Recap: Generic Parallel Architecture

- A generic modern multiprocessor

Node: processor(s), memory system, plus communication assist
  - Network interface and communication controller
  - Scalable network
  - Convergence allows lots of innovation, now within framework
    - Integration of assist with node, what operations, how efficiently...

© C. Xu 2001-2005
Interconnection Networks

- Static vs Dynamic networks
  - Static: built out of point-to-point communication links between processors (aka direct networks)
    - Usually associated to message passing architectures
    - Examples: completely-/star-connected, linear array, ring, mesh, hypercube
  - Dynamic: built out of links and switches (aka indirect networks)
    - Usually associated to shared address space architectures
    - Examples: crossbar, bus-based, multistage

Goal: Bandwidth and Latency

Latencies remain reasonable only as long as application BW requirement is much smaller than BW available on machine.
Topological Properties

- **Degree** --- number of incident nodes
- **Diameter** --- maximum routing distance
- **Bisection bandwidth**: sum of bandwidth of smallest set of links that partition the network into two halves
- **Routing distance** --- number of links on route
- **Average distance** --- average routing distance over all pairs of nodes
- **Scalability** --- the ability to be modularly expandable with a scaleable performance
- **Partitionable** --- whether a subgraph keeps the same properties
- **Symmetric**: uniform traffic vs hot-spot
- **Fault tolerance**

Static Interconnection Networks

- Completely connected networks
- Star-connected networks
- Linear Array
- Mesh
- Hypercube Network
Linear Arrays and Rings

- Linear Array
  - Diameter of an array of size $n$ nodes?
  - Average Distance?
  - Bisection bandwidth?
  - Node labeling and Routing Algorithm:
    - For linear array: `next route(myid, src, dest){... ... }`
    - For bidirectional rings: ??
- Examples: FDDI, SCI, FiberChannel Arbitrated Loop

2-D Meshes and Tori

- Examples: Intel Paragon (2d mesh), Cray T3D (3d torus)
- For a 2d mesh of size $n \times n$
  - Diameter? Bisection bandwidth? Average Distance?
  - X-Y Routing: Labeling each node in a pair of integers (i,j).
    A message is routed first along the X dimension until it reaches the column of the destination node and then along the Y dimension until it reaches its destination
  - nid `route(myid, src, dest){ ... ... }`
Multidimensional Meshes and Tori

- $d$-dimensional array
  - $n = k_{d-1} \times \ldots \times k_0$ nodes
  - described by $d$-vector of coordinates $(i_{d-1}, \ldots, i_0)$
- $d$-dimensional $k$-ary mesh: $N = k^d$
  - $k = d^{\sqrt{N}}$
  - described by $d$-vector of radix $k$ coordinate
- $d$-dimensional $k$-ary torus (or $k$-ary $d$-cube)?

Hypercubes

- Also called binary $n$-cubes. # of nodes = $N = 2^n$.
- Degree: $n = \log N$
- Distance $O(\log N)$ Hops
- Good bisection BW
- Examples: SGI Origin 2000
Hypercube Labeling and Routing

• Binary representation
• Distance?
  – xor operation
• E-cube routing

![Diagram of hypercubes]

E-Cube Routing

• Dimension-ordered routing (extension of XY routing)
• Deterministic and minimal message routing

![Diagram of E-cube routing]

Figure 2.28. Routing a message from node $P_2$ (010) to node $P_d$ (111) in a three-dimensional hypercube using E-cube routing.


**Hypercube Properties**

- One node connected to \( d \) others
- One bit difference in labels \( \Leftrightarrow \) direct link
- One hyper can be partitioned in two \((d-1)\) hypers
- The Hamming distance = shortest path length
  - Hamming distance = \# of bits that are difference in source and dest (binary addresses of the two nodes) = \# of nodes in source xor dest
- Each node address contains \( d \) bits
  - fixing \( k \) of these bits, the nodes that differ in the remaining \((d-k)\) for a \((d-k)\) subcube of \( 2^(d-k) \) nodes. There are \( 2^k \) such subcubes.

**K-ary d-cubes**

- A k-ary d-cubes is a d-dimensional mesh with \( k \) elements along each dimension
  - \( k \) is radix, \( d \) is dimension
  - built from k-ary \((d-1)\)-cubes by connecting the corresponding processors into a ring
- Some of the other topologies are particular instances of the k-ary d-cubes:
  - A ring of \( n \) nodes is a n-ary 1-cube
  - A two-dimensional \( n \times n \) torus is a n-ary 2-cube
### Topology Summary

<table>
<thead>
<tr>
<th>Topology</th>
<th>Degree</th>
<th>Diameter</th>
<th>Ave Dist</th>
<th>Bisection</th>
<th>$D (D_{ave}) \oplus P=1024$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1D Array</td>
<td>2</td>
<td>$N-1$</td>
<td>$N/3$</td>
<td>1</td>
<td>huge</td>
</tr>
<tr>
<td>1D Ring</td>
<td>2</td>
<td>$N/2$</td>
<td>$N/4$</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2D Mesh</td>
<td>4</td>
<td>$2 (N^{1/2} - 1)$</td>
<td>$2/3 N^{1/2}$</td>
<td>$N^{1/2}$</td>
<td>63 (21)</td>
</tr>
<tr>
<td>2D Torus</td>
<td>4</td>
<td>$N^{1/2}$</td>
<td>$1/2 N^{1/2}$</td>
<td>$2N^{1/2}$</td>
<td>32 (16)</td>
</tr>
<tr>
<td>k-ary n-cube</td>
<td>$2n$</td>
<td>$nk/2$</td>
<td>$nk/4$</td>
<td>$nk/4$</td>
<td>15 (7.5) $\oplus n=3$</td>
</tr>
<tr>
<td>Hypercube</td>
<td>$n = \log N$</td>
<td>$n$</td>
<td>$n/2$</td>
<td>$N/2$</td>
<td>10 (5)</td>
</tr>
</tbody>
</table>

© C. Xu 2001-2005

---

### Bus-based Networks

- A bus is a shared communication link, using one set of wires to connect multiple processing elements.
  - Processor/Memory bus, I/O bus
  - Processor/Processor bus
- Very simple concept, its major drawback is that bandwidth does not scale up with the number of processors
  - cache can alleviate problem because reduce traffic to memory

© C. Xu 2001-2005
Crossbar Switching Networks

- Digital analogous of a switching board
  - allows connection of any of p processors to any of b memory banks
  - Examples: Sun Ultra HPC 1000, Fujitsu VPP 500, Myrinet switch
- Main drawback: complexity grows as P^2
  - too expensive for large p
- Crossbar [Bus] network is the dynamic analogous of the completed [star] network

Multistage Interconnection Network

- A good compromise between cost and performance
  - More scalable in terms of cost than crossbar, more scalable in terms of performance than bus
  - Popular schemes include omega and butterfly networks

Figure 2.9 The schematic of a typical multistage interconnection network.
Omega Network

- Perfect Shuffle; Log p stages each with p/2 switches

Routing in Omega Network

Routing algorithm
- At each stage, look at the corresponding bit (starting with the msb) of the source and dest address
- If the bits are the same, messages passes through, otherwise crossed-over

Example of blocking: either (010 to 111) or (110 to 100) has to wait until link AB is free
Butterflies

- Tree with lots of roots!
- $N \log N$ (actually $N/2 \times \log N$)
- Exactly one route from any source to any destination
- Bisection $N/2$ vs $n^{(d-1)/d}$

Tree Structures

- Static vs Dynamic Trees

© C. Xu 2001-2005
Tree Properties

- Diameter and ave distance logarithmic
  - k-ary tree, height \( d = \log_k N \)
  - address specified d-vector of radix k coordinates describing path down from root

- Fixed degree
- H-tree space is \( O(N) \) with \( O(\sqrt{N}) \) long wires
- Bisection BW?

- Fat Tree
  - Example: CM-5

Benes network and Fat Tree

- Back-to-back butterfly can route all permutations
  - off line
- What if you just pick a random mid point?
Relationship Between Butterfly and Hypercubes

- Wiring is isomorphic
- Except that Butterfly always takes log n steps

Real Machines

<table>
<thead>
<tr>
<th>Machine</th>
<th>Topology</th>
<th>Cycle Time (ns)</th>
<th>Channel Width (bits)</th>
<th>Routing Delay (cycles)</th>
<th>Flit (data bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>nCUBE/2</td>
<td>Hypercube</td>
<td>25</td>
<td>1</td>
<td>40</td>
<td>32</td>
</tr>
<tr>
<td>TMC CM-5</td>
<td>Fat-Tree</td>
<td>25</td>
<td>4</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>IBM SP-2</td>
<td>Banyan</td>
<td>25</td>
<td>8</td>
<td>5</td>
<td>16</td>
</tr>
<tr>
<td>Intel Paragon</td>
<td>2D Mesh</td>
<td>11.5</td>
<td>16</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>Meiko CS-2</td>
<td>Fat-Tree</td>
<td>20</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>CRAY T3D</td>
<td>3D Torus</td>
<td>6.67</td>
<td>16</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>DASH</td>
<td>Torus</td>
<td>30</td>
<td>16</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>J-Machine</td>
<td>3D Mesh</td>
<td>31</td>
<td>8</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>Monsoon</td>
<td>Butterfly</td>
<td>20</td>
<td>16</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>SGI Origin</td>
<td>Hypercube</td>
<td>2.5</td>
<td>20</td>
<td>16</td>
<td>160</td>
</tr>
<tr>
<td>Myricom</td>
<td>Arbitrary</td>
<td>6.25</td>
<td>16</td>
<td>50</td>
<td>16</td>
</tr>
</tbody>
</table>

- Wide links, smaller routing delay
- Tremendous variation
Switches

- Basic Switch Organization
  - Input ports, output ports, crossbar internal switch, buffer, control

![Switch Diagram]

Basic Switching Strategies

- Circuit Switching
  - establish a circuit from source to destination

- Packet Switching
  - Store and Forward (SF)
    - move entire packet one hop toward destination
    - buffer till next hop permitted
    - e.g. Internet
  - Virtual Cut-Through and Wormhole
    - pipeline the hops: switch examines the header, decides where to send the message, and then starts forwarding it immediately
    - Virtual Cut-Through: buffer on blockage
    - Wormhole: leave message spread through network on blockage
Store-and-Forward (SF)

- Performance Model
  - Startup time $T_s$: the time required to handle a msg at the sending/receiving nodes;
  - Per-hop time $T_h$: the transfer time of the msg header in a link
  - Per-word transfer time $T_w$: if the link bw is $r$, then $T_w = 1/r$
- Latency for a message of size $m$ words to be transmitted through $l$ links:
  \[
  T_{\text{comm}} = T_s + (mT_w + T_h)l
  \]

Cut-Through and Wormhole

- Each message is broken into fixed units called flow control digits (flits)
- Flits contain no routing information
- They follow the same path established by a header.
- A message of size $m$ words traverses $l$ links:
  \[
  T_{\text{comm}} = T_s + lT_h + mT_w
  \]
Routing

- Recall: routing algorithm determines
  - which of the possible paths are used as routes
  - how the route is determined
  - $R: N \times N \rightarrow C$, which at each switch maps the destination node $n_d$ to the next channel on the route

- Issues:
  - Routing mechanism
    - arithmetic
    - source-based port select
    - table driven
    - general computation
  - Properties of the routes
  - Deadlock free
Routing Mechanism

- need to select output port for each input packet
  - in a few cycles
- Simple arithmetic in regular topologies
  - ex: \( \Delta x, \Delta y \) routing in a grid
    - west (-x) \( \Delta x < 0 \)
    - east (+x) \( \Delta x > 0 \)
    - south (-y) \( \Delta x = 0, \Delta y < 0 \)
    - north (+y) \( \Delta x = 0, \Delta y > 0 \)
    - processor \( \Delta x = 0, \Delta y = 0 \)
- Reduce relative address of each dimension in order
  - Dimension-order routing in k-ary d-cubes
  - e-cube routing in n-cube

Properties of Routing Algorithms

- Deterministic
  - route determined by (source, dest), not intermediate state (i.e. traffic)
- Adaptive
  - route influenced by traffic along the way
- Minimal
  - only selects shortest paths
- Deadlock free
  - no traffic pattern can lead to a situation where no packets mover forward
Deadlock-Free Routing

- How can it arise?
  - necessary conditions:
    - shared resource
    - incrementally allocated
    - non-preemptive
- How do you avoid it?
  - constrain how channel resources are allocated
  - ex: dimension-ordered routing
- How to prove that a routing algorithm is deadlock free
  - resources are logically associated with channels
  - messages introduce dependences between resources as they move forward
  - need to articulate the possible dependences that can arise between channels
  - show that there are no cycles in Channel Dependence Graph
    - find a numbering of channel resources such that every legal route follows a monotonic sequence
    => no traffic pattern can lead to deadlock
  - network need not be acyclic, on channel dependence graph
Example: k-ary 2D array

- Theorem: x,y routing is deadlock free
- Numbering

```
  0  1  2  3  03
 01 02 03 03 03
 03 03 03 03 03
```

- any routing sequence: x direction, turn, y direction is increasing

Breaking deadlock with virtual channels

Packet switches from lo to hi channel
Flow Control

- Multiple streams trying to use the same link at same time
  - Ethernet/LAN: collision detection and retry after delay
  - TCP/WAN: buffer, drop, adjust rate
  - any solution must adjust to output rate
- Flow control in parallel computers
  - Link-level control: Block in place
  - The dest buffer may not be available to accept transfers; this may cause the buffer at sources to fill and exert back pressure in end-to-end flow control.

Network characterization

- Topology (what)
  - physical interconnection structure of the network graph
  - direct vs indirect
- Routing Algorithm (which)
  - restricts the set of paths that msgs may follow
- Switching Strategy (how)
  - how data in a msg traverses a route
  - circuit switching vs. packet switching
- Flow Control Mechanism (when)
  - when a msg or portions of it traverse a route
  - what happens when traffic is encountered?
  - Interplay of all of these determines performance