Lecture 17:
Interconnection Networks

Parallel Computer Architecture and Programming
CMU 15-418/15-618, Spring 2014

Michael Papamichael

Material partially based on Onur Mutlu’s 18-742 lecture slides
Tunes

Phantogram

Let Me Go

(Eyelid Movies)

“A song for all those packets needlessly waiting due to head-of-line blocking.”

- Josh Carter
What are interconnection networks used for?

- To connect
  - Processors and processors
  - Processors and memories
  - Processors and caches
  - Caches and caches
  - I/O devices
Basic system design from previous lectures

Bus interconnect:
All nodes connected by a shared set of wires

Request bus:
- cmd + address
  - e.g., 40 bits
- Response bus:
  - data
    - e.g., 256 bits
  - Response tag
    - 3 bits
Basic system design from previous lectures

Interconnection Network

Processor

Cache

Bus clients (interconnect nodes)

Memory

Processor

Cache

Bus interconnect:

All nodes connected by a shared set of wires

| Request bus: cmd + address | e.g., 40 bits |
| Response bus: data | e.g., 256 bits |
| Response tag | 3 bits |
Why are interconnection networks important?

- **Scalability**
  - How large of a system can you build?
  - How easily can you add more nodes (e.g. cores)

- **Performance and energy efficiency**
  - How fast can cores, caches and memory communicate?
  - How long are the latencies to memory?
  - How much energy is spent on communication?
Scaling to larger numbers of nodes
On a single chip!

- Intel core i7 (4-CPU cores, + GPU core)
- Tilera GX 64-core chip
- Intel Xeon Phi (61-core x86 chip)
- Oracle 16-core SPARC T5
Today’s agenda

- Interconnection Networks
  - Basics
  - Topology
  - Buffering and Flow control
  - Research Topics
Terminology

- **Node**
  - A network endpoint connected to a router/switch (e.g. cores, memories)

- **Switch/router**
  - Connects fixed set of inputs to set of outputs

- **Link**
  - A bundle of wires carrying a signal

- **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 the network
Design issues

- **Topology**
  - How switches are wired
  - Affects routing throughput, latency, complexity/cost of implementation

- **Routing**
  - How does message get from source to destination
  - Can be static (predetermined path) or adaptive based on network conditions

- **Buffering and flow control**
  - What data is stored in the network? packets, partial packets? etc.
  - How does the network manage buffer space?
  - Tightly coupled with routing strategy
Interconnect properties

- 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
More properties

- 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
Even more properties

- **Bisection bandwidth**
  - Often used to describe network performance
  - Cut network in half and sum bandwidth of links severed
    (Min # channels spanning two halves) * (BW of each channel)
  - Meaningful only for recursive topologies
  - Can be misleading; 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.
  - Rearrangeable non-blocking: Same as non-blocking but might require rearranging connections when switching from one permutation to another.
Blocking vs. non-blocking example

- Is this network blocking or non-blocking?
  - Consider messages from 3-to-7 and 1-to-6. Blocking!!!
Network Performance

- Load-Latency behavior
  - Can heavily depend on traffic pattern

Latency

Load - Offered Traffic (bits/sec)

Zero load or idle latency (topology + routing + flow control)

Min latency given by topology

Min latency given by routing algorithm

Saturation throughput (given by flow control)

Throughput given by routing

Throughput given by topology
Today’s agenda

- Interconnection Networks
  - Basics
  - Topology
  - Buffering and Flow control
  - Research Topics
Topology

- Many options!
  - Bus
  - Crossbar
  - Ring
  - Tree
  - Omega
  - Hypercube
  - Mesh
  - Torus
  - Butterfly
  - ...
Bus

- **Good**
  - Simple
  - Cost effective for small number of nodes
  - Easy to implement coherence (snooping)

- **Bad**
  - High contention: all nodes contend for shared bus
  - Limited bandwidth: all nodes communicate over same wires
  - High electrical load → Low frequency, high power
Crossbar

- Every node connected to every other node (non-blocking, indirect)

- Good
  - Low latency
  - High bandwidth

- Bad
  - Expensive
  - Not scalable: $O(N^2)$ switches
  - Difficult to arbitrate at scale

Crossbar scheduling algorithms and efficient hardware implementations are still active research areas. Nick McKeown (Stanford Prof), who developed one of the most well-known hardware-friendly algorithms called iSLIP, devoted his entire PhD on studying crossbar scheduling algorithms.
Crossbars are used in Sun/Oracle CMPs

- Two crossbars for connecting cores to last-level cache banks and vice-versa

Sun SPARC T2 (8 cores, 8 L2 cache banks)

Oracle SPARC T5 (16 cores, 8 L3 cache banks)
Ring

- **Good**
  - Simple
  - Cheap: $O(N)$

- **Bad**
  - Scalability: bisection bandwidth remains constant as nodes are added
  - High latency: $O(N)$

- **Used in Intel and IBM chips**
  - Intel Larrabee, Core i7, Xeon Phi
  - IBM Cell processor
Mesh

- Direct network
- Echoes locality in grid-based applications
- $O(N)$ cost
- Average latency: $O(\sqrt{N})$
- Easy to lay out on chip: regular and equal-length links
- Path diversity: many ways for message to get from one node to another
- Used by:
  - Tilera processors
  - Some Intel prototype chips

The mesh is one of the most popular topologies studied in computer architecture research. At this year’s International Solid-State Circuits Conference Intel presented a 22nm prototype chip of a 16x16 2D mesh (256 nodes) that can achieve an aggregate throughput of 20.2 terabits per second.
Torus

- Mesh topology’s characteristics different based on whether node is near edge or middle of network (torus topology introduces new links to avoid this problem)
- Still $O(N)$ cost, but higher cost than 2D grid
- Higher path diversity and bisection BW than mesh
- Higher complexity
  - Harder to layout on chip
  - Unequal link lengths
### Trees

- Planar, hierarchical topology
- Good when traffic has locality
- Latency: $O(\log N)$
- Root can become a bottleneck
  - Use “fat trees” (higher bandwidth on links near root) to alleviate root bandwidth problem

---

The Fat Tree network was invented by MIT Prof. Charles Leiserson and was used as the interconnection network in the Thinking Machines CM-5 supercomputer developed in the 90s. A CM-5 was featured in the movie Jurassic Park.
Hypercube

- Low latency: $O(\lg N)$
- Radix: $O(\lg N)$
- Number of links $O(N \lg N)$

- A 6-D hyper cube was used in the Cosmic Cube 64-core parallel computer developed at Caltech in the 80s.
Multi-stage logarithmic

- Indirect network with multiple layers of switches between terminals
- Cost: $O(N \log N)$
- Latency: $O(\log N)$
- Many variations: Omega, butterfly, Clos networks, etc...

![Omega Network Diagram]
# Review: network topologies

<table>
<thead>
<tr>
<th>Topology</th>
<th>Crossbar</th>
<th>Multistage Logarithm.</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 (not always, this specific one)</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
  - Interconnect Basics
  - Topology
  - Buffering and Flow control
  - Research Topics
Circuit switching vs. packet switching

- Circuit switching sets up a full path (acquires all resources) between sender and receiver prior to sending a message
  - Establish route (reserve links) then send data
  - Higher bandwidth transmission (no link management overhead)
  - Overhead to set up/tear down path
  - Reserving links can result in low utilization

- Packet switching makes routing decisions per packet
  - Route each packet individually (possibly over different network links)
  - Opportunity to use link for a packet whenever link is idle
  - Overhead due to dynamic switching logic during transmission
  - No setup/tear down overhead
Packet format

- **Header**
  - routing and control information
  - at start so router can start forwarding early

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

- **Tail**
  - contains control information, e.g. error code
  - at end of packet so it can be generated on the way out
Handling contention

- Options:
  - Buffer one packet, send it over link later
  - Drop one packet
  - Reroute one packet (deflection)

- In this lecture: we only consider buffering

Scenario: two packets need to be routed onto the same outbound link at the same time

Recent research has looked at using bufferless networks with deflection routing as a power efficient interconnect for chip multiprocessors.
Circuit-switched routing

- **High-granularity resource allocation**
  - Main idea: **pre-allocate** all resources (links across multiple switches) along entire network path for a message (“setup a flow”)

- **Costs**
  - Needs setup phase (“probe”) to set up the path (and to tear it down and release the resources when message complete)
  - Lower link utilization. Transmission of two messages cannot share same link (even if some resources on a preallocated path are no longer utilized during a transmission)

- **Benefits**
  - No contention during transmission due to preallocation, so no need for buffering
  - Arbitrary message sizes (once path is set up, send data until done)
Store-and-forward (packet-based routing)

- Packet copied entirely into network switch before moving to next node
- Flow control unit is an entire packet
- Leads to high per-packet latency
- Requires buffering for entire packet in each node

Can we do better?
Cut-through flow control (also packet-based)

- Switch starts forwarding data on next link as soon as header is received (header carries packet info, e.g., destination, size)
- Result: reduced transmission latency
- Still requires buffering for entire packet in each router (like store-and-forward)
- Worst case: entire message is absorbed into a buffer in a switch (degenerates to store-and-forward)

- What if packets are large? Can we do better?
Wormhole flow control

- Packets broken into smaller units called “flits”
  - Flit: (“flow control digit”) a unit of flow control in the network
  - Flits become minimum granularity of routing/buffering
Wormhole flow control

- Flits sent in a “wormhole” fashion
- Routing information only in head
- Body flits follows head, tail flit follows body
- If head flit blocks, rest of packet stops
- Completely pipelined transmission
  - For long messages, latency almost entirely independent of network distance

Example: Five-flit packet sent using wormhole flow control

- How does body/tail know where to go?
Wormhole Flow Control

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

- Limitations
  - Occupies resources across multiple routers
  - Suffers from head of line blocking
  - If head flit cannot move due to contention, another worm cannot proceed even though links may be idle

![Diagram showing Wormhole Flow Control with Input Queues, Switching Fabric, and Outputs with HOL Blocking and Idle! annotations.](image-url)
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
Head-of-line blocking

- A worm can be before another in the router input buffer
- Due to FIFO nature, the second worm cannot be scheduled - even though it may need to access another output port!

Karo et al., “Input Versus Output Queuing on a Space-Division Packet Switch,” IEEE Transactions on Communications 1987
Virtual Channel Flow Control

- Multiplex multiple operations over a single physical channel
- Divide switch’s input buffer into multiple buffers sharing a single physical channel
- Reduces head-of-line blocking

Virtual Channel Flow Control

- Multiplex multiple operations over single physical channel
- Divide switch’s input buffer into multiple buffers sharing a single physical channel
- Reduces head-of-line blocking

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
Credit-based flow control

- **Round-trip credit delay:**
  - Time between when buffer empties and when next flit can be processed from that buffer entry
- **Significant throughput degradation if there are few buffers**
- **Important to size buffers to tolerate credit turn-around**
On/Off (XON/XOFF) Flow Control

- Downstream has on/off signal to upstream

- $F_{	ext{off}}$ set to prevent flits arriving before $t4$ from overflowing

- $F_{	ext{on}}$ set so that Node 2 does not run out of flits between $t5$ and $t8$
**Review: flow control**

- **Store and Forward**
  - Blocked by other packets

- **Cut Through / Wormhole**
  - Buffer full: blue cannot proceed
  - Channel idle but red packet blocked behind blue

**Any other issues?**

- **Head-of-Line Blocking**
  - Red holds this channel: channel remains idle until read proceeds

- **Use Virtual Channels**
  - Blocked by other packets
Review: flow control

Store and Forward

- S
- D
- Shrink Buffers
- Reduce latency

Cut Through / Wormhole

- S
- D

Any other issues?

Head-of-Line Blocking

Use Virtual Channels

Blocked by other packets

Buffer full: blue cannot proceed

CMU 15-418, Spring 2014
Some research topics

- **Energy efficiency**
  - Interconnect can consume a lot of power (~35% of total chip power in MIT RAW chip)
  - Bufferless networks are one energy-efficient alternative to buffered networks
    (related work on bufferless networks at CMU: BLESS, CHIPPER, MIN-BD)
  - Other ideas: turn on/off parts of the network, have separate fast/slow networks

- **Prioritization and Quality of Service (QoS) guarantees**
  - Prioritize packets based to improve multiprocessor performance
  - Throttle endpoints (e.g. cores) based on network feedback
    (related work at CMU: Aergia, HAT, Application-aware packet prioritization for on-chip networks)

- **New and emerging technologies**
  - Die stacking (3D chips)
  - Photonic Networks-on-Chip (use optical waveguides instead of wires)
  - Reconfigurable devices (FPGAs): create custom interconnect tailored to application
    (related CMU projects: CONNECT, CoRAM, Shrinkwrap)

- **Reliability and fault tolerance**
  - Routing algorithms that dynamically adapt based on interconnect faults
Summary

- The performance of the interconnection network in a modern multi-processor is absolutely critical to overall system performance
  - Buses do not scale to many nodes
  - Historically interconnect was off-chip network connecting sockets, boards, racks
  - Today, all these issues apply to the design of on-chip networks

- Network topologies differ in performance, cost, complexity tradeoffs
  - e.g., crossbar, ring, mesh, torus, multi-stage network, fat tree, hypercube

- Challenge: efficiently routing data through network
  - Interconnect is a precious resource (communication is expensive!)
  - Flit-based flow control: fine-grained flow control to make good use of available link bandwidth
  - If interested, much more to learn about (not discussed in this class): ensuring quality-of-service, prioritization, reliability, deadlock, livelock, etc.