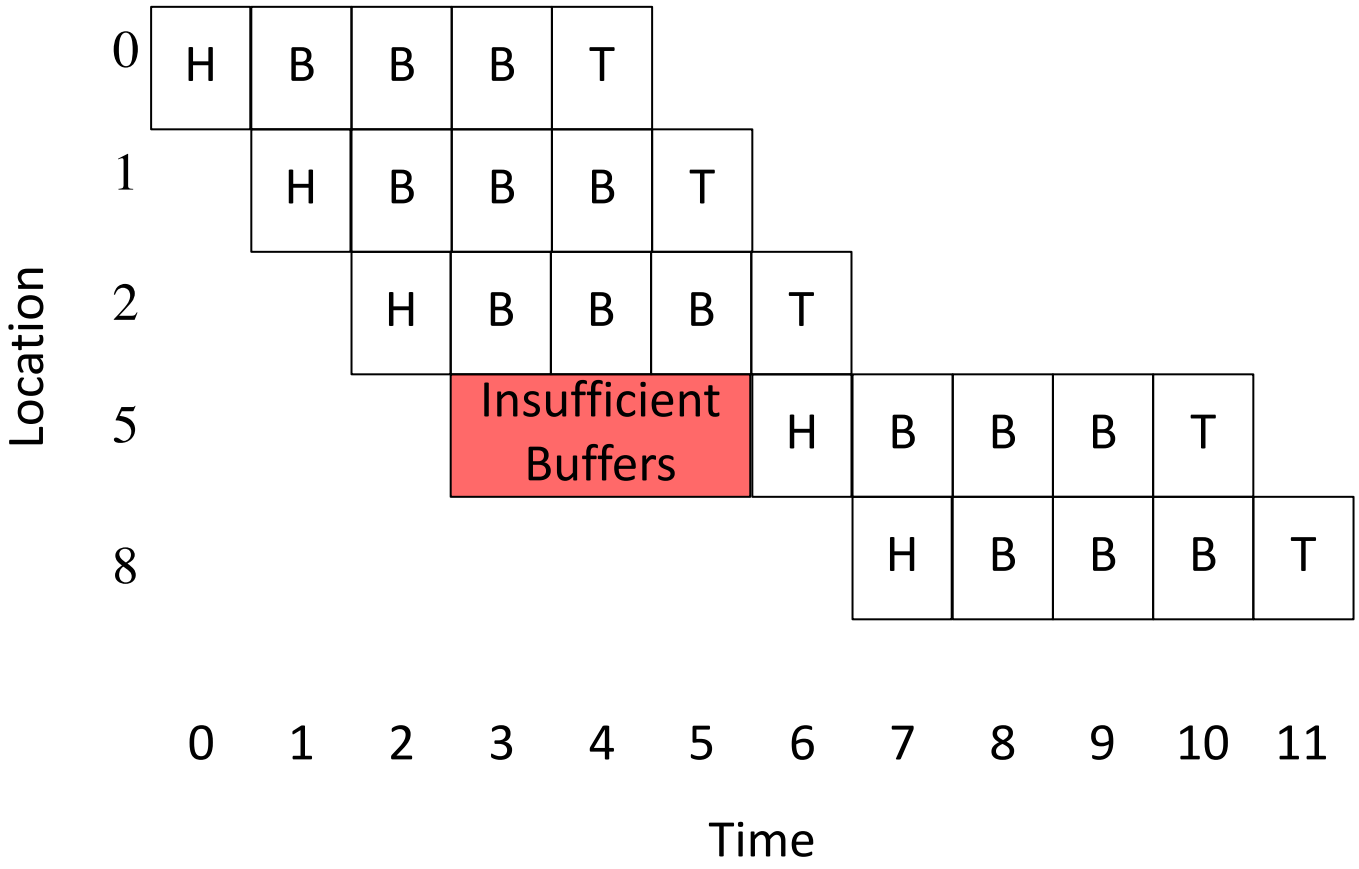


Virtual Cut Through



- Throughput suffers from inefficient buffer allocation

Time-Space Diagram: VCT (2)



Flit-Level Flow Control

- Help routers meet tight area/power constraints

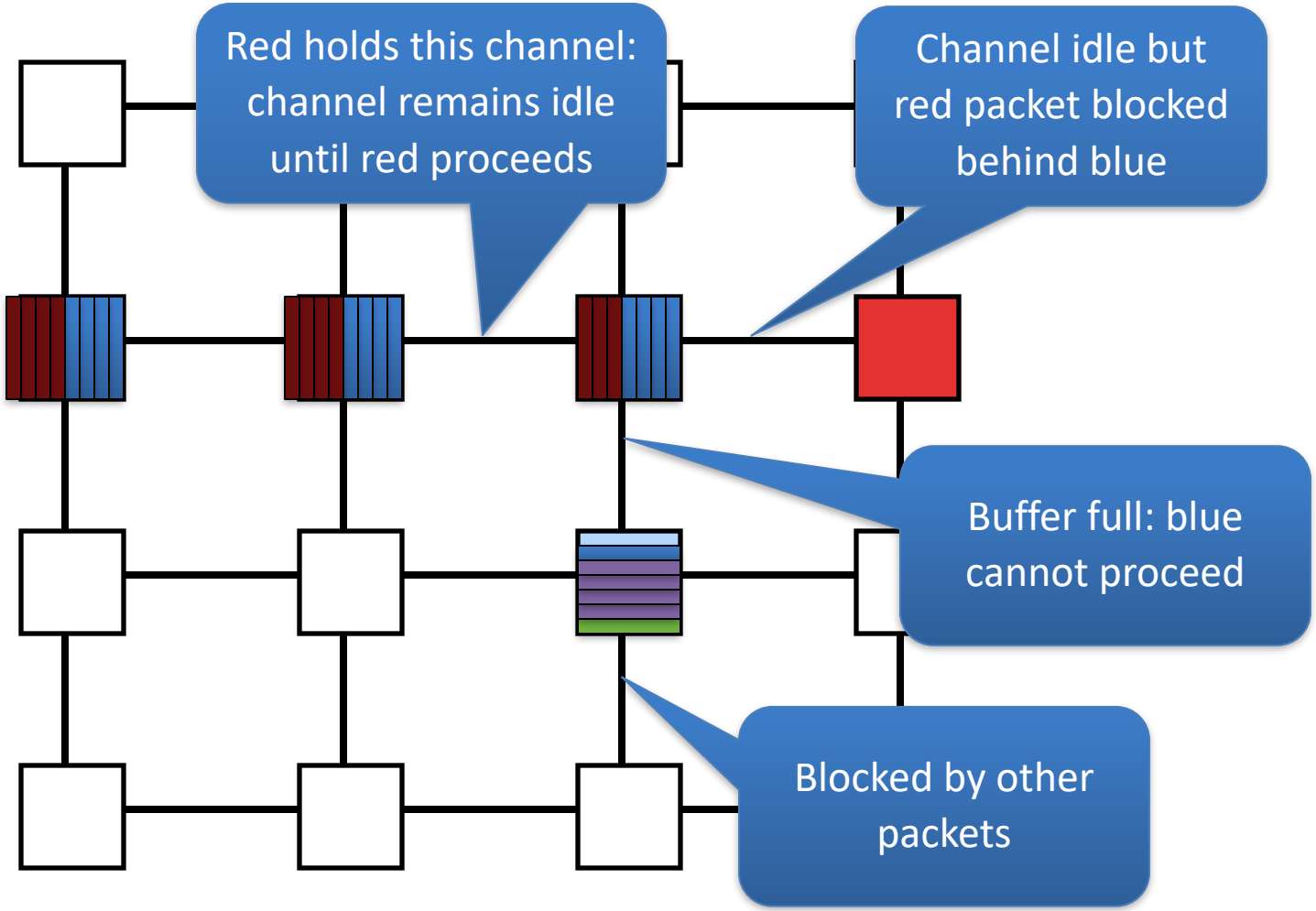
- Flit can proceed to next router when there is buffer space available for that **flit**
  - Improves over SAF and VCT by allocating buffers on a flit-by-flit basis

Wormhole Flow Control

- Pros
  - More efficient buffer utilization (good for on-chip)
  - Low latency

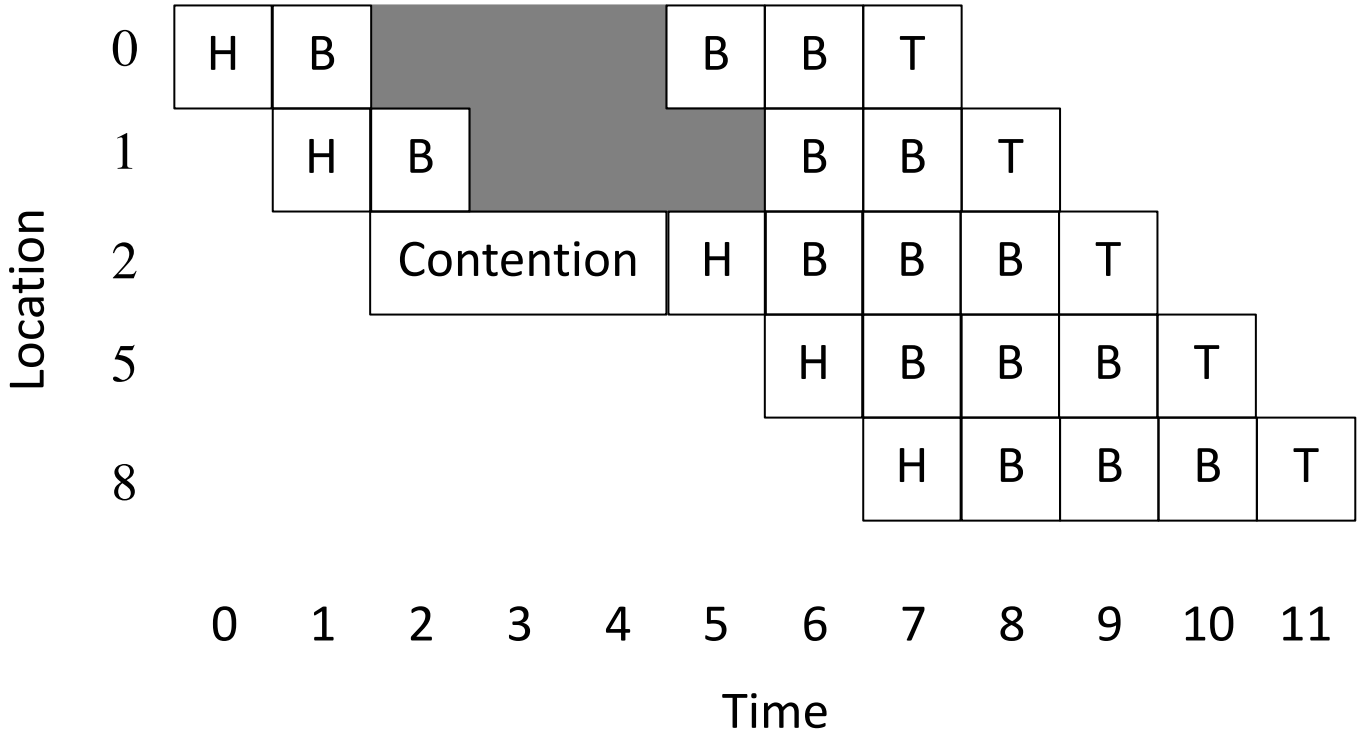
- Cons
  - Poor link utilization: if head flit becomes blocked, all links spanning length of packet are idle
    - Cannot be re-allocated to different packet
    - Suffers from **head of line** (HOL) blocking

Wormhole Example



- 6 flit buffers/input port

Time-Space Diagram: Wormhole



Virtual Channels

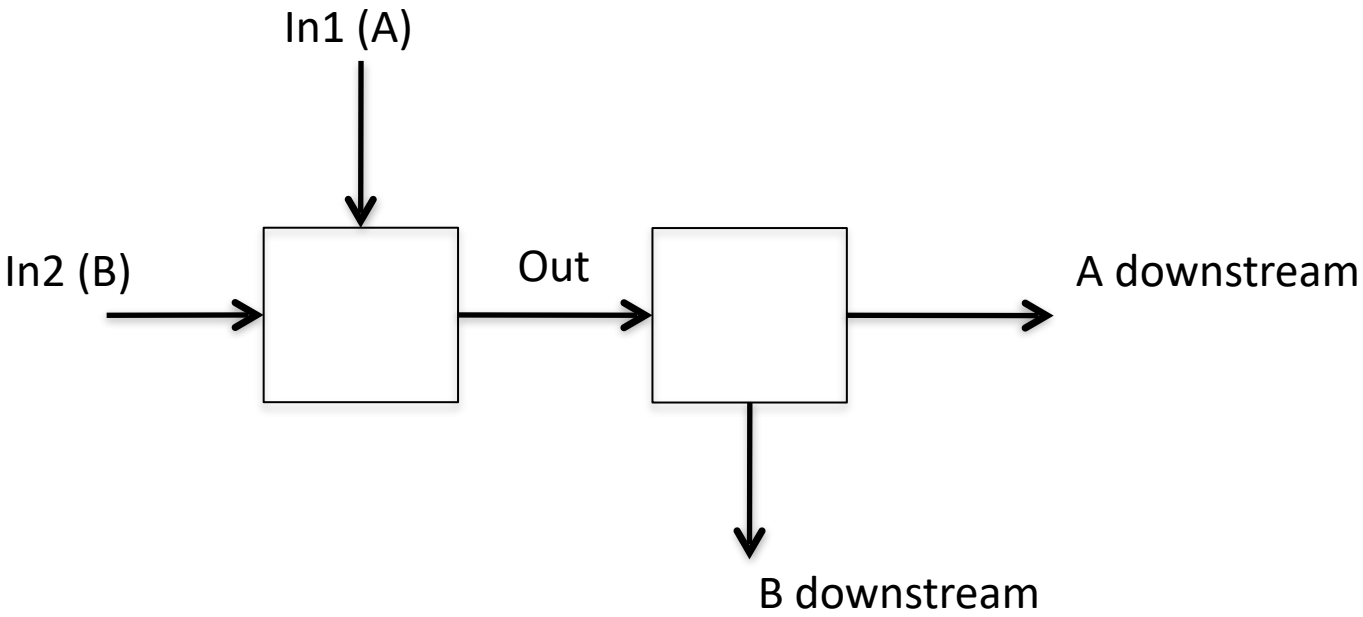
- First proposed for deadlock avoidance
  - We'll come back to this

- Can be applied to any flow control
  - First proposed with wormhole

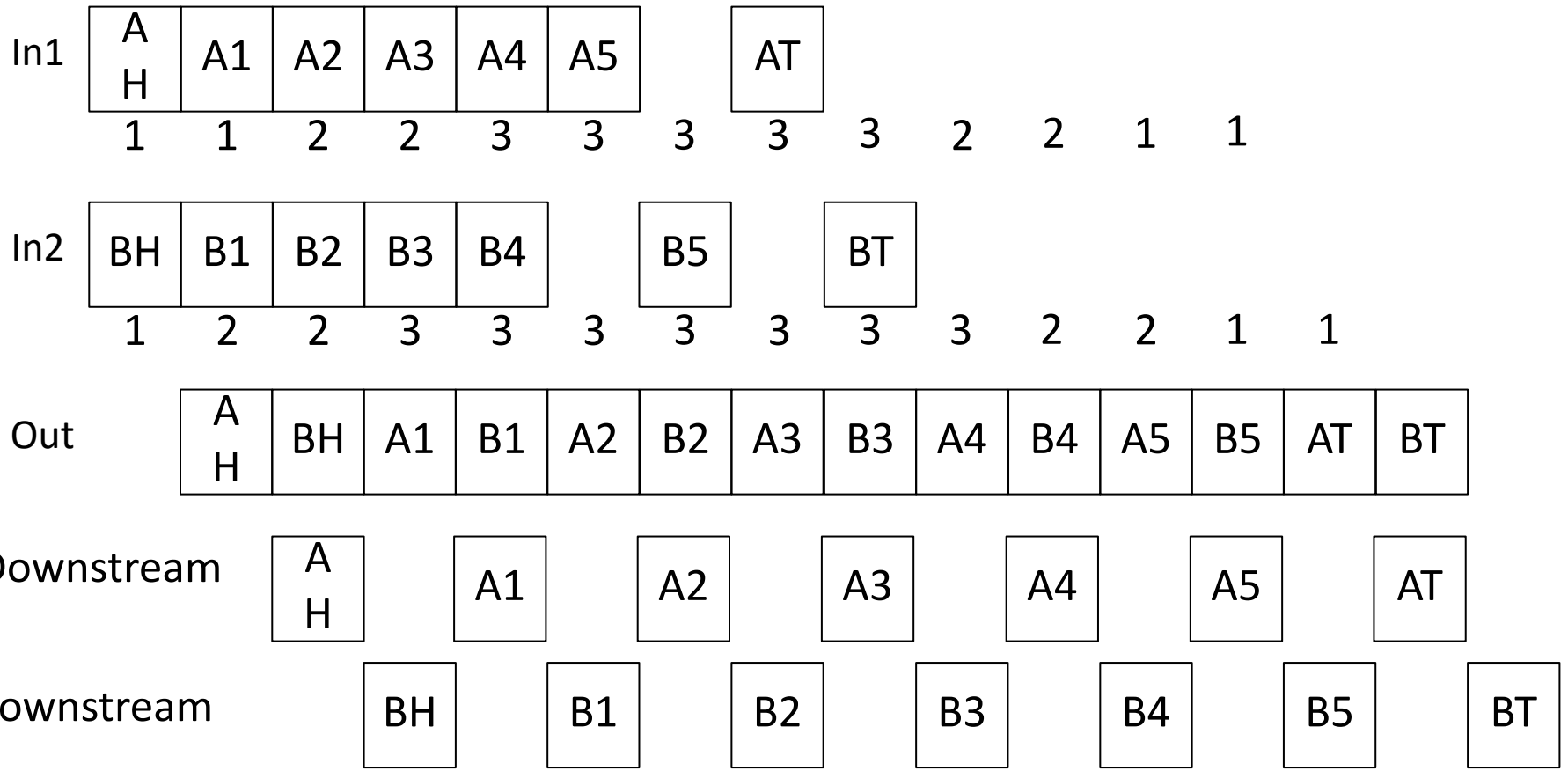
Virtual Channel Flow Control

- Virtual channels used to combat **HOL** blocking in wormhole
- Virtual channels: **multiple** flit queues per input port
  - Share **same** physical link (channel)
- Link utilization improved
  - Flits on different VC can pass blocked packet

Virtual Channel Flow Control (2)



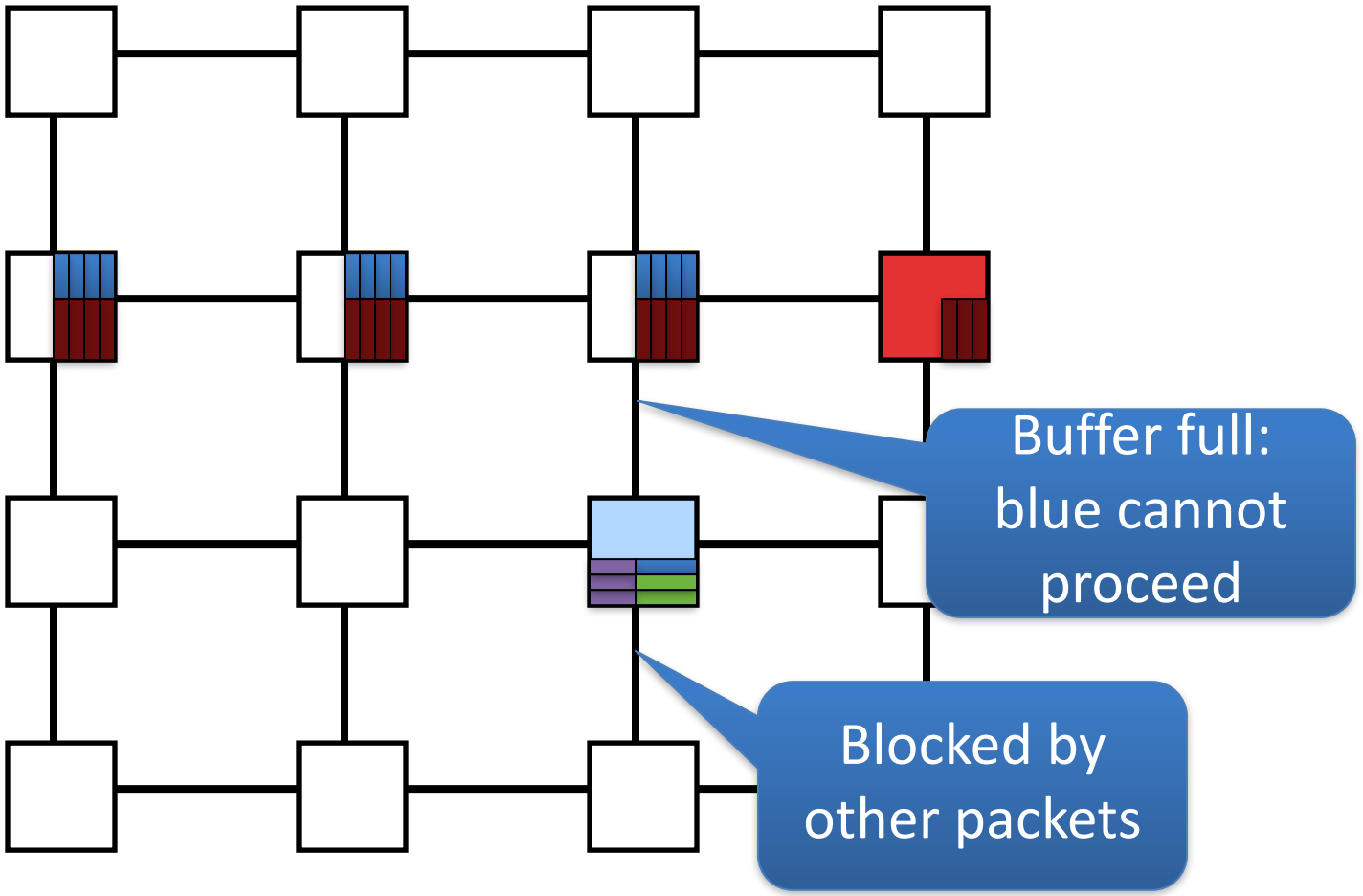
Virtual Channel Flow Control (3)



Virtual Channel Flow Control (3)

- Packets compete for VC on flit by flit basis
- In example: on downstream links, flits of each packet are available every other cycle
- Upstream links throttle because of limited buffers
- Does not mean links are idle
  - May be used by packet allocated to other VCs

Virtual Channel Example



- 6 flit buffers/input port
- 3 flit buffers/VC

Summary of techniques

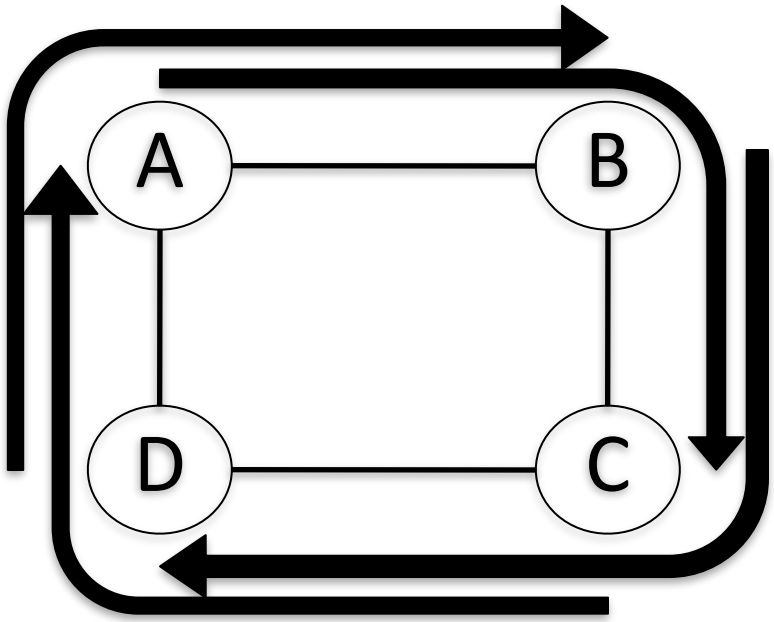
	Links reserved at granularity of	Buffers reserved at granularity of	Comments
Circuit-Switching	Messages	N/A (buffer-less)	Setup & Ack
Store and Forward	Packet	Packet	Head flit waits for tail
Virtual Cut Through	Packet	Packet	Head can proceed
Wormhole	Packet	Flit	HOL
Virtual Channel	Flit	Flit	Interleave flits of different packets

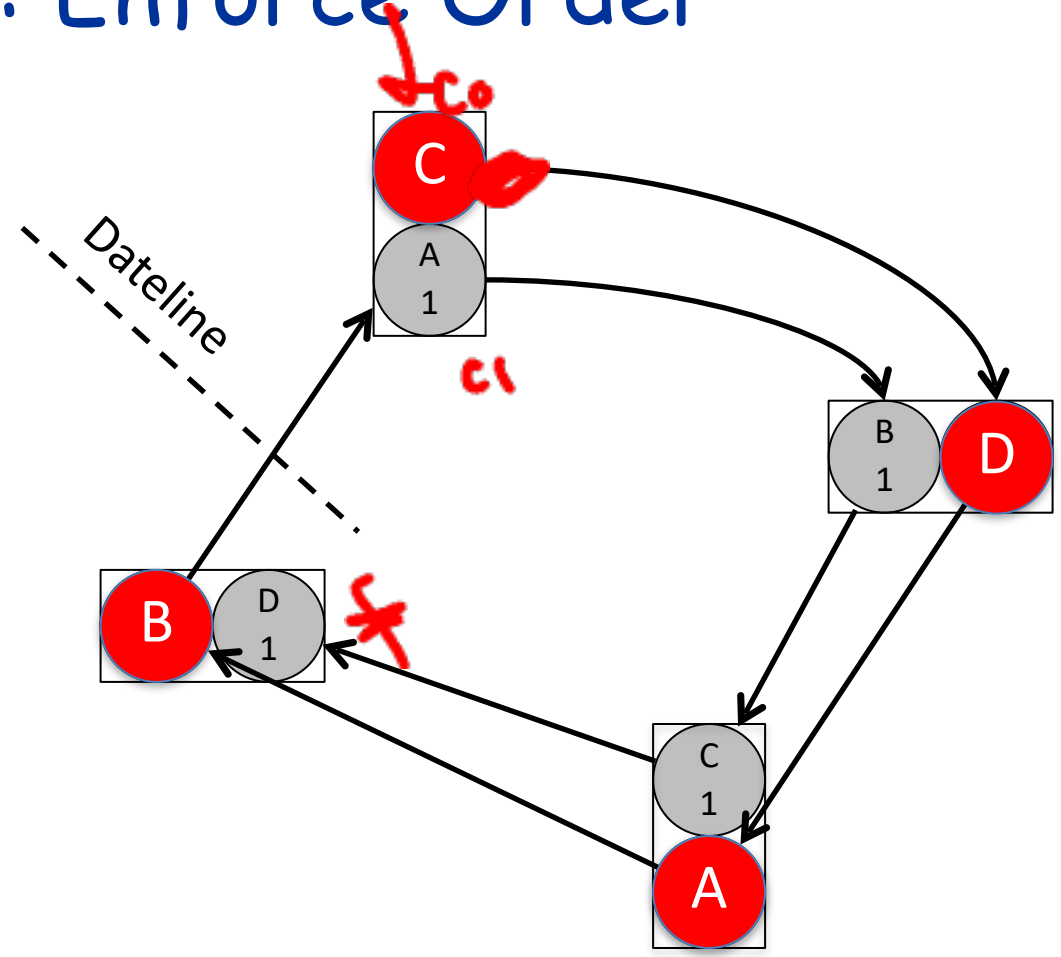
Deadlock

Deadlock

- Using flow control to guarantee deadlock freedom gives more **flexible** routing
  - Recall: routing restrictions needed for deadlock freedom
- If routing algorithm is not deadlock free
  - VCs can break resource cycle
- Each VC is time-multiplexed onto physical link
  - Holding VC implies holding associated buffer queue
  - Not tying up physical link resource
- Enforce order on VCs

Deadlock: Enforce Order





- All message sent through VC 0 until cross dateline
- After dateline, assigned to VC 1
  - Cannot be allocated to VC 0 again

Deadlock: Escape VCs

- Enforcing order lowers VC utilization
  - Previous example: VC 1 **underutilized**

- Escape Virtual Channels
  - Have 1 VC that is deadlock free
  - Example: VC 0 uses DOR, other VCs use arbitrary routing function
  - Access to VCs arbitrated fairly: packet always has chance of landing on escape VC

- Assign different message classes to different VCs to prevent protocol level deadlock
  - Prevent req-ack message cycles

Buffer Backpressure

Buffer Backpressure

- Need mechanism to prevent buffer **overflow**
  - Avoid dropping packets
  - Upstream nodes need to know buffer availability at downstream routers
- Significant impact on throughput achieved by flow control
- Two common mechanisms
  - Credits
  - On-off
- Credit-based generally works better
  - On-chip, wires are cheaper than buffers

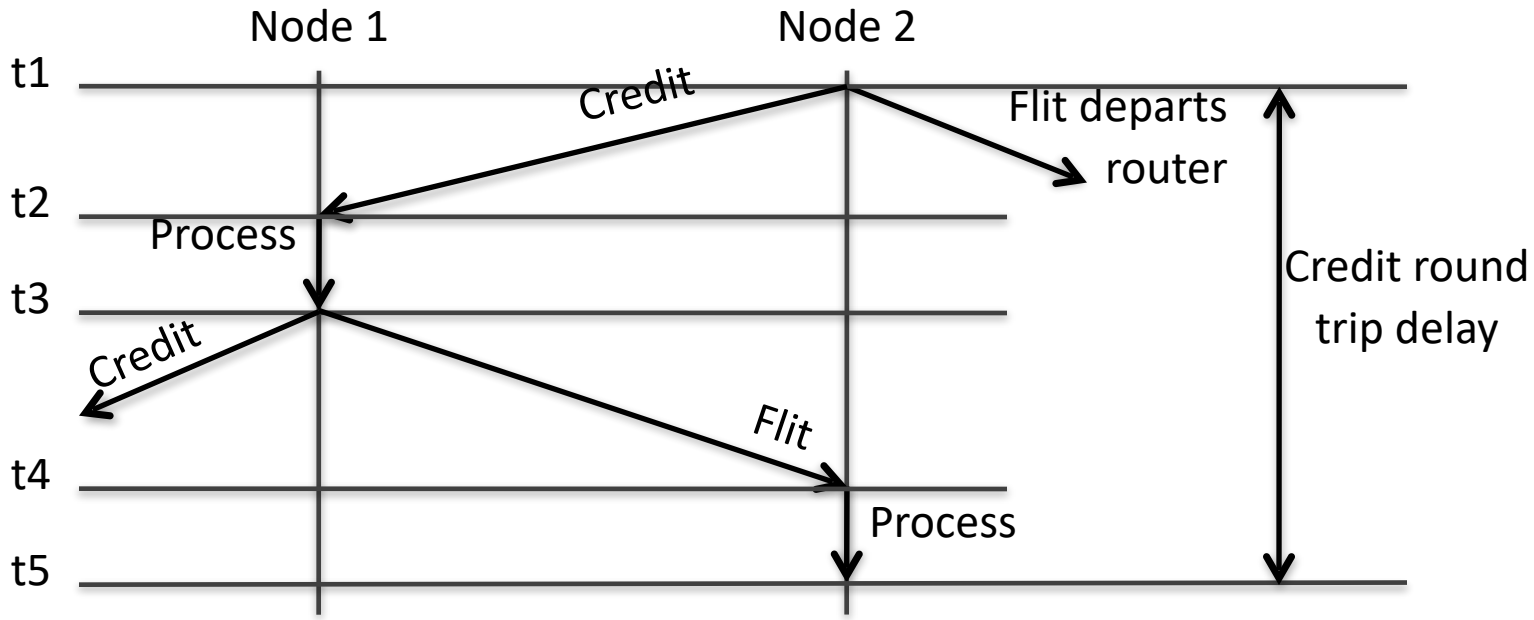
Credit-Based Flow Control

- Upstream router stores credit counts for each downstream VC

- Upstream router
  - When flit forwarded
    - Decrement credit count
  - Count == 0, buffer full, stop sending

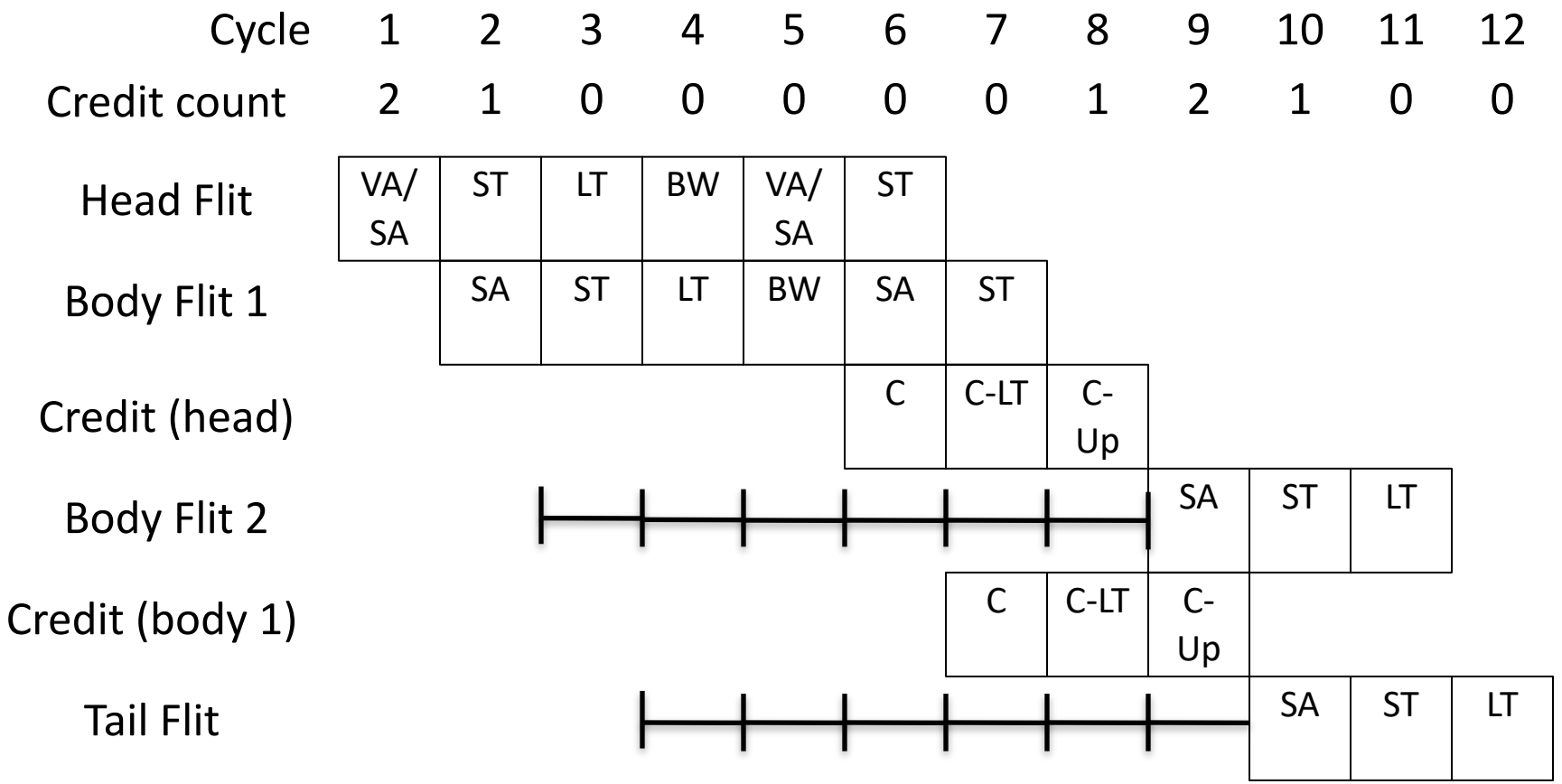
- Downstream router
  - When flit forwarded and buffer freed
    - Send credit to upstream router
    - Upstream increments credit count

Credit Timeline



- Round-trip credit delay:
  - Time between when buffer empties and when next flit can be sent from that buffer entry
  - If only single entry buffer, would result in significant throughput degradation
  - Important to size buffers to tolerate credit turn-around

Buffer Utilization



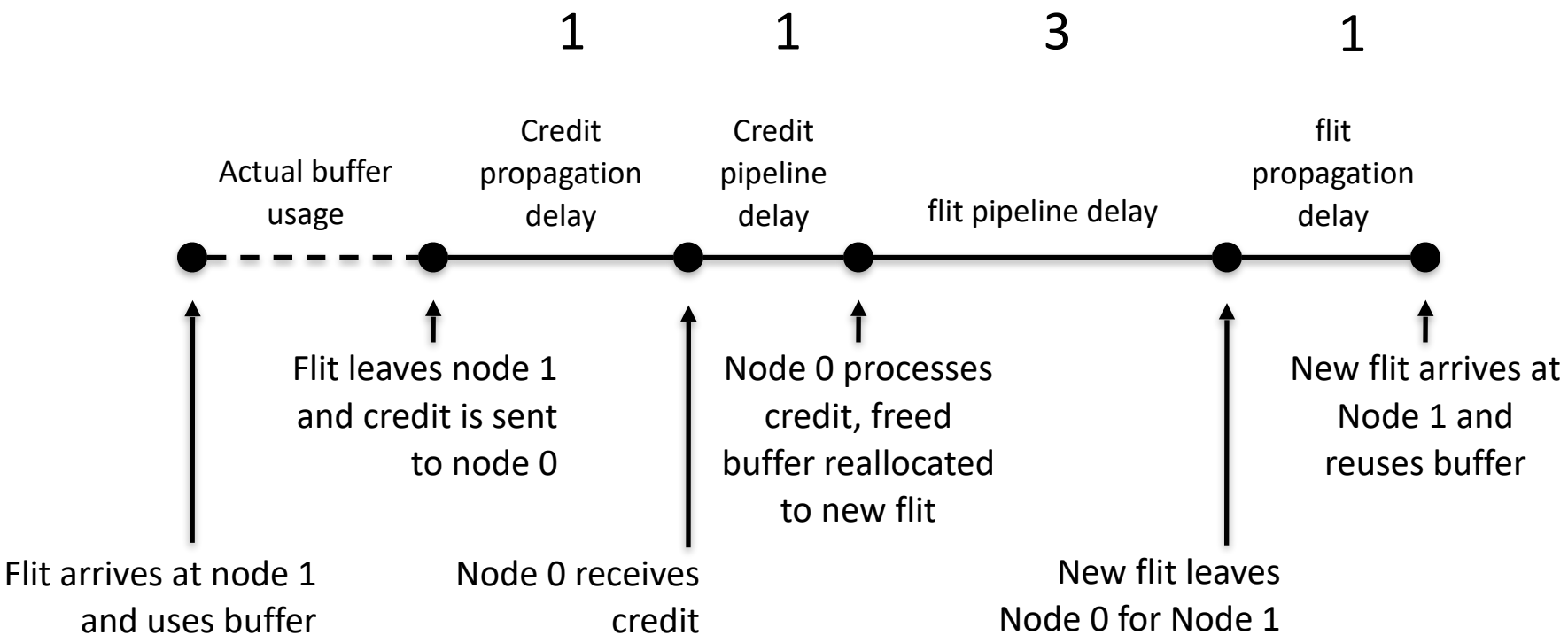
Buffer Sizing

- Prevent backpressure from limiting throughput
  - Buffers must hold # of flits >= turnaround time

- Assume:
  - 1 cycle propagation delay for data and credits
  - 1 cycle credit processing delay
  - 3 cycle router pipeline

- At least 6 flit buffers

Actual Buffer Usage & Turnaround Delay

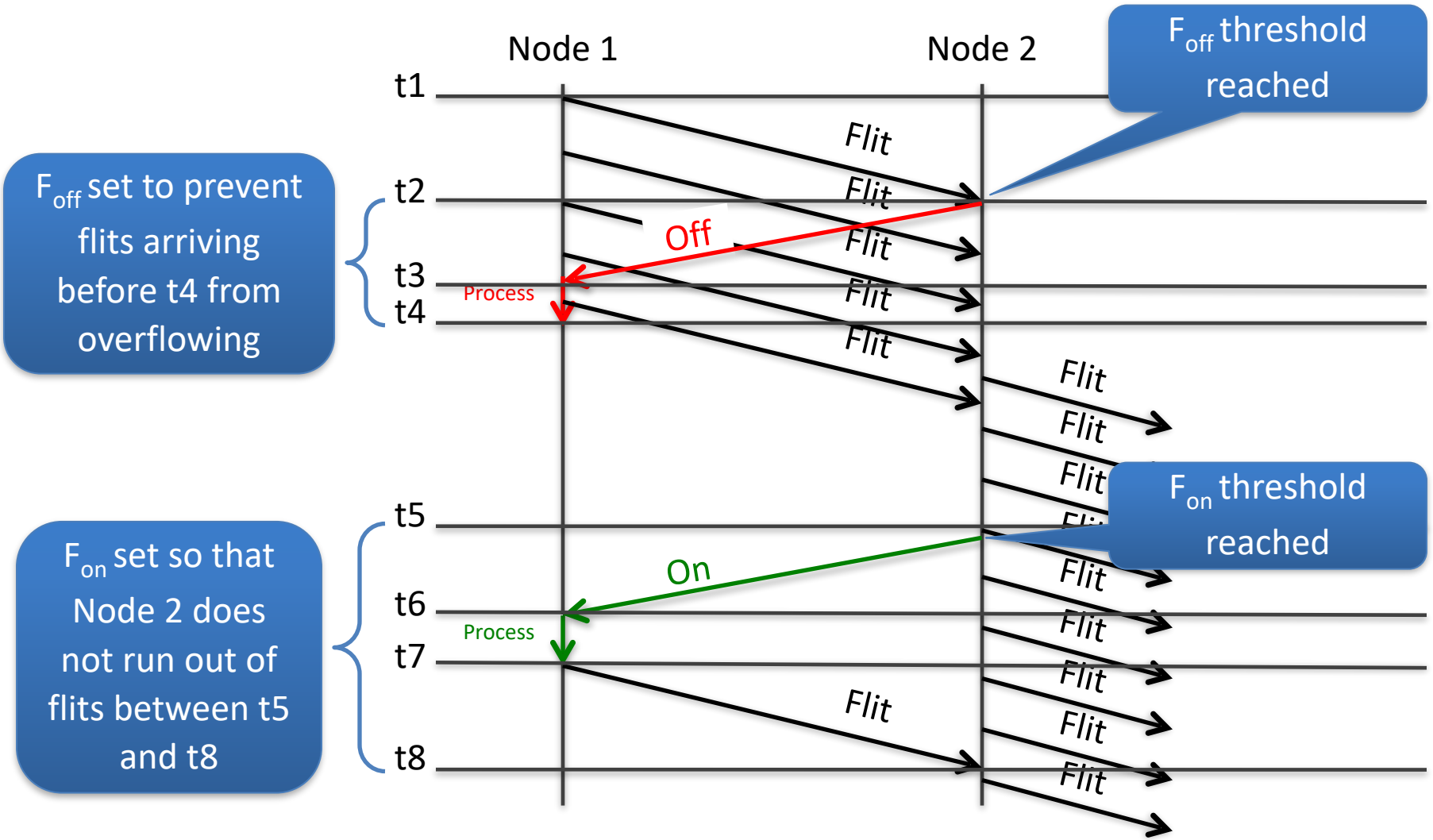


- Would need 6 buffer entries to cover turnaround time

On-Off Flow Control

- Credit: requires upstream signaling for every flit
- On-off: decreases upstream signaling
- Off signal
  - Sent when number of free buffers falls below threshold F_{off}
- On signal
  - Sent when number of free buffers rises above threshold F_{on}

On-Off Timeline



- Less signaling but more buffering
  - On-chip buffers more expensive than wires

Flow Control Summary

- On-chip networks require techniques with lower buffering requirements
  - Wormhole or Virtual Channel flow control
- Avoid dropping packets in on-chip environment
  - Requires buffer backpressure mechanism
- Complexity of flow control impacts router microarchitecture