Lecture 18: Interconnection Networks
Announcements

- Project deadlines:
  - Mon, April 2: project proposal: 1-2 page writeup
  - Fri, April 20: project checkpoint: 1-2 page writeup
  - Thurs, May 10: final presentations + final writeup
Today’s Agenda

- Interconnection Networks
  - Introduction and Terminology
  - Topology
  - Buffering and Flow control
Inteconnection Network Basics

▪ **Topology**
  - Specifies way switches are wired
  - Affects routing, reliability, throughput, latency, building ease

▪ **Routing**
  - How does a message get from source to destination
  - Static or adaptive

▪ **Buffering and Flow Control**
  - What do we store within the network?
    - Entire packets, parts of packets, etc?
  - How do we manage and negotiate buffer space?
    - How do we throttle during oversubscription?
  - Tightly coupled with routing strategy
Terminology

▪ Network interface
  - Connects endpoints (e.g. cores) to network.
  - Decouples computation/communication

▪ Links
  - Bundle of wires that carries a signal

▪ Switch/router
  - Connects fixed number of input channels to fixed number of output channels

▪ Channel
  - A single logical connection between routers/switches
More Terminology

- **Node**
  - A network endpoint connected to a router/switch

- **Message**
  - Unit of transfer for network clients (e.g. cores, memory)

- **Packet**
  - Unit of transfer for network

- **Flit**
  - Flow control digit
  - Unit of flow control within network
Some More Terminology

- **Direct or Indirect Networks**
  - Endpoints sit “inside” (direct) or “outside” (indirect) the network
  - E.g. mesh is direct; every node is both endpoint and switch

![Diagram showing indirect and direct networks](image)
Today’s Agenda

- Interconnection Networks
  - Introduction and Terminology
  - Topology
  - Buffering and Flow control
Properties of a Topology/Network

- Regular or Irregular
  - regular if topology is regular graph (e.g. ring, mesh)

- Routing Distance
  - number of links/hops along route

- Diameter
  - maximum routing distance

- Average Distance
  - average number of hops across all valid routes
Properties of a Topology/Network

- **Bisection Bandwidth**
  - Often used to describe network performance
  - Cut network in half and sum bandwidth of links severed
    - \((\text{Min # channels spanning two halves}) \times (\text{BW of each channel})\)
  - Meaningful only for recursive topologies
  - Can be misleading, because does not account for switch and routing efficiency

- **Blocking vs. Non-Blocking**
  - If connecting any permutation of sources & destinations is possible, network is non-blocking; otherwise network is blocking.
Many Topology Examples

- Bus
- Crossbar
- Ring
- Tree
- Omega
- Hypercube
- Mesh
- Torus
- Butterfly
- ...
Bus

+ Simple
+ Cost effective for a small number of nodes
+ Easy to implement coherence (snooping)

- Not scalable to large number of nodes
  (limited bandwidth, electrical loading \(\rightarrow\) reduced frequency)

- High contention
Crossbar

- Every node connected to all others (non-blocking)
- Good for small number of nodes
  + Low latency and high throughput
  - Expensive
  - Not scalable $\rightarrow O(N^2)$ cost
  - Difficult to arbitrate

Core-to-cache-bank networks:
- IBM POWER5
- Sun Niagara I/II
Ring

+ Cheap: $O(N)$ cost
- High latency: $O(N)$
- Not easy to scale
  - Bisection bandwidth remains constant

Used in:
- Intel Larrabee/Core i7
- IBM Cell
Mesh

- O(N) cost
- Average latency: O(sqrt(N))
- Easy to layout on-chip: regular & equal-length links
- Path diversity: many ways to get from one node to another

- Used in:
  - Tilera 100-core CMP
  - On-chip network prototypes
Torus

- Mesh is not symmetric on edges: performance very sensitive to placement of task on edge vs. middle
- Torus avoids this problem
  + Higher path diversity (& bisection bandwidth) than mesh
  - Higher cost
  - Harder to lay out on-chip
    - Unequal link lengths
Trees

Planar, hierarchical topology
Latency: $O(\log N)$
Good for local traffic
+ Cheap: $O(N)$ cost
+ Easy to Layout
- Root can become a bottleneck
  Fat trees avoid this problem (CM-5)
Hypercube

- Latency: $O(\log N)$
- Radix: $O(\log N)$
- #links: $O(N\log N)$

+ Low latency
- Hard to lay out in 2D/3D

- Used in some early message passing machines, e.g.:
  - Intel iPSC
  - nCube
Multistage Logarithmic Networks

- Idea: Indirect networks with multiple layers of switches between terminals
- Cost: $O(N \log N)$, Latency: $O(\log N)$
- Many variations (Omega, Butterfly, Benes, Banyan, …)
- E.g. Omega Network:

Q: Blocking or non-blocking?
### Review: Topologies

<table>
<thead>
<tr>
<th>Topology</th>
<th>Crossbar</th>
<th>Multistage Logarith.</th>
<th>Mesh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct/Indirect</td>
<td>Indirect</td>
<td>Indirect</td>
<td>Direct</td>
</tr>
<tr>
<td>Blocking/ Non-blocking</td>
<td>Non-blocking</td>
<td>Blocking</td>
<td>Blocking</td>
</tr>
<tr>
<td>Cost</td>
<td>$O(N^2)$</td>
<td>$O(N \log N)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>Latency</td>
<td>$O(1)$</td>
<td>$O(\log N)$</td>
<td>$O(\sqrt{N})$</td>
</tr>
</tbody>
</table>
Today’s Agenda

- Interconnection Networks
  - Introduction and Terminology
  - Topology
  - Buffering and Flow control
Circuit vs. Packet Switching

- Circuit switching sets up full path
  - Establish route then send data
  - (no one else can use those links)
  - faster and higher bandwidth
  - setting up and bringing down links slow

- Packet switching routes per packet
  - Route each packet individually (possibly via different paths)
  - if link is free can use
  - potentially slower (must dynamically switch)
  - no setup, bring down time
Packet Switched Networks: Packet Format

- **Header**
  - routing and control information

- **Payload**
  - carries data (non HW specific information)
  - can be further divided (framing, protocol stacks…)

- **Error Code**
  - generally at tail of packet so it can be generated on the way out

![Packet Format Diagram]

(CMU 15-418, Spring 2012)
Handling Contention

- Two packets trying to use the same link at the same time
- What do you do?
  - Buffer one
  - Drop one
  - Misroute one (deflection)
- We will only consider buffering in this lecture
Flow Control Methods

- Circuit switching
- Store and forward (Packet based)
- Virtual Cut Through (Packet based)
- Wormhole (Flit based)
Circuit Switching Revisited

- Resource allocation granularity is high

- Idea: Pre-allocate resources across multiple switches for a given “flow”

- Need to send a probe to set up the path for preallocation

  + No need for buffering
  + No contention (flow’s performance is isolated)
  + Can handle arbitrary message sizes

- Lower link utilization: two flows cannot use the same link
- Handshake overhead to set up a “circuit”
Store and Forward Flow Control

- Packet based flow control
- Store and Forward
  - Packet copied entirely into network router before moving to the next node
  - Flow control unit is the entire packet
- Leads to high per-packet latency
- Requires buffering for entire packet in each node

Can we do better?
Cut through Flow Control

- Another form of packet based flow control
- Start forwarding as soon as header is received and resources (buffer, channel, etc) allocated
  - Dramatic reduction in latency
- Still allocate buffers and channel bandwidth for full packets

- What if packets are large?
Cut through Flow Control

- What to do if output port is blocked?
  - Lets the tail continue when the head is blocked, absorbing the whole message into a single switch.
    - Requires a buffer large enough to hold the largest packet.
  - Degenerates to store-and-forward with high contention

- Can we do better?
Wormhole Flow Control

- Packets broken into (potentially) smaller flits (buffer/bw allocation unit)
- Flits are sent across the fabric in a wormhole fashion
  - Body follows head, tail follows body
  - Pipelined
  - If head blocked, rest of packet stops
  - Routing (src/dest) information only in head

- How does body/tail know where to go?
- Latency almost independent of distance for long messages
Wormhole Flow Control

- Advantages over “store and forward” flow control
  + Lower latency
  + More efficient buffer utilization

- Limitations
  - Suffers from head-of-line (HOL) blocking
    - If head flit cannot move due to contention, another worm cannot proceed even though links may be idle
Head-of-Line Blocking

- Red holds this channel: channel remains idle until read proceeds.
- Channel idle but red packet blocked behind blue.
- Buffer full: blue cannot proceed.
- Blocked by other packets.
Virtual Channel Flow Control

- Idea: Multiplex multiple channels over one physical channel
- Reduces head-of-line blocking
- Divide up the input buffer into multiple buffers sharing a single physical channel
Virtual Channel Flow Control

- Buffer full: blue cannot proceed
- Blocked by other packets
Other Uses of Virtual Channels

- **Deadlock avoidance**
  - Enforcing switching to a different set of virtual channels on some “turns” can break the cyclic dependency of resources
  - **Escape VCs**: Have at least one VC that uses deadlock-free routing. Ensure each flit has fair access to that VC.
  - **Protocol level deadlock**: Ensure request and response packets use different VCs → prevent cycles due to intermixing of different packet classes

- **Prioritization of traffic classes**
  - Some virtual channels can have higher priority than others
Communicating Buffer Availability

- Credit-based flow control
  - Upstream knows how many buffers are downstream
  - Downstream passes back credits to upstream
  - Significant upstream signaling (esp. for small flits)

- On/Off (XON/XOFF) flow control
  - Downstream has on/off signal to upstream

- Ack/Nack flow control
  - Upstream optimistically sends downstream
  - Buffer cannot be deallocated until ACK/NACK received
  - Inefficiently utilizes buffer space
Review: Flow Control

Store and Forward

- Red holds this channel: channel remains idle until read proceeds
- Blocked by other packets

Cut Through / Wormhole

- Shrink Buffers
- Reduce latency

Any other issues?

Head-of-Line Blocking

Use Virtual Channels

Channel idle but red packet blocked behind blue

Buffer full: blue cannot proceed
Review: Flow Control

Store and Forward

Cut Through / Wormhole

Shrink Buffers
Reduce latency

Any other issues?

Head-of-Line Blocking
Use Virtual Channels

Buffer full: blue cannot proceed
Blocked by other packets