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

**Suppose Router Delay = 1, Link Delay = 1**

<table>
<thead>
<tr>
<th>Topology</th>
<th>Latency (hops) (A→B)</th>
<th>Throughput (msg/cycle) (A→B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XY</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Routing (XY)</th>
<th>Latency (hops) (A→B)</th>
<th>Throughput (msg/cycle) (A→B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XY</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Flow Control</th>
<th>Latency (hops) (A→B)</th>
<th>Throughput (msg/cycle) (A→B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Rₐ + Lₐₜₐₜ + Rₙ) + Lₜₐₜₜ (Rₙ)</td>
<td>3</td>
<td>1/2</td>
</tr>
</tbody>
</table>

Case I: One buffer at C
Case II: D→B msgs

Case II: D sends 4 messages, 1 cycle break, 4 messages, 1 cycle break...C prioritizes straight over turning traffic
**Allocation Granularity: Messages, Packets, and Flits**

- **Message**
  - **Header**
  - **Payload**

- **Packet**
  - **Route**
  - **Seq#**
  - **Head Flit**
  - **Body Flit**
  - **Tail Flit**

- **Sequence in the message**

- **Flit**
  - **Route**
  - **Type**
  - **VCID**

- **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)**

<table>
<thead>
<tr>
<th>Route</th>
<th>Type</th>
<th>VCid</th>
<th>Addr</th>
<th>Bytes 0-15</th>
<th>Bytes 16-31</th>
<th>Bytes 32-47</th>
<th>Bytes 48-63</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Head Flit</td>
<td>Body Flits</td>
<td>Tail Flit</td>
<td></td>
</tr>
</tbody>
</table>

**Cache Line Request**

<table>
<thead>
<tr>
<th>Route</th>
<th>Type</th>
<th>VCid</th>
<th>Addr</th>
<th>Cmd</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Head_Tail Flit</td>
</tr>
</tbody>
</table>

- **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
When there is contention

- Significant wait time
- Message from 1 → 2 must wait

Wait till Data transmission from 0 complete!
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

Time to setup+ack circuit from 0 to 8

Time setup from 2 to 8 is blocked
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
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

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

Total delay = 1 cycle per hop x 3 hops + serialization = 6 cycles

Allocate 4 flit-sized buffers before head proceeds
TIME-SPACE DIAGRAM: VIRTUAL CUT-THROUGH
VIRTUAL CUT-THROUGH EXAMPLE (2)

Throughput suffers from inefficient buffer allocation.

Cannot proceed because only 2 flit buffers available.
TIME-SPACE DIAGRAM: VIRTUAL CUT-THROUGH (2)

The diagram illustrates a time-space diagram for virtual cut-through flow control. The diagram shows the progression of packets (denoted by 'H', 'B', 'B', and 'T') through different locations and time slots. Locations are represented vertically, with '0' at the top and '8' at the bottom. Time is represented horizontally from '0' to '11'. The shaded red box indicates an area where there are insufficient buffers to accommodate the packets, highlighting potential congestion points in the flow control mechanism.
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
Red holds this channel: channel remains idle until red proceeds

Channel idle but red packet blocked behind blue

6 flit buffers/input port

Dest for Red

Buffer full: blue cannot proceed

Blocked by other packets

“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

![Diagram showing time-space visualization of wormhole with location and time axes. The diagram illustrates the concept of contentions in a network context.](image-url)
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

Node 1

Node 2

Node 3

virtual channel

idle
chan p

idle
chan q

Blocked

A

B

B

A

Node 1

Node 2

Node 3
VCS DECOUPLE DEPENDENCY BETWEEN BUFFER AND CHANNEL

Node 1

Node 2

Node 3

 blocked

chan p

chan q
VIRTUAL CHANNEL FLOW CONTROL EXAMPLE

Dest for Blue

Buffer full: blue cannot proceed

Blocked by other packets

Dest for Red

6 flit buffers/input port
3 flit buffers/VC
### TIME-SPACE DIAGRAM: VC FLOW CONTROL

**With Fair Interleaving**

<table>
<thead>
<tr>
<th></th>
<th>In1</th>
<th></th>
<th>In2</th>
<th></th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AH</td>
<td>A1</td>
<td>A2</td>
<td>A3</td>
<td>A4</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>BH</td>
<td>B1</td>
<td>B2</td>
<td>B3</td>
<td>B4</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

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

**A Downstream**

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AH</td>
</tr>
<tr>
<td>A1</td>
</tr>
<tr>
<td>A2</td>
</tr>
<tr>
<td>A3</td>
</tr>
</tbody>
</table>

**B Downstream**

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BH</td>
</tr>
<tr>
<td>B1</td>
</tr>
<tr>
<td>B2</td>
</tr>
<tr>
<td>B3</td>
</tr>
</tbody>
</table>
**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.

In1

<table>
<thead>
<tr>
<th></th>
<th>A_H</th>
<th>A_1</th>
<th>A_2</th>
<th>A_3</th>
<th>A_4</th>
<th>A_5</th>
<th>A_6</th>
<th>A_T</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

In2

<table>
<thead>
<tr>
<th></th>
<th>B_H</th>
<th>B_1</th>
<th>B_2</th>
<th>B_3</th>
<th>B_4</th>
<th>B_5</th>
<th>B_6</th>
<th>B_T</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Out

<table>
<thead>
<tr>
<th></th>
<th>A_H</th>
<th>B_H</th>
<th>A_1</th>
<th>B_1</th>
<th>A_2</th>
<th>B_2</th>
<th>A_3</th>
<th>B_3</th>
<th>A_4</th>
<th>B_4</th>
<th>A_5</th>
<th>B_5</th>
<th>A_6</th>
<th>B_6</th>
<th>A_T</th>
<th>B_T</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

A downstream

<table>
<thead>
<tr>
<th></th>
<th>A_H</th>
<th>A_1</th>
<th>A_2</th>
<th>A_3</th>
<th>A_4</th>
<th>A_5</th>
<th>A_6</th>
<th>A_T</th>
</tr>
</thead>
</table>

B downstream

<table>
<thead>
<tr>
<th></th>
<th>B_H</th>
<th>B_1</th>
<th>B_2</th>
<th>B_3</th>
<th>B_4</th>
<th>B_5</th>
<th>B_6</th>
<th>B_T</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>Links</th>
<th>Buffers</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circuit-Switching</td>
<td>Messages</td>
<td>N/A (buffer-less)</td>
</tr>
<tr>
<td>Store and Forward</td>
<td>Packet</td>
<td>Packet</td>
</tr>
<tr>
<td>Virtual Cut Through</td>
<td>Packet</td>
<td>Packet</td>
</tr>
<tr>
<td>Wormhole</td>
<td>Packet</td>
<td>Flit</td>
</tr>
<tr>
<td>Virtual Channel</td>
<td>Flit</td>
<td>Flit</td>
</tr>
</tbody>
</table>
DESIGNING A FLOW CONTROL PROTOCOL: MANAGING BUFFERS AND CONTENTION
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?*
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”

Arbitration Result
(Send input if no traffic on ring)

Arbiter: Decides who uses the output link.

Arbitration:
Centralized or Distributed?

Centralized
• Bus
• Crossbar

Distributed
• Ring
• Mesh
3. What should a flit do if its output is blocked?
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
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*
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”)

![Diagram of flow control options](image)
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

$N_{\text{threshold}}$ set to 3 to prevent flits departing Node 1 before $t_4$ from overflowing.

$N_{\text{total}}$ set so that Node 2 does not run out of flits to send between $t_5$ and $t_8$.

$N_{\text{threshold}}$ reached

On/Off Delay = 2

Process Delay = 1

$N_{\text{threshold}} + 1$ reached

Flit Delay = 1
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 $N_{\text{threshold}}$ 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**

<table>
<thead>
<tr>
<th>t1</th>
<th>Node 0</th>
<th>Node 1</th>
<th>Node 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Credits=4</td>
<td>Credits=1</td>
<td>Credits=2</td>
</tr>
<tr>
<td>t2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Credit</td>
<td>Flit</td>
<td></td>
</tr>
<tr>
<td>t3</td>
<td>Process credit</td>
<td>Credits=0</td>
<td>Process Flit</td>
</tr>
<tr>
<td>t4</td>
<td>Credits=5</td>
<td></td>
<td>Credits=1</td>
</tr>
<tr>
<td>t5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Credits 4 at Node 0 at t1
- Credit sent from Node 0 to Node 1 at t2
- Flit sent from Node 1 to Node 2 at t3
- Process credit from Node 1 to Node 0 at t3
- Credits 1 at Node 0 at t5
BACKPRESSURE SIGNALING MECHANISMS

On/Off Flow Control

Pros
- Low overhead: one-bit signal

Cons
- Inefficient buffer utilization – have to design assuming worst case of $N_{\text{threshold}}$ 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**

```
<table>
<thead>
<tr>
<th>t1</th>
<th>Node 0</th>
<th>Node 1</th>
<th>Node 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credits=4</td>
<td></td>
<td>Credits=1</td>
<td>Credits=2</td>
</tr>
<tr>
<td>t2</td>
<td></td>
<td>Credit</td>
<td></td>
</tr>
<tr>
<td>t3</td>
<td>Process credit</td>
<td></td>
<td>Process Flit</td>
</tr>
<tr>
<td>Credits=5</td>
<td>or Credit Round Trip Time</td>
<td>Credits=0</td>
<td></td>
</tr>
<tr>
<td>t4</td>
<td>Credits=5</td>
<td></td>
<td>Credits=1</td>
</tr>
<tr>
<td>t5</td>
<td>No flit can be sent into this buffer during this delay</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Turnaround Time**

To prevent backpressure from limiting throughput, number of buffers >= turnaround time
**“BUFFER TURNAROUND TIME”**

1. Flit arrives at node 1 and uses buffer
2. Flit leaves node 1 and credit is sent to node 0
3. Node 0 receives credit, freed buffer reallocated to new flit
4. New flit leaves Node 0 for Node 1
5. New flit arrives at Node 1 and reuses buffer

Actual buffer usage
Credit propagation delay
Credit pipeline delay
Flit pipeline delay
Flit propagation delay

How many buffers needed?

1+1+3+1 = 6

How many buffers needed in on/off flow-control?

6 + 2 = 8
(off propagation + processing)
BUT THIS IS INEFFICIENT

Flit arrives at node 1 and uses buffer

Flit leaves node 1 and credit is sent to node 0

Node 0 receives credit

Node 0 processes credit, freed buffer reallocated to new flit

New flit leaves Node 0 for Node 1

New flit arrives at Node 1 and reuses buffer

See: Flit Rsvn Flow Control, HPCA 2000