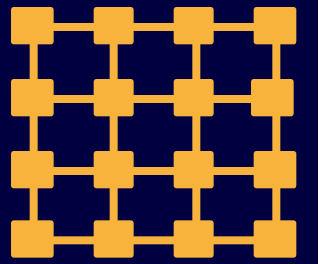
ECE 8823 A / CS 8803 - ICN
Interconnection Networks
Spring 2017



http://tusharkrishna.ece.gatech.edu/teaching/icn_s17/

Lecture 7: Flow Control - I

Tushar Krishna

Assistant Professor
School of Electrical and Computer Engineering
Georgia Institute of Technology

tushar@ece.gatech.edu

Acknowledgment: Slides adapted from Univ of Toronto ECE 1749 H (N Jerger)

Network Architecture

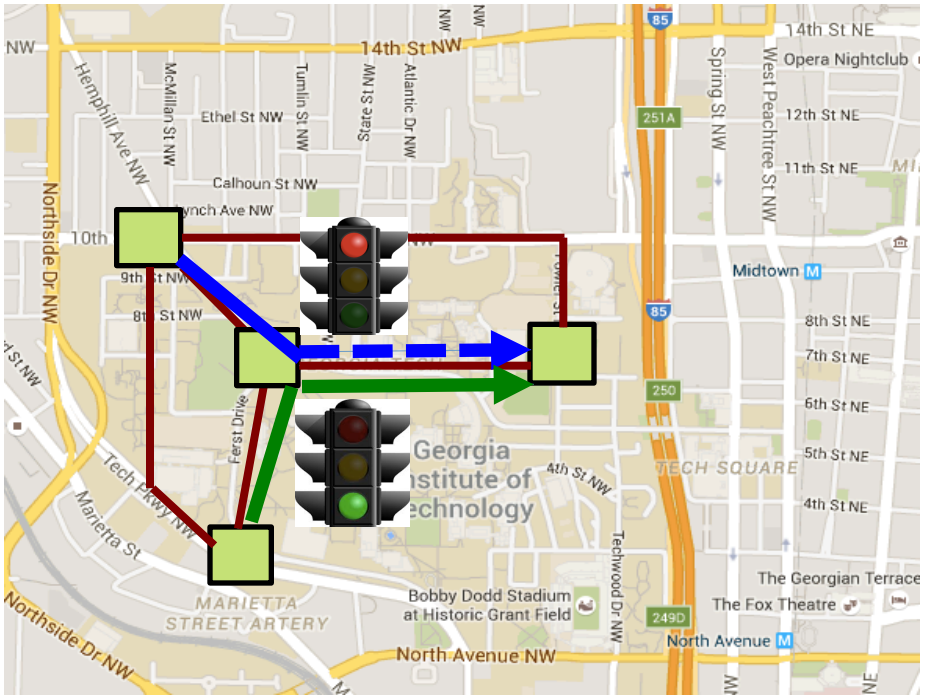
- **Topology**
  - How to connect the nodes
  - ~Road Network
- **Routing**
  - Which path should a message take
  - ~Series of road segments from source to destination
- **Flow Control**
  - When does the message have to stop/proceed
  - ~Traffic signals at end of each road segment
- **Router Microarchitecture**
  - How to build the routers
  - ~Design of traffic intersection (number of lanes, algorithm for turning red/green)

Flow Control

Once the topology and route are fixed, flow control determines the ***allocation of network resources*** (channel bandwidth, buffer capacity, and control state) to packets as they traverse the network

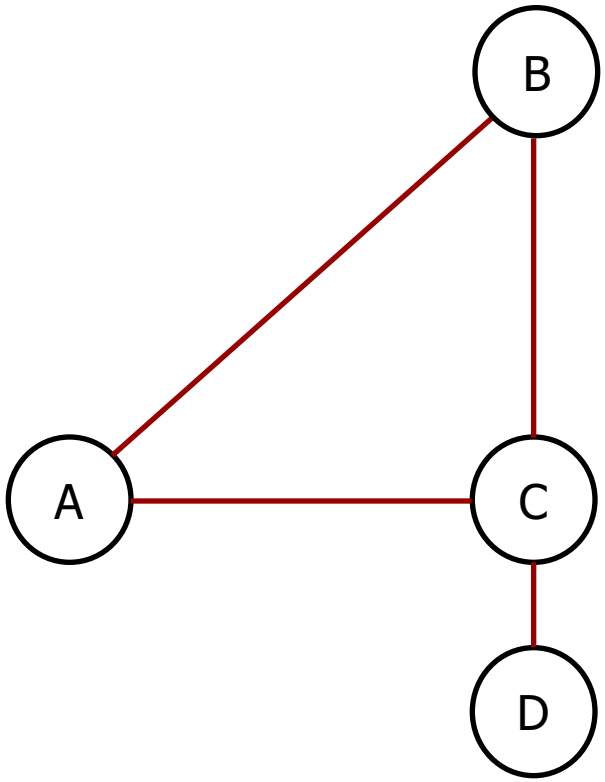
== resolution of contention between packets requesting the same resource

~Traffic Signals / Stop signs at end of each road segment



Why Flow Control matters?

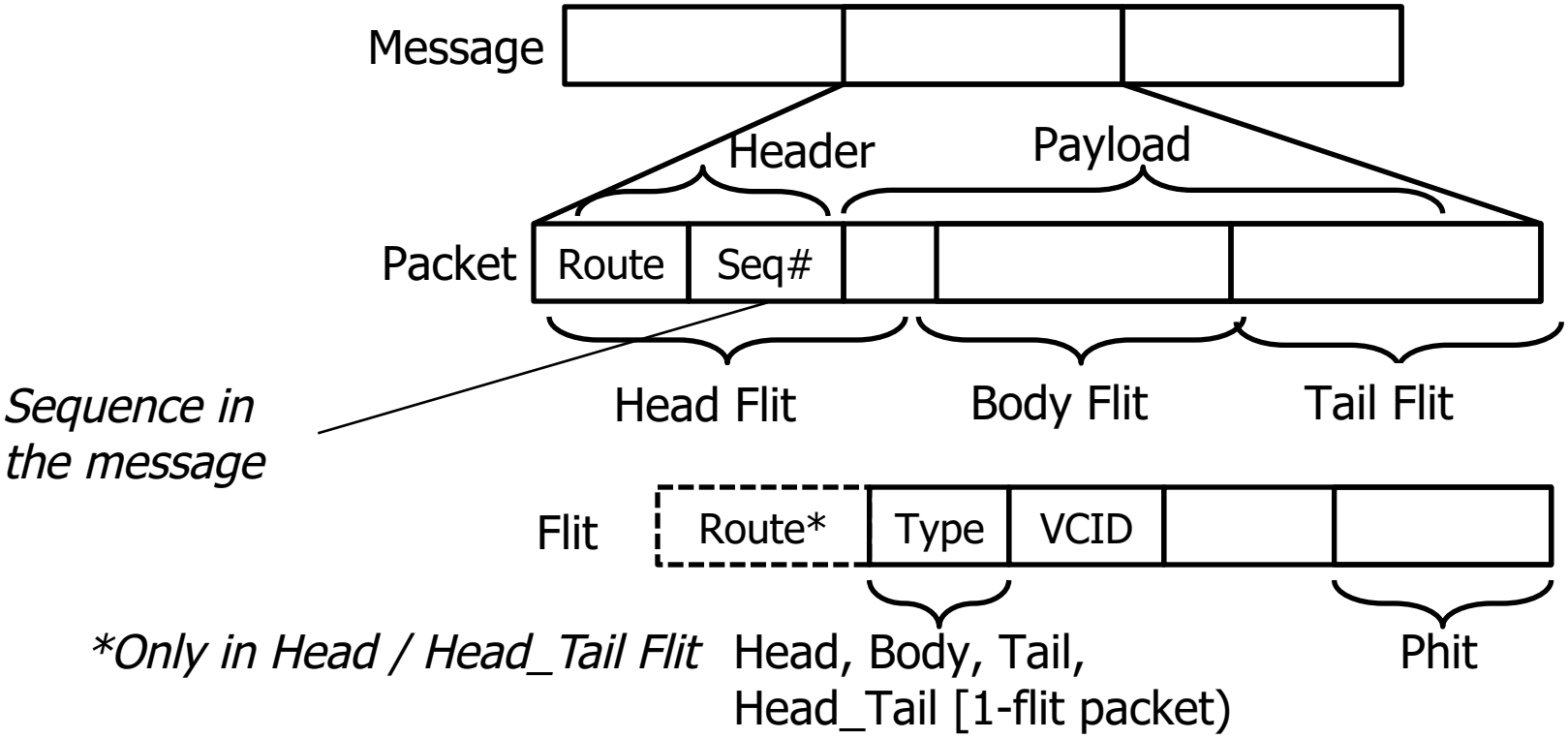
Flow control can single-handedly determine performance, however efficient the topology or routing algorithm might be



	Latency (hops) (A→B)	Throughput (msg/cycle) (A→B)
Topology	**1**	**1**
Routing (XY)	**2**	**1**
Flow Control Case I: One buffer at C Case II: Four D→B msgs	**3** **$(R_A +) L_{AC} + R_C + L_{CB} (+ R_B)$**	 **1/2** **1/5**

Suppose Router Delay = 1, Link Delay = 1

Allocation Granularity: Messages, Packets, and Flits



Off-chip (SANs)

Messages could be B/KB/MB of data

Flits have to be sent serially as multiple phits (limited by **pins**)

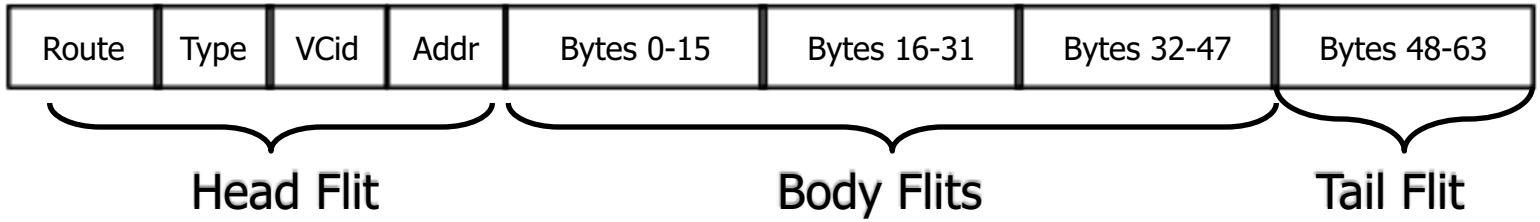
On-chip (NoC)

Message = Packet

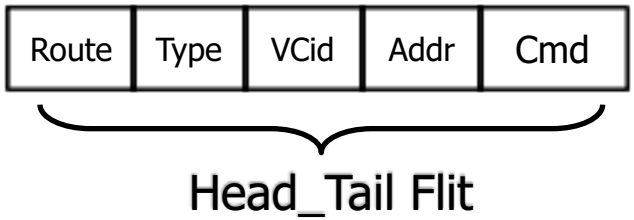
Flit = Phit (**abundant on-chip wires**)

Packet Sizes in NoCs

Cache line (Data)



Cache Line Request



64B Cache Line
~128-bit flits (i.e., link width)

1 control flit (cache line req)
5 data flits (cache line data)

All flits of a packet take same route and have the same VCid

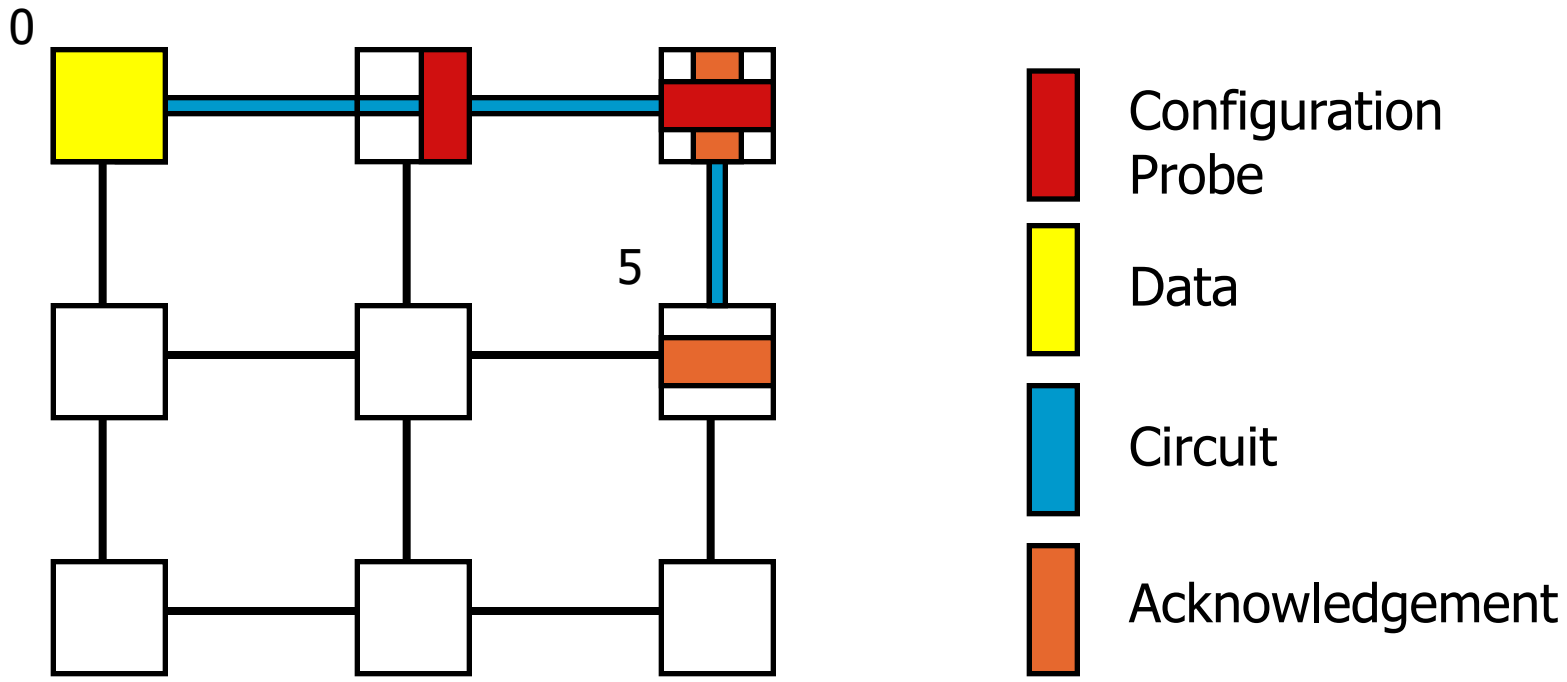
Flow Control based on Allocation Granularity

- Message-based Flow Control
  - E.g., Circuit Switching
- Packet-based Flow Control
  - E.g., Store and Forward, Virtual Cut-Through
- Flit-based Flow Control
  - E.g., Wormhole, Virtual Channel

Message-based Flow Control

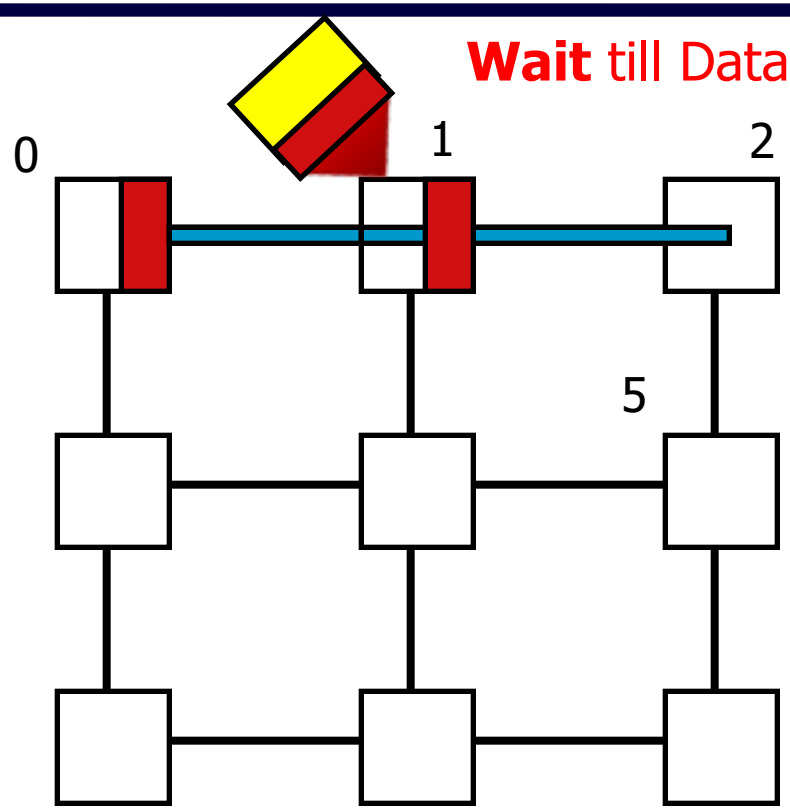
- Coarsest Granularity
- Circuit-switching
  - Setup entire path before sending message
    - Reserve all channels from source to destination using a setup probe
  - Once setup complete, send Data through the channels
    - Buffers not needed at routers as no contention
  - Tear down the circuit once transmission complete

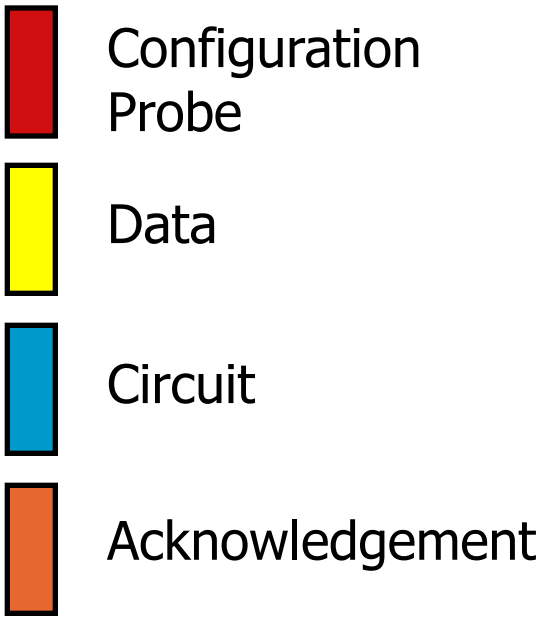
Circuit Switching Example



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

Handling Contention





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

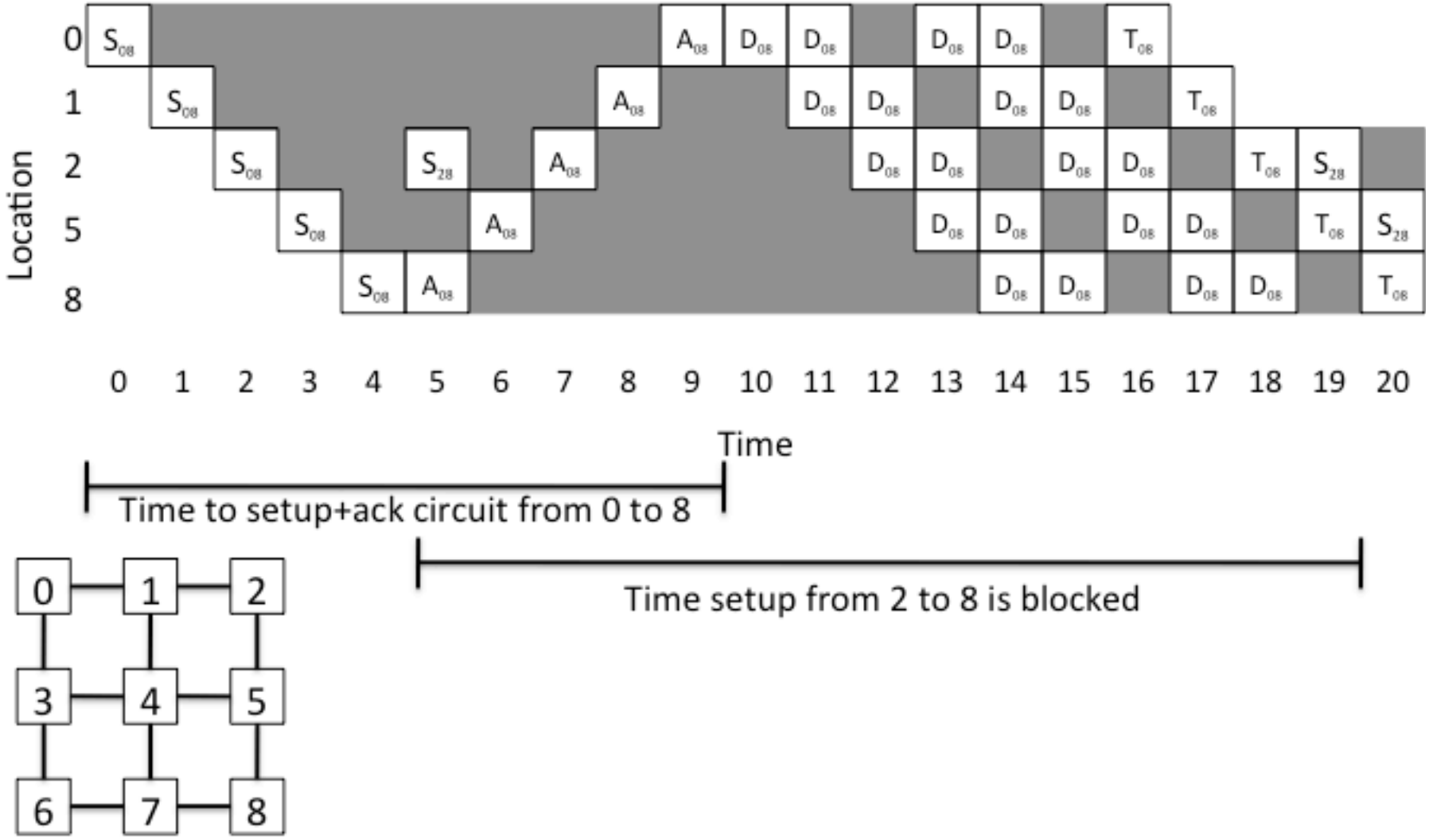
Challenges with Circuit-Switching

- Loss in bandwidth (throughput)
  - Throughput can suffer due to **setup** and **transfer** time for circuits
    - Links are idle until setup is complete
    - No other message can use links until transfer is complete
- Latency overhead in setup if the amount of data being transferred is small

Circuit-Switching in NoCs?

- Cache Line = 64B
  - Suppose
    - Channel Width = 128b => 64x8/128 = 4 chunks
    - 3-hop traversal with 1-cycle per hop
  - Setup = 3 cycles
  - ACK = 3 cycles
  - Data Transfer Time = 3 (for first chunk) + 3 (remaining chunks) = 6 cycles
  - Total Time = 12 cycles
    - Half of this went in circuit setup!
- Hybrid Circuit-Packet Switching
  - "Jerger et. al, "Circuit Switched Coherence", NOCS 2008

Time-Space Diagram: Circuit Switching



Packet-based Flow Control

- "Packet Switching"
  - Break messages into packets
  - Interleave packets on links
    - Better utilization
  - Requires per-node buffering to store packets in-flight waiting for output channel

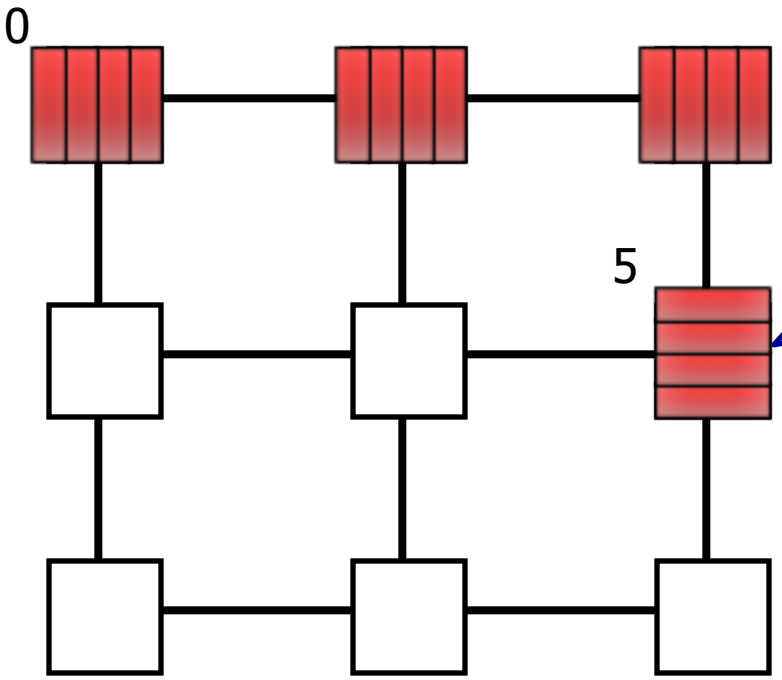
- Two techniques
  - Store and Forward
  - Virtual Cut-Through

Packet-based: 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

Store and Forward Example

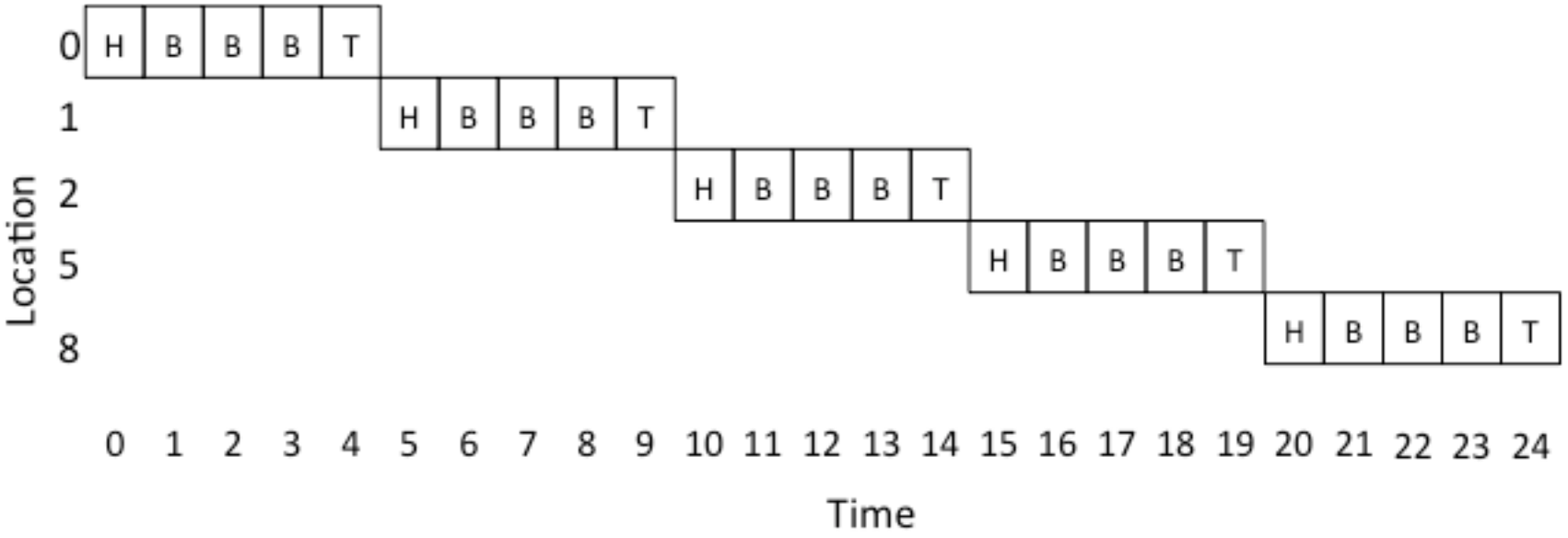


Total delay = 4 cycles per hop x 3 hops = 12 cycles

Not suitable on-chip.
Why?

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

Time-Space Diagram: Store and Forward

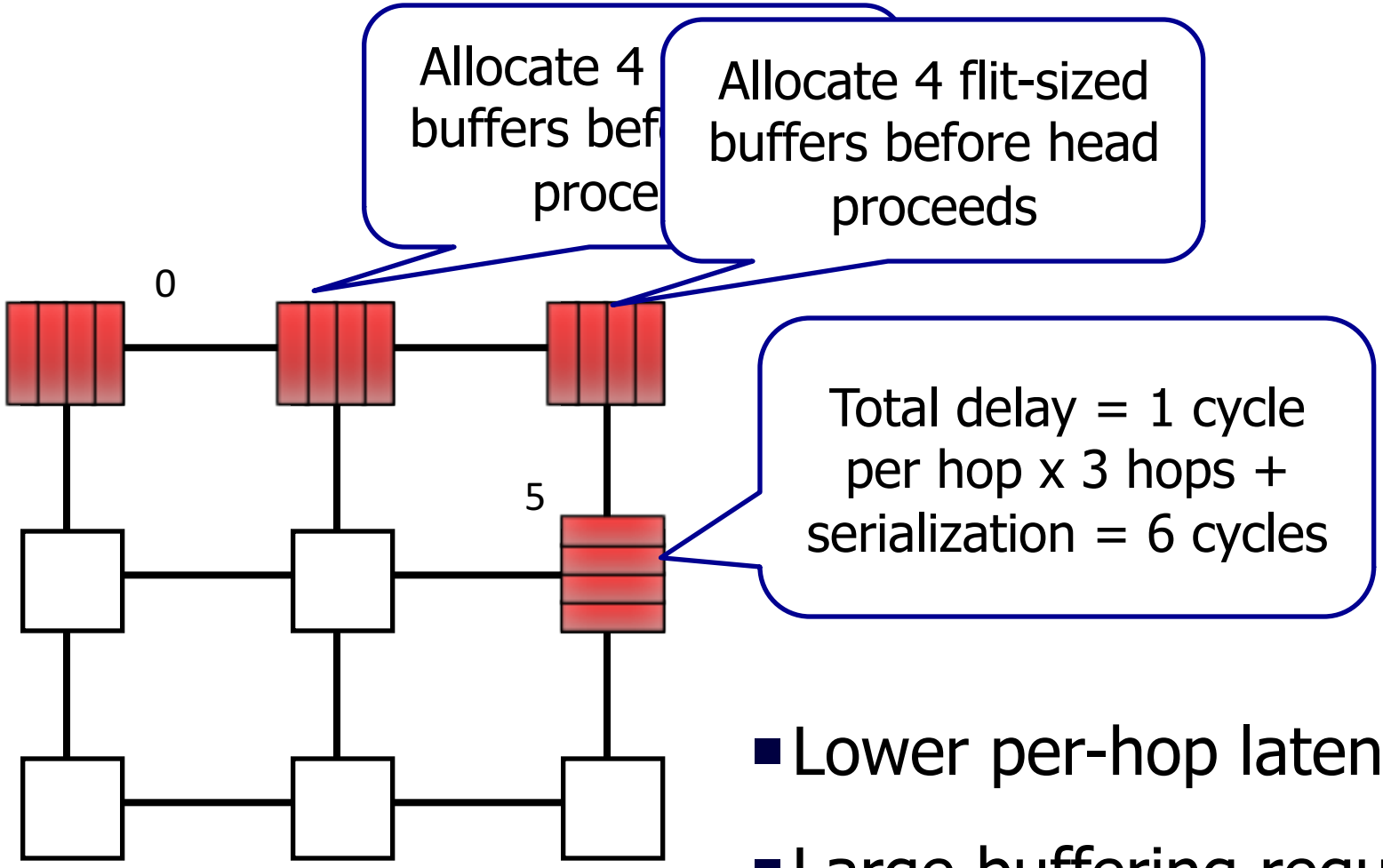


Packet-based: Virtual Cut-Through

- Links and Buffers allocated to **entire** packets

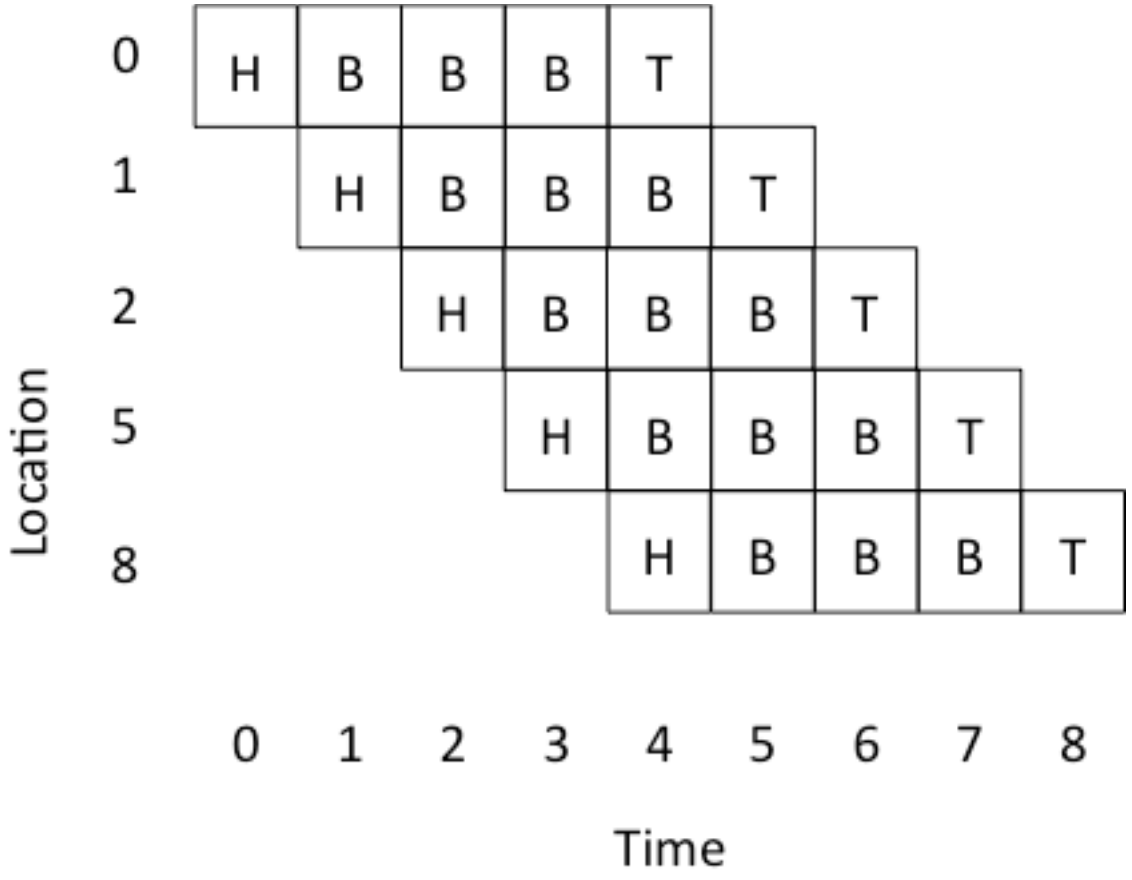
- Flits can proceed to next hop before tail flit has been received by current router
  - But only if next router has enough buffer space for **entire** packet

Virtual Cut-Through Example

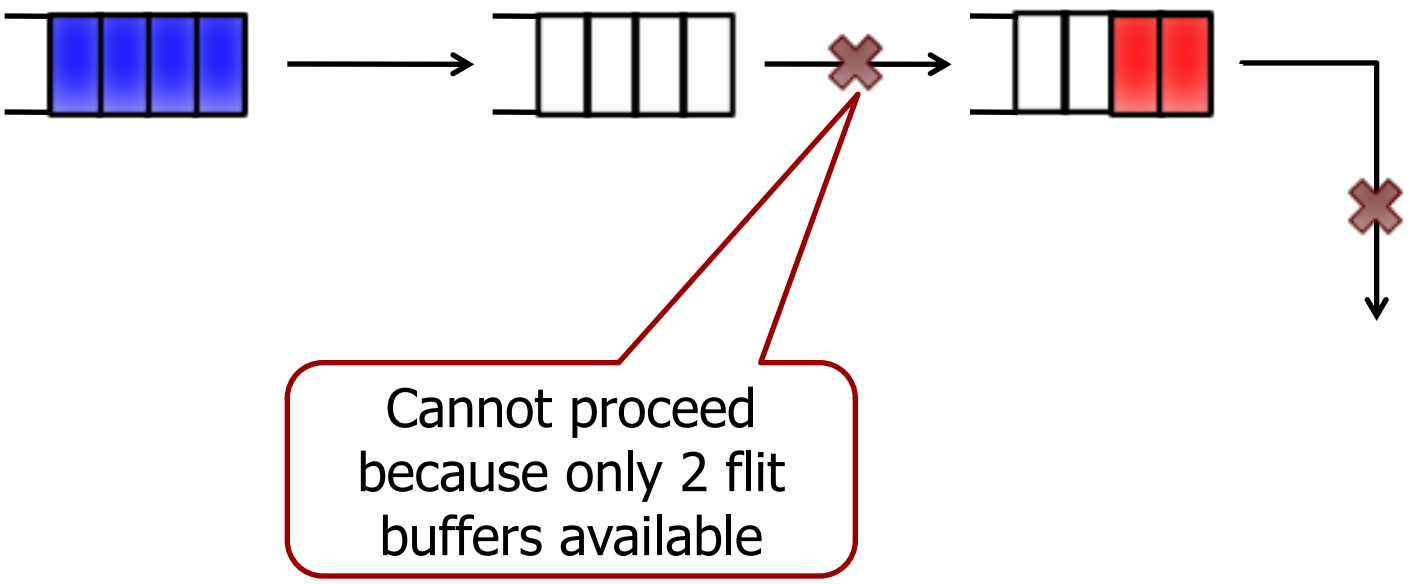


- Lower per-hop latency
- Large buffering required

Time-Space Diagram: Virtual Cut-Through

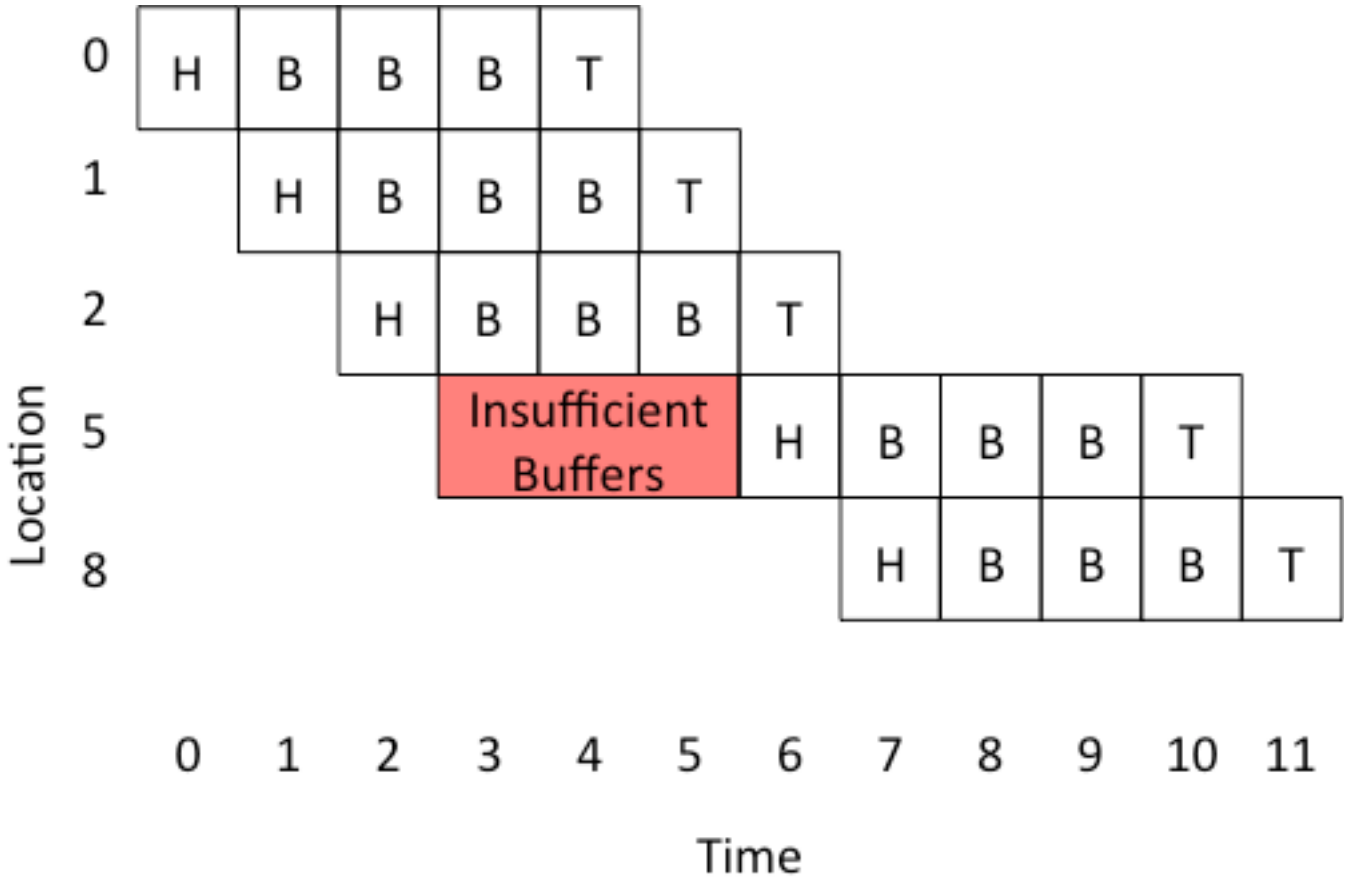


Virtual Cut-Through Example (2)



Throughput suffers from inefficient buffer allocation

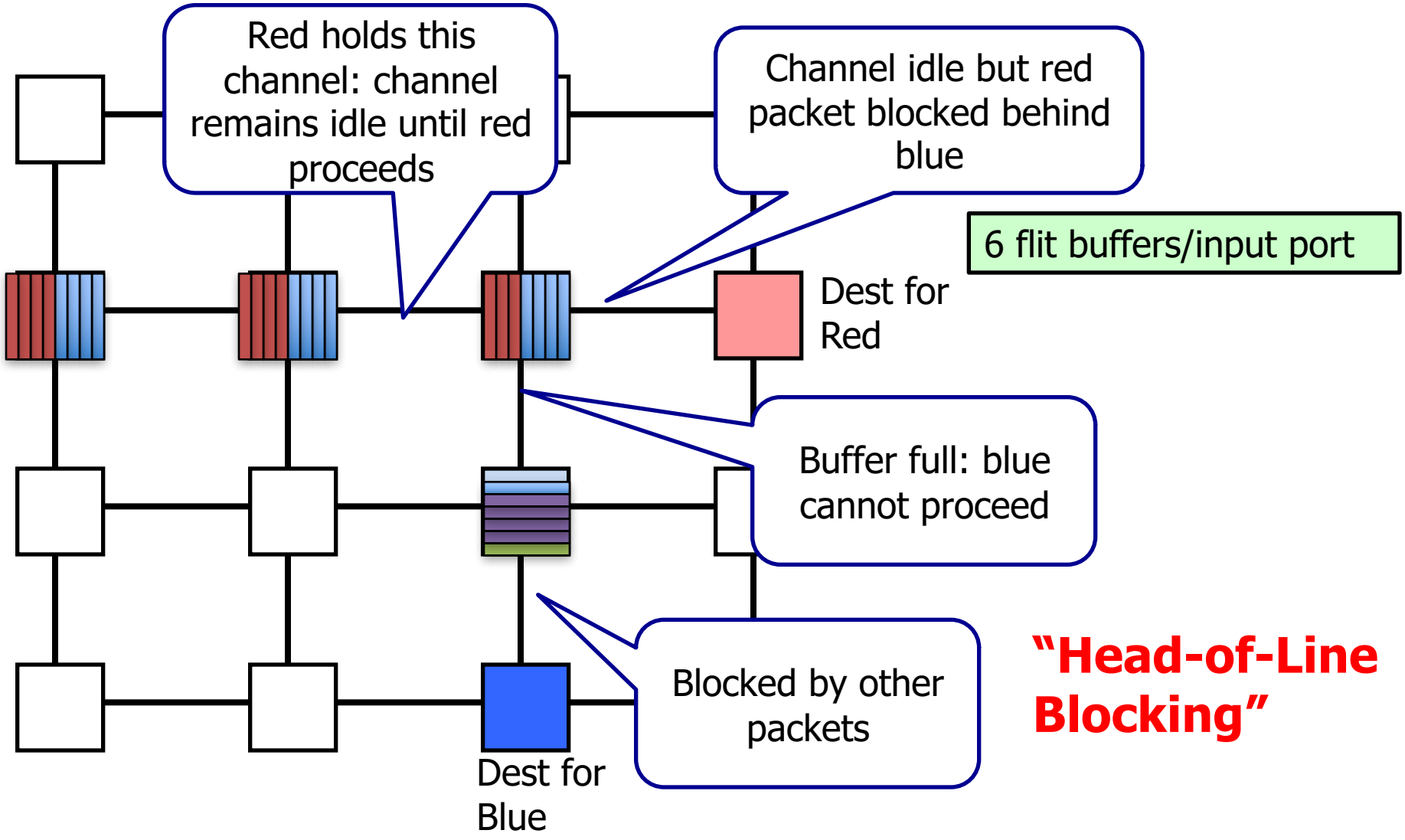
Time-Space Diagram: Virtual Cut-Through (2)



Flit-level Flow Control

- Like VCT, flit can proceed to next router before entire packet arrives
  - Unlike VCT, flit can proceed as soon as there is sufficient buffering for that **flit**
- Buffers allocated per flit rather than per packet
  - Routers do not need to have packet-sized buffers
  - Help routers meet tight area/power constraints
- Two techniques
  - Wormhole – link allocated per packet
  - Virtual Channel – link allocated per flit

Wormhole Flow Control Example



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

Time-Space Diagram: Wormhole