Lecture 25: Interconnection Networks

• Topics: communication latency, centralized and decentralized switches, routing, deadlocks (Appendix E)

• Review session, Wednesday Dec 1st, 10-12, LCR (MEB 3147)

• Final exam reminders
  • Come early, 10:35 – 12:15
  • Same rules as first midterm, open books/notes/…,
  • Can use calculators and laptops (no search or internet)
  • 20% from first midterm material; remaining 80% from caches, multiprocs, TM
  • 20% new problems
  • Attempt every question
Topologies

• Internet topologies are not very regular – they grew incrementally

• Supercomputers have regular interconnect topologies and trade off cost for high bandwidth

• Nodes can be connected with
  ➢ centralized switch: all nodes have input and output wires going to a centralized chip that internally handles all routing
  ➢ decentralized switch: each node is connected to a switch that routes data to one of a few neighbors
Centralized Crossbar Switch
Centralized Crossbar Switch
Crossbar Properties

• Assuming each node has one input and one output, a crossbar can provide maximum bandwidth: N messages can be sent as long as there are N unique sources and N unique destinations

• Maximum overhead: $WN^2$ internal switches, where $W$ is data width and $N$ is number of nodes

• To reduce overhead, use smaller switches as building blocks – trade off overhead for lower effective bandwidth
Switch with Omega Network

Diagram showing a switch with Omega Network with nodes labeled P0 to P7 and binary routing paths from input to output.
Omega Network Properties

• The switch complexity is now $O(N \log N)$

• Contention increases: $P_0 \rightarrow P_5$ and $P_1 \rightarrow P_7$ cannot happen concurrently (this was possible in a crossbar)

• To deal with contention, can increase the number of levels (redundant paths) – by mirroring the network, we can route from $P_0$ to $P_5$ via $N$ intermediate nodes, while increasing complexity by a factor of 2
Tree Network

- Complexity is $O(N)$
- Can yield low latencies when communicating with neighbors
- Can build a fat tree by having multiple incoming and outgoing links
Bisection Bandwidth

• Split N nodes into two groups of N/2 nodes such that the bandwidth between these two groups is minimum: that is the bisection bandwidth

• Why is it relevant: if traffic is completely random, the probability of a message going across the two halves is ½ – if all nodes send a message, the bisection bandwidth will have to be N/2

• The concept of bisection bandwidth confirms that the tree network is not suited for random traffic patterns, but for localized traffic patterns
Distributed Switches: Ring

- Each node is connected to a 3x3 switch that routes messages between the node and its two neighbors.
- Effectively a repeated bus: multiple messages in transit.
- Disadvantage: bisection bandwidth of 2 and N/2 hops on average.
Distributed Switch Options

- Performance can be increased by throwing more hardware at the problem: fully-connected switches: every switch is connected to every other switch: $N^2$ wiring complexity, $N^2/4$ bisection bandwidth

- Most commercial designs adopt a point between the two extremes (ring and fully-connected):
  - Grid: each node connects with its N, E, W, S neighbors
  - Torus: connections wrap around
  - Hypercube: links between nodes whose binary names differ in a single bit
Topology Examples

Criteria | Bus | Ring | 2Dtorus | 6-cube | Fully connected
---|---|---|---|---|---
Performance | | | | | |
Bisection bandwidth | | | | | |
Cost | | | | | |
Ports/switch | | | | | |
Total links | | | | | |
## Topology Examples

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Bus</th>
<th>Ring</th>
<th>2Dtorus</th>
<th>6-cube</th>
<th>Fully connected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bisection bandwidth</td>
<td>1</td>
<td>2</td>
<td>16</td>
<td>32</td>
<td>1024</td>
</tr>
<tr>
<td>Cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ports/switch</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>64</td>
</tr>
<tr>
<td>Total links</td>
<td>1</td>
<td>128</td>
<td>192</td>
<td>256</td>
<td>2080</td>
</tr>
</tbody>
</table>
k-ary d-cube

- Consider a k-ary d-cube: a d-dimension array with k elements in each dimension, there are links between elements that differ in one dimension by 1 (mod k)

- Number of nodes \( N = k^d \)

<table>
<thead>
<tr>
<th>Number of switches</th>
<th>Avg. routing distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch degree</td>
<td>Diameter</td>
</tr>
<tr>
<td>Number of links</td>
<td>Bisection bandwidth</td>
</tr>
<tr>
<td>Pins per node</td>
<td>Switch complexity</td>
</tr>
</tbody>
</table>

Should we minimize or maximize dimension?
Consider a k-ary d-cube: a d-dimension array with k elements in each dimension, there are links between elements that differ in one dimension by 1 (mod k).

Number of nodes $N = k^d$ (with no wraparound)

- Number of switches: $N$
- Switch degree: $2d + 1$
- Number of links: $Nd$
- Pins per node: $2wd$
- Avg. routing distance: $d(k-1)/2$
- Diameter: $d(k-1)$
- Bisection bandwidth: $2wk^{d-1}$
- Switch complexity: $(2d + 1)^2$

Should we minimize or maximize dimension?
Routing

• Deterministic routing: given the source and destination, there exists a unique route

• Adaptive routing: a switch may alter the route in order to deal with unexpected events (faults, congestion) – more complexity in the router vs. potentially better performance

• Example of deterministic routing: dimension order routing: send packet along first dimension until destination co-ord (in that dimension) is reached, then next dimension, etc.
Deadlock

- Deadlock happens when there is a cycle of resource dependencies – a process holds on to a resource (A) and attempts to acquire another resource (B) – A is not relinquished until B is acquired
Deadlock Example

Each message is attempting to make a left turn – it must acquire an output port, while still holding on to a series of input and output ports.
Deadlock-Free Proofs

• Number edges and show that all routes will traverse edges in increasing (or decreasing) order – therefore, it will be impossible to have cyclic dependencies

• Example: k-ary 2-d array with dimension routing: first route along x-dimension, then along y
Breaking Deadlock I

• The earlier proof does not apply to tori because of wraparound edges

• Partition resources across multiple virtual channels

• If a wraparound edge must be used in a torus, travel on virtual channel 1, else travel on virtual channel 0
Breaking Deadlock II

• Consider the eight possible turns in a 2-d array (note that turns lead to cycles)

• By preventing just two turns, cycles can be eliminated

• Dimension-order routing disallows four turns

• Helps avoid deadlock even in adaptive routing

West-First  North-Last  Negative-First  Can allow deadlocks
Title

- Bullet