Lecture 3: Flow-Control

Tushar Krishna

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

tushar@ece.gatech.edu

Acknowledgment: Some slides adapted from Univ of Toronto ECE 1749 H (N Jerger) and MIT 6.883 (L-S Peh)
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.

Suppose Router Delay = 1, Link Delay = 1

<table>
<thead>
<tr>
<th></th>
<th>Latency (hops) (A→B)</th>
<th>Throughput (msg/cycle) (A→B)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Topology</strong></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>Routing (XY)</strong></td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td><strong>Flow Control</strong></td>
<td><strong>3</strong> $(R_A + L_{AC} + R_C + L_{CB} (+ R_B))$</td>
<td>1/2 1/5</td>
</tr>
</tbody>
</table>

- **Case I:** One buffer at C
- **Case II:** 4 D→B msgs.
  Priority at C: straight>turn
Allocation Granularity: Messages, Packets, and Flits

Sequence in the message

- **Head Flit**
- **Body Flit**
- **Tail Flit**

*Only in Head / Head_Tail Flit*

- Head, Body, Tail, Head_Tail [1-flit packet]
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></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Head Flit
- Body Flits
- Tail Flit

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></td>
</tr>
</tbody>
</table>

- Head_Tail Flit

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

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
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

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

Not suitable on-chip. Why?

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

Allocate 4 flit-sized buffers before head proceeds

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

Red holds this channel: channel remains idle until red proceeds

Channel idle but red packet blocked behind blue

Dest for Red

6 flit buffers/input port

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
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
cchan p

Blocking

Node 1

Node 2

Node 3
VCs decouple dependency between buffer and channel
Virtual Channel Flow Control Example

Buffer full: blue cannot proceed

Blocked by other packets

Dest for Blue

Dest for Red

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

A Downstream

B Downstream

© Tushar Krishna, School of ECE, Georgia Tech
July 13, 2017
### Summary of Techniques

<table>
<thead>
<tr>
<th></th>
<th>Links</th>
<th>Buffers</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Circuit-Switching</strong></td>
<td>Messages</td>
<td>N/A (buffer-less)</td>
<td>Setup &amp; Ack</td>
</tr>
<tr>
<td><strong>Store and Forward</strong></td>
<td>Packet</td>
<td>Packet</td>
<td>Head flit waits for tail</td>
</tr>
<tr>
<td><strong>Virtual Cut Through</strong></td>
<td>Packet</td>
<td>Packet</td>
<td>Head can proceed</td>
</tr>
<tr>
<td><strong>Wormhole</strong></td>
<td>Packet</td>
<td>Flit</td>
<td>HOL</td>
</tr>
<tr>
<td><strong>Virtual Channel</strong></td>
<td>Flit</td>
<td>Flit</td>
<td>Interleave flits of different packets</td>
</tr>
</tbody>
</table>
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?
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?
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

```c
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 \textit{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

<table>
<thead>
<tr>
<th>Node 1</th>
<th>Node 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td></td>
</tr>
<tr>
<td>t2</td>
<td>Flit</td>
</tr>
<tr>
<td>t3</td>
<td>Off</td>
</tr>
<tr>
<td>t4</td>
<td>Flit</td>
</tr>
<tr>
<td>t5</td>
<td></td>
</tr>
<tr>
<td>t6</td>
<td>On</td>
</tr>
<tr>
<td>t7</td>
<td>Flit</td>
</tr>
<tr>
<td>t8</td>
<td></td>
</tr>
</tbody>
</table>

- **$N_{\text{threshold}}$ set to 3** to prevent flits departing Node 1 before t4 from overflowing.
- **$N_{\text{total}}$ set so that Node 2 does not run out of flits to send between t5 and t8.**

**Notes:**
- **On/Off Delay = 2**
- **Process Delay = 1**
- **Flit Delay = 1**
- **$N_{\text{threshold}} + 1$ reached**
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 $N_{\text{threshold}}$ 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

Node 0  Node 1  Node 2

\[ \text{Credits} = 4 \]  \[ \text{Credits} = 1 \]  \[ \text{Credits} = 2 \]

\[ \text{Credits} = 0 \]

\[ \text{Credits} = 5 \]

\[ \text{Process credit} \]

\[ \text{Process credit} \]

\[ \text{Process Flit} \]

\[ \text{Process Flit} \]

\[ \text{Process Flit} \]
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
Active Research in Flow-Control

- Efficient management of buffers and links
  - Hybrid circuit and packet-switched networks

- Quality-of-Service over shared NoC
  - Multi-threaded applications
  - Multi-programmed workloads
  - Real-time processes