ECE 6115 / CS 8803 - ICN
Interconnection Networks for High Performance Systems
Spring 2020

FLOW-CONTROL

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

tushar@ece.gatech.edu

NETWORK ARCHITECTURE

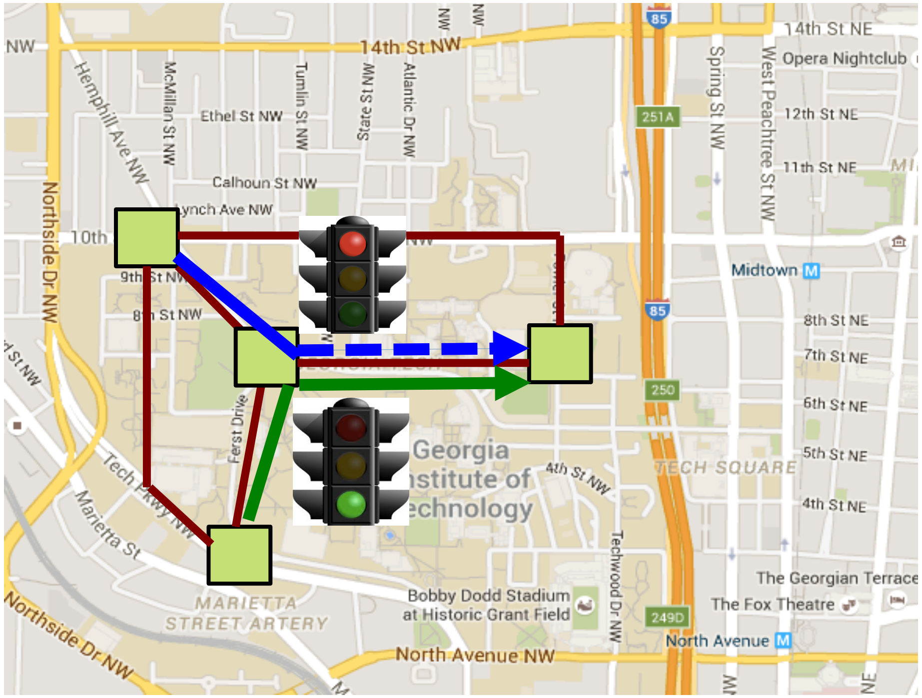
- **Topology**
  - How to connect the nodes
  - ~Road Network
- **Routing**
  - Which path should a message take
  - ~Series of road segments from source to destination
- **Flow Control**
  - When does the message have to stop/proceed
  - ~Traffic signals at end of each road segment
- **Router Microarchitecture**
  - 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

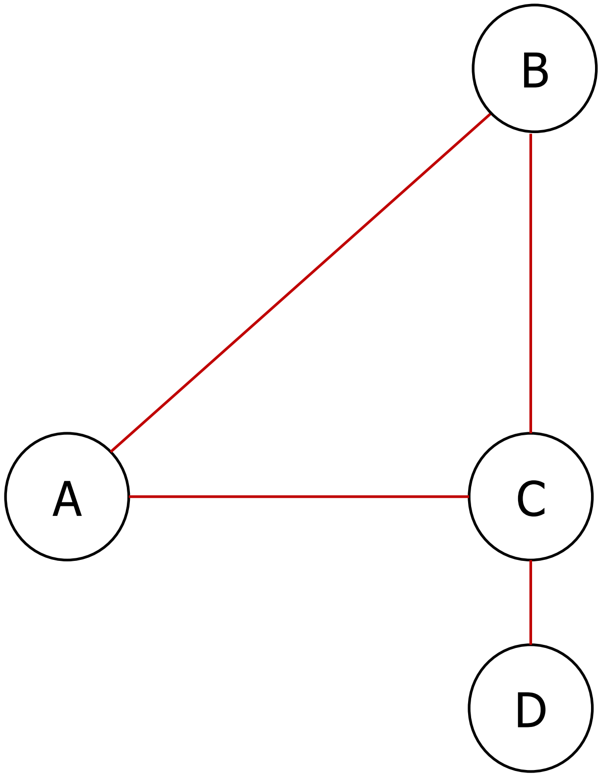
== resolution of contention between packets requesting the same resource

~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

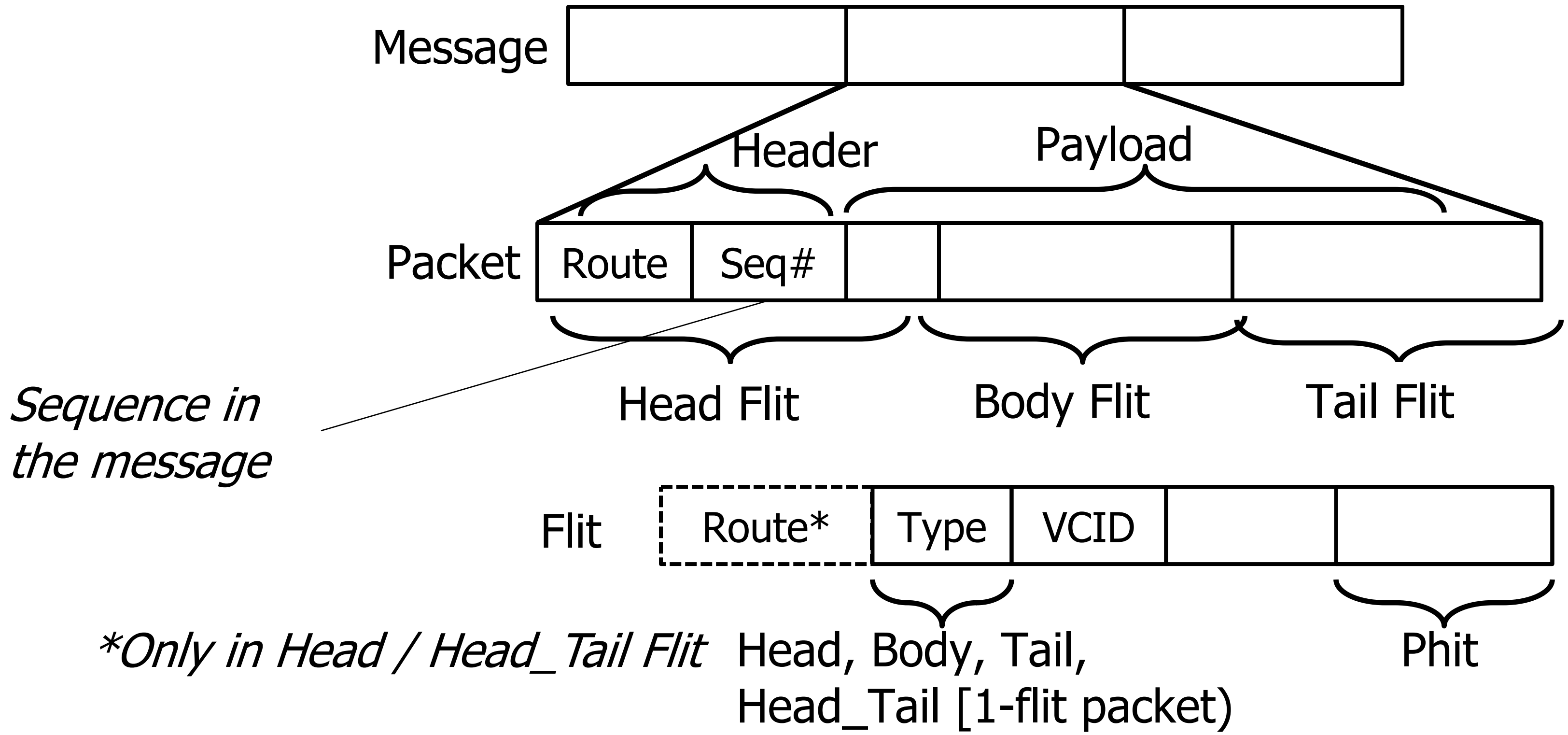


Case II: D sends 4 messages, 1 cycle break, 4 messages, 1 cycle break…C prioritizes straight over turning traffic

	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: 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

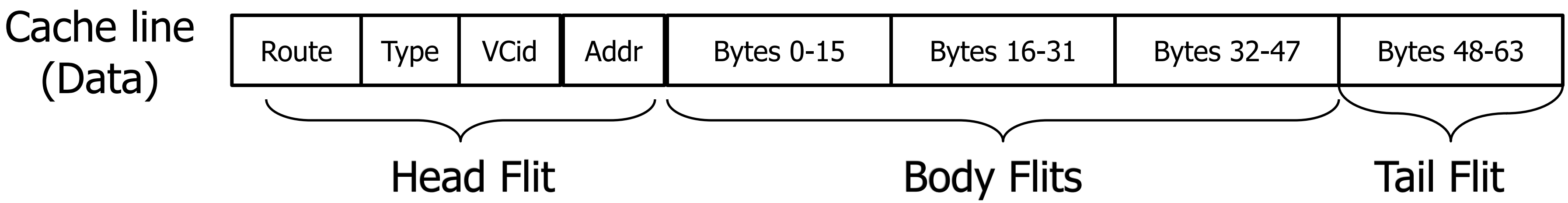
ALLOCATION GRANULARITY: MESSAGES, PACKETS, AND FLITS

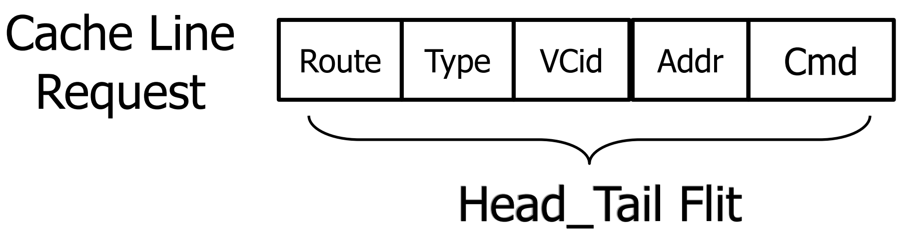


Off-chip (SANs)
Messages could be B/KB/MB of data
Flits have to be sent serially as multiple phits (limited by **pins**)

On-chip (NoC)
Message = Packet
Flit = Phit (**abundant on-chip wires**)

PACKET SIZES IN NOCS





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

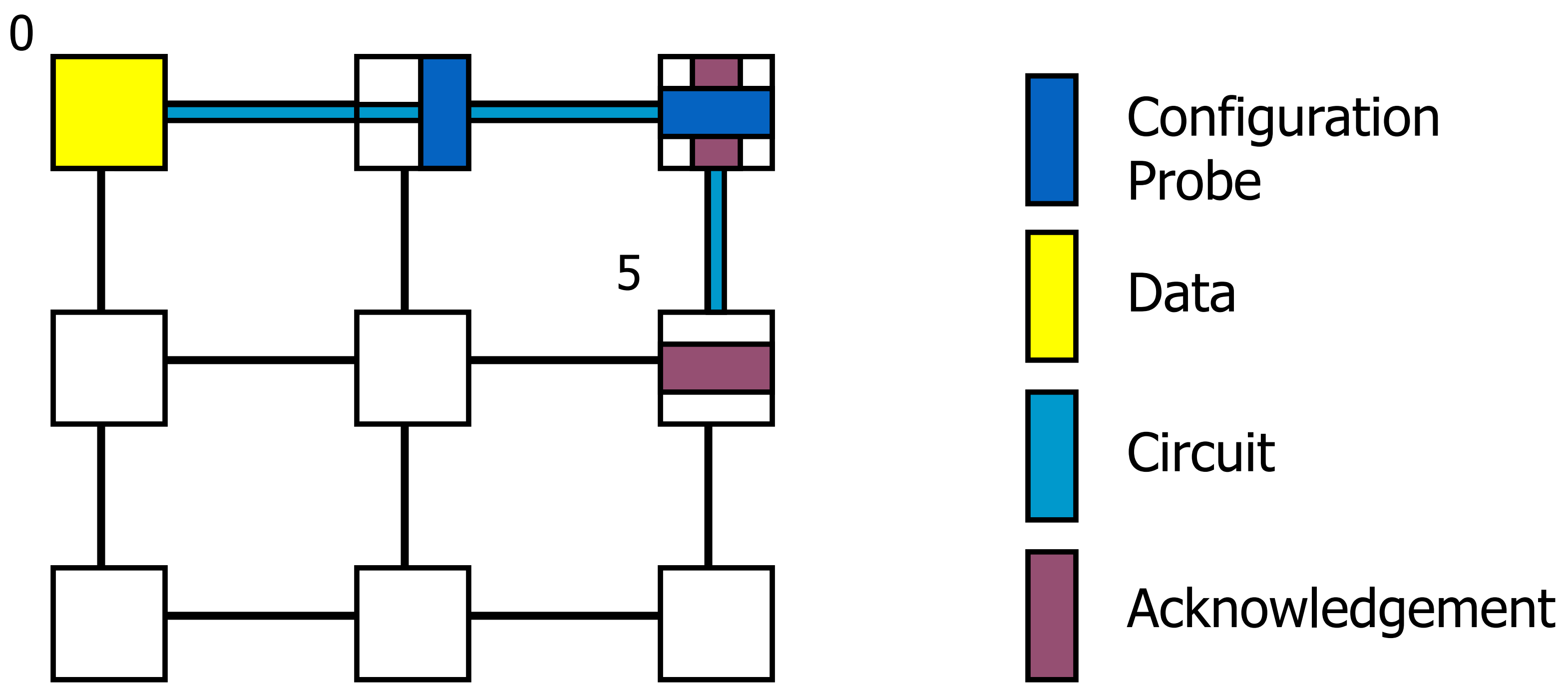
FLOW CONTROL BASED ON ALLOCATION GRANULARITY

- 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

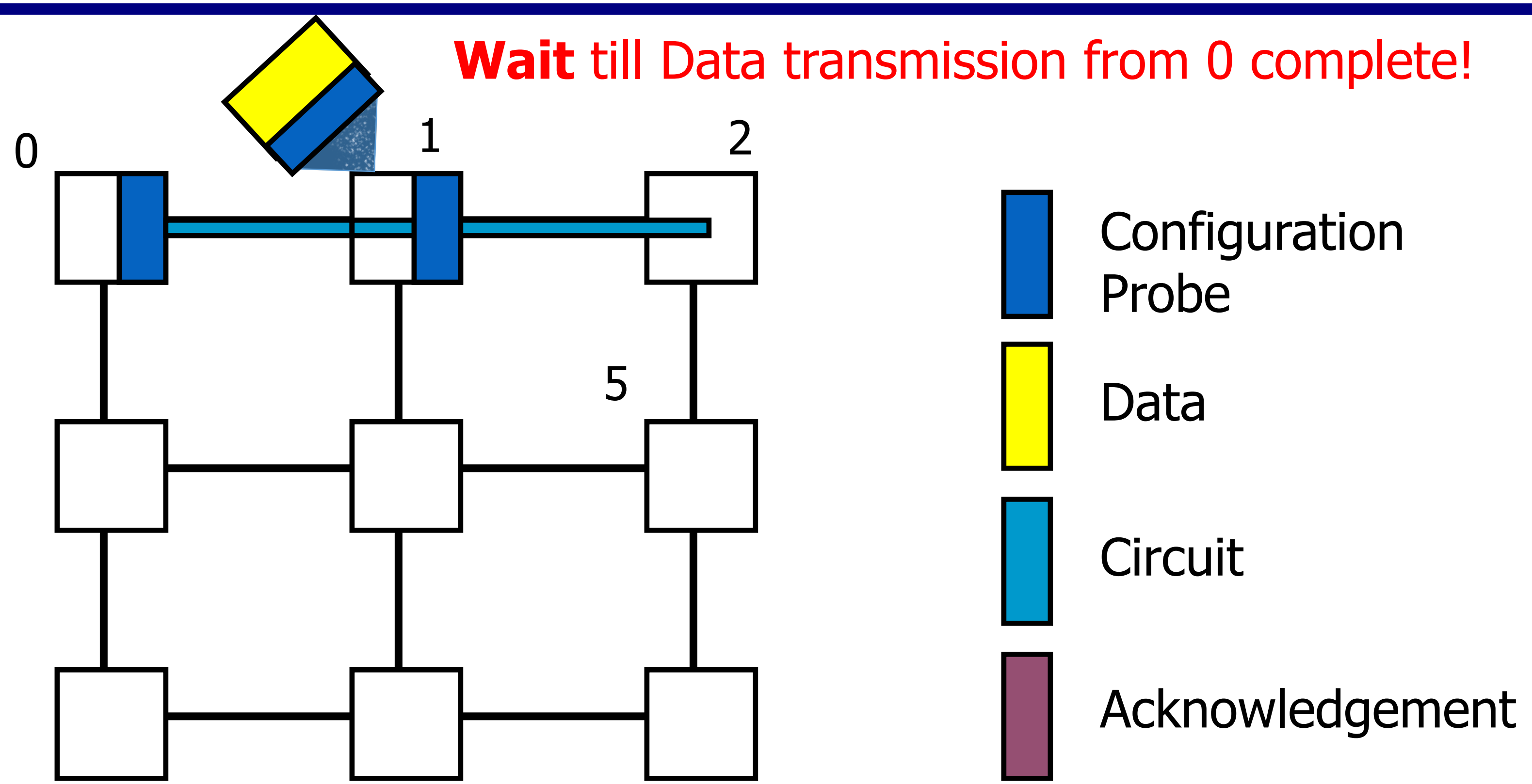
CIRCUIT SWITCHING EXAMPLE



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

HANDLING CONTENTION

Wait till Data transmission from 0 complete!



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

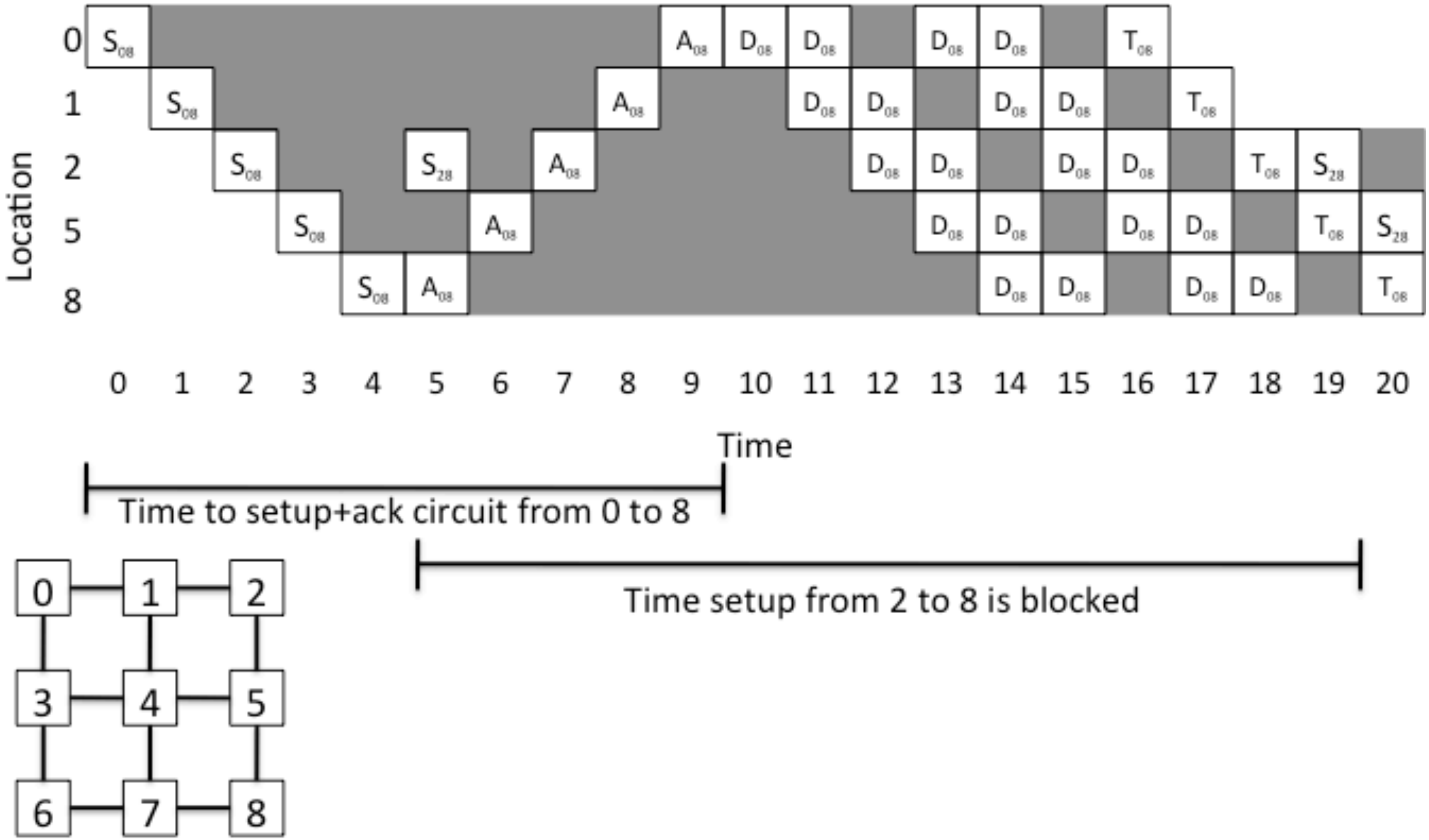
CHALLENGES WITH CIRCUIT-SWITCHING

- 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

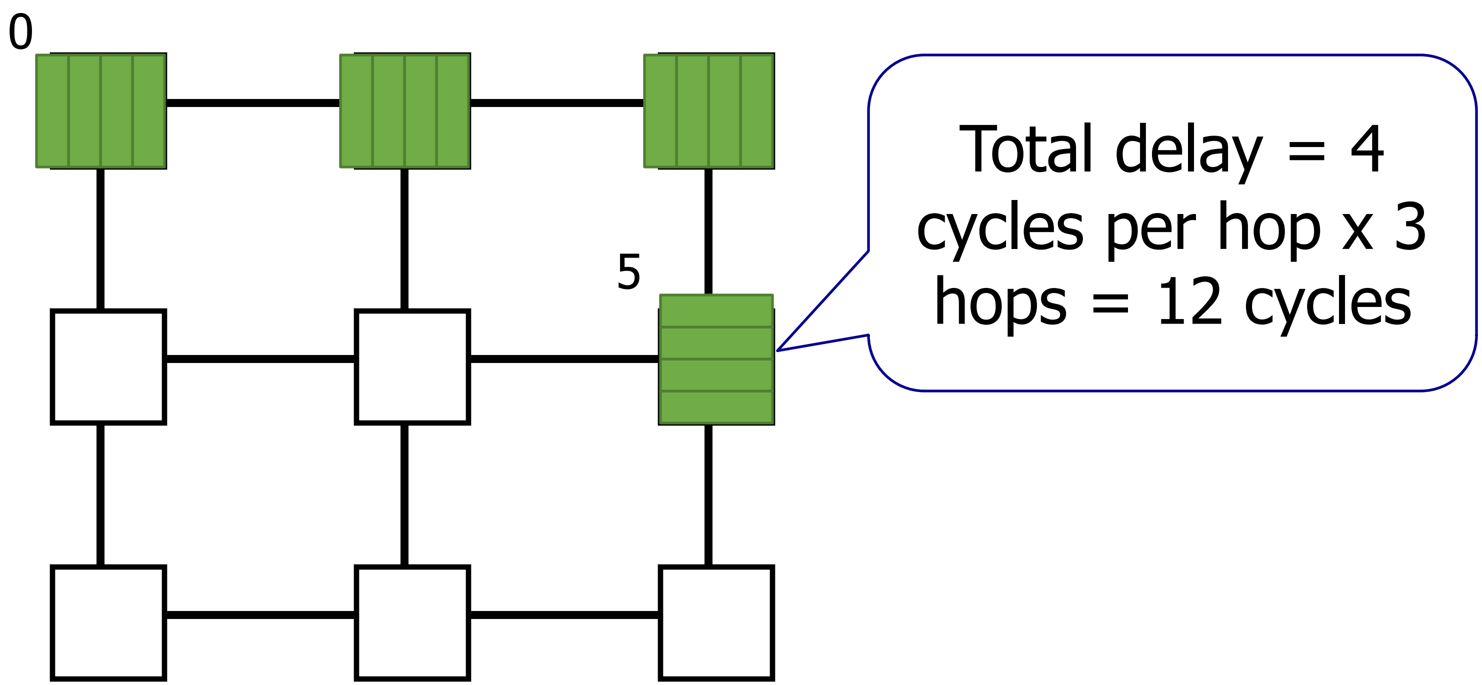
- Two techniques
  - Store and Forward
  - Virtual Cut-Through

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

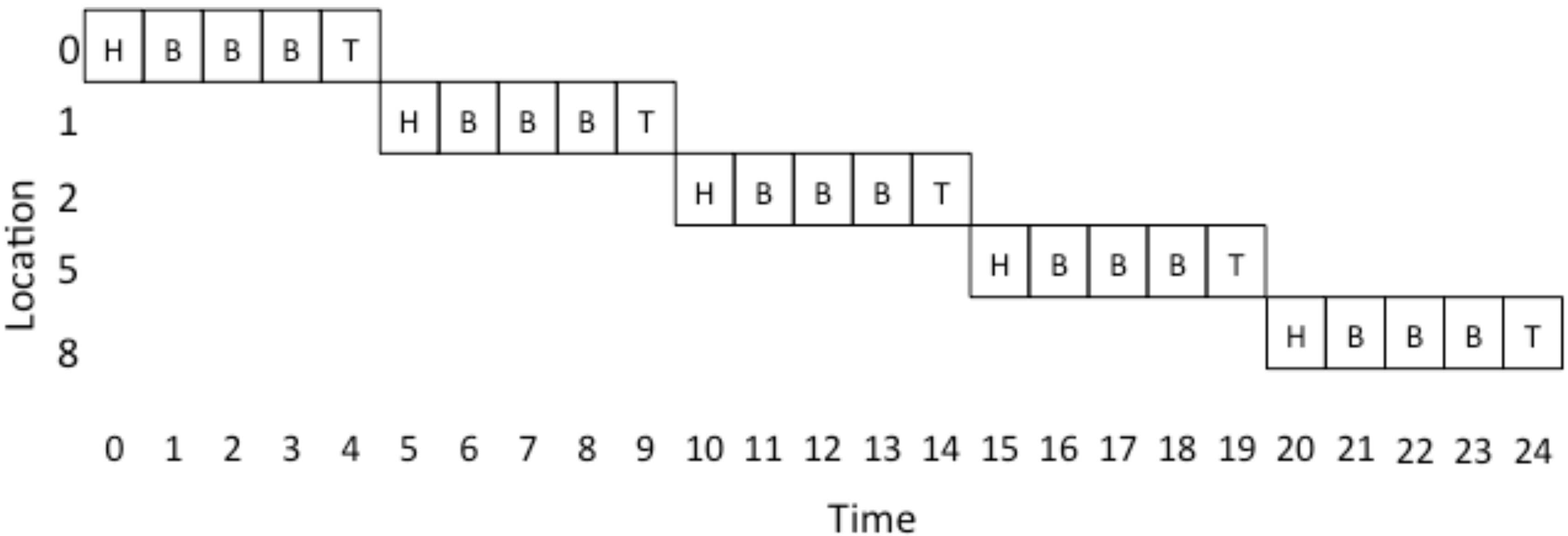
STORE AND FORWARD EXAMPLE



Not suitable on-chip.
Why?

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

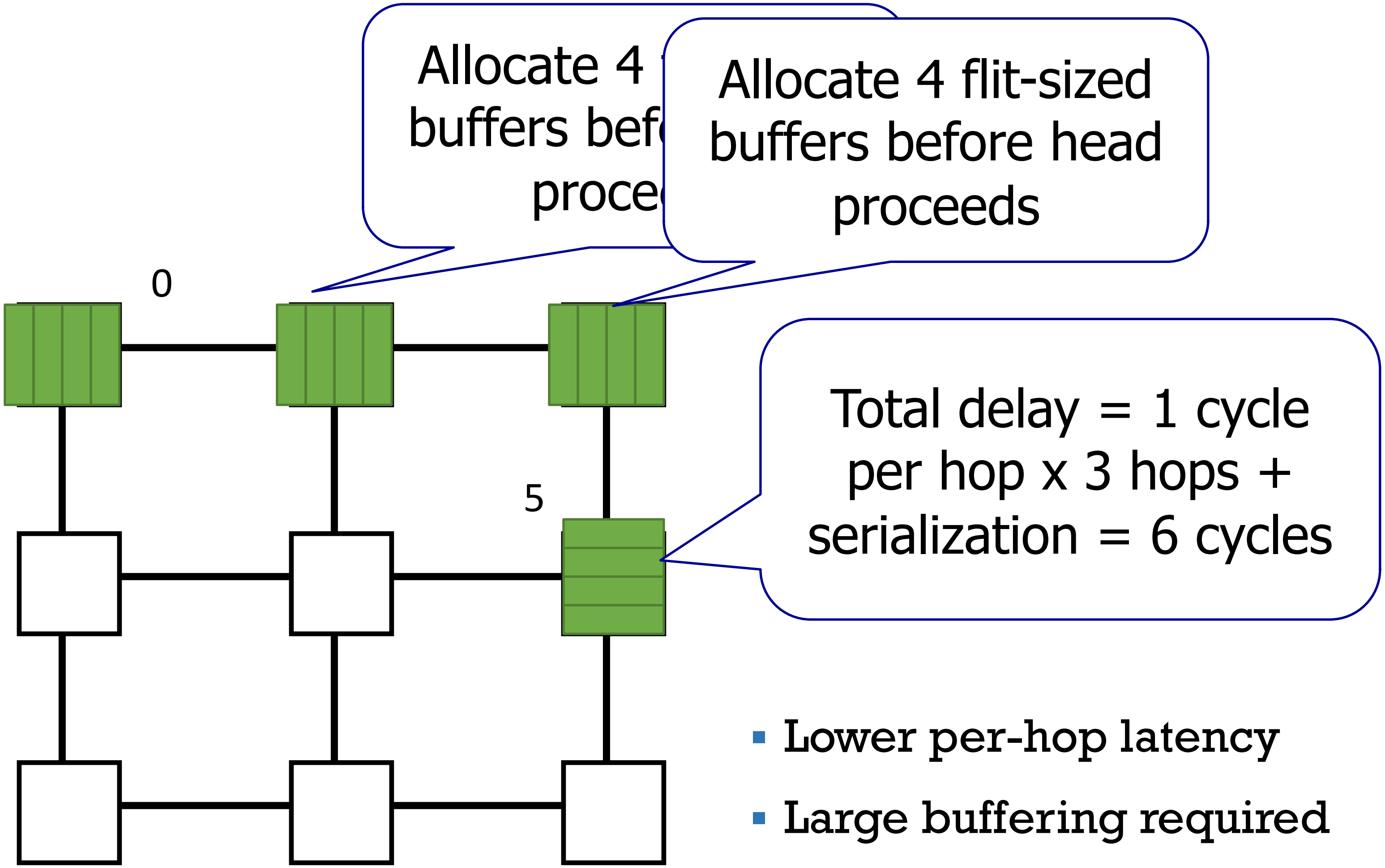
TIME-SPACE DIAGRAM: STORE AND FORWARD



PACKET-BASED: VIRTUAL CUT-THROUGH

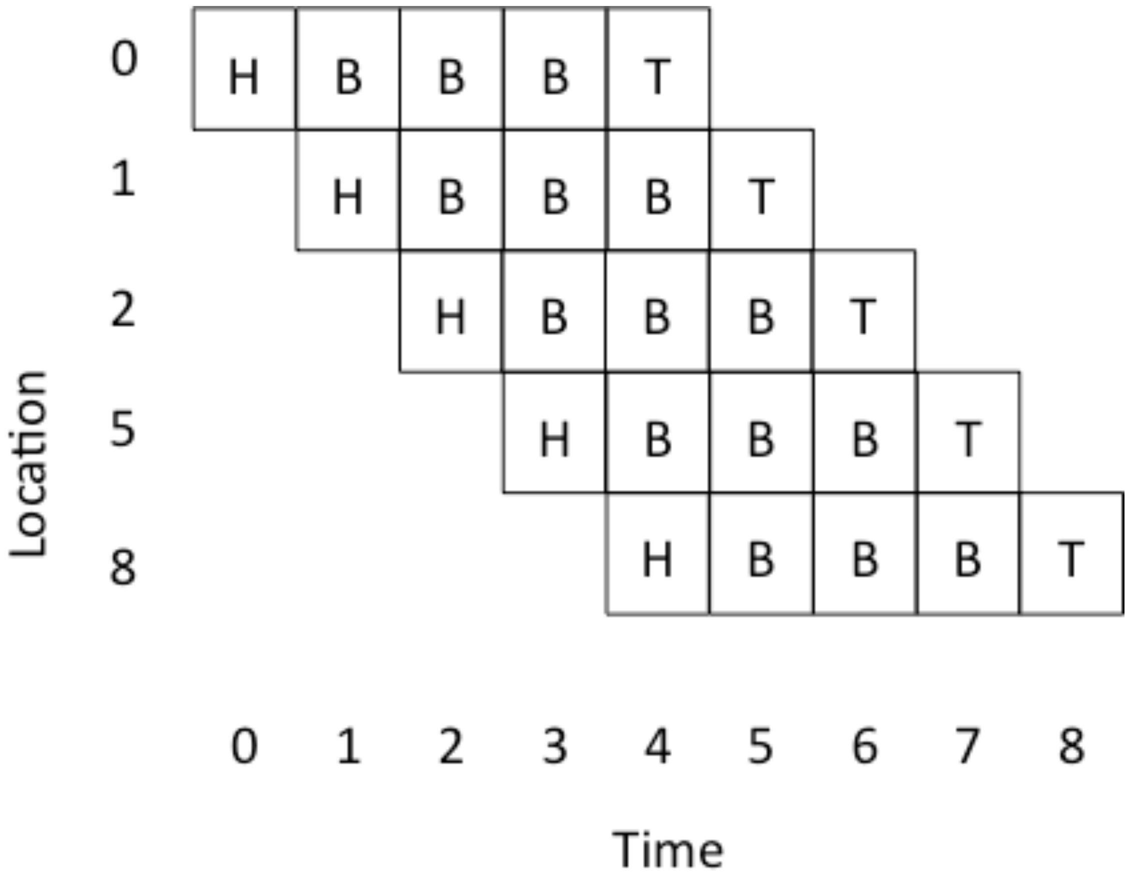
- Links and Buffers allocated to **entire** packets
- Flits can proceed to next hop before tail flit has been received by current router
  - But only if next router has enough buffer space for **entire** packet

VIRTUAL CUT-THROUGH EXAMPLE

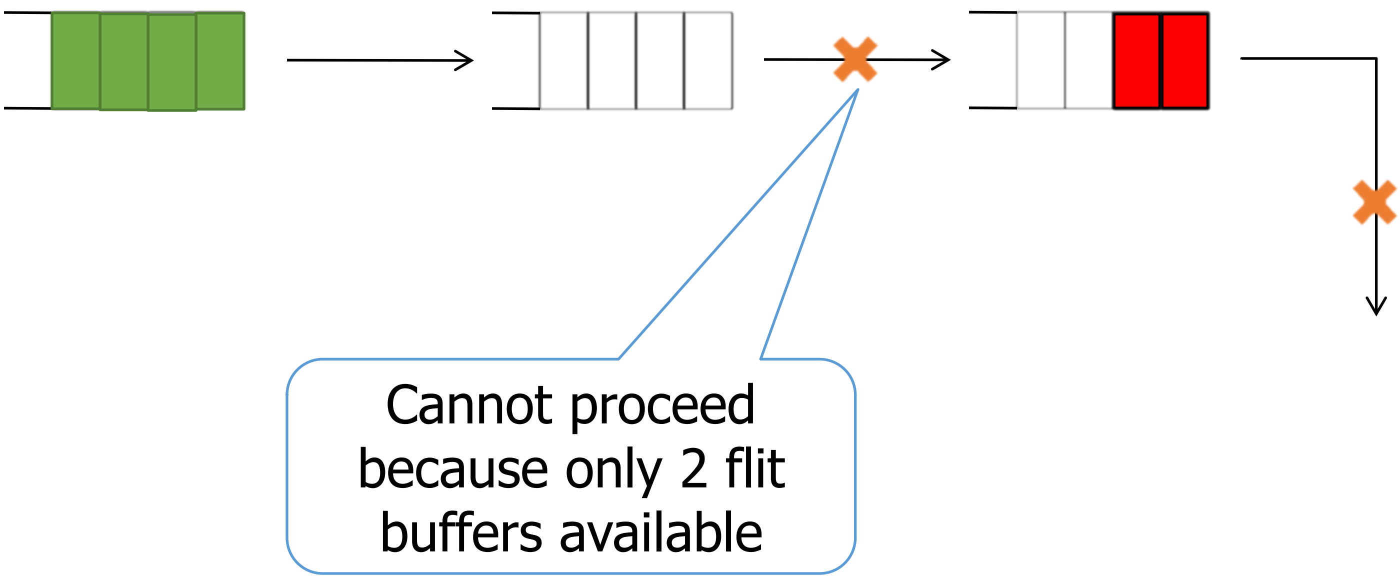


- Lower per-hop latency
- Large buffering required

TIME-SPACE DIAGRAM: VIRTUAL CUT-THROUGH

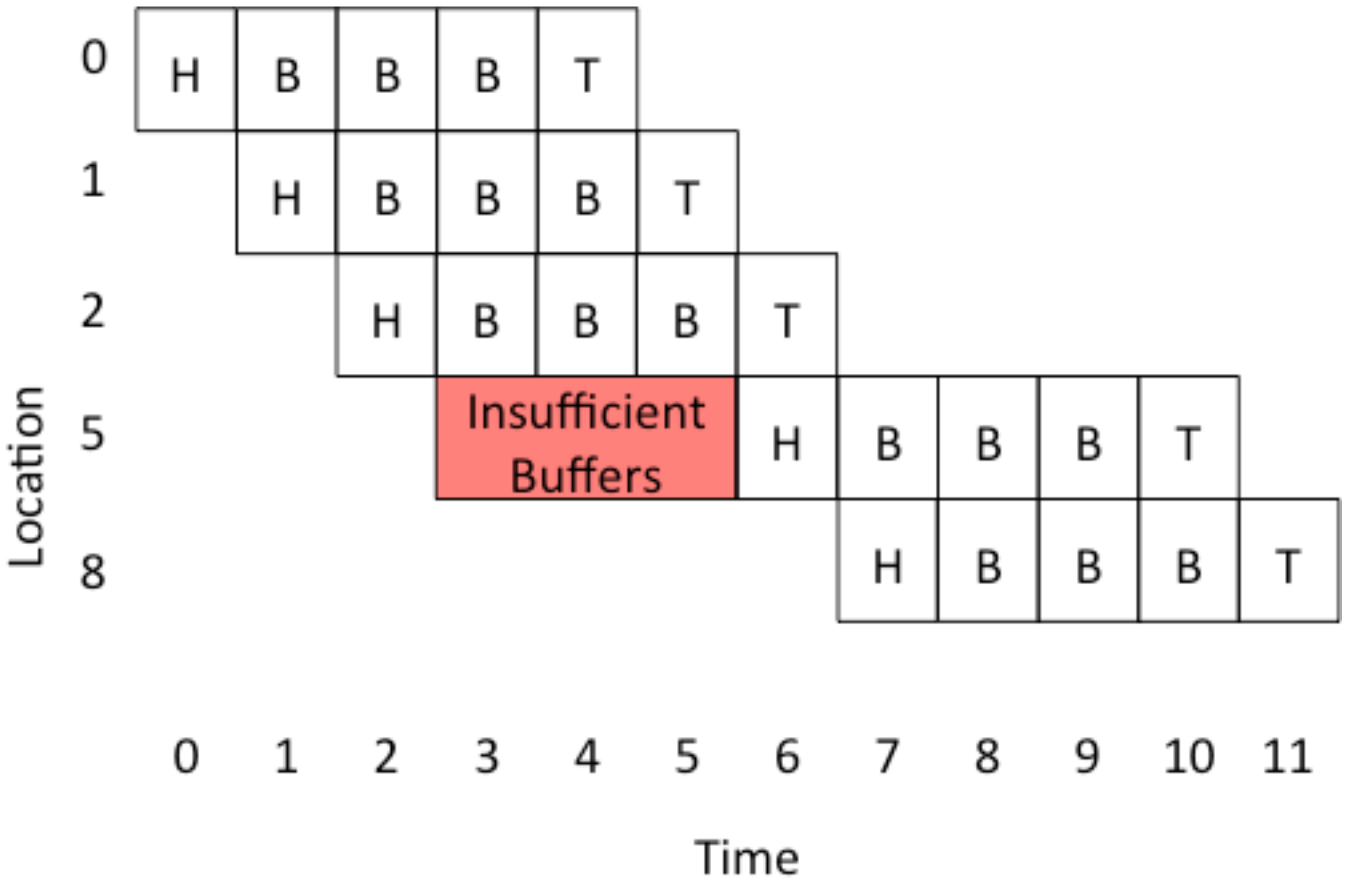


VIRTUAL CUT-THROUGH EXAMPLE (2)



Throughput suffers from inefficient buffer allocation

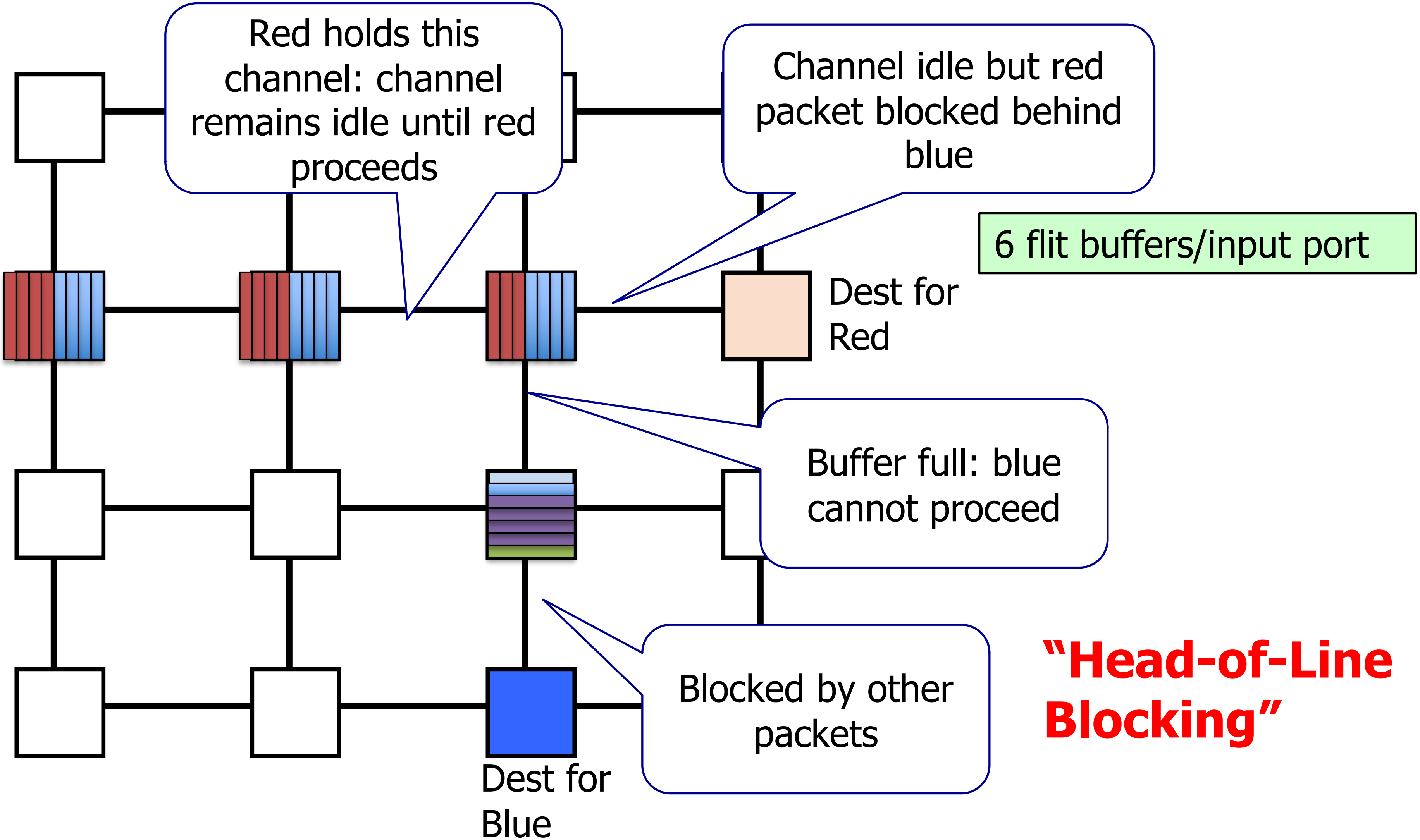
TIME-SPACE DIAGRAM: VIRTUAL CUT-THROUGH (2)



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

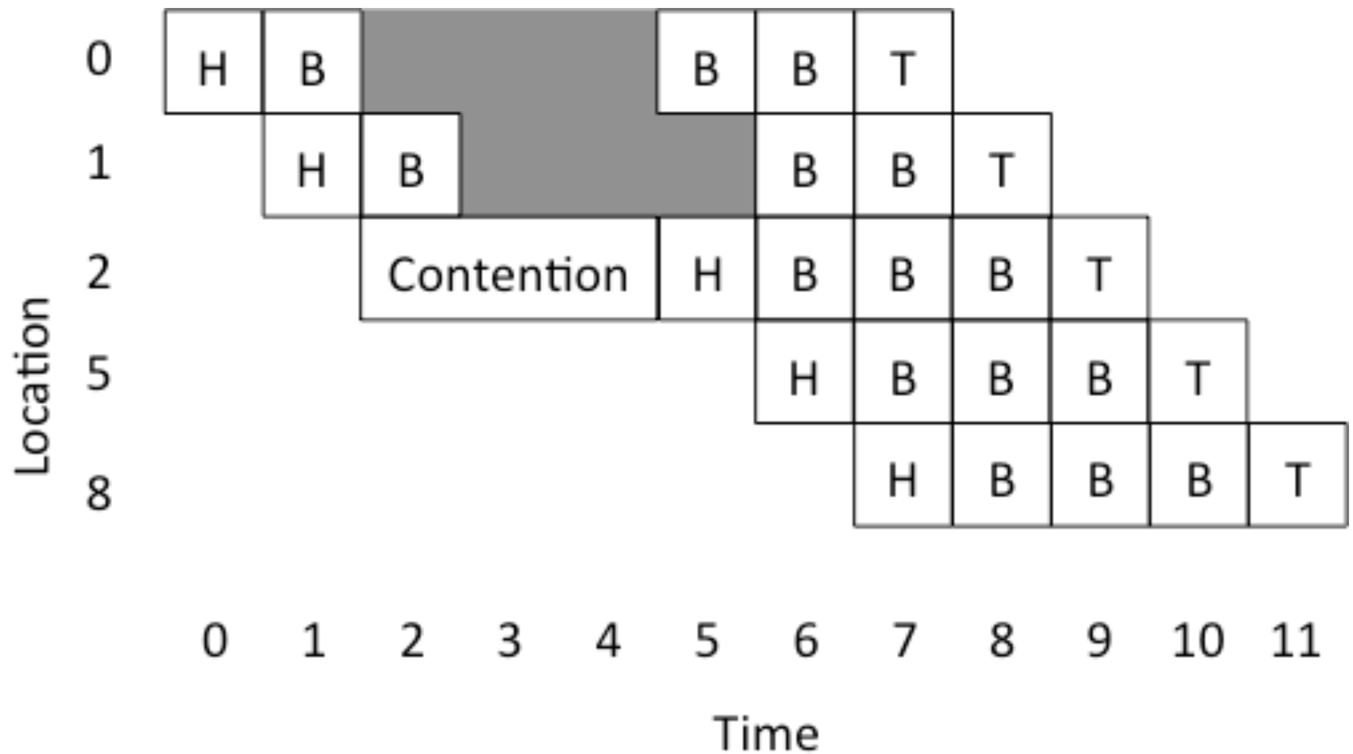


"Head-of-Line Blocking"

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

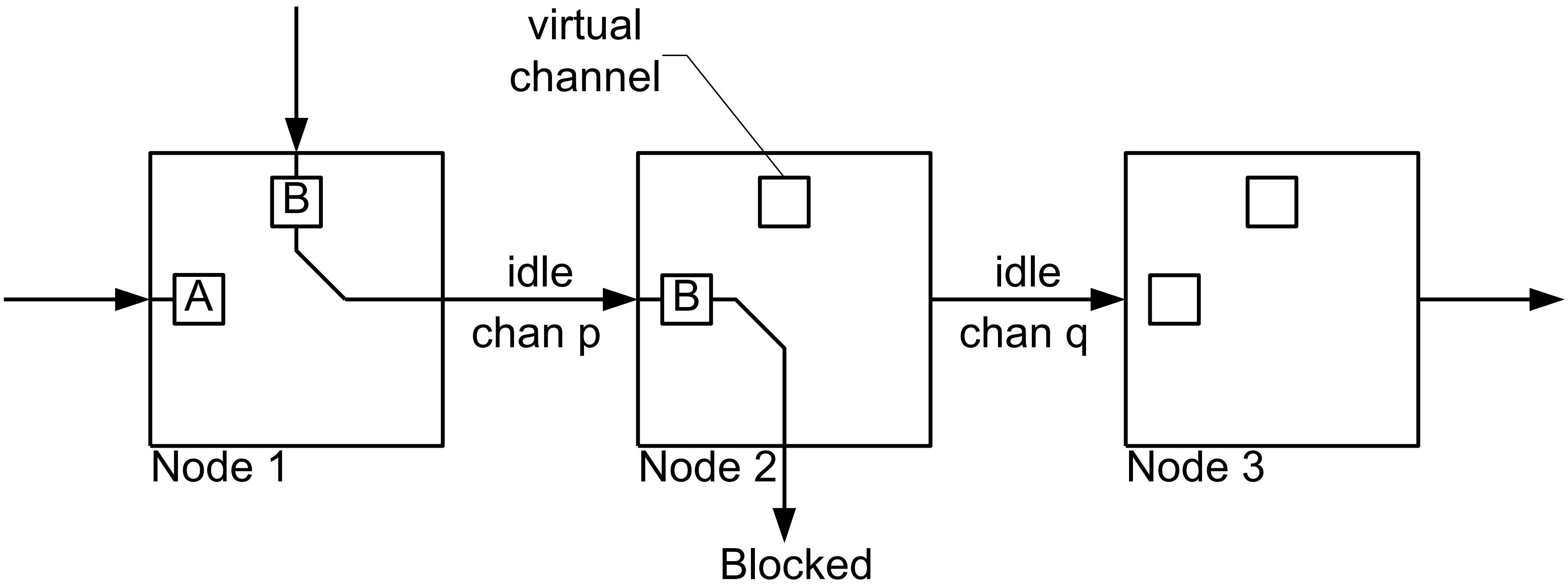
TIME-SPACE DIAGRAM: WORMHOLE



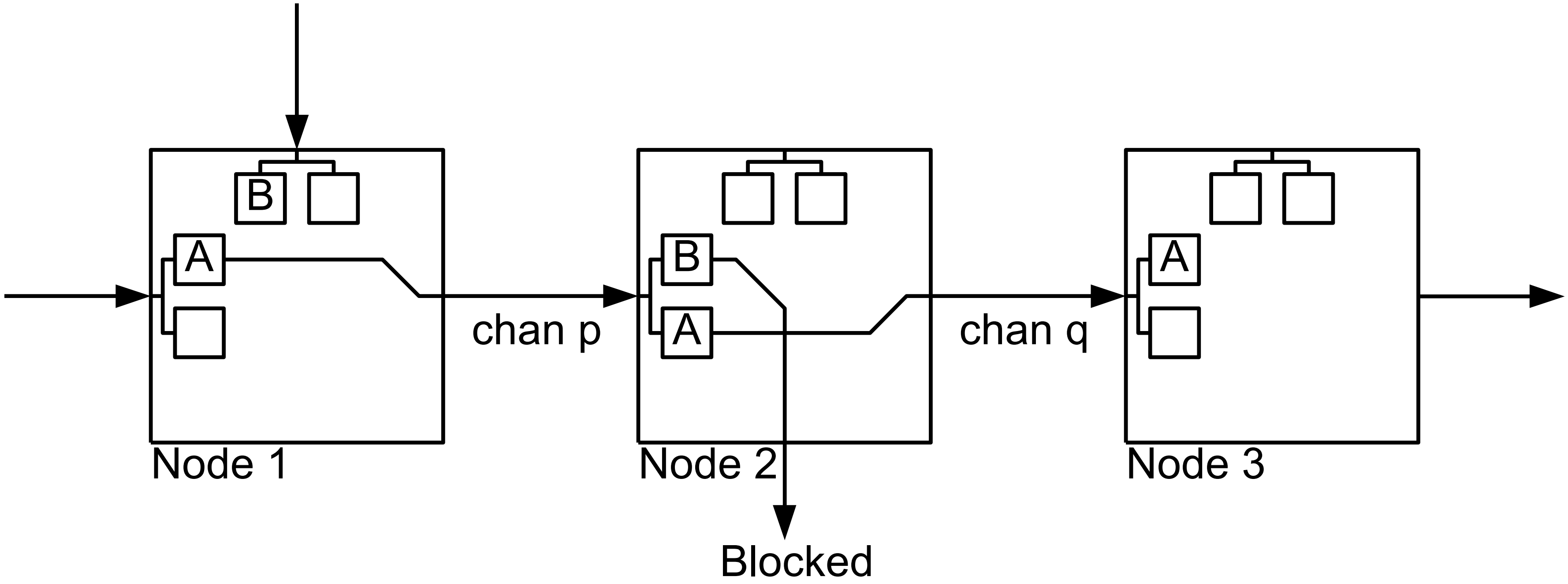
VIRTUAL CHANNEL FLOW CONTROL

- Like lanes on a highway
  - Flits on different VC can pass blocked packet
  - Link utilization improved
- Dual Use
  - Deadlock avoidance
  - Avoid Head-of-Line blocking
- Virtual channel implementation: multiple flit queues per input port
  - Share same physical link (channel)

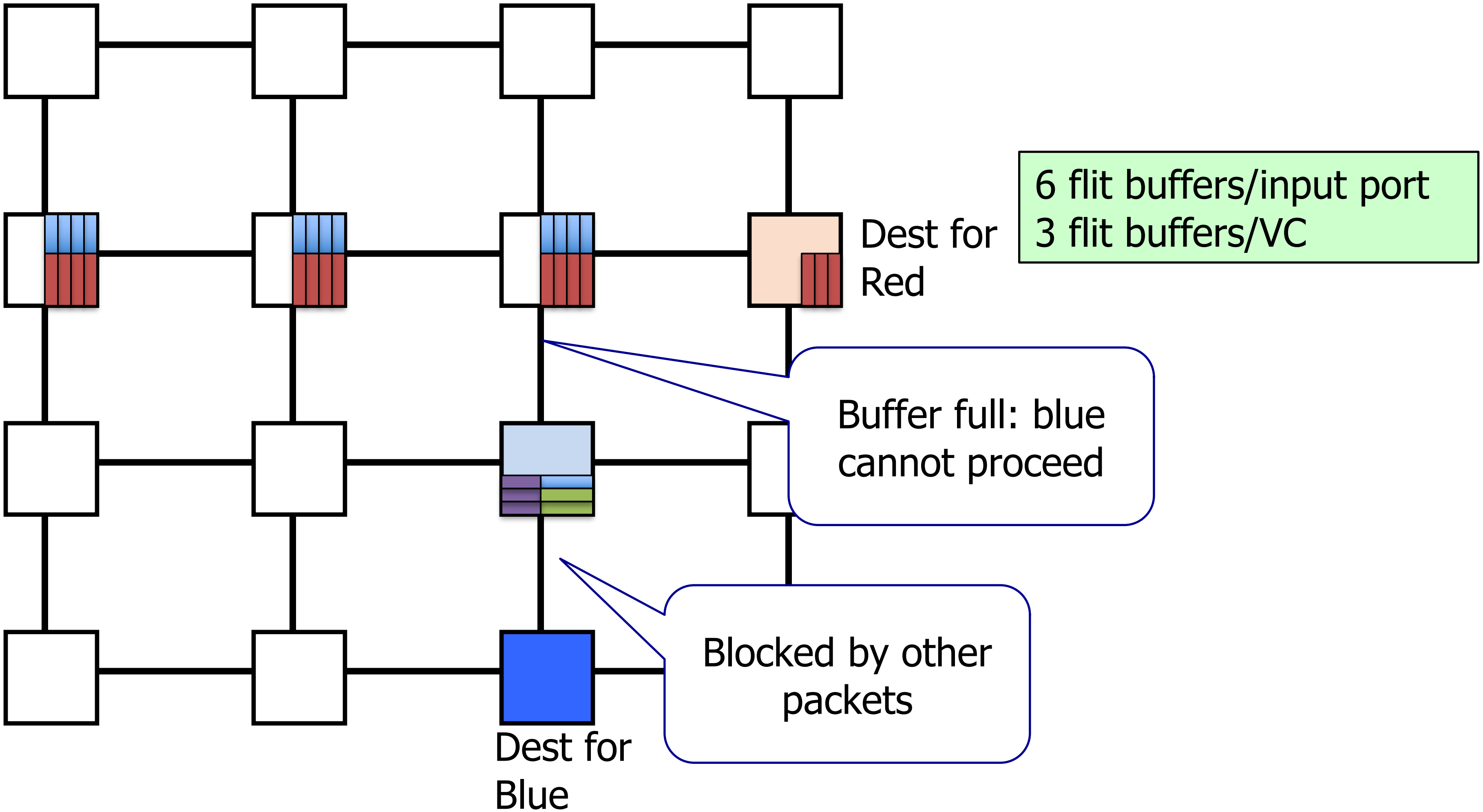
BLOCKING IN WORMHOLE FLOW CONTROL



VCS DECOUPLE DEPENDENCY BETWEEN BUFFER AND CHANNEL



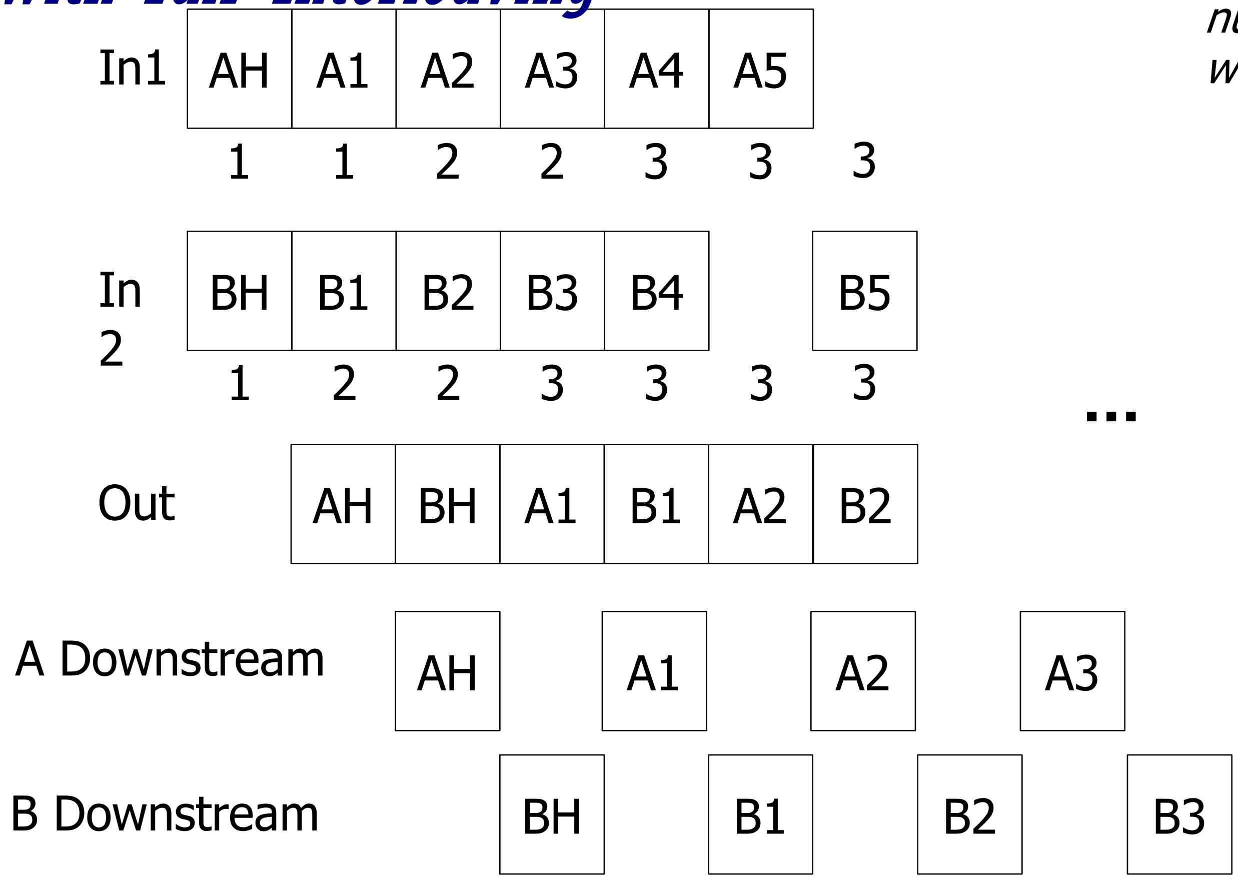
VIRTUAL CHANNEL FLOW CONTROL EXAMPLE



TIME-SPACE DIAGRAM: VC FLOW CONTROL

With Fair Interleaving

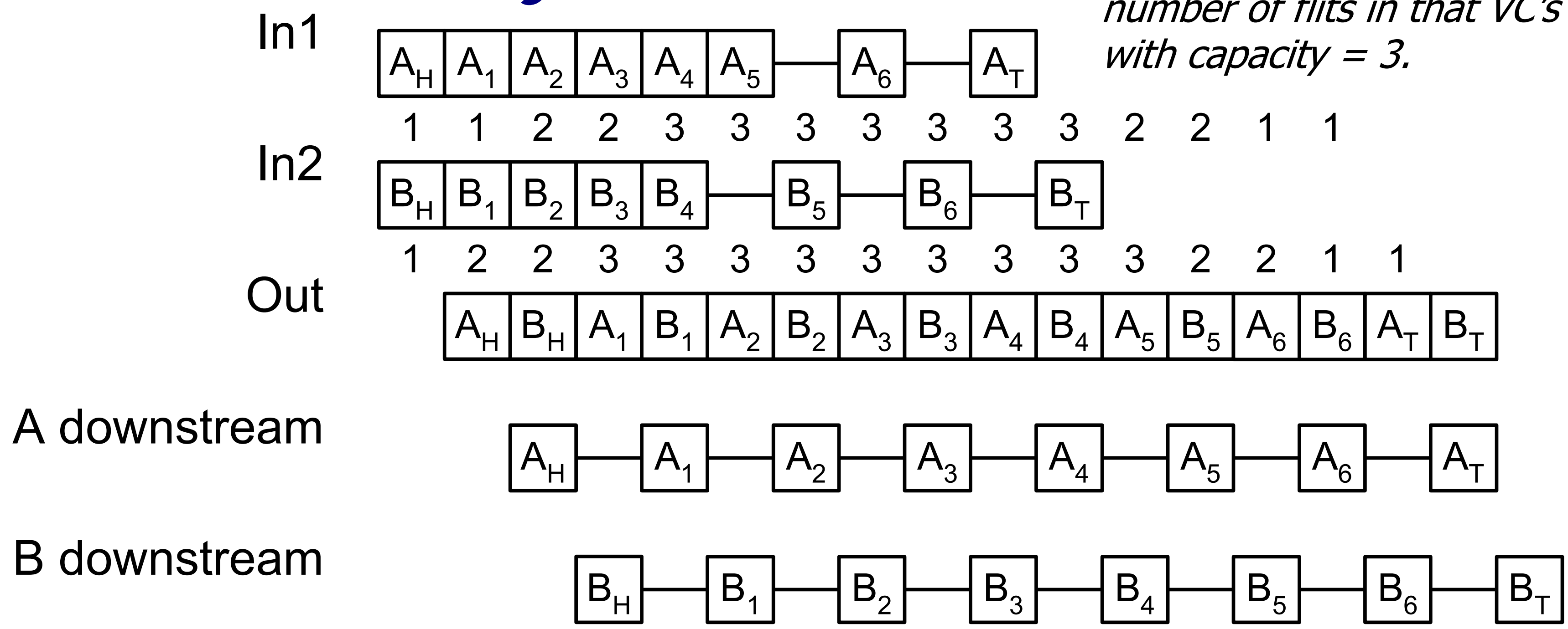
Numbers under the buffers show number of flits in that VC's buffer, with capacity = 3.



TIME-SPACE DIAGRAM: VC FLOW CONTROL

With Fair Interleaving

Numbers under the buffers show number of flits in that VC's buffer, with capacity = 3.

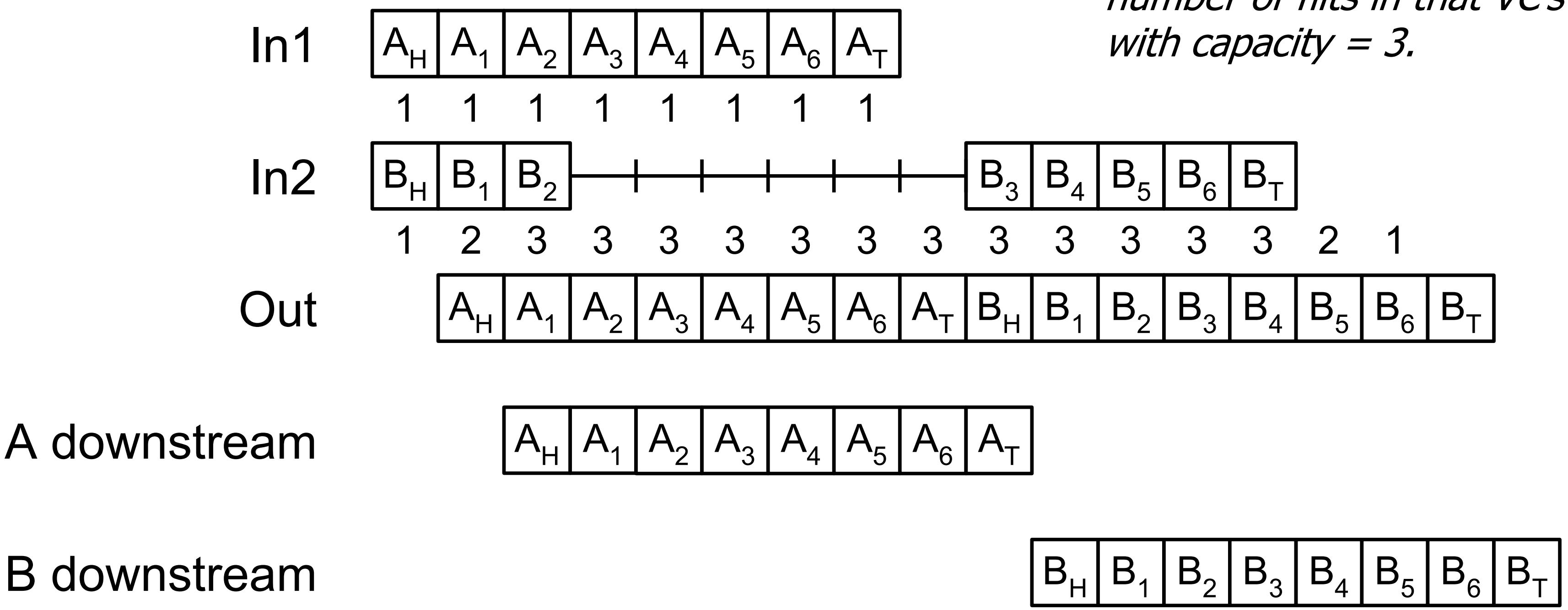


Latency of both packets got impeded due to fair interleaving!

TIME-SPACE DIAGRAM: VC FLOW CONTROL

With Winner-Takes-All

Numbers under the buffers show number of flits in that VC's buffer, with capacity = 3.



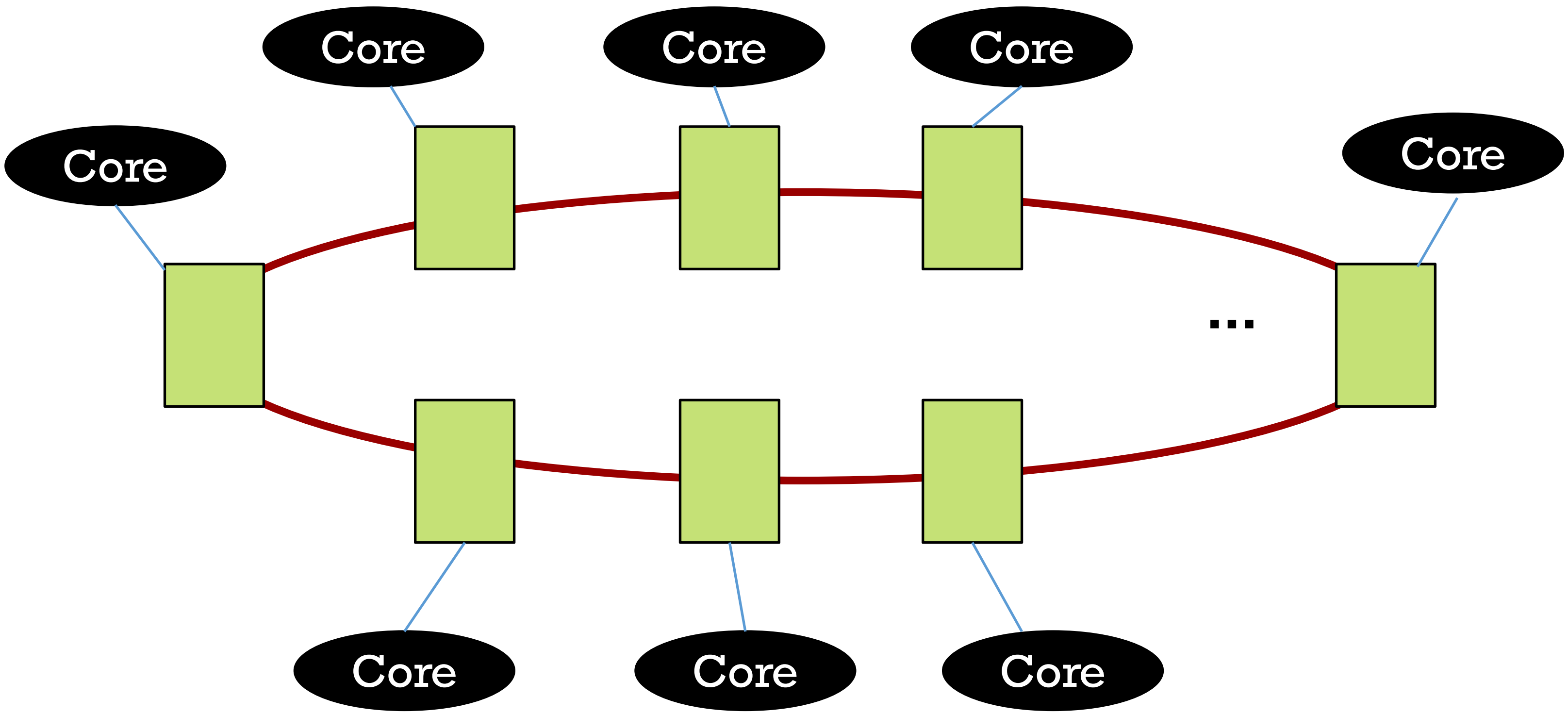
Latency of packet A goes down by 7 cycles. (zero contention latency)
Latency of packet B is unaffected
(contention latency = serialization latency of packet A)

SUMMARY OF TECHNIQUES

	Links	Buffers	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

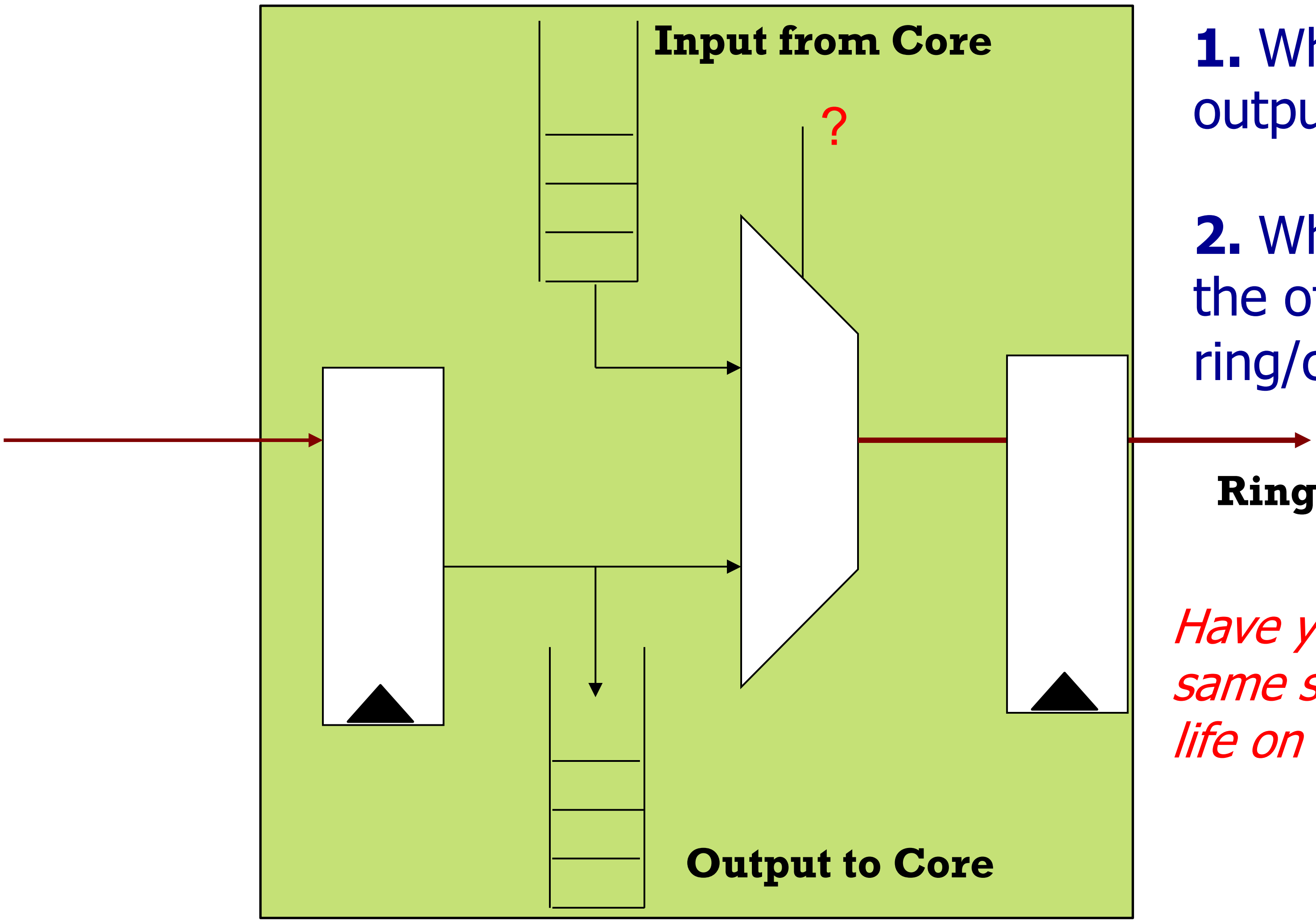
DESIGNING A FLOW CONTROL PROTOCOL: MANAGING BUFFERS AND CONTENTION

SUPPOSE WE HAVE A RING ...



For a Mesh, the analysis will be similar, with 5 ports (North, South, East, West, Core) instead of 2 (Ring, Core) ports

FLOW CONTROL PROTOCOL



1. Who should use output link?

2. What to do with the other flit (from ring/core)

Have you seen this same situation in real life on a road network?

ROTARY

1. Who should use output link?

Traffic already on ring has priority

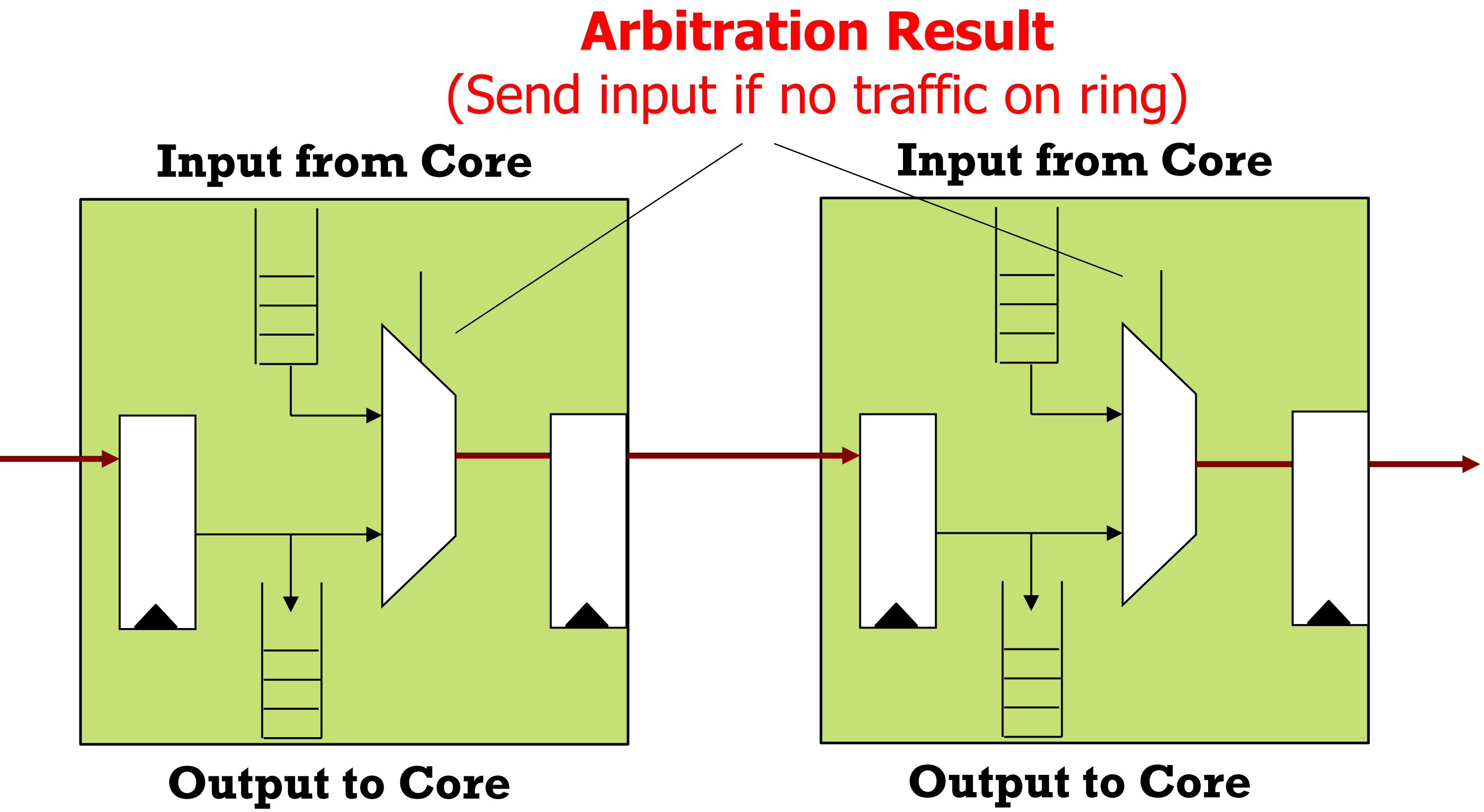
2. What to do with the other flit (from ring/core)

Wait

FLOW CONTROL PROTOCOL

This is known as "arbitration"
The control structure is called an "arbiter"

Arbiter: Decides who uses the output link.



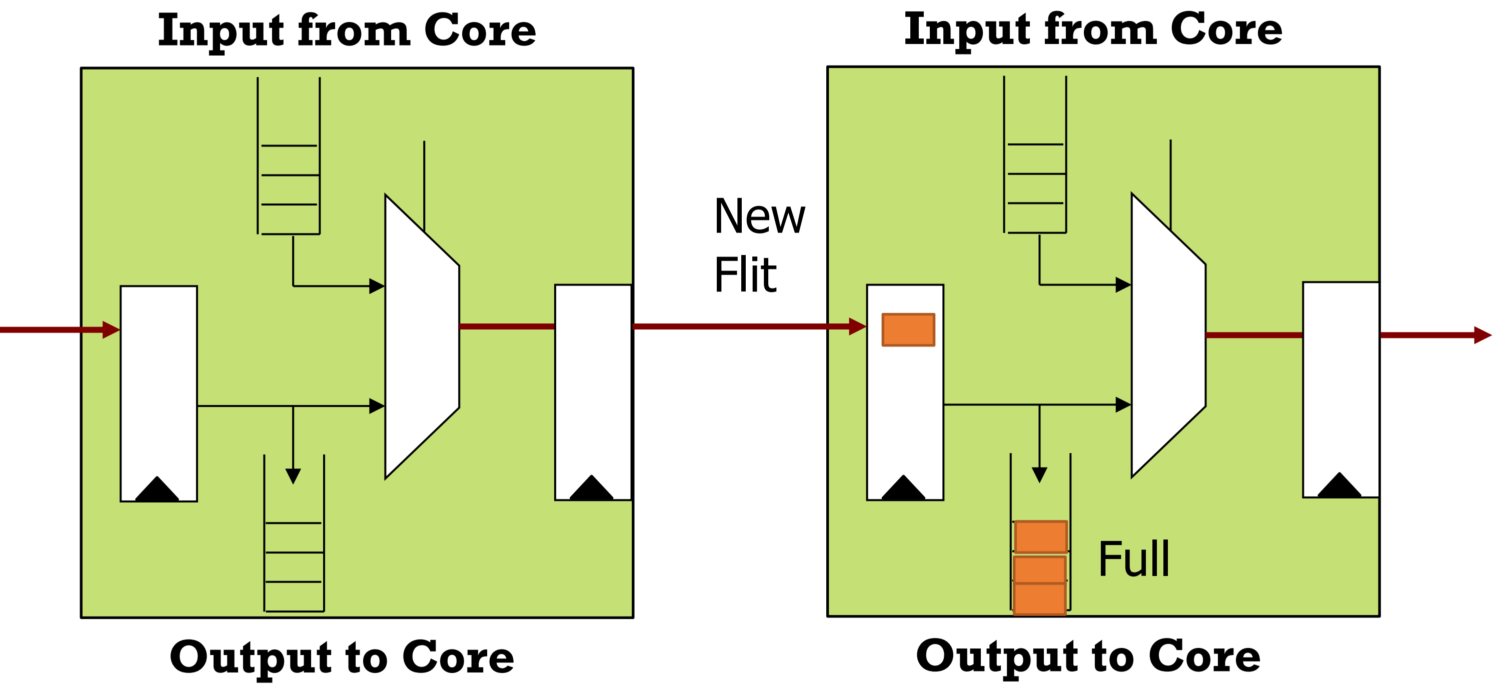
Arbitration:
Centralized or Distributed?

Centralized
- Bus
- Crossbar

Distributed
- Ring
- Mesh

FLOW CONTROL PROTOCOL

3. What should a flit do if its output is blocked?



FLOW CONTROL OPTIONS

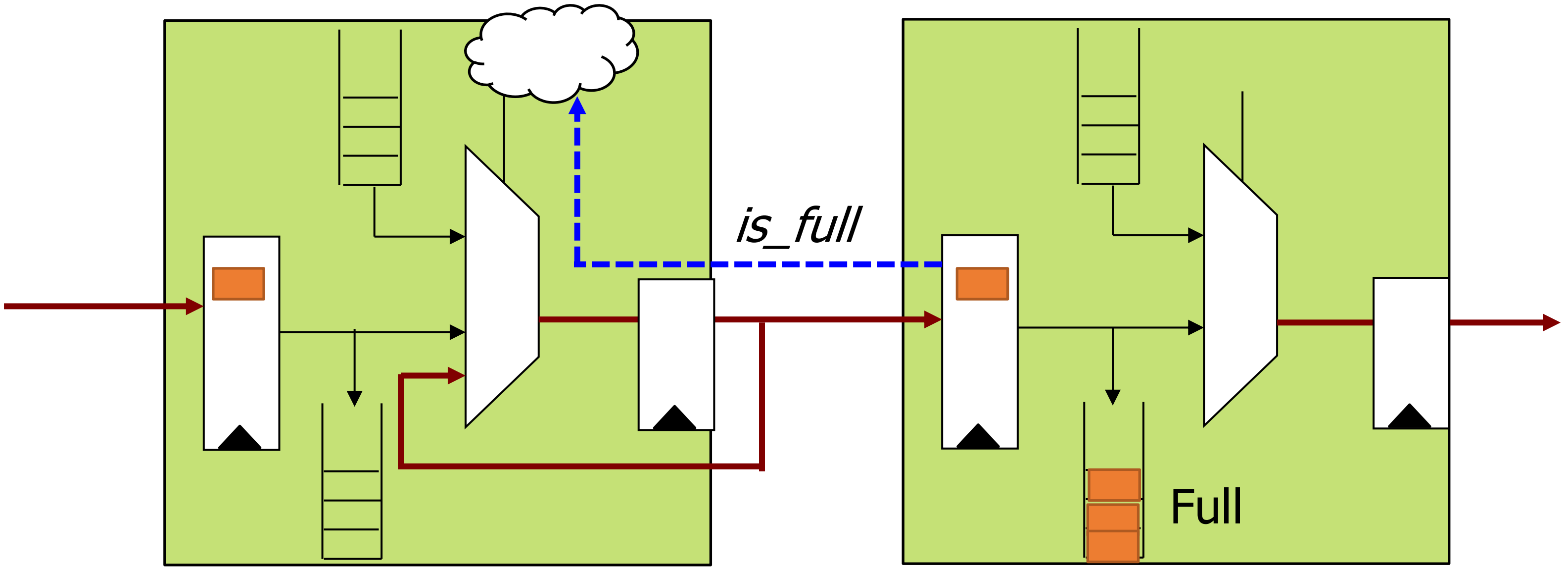
- What should a flit do if its output is blocked?
  - **Option 1:** Drop!
    - Send a NACK back for dropped packet or have a timeout
      - Source retransmits
      - Implicit congestion control
    - Flow control protocol on the Internet
    - **Advantage: can be bufferless!**
    - Challenges?
      - Latency and energy overhead of re-transmitting more than that of buffering so not preferred on-chip

FLOW CONTROL OPTIONS

- What should a flit do if its output is blocked?
  - **Option 2:** Misroute!
    - As long as N input ports and N output ports, can send flit out of some other output port
      - called "bouncing" on a ring
    - **Advantage: can be bufferless!**
    - Challenges
      - Energy
        - Routes become non-minimal – more energy consumption at router latches and on links
      - Performance
        - Non-minimal routes – can lead to longer delays
      - Correctness
        - **Pt-to-Pt ordering violation** inside protocol
          - *Need mechanism to misroute subsequent packets from same source*
        - **Livelock!** – cannot *guarantee* forward progress
          - *Need to restrict number of misroutes of same packet*

FLOW CONTROL OPTIONS

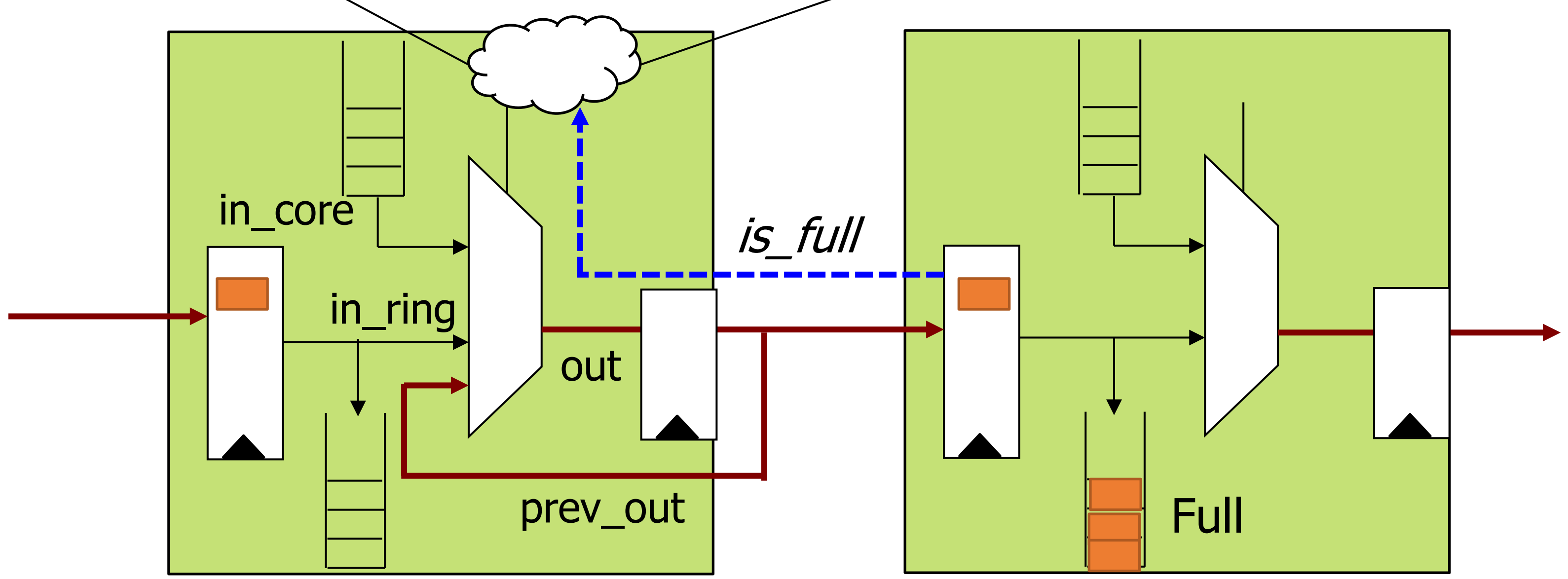
- What should a flit do if its output is blocked?
  - **Option 3:** Wait!
    - How? What about flit at previous router?
      - Signal back that it should wait too ("Backpressure")



ARBITRATION LOGIC

```
if (is_full)
    out = prev_out;
else if (in_ring.valid)
    out = in_ring;
else if (in_core.valid)
    out = in_core;
else
    out = 0;
```

Note: if we use VC flow control, some other flit going into a VC that is not blocked can use the link



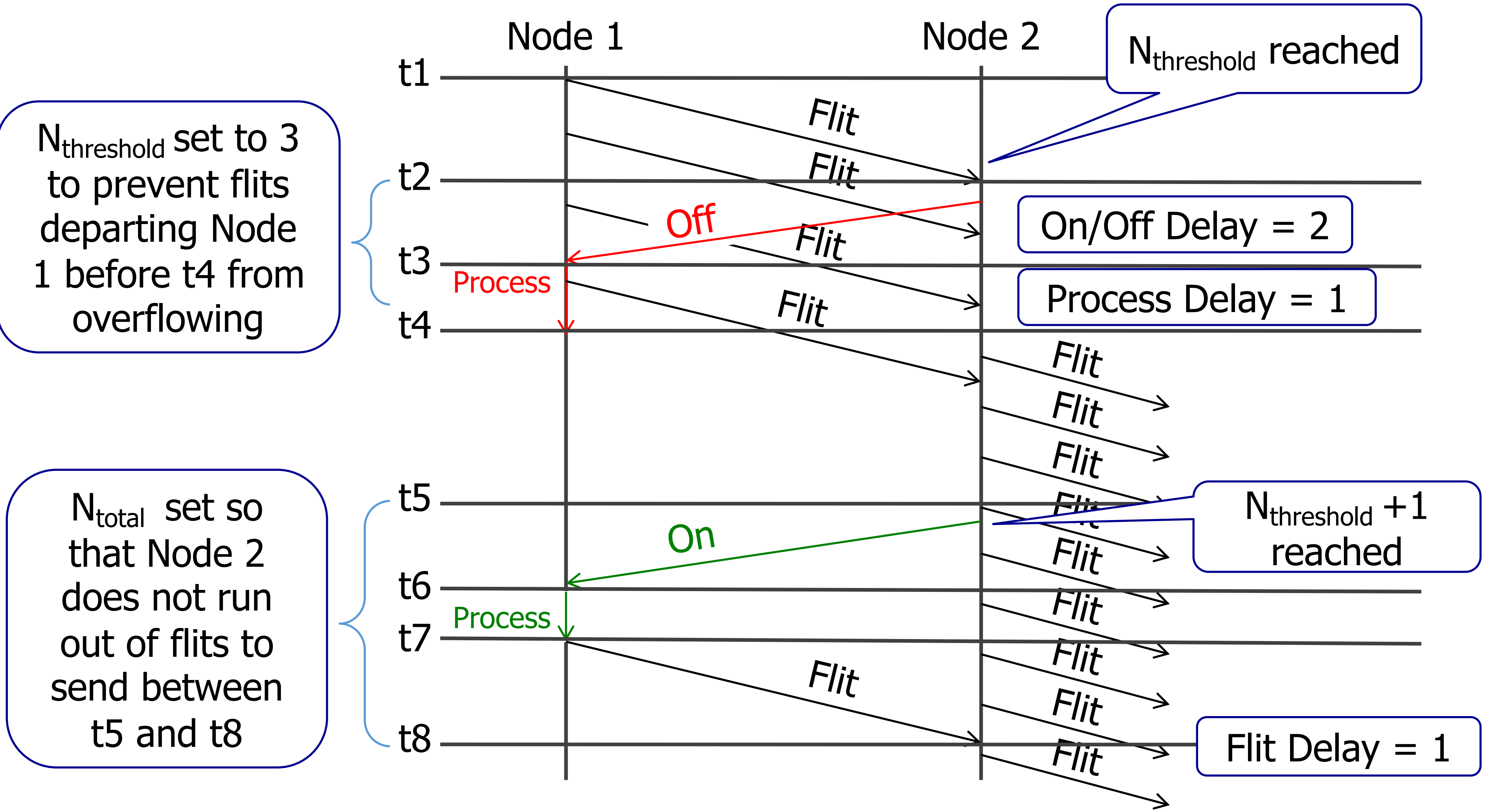
BACKPRESSURE SIGNALING MECHANISMS

- **On/Off Flow Control**
  - downstream router signals if it can receive or not
- **Credit-based Flow Control**
  - upstream router tracks the number of free buffers available at the downstream router

ON/OFF FLOW CONTROL

- Downstream router sends a 1-bit on/off if it can receive or not
  - Upstream router sends only when it sees on
- Any potential challenge?
  - Delay of on/off signal
  - By the time the on/off signal reaches upstream, there might already be flits in flight
  - Need to send the off signal *once the number of buffers reaches a threshold* such that all potential in-flight flits have a free buffer

ON/OFF TIMELINE WITH N BUFFERS



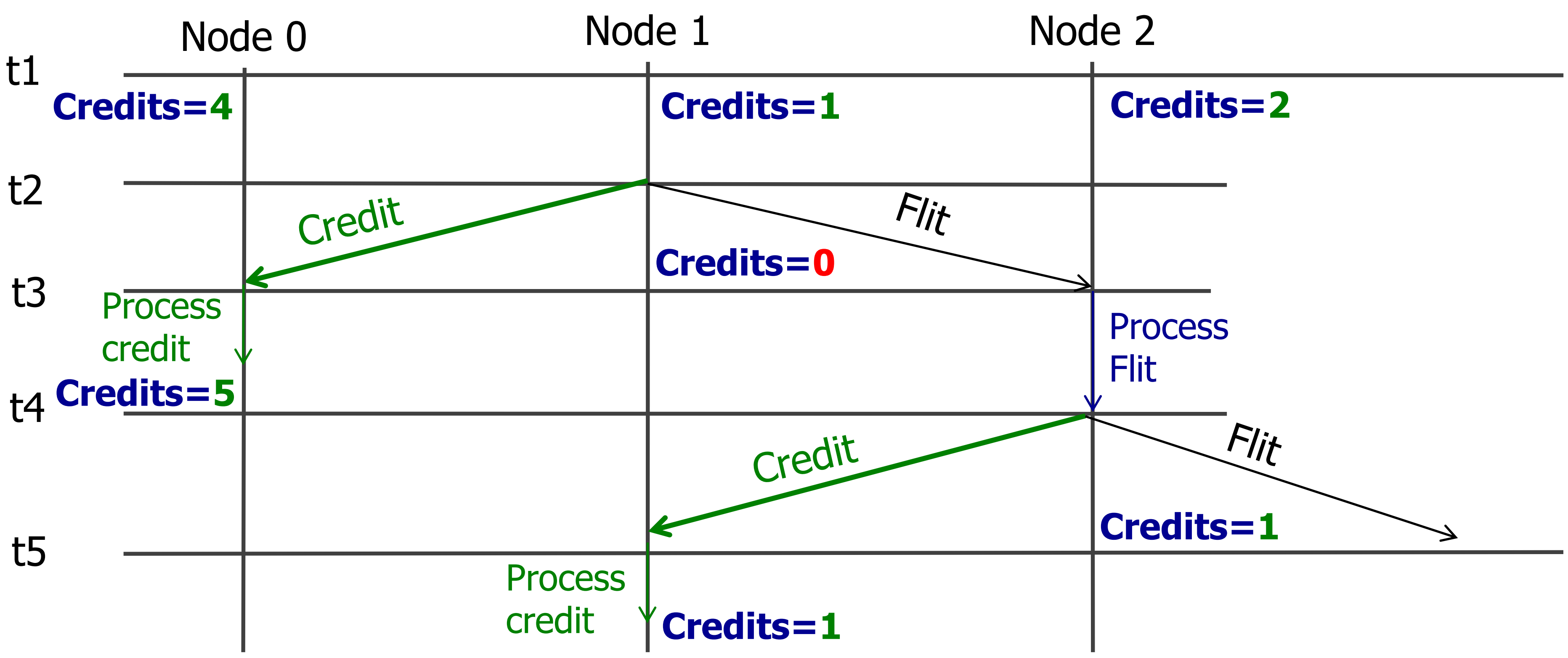
BACKPRESSURE SIGNALING MECHANISMS

- **On/Off Flow Control**
  - Pros
    - Low overhead: one-bit signal from downstream to upstream node, only switches when threshold crossed
  - Cons
    - Inefficient buffer utilization – have to design assuming worst case of $N_{threshold}$ flights in flight

CREDIT-BASED FLOW CONTROL

- **Upstream router** tracks the **number of free buffers available at the downstream router**
  - Upstream router sends only if credits > 0
- When should credit be decremented at upstream router?
  - When a flit is sent to the downstream router
- When should credit be incremented at upstream router?
  - When a flit leaves the downstream router

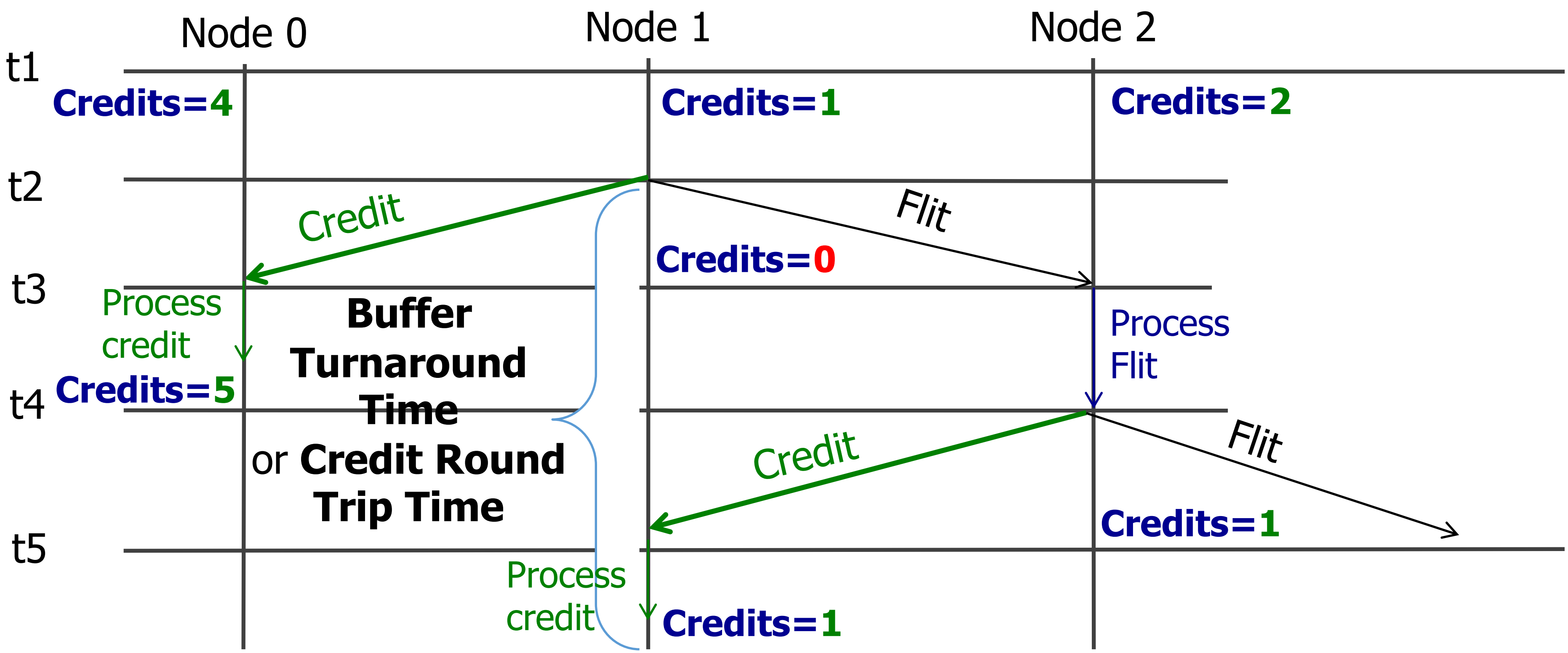
CREDIT TIMELINE



BACKPRESSURE SIGNALING MECHANISMS

- **On/Off Flow Control**
  - Pros
    - Low overhead: one-bit signal
  - Cons
    - Inefficient buffer utilization – have to design assuming worst case of $N_{threshold}$ flights in flight
- **Credit Flow Control**
  - Pros
    - Each buffer fully utilized - an keep sending till credits are zero (unlike on/off)
  - Cons
    - More signaling – need to signal upstream for every flit

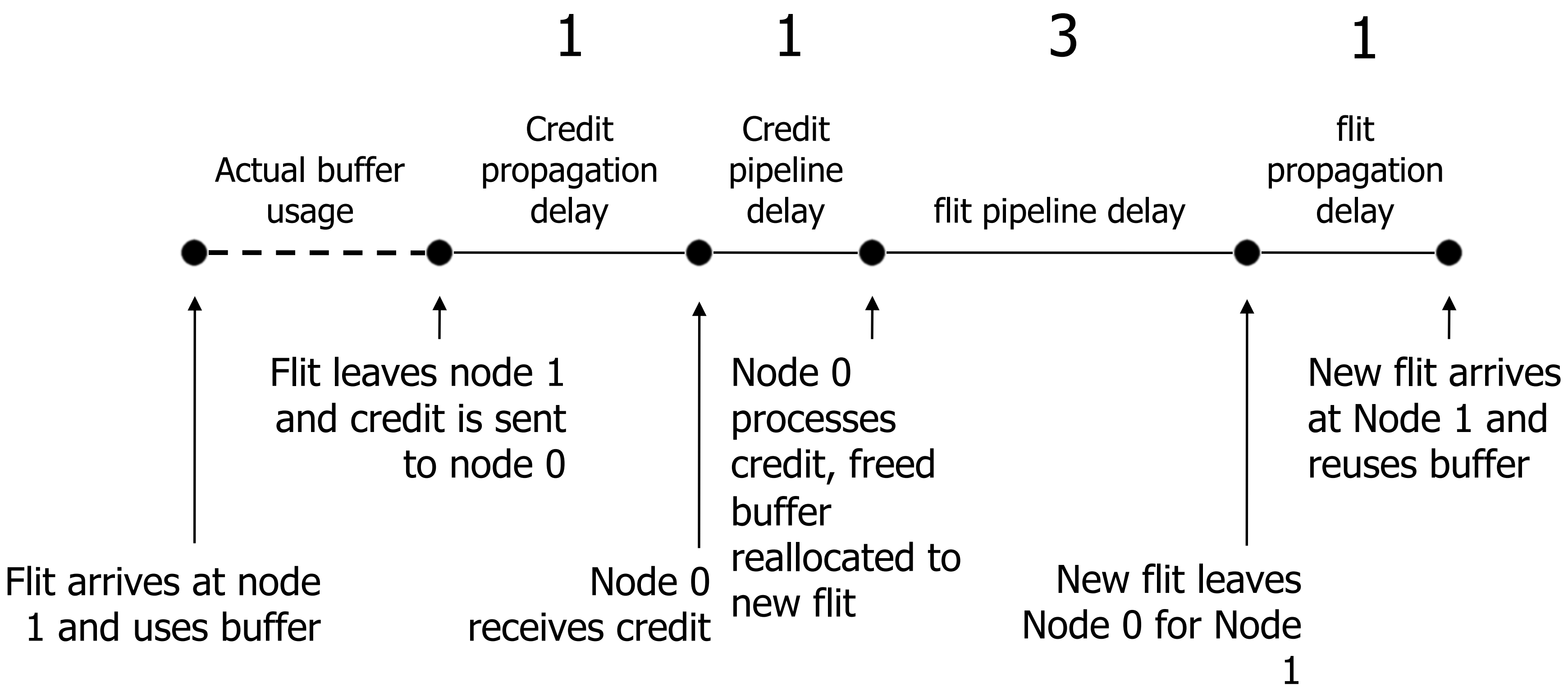
BACKPRESSURE AND BUFFER SIZING



No flit can be sent into this buffer during this delay

To prevent backpressure from limiting throughput, number of buffers >= turnaround time

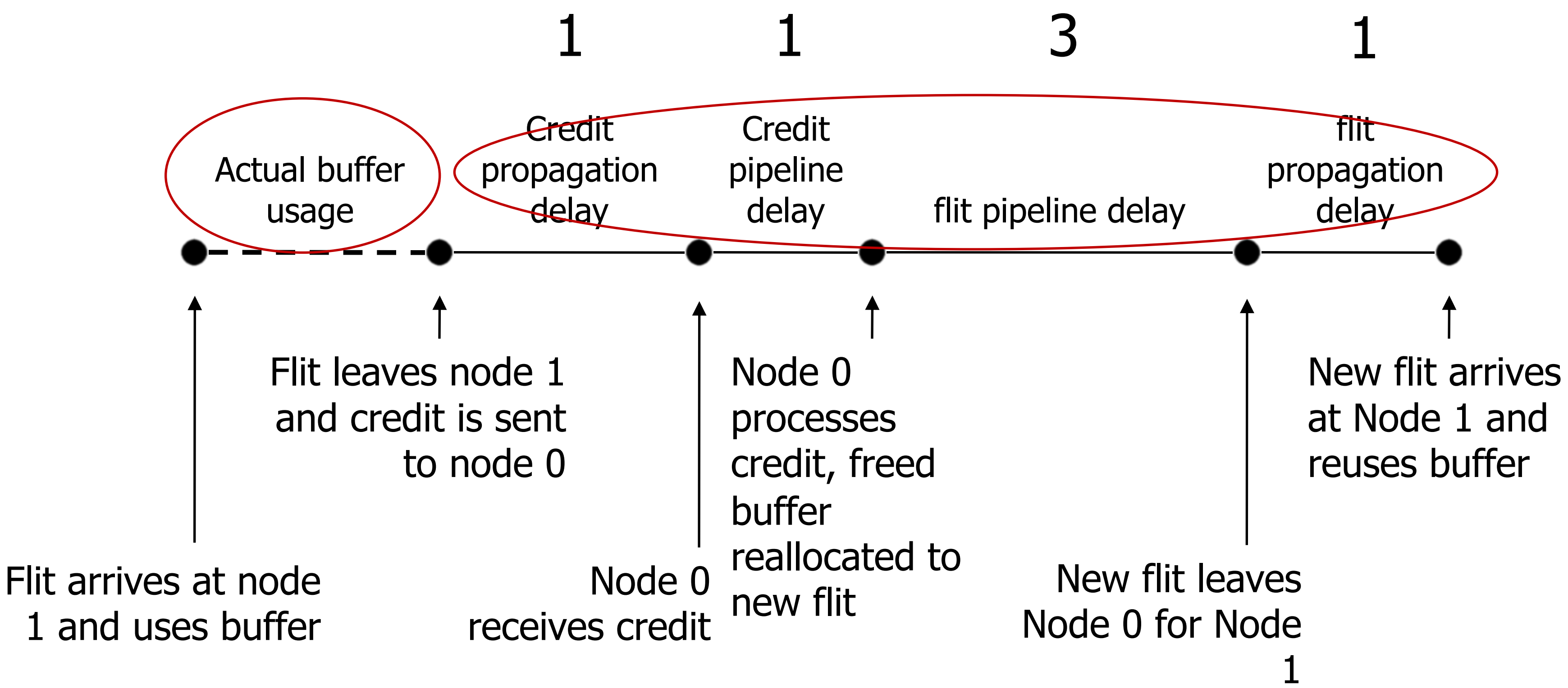
"BUFFER TURNAROUND TIME"



How many buffers needed? 1+1+3+1 = 6

How many buffers needed in on/off flow-control? 6 + 2 = 8 (off propogation + processing)

But this is inefficient



See: Flit Rsvn Flow Control, HPCA 2000