Supercomputer Interconnection Networks

CS 740 Class Talk
April 19, 1995

Presentation by
Ariel Tamches
6366 CS
`tamches@cs.wisc.edu`

Outline Of Talk

What makes a good supercomputer network?

Survey of supercomputer interconnection networks

The CM-5 interconnection networks

Interconnection Networks — Needs

- Low latency node-to-node message passing, optimized for small messages.
- Low latency synchronization.
- Scalability
  - Price scalability: network cost/complexity proportional to #CPUs.
  - Performance scalability: want latency to degrade very slowly, *even under heavy load*.

Why Low Latency?

LANs advertise "peak bandwidth" numbers.
They are useless for our purposes, because

- Peak means "less than"
- Message xfer time = $\textit{latency(first-bit)} + \frac{numbytes}{bandwidth}$.
  - Near-peak *only* when second term dominates first.
  - Supercomputers usually pass *very* short messages, so *first* term dominates!

Supercomputers pass short messages (an example)

Supercomputer memory is distributed, but programmers want the illusion of shared memory.

Shared memory is possible

- Pick sharing granularity (e.g. cache block).
- Divide virtual memory into an array of these chunks.
- 1 CPU may have write access to chunk x at a time.
- Any CPUs may share read access to chunk x at a time.

On read access fault to a chunk

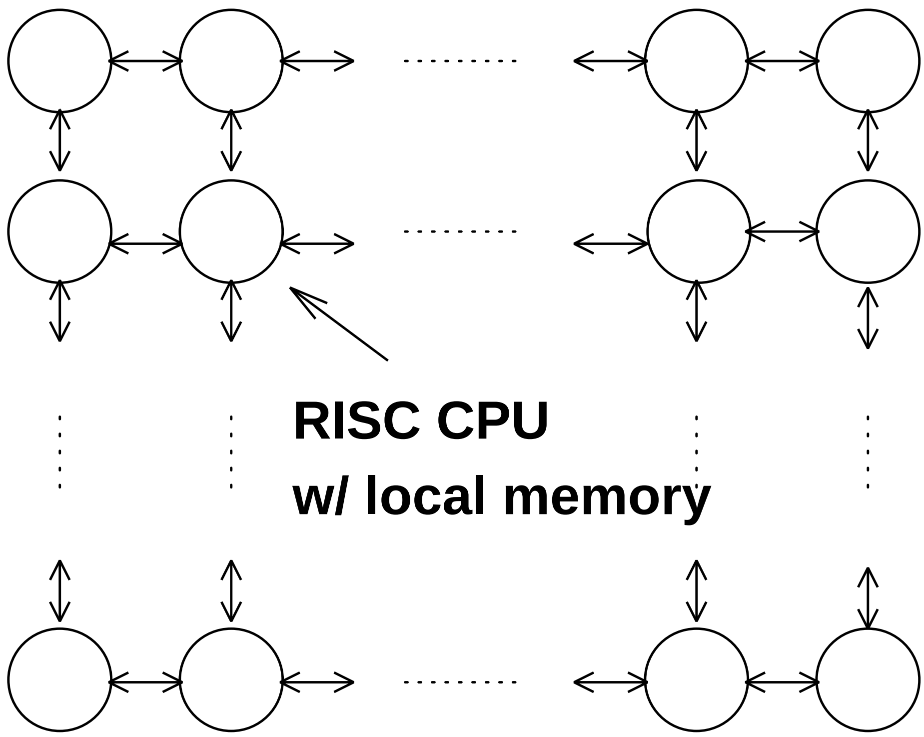
- Obtain copy from any CPU having it.

On write access fault

- Multicast invalidation to CPU(s) having it; obtain possibly-updated copy.

$\Rightarrow$ Bottom line: *very* short messages are being sent back and forth (about 32 bytes)

Static Network Example



Static Networks

- Processors directly connected by wires
- CPU x can only send to neighbors
  - Need software to route all other messages
  - Beware OS kernel and context switch overheads!

We can dismiss some static networks:

Ring:

+ Passes price scalability test (ICN cost $\propto n$).

— Fails performance scalability test.

Fully connected

+ Passes performance scalability test.

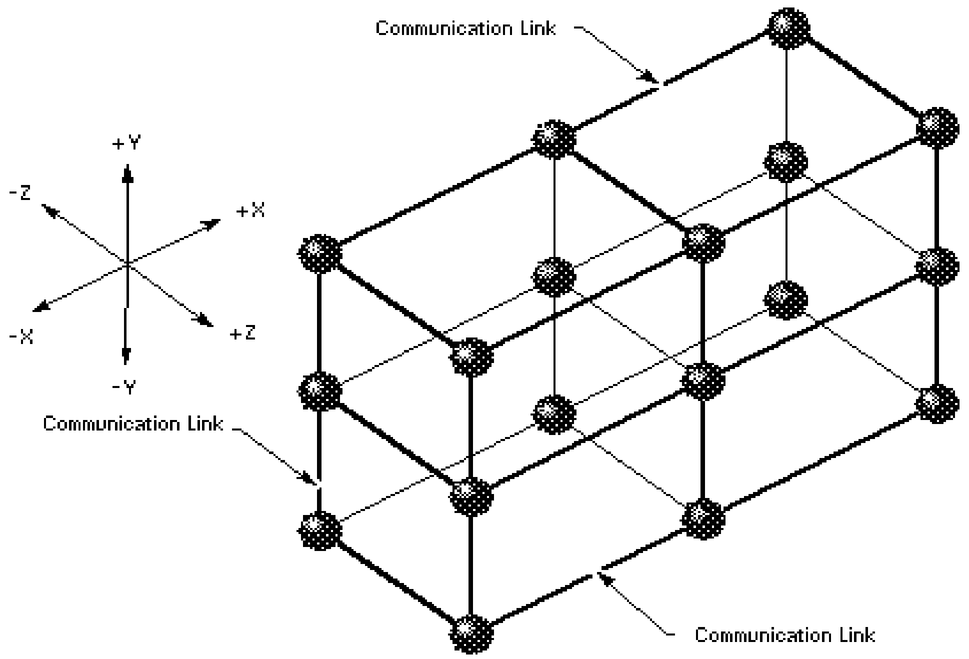
— Fails price scalability test (ICN cost $\propto n^2$).

Static Networks: Meshes

k-D mesh has n total CPUs, each of degree $2k$.

Cray T3D has 3-D mesh ("torus"):

- Each CPU connects to 6 others.



(Software) routing done 1 dimension at a time:

- Example: (a, b, c) to (x, y, z):
  $(a, b, c) \Rightarrow (x, b, c) \Rightarrow (x, y, c) \Rightarrow (x, y, z)$

\+ Passes price scalability test; # wires $\propto kn$.

— Fails performance scalability test; latency $\propto \sqrt[k]{n}$.

— Bottlenecks quickly under moderate load (hard to evenly distribute load).

Static Networks: Hypercubes

Designed as an improvement to meshes.
k-D hypercube has $n = 2^k$ nodes, each with degree k.

Assign distinct binary numbers to each node.
Two nodes connected iff they differ by 1 bit.

Evaluation of Hypercubes:

— Fails price scalability test; nodes have degree lgN, which is too expensive.

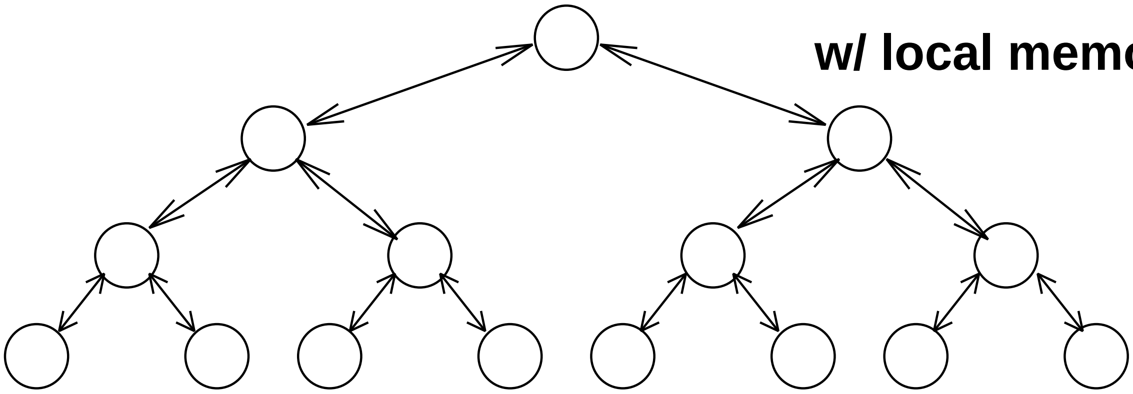
+ Routing latency $\propto k(lgN)$.

— Like meshes, congests quickly under contention.

Static Networks: Binary Trees

Starting point for understanding CM-5 data network.

Each circle is a CPU w/ local memory



+ (Software) routing easy.

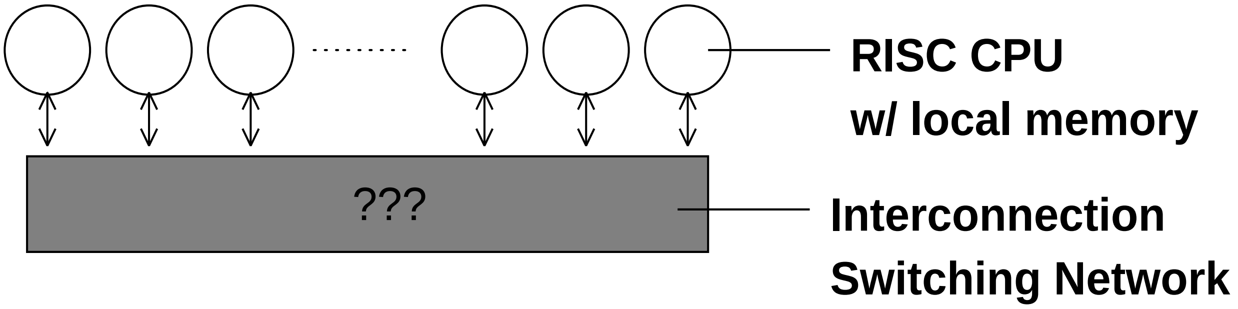
+ Passes price scalability test easily; # wires $\propto n$.

+ *Almost* passes performance scalability test; latency appears $\propto lgN$! But...

— Like meshes and hypercubes, congests easily under contention. Root node will become *very* congested.

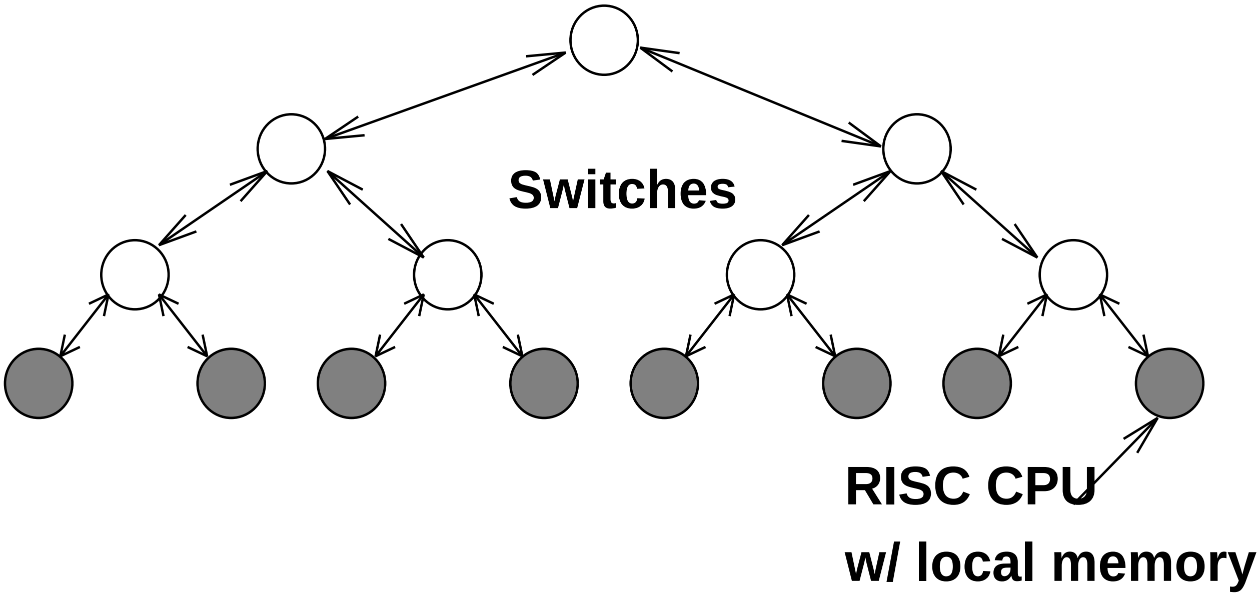
Dynamic Networks

- Have switching network, so any two nodes can communicate in hardware.



- Generally superior to static networks, despite extra hardware cost.
  - No need for software routing.
  - Not inherently difficult to distribute load.
- Naive dynamic network:
  - Shared bus — fails performance scalability test; bus can accomodate 1 message at a time, so only one CPU can use the network at a time.

Dynamic Networks: Binary Trees



+ Routing is easy; travel up to lowest common ancestor, and descend.
+ Passes price scalability test; $\propto n$ switches, $\propto n$ links.
+ Potentially, latency $\propto lg(n)$.

— Nodes nearer to root get a disproportionate share of network traffic, leading to congestion. Does not pass performance scalability test under heavy load.

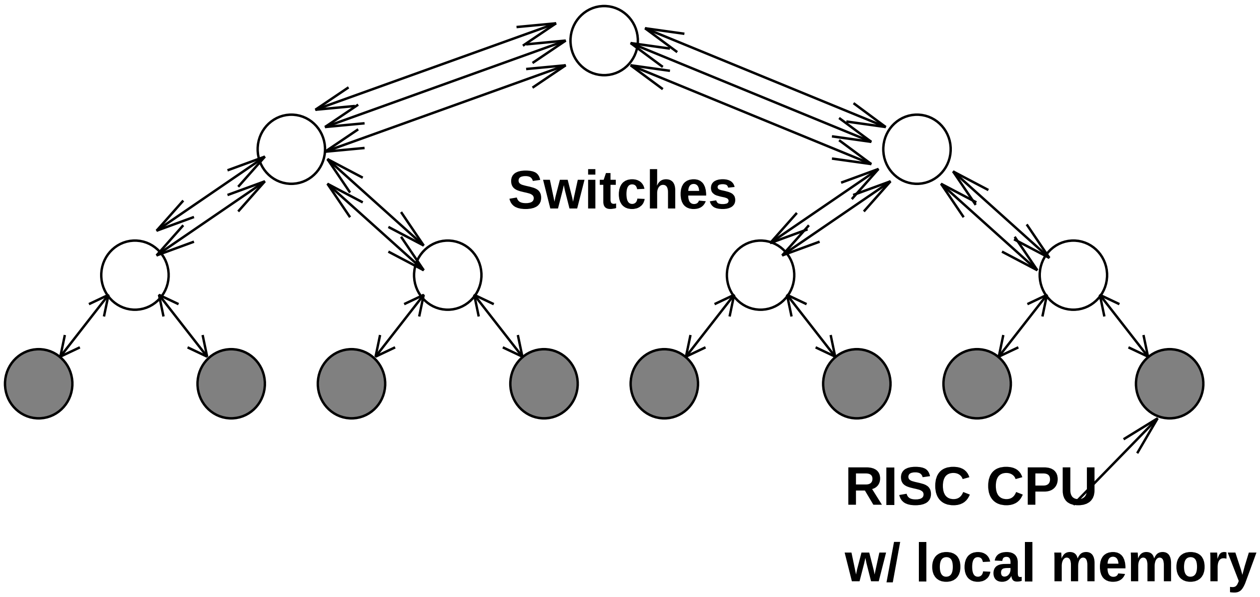
Dynamic Binary Fat-Tree Network

How to fix dynamic binary tree congestion problem?

Spend extra money to make nodes nearer the top faster.

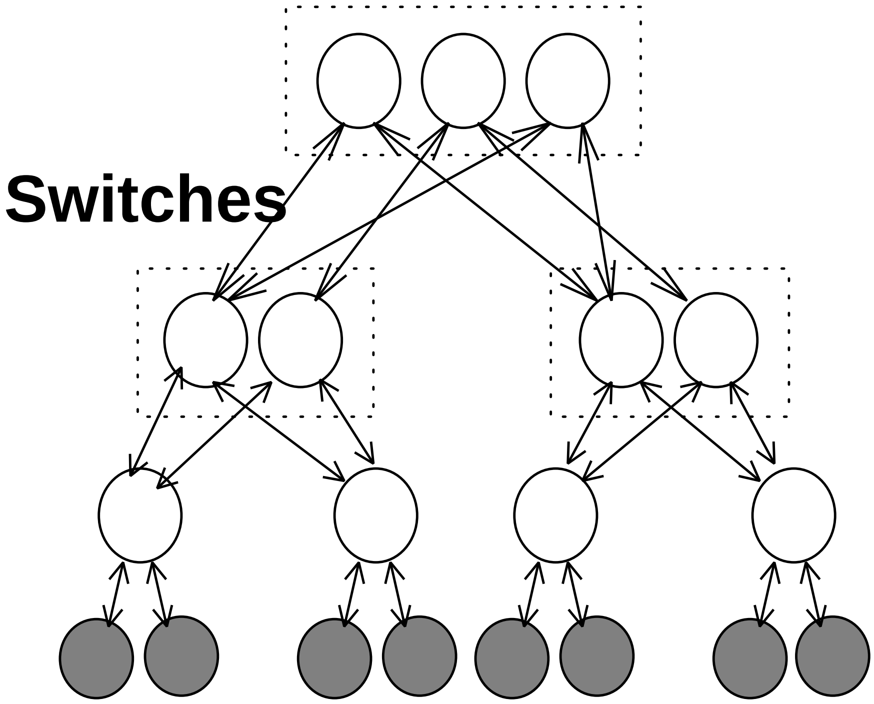
- Can still pass price scalability test
  (such nodes are relatively few)

Provide extra data paths through such switches and extra links to their child switches.



Dynamic Binary Fat-Tree (cont'd)

Switch internals change (they really become multiple switches):



Dynamic Binary Fat-Tree Properties

- Build a fat-tree of n processors with root capacity (RC) between $n^{2/3}$ and n and level-k capacity of $\min(\frac{n}{2^k}, \frac{RC}{2^{2k/3}})$. The fat-tree
  - Has capacities that grow at rate $\sqrt[3]{4}$ for levels above $3lg\frac{n}{RC}$, and that double for other levels.
  - Can be built in $O((RC \times lg\frac{n}{RC})^{3/2})$ volume.
  - Can be built in $O(n \times lg\frac{RC^3}{n^2})$ components.
- Any such fat-tree can route *any* message, under *any* load, within a $lg^3 n$ factor of *any* other network with the same number of nodes and volume.
- During time interval t, a message set M puts at most $O(t \times lg(\frac{n}{V^{2/3}})) = O(t \times RC)$ load on any one switch.

Cut-through / Wormhole Routing

Routing choices in a dynamic network:

Circuit Switching (CS)

- Establish circuit (reserve all switches on path), then send contention-free with full pipelining.

Store-and-forward (SAF) switching

- *Entire* msg sent 1 link, buffered, and forwarded.
- Can't pipeline msg across multiple links.

Cut-through switching

- When msg head arrives at switch
  - If next link free, *immediately* begin forwarding.
  - Else, store-and-forward as usual.

Cut-through / Wormhole (cont'd)

Cut-through is a *big* latency win

+ No circuit setup time, like SAF.

+ Can pipeline across multiple links, like CS.

+ Links released when last word forwarded, like SAF.

— What to do on CRC failure?

Why don't "normal" LANs use cut-through?

- SAF can utilize all links when given many packets.
- But cut-through superior for 1-packet messages.

Wormhole Routing:

- Minimal variant of cut-through. Flow control allows stopping reception *in the middle* of a message.

+ Don't need any buffer space at links.

— Leads to deadlock issues.

CM-5 uses wormhole routing.

Connection Machine CM-5

- From 32 to 16384 SPARC CPUs @ 33 MHz.
- $\geq$ 32 MB per node.

- Two message passing interconnection networks.
  - Data network (standard cpu-to-cpu msg passing)
  - Control network (broadcasts, synchronization)

- First supercomputer to optimize for small messages.

CM-5 Partitions

- 1 outside control processor.
- $\geq$ 32 processing nodes.
- *Dedicated* subtrees of data & control nets.
- Think of each partition as 1 supercomputer.

Whenever running, programs get a dedicated partition.

At Wisconsin:
`mendota` (32 nodes) + **`monona`** (32 nodes) or
`mendota` (64 nodes)

- No OS overhead
  - Data and control network functions don't go to kernel! A *big* win.
  - Safe because we "own" the entire paritition while running. But, context switching becomes tricky.

CM-5 Network Interface

Level of indirection between processors and networks.

- Several incoming & outgoing queues for each node.
- User unaware of network architecture.
- 0-based partition-relative node addressing.

Consequences

\+ CPUs & network can be independently upgraded.

— Network must guarantee delivery (flow control at link level), complicating the network.

\+ Messages sent asynchronously ("fire and forget").

- Programs made faster and simpler.

CM-5 Data Network

Dynamic 4-ary fat-tree interconnection network.

- 4 children per switch.
- 2 parents per node (lowest two levels)
- 4 parents per node (all higher levels)

- Messages contain
  - Routing info (# levels up, route to take down)
  - Data length, in words (1-5)
  - Tag (4 bits)
  - Data (up to 20 bytes)
  - CRC

- Sending one message empirically takes 3–7 μsecs.
- Nodes have 40Mbytes/sec raw bandwidth into and out of data network.
  - A useful number only when lots being sent.

CM-5 Data Network (cont'd)

- Exist several parent choices when routing upwards.
  - Theorum: choosing randomly among currently-available parents is optimal.

Data network does *not* guarantee in-order delivery of messages.

+ Lets data network treat messages independently & concentrate on sending each as fast as possible.

— A burden when logical message must be sent as several 20-byte messages (software SAR).

Wormhole Routing Considerations

Due to cut-through / wormhole routine, message head may be several links away when CRC error detected.

- First switch to detect CRC error treats as actual error, and retransmits.
- Then, it appends complement of CRC. Switches seeing CRC followed by complement don't retransmit.
- Duplicate messages detected by data-net chip counters, which should be equal at checkpoints such as context switches. Else, hardware error is reported.
- Same for lost messages.

Data Network Deadlock Avoidance

Data network guarantees to accept & deliver all messages *provided* nodes eventually consume messages delivered to them.

+ Deadlock impossible if contract obeyed by nodes.

— Many algorithms cannot uphold contract given bounded FIFO queues. May fail to progress due to backlogged FIFO. When so, must *not* block or retry repeatedly (must periodically check for and accept delivered messages from incoming FIFO).

Solution 1

- Given FIFO size $\propto n$.
- Node X doesn't send to Y until Y tells X it has room.

+ Avoids deadlock by ensuring bounded # msgs outstanding between any 2 nodes.

— Complex software overhead.

Solution 2

- Given 2 in and 2 out FIFOs per node (CM-5 has).
- Send requests on "left" port; recv responses on "right" port.

Control Network

- Fast broadcasts
- Fast combining

A dynamic binary, but *not* a fat tree. Why not?

- We have entire partition to ourselves; never want to do more than one broadcast or combining operation simultaneously.

Reduction

- Combines 32-bit values from all nodes (OR, XOR, etc.) and delivers to all nodes

Forward scan

- Apply (OR, XOR, etc.) on 32-bit value for nodes 0 thru $i - 1$ and deliver to node i, $\forall i$.

Router-done

- Lets nodes know when data network is empty.

Context Switching

Partitions can run in timeshared mode.
Control processor signals context switch with a broadcast through the control network.

On context switch, *entire* partition state must be saved.

State exists in:

- Network interface chip
  - Easy; NI chips attached to respective nodes
- Data network (msgs in transit)
  - Can't wait for msgs to be delivered (may be forever if user deadlocks)
  - Messages in data net chip in "all-fall-down" mode are routed down on a hardwired path to some node, which saves the state.
  - On resume, system resends message
- Control network (i.e. broadcasts in transit)
  - Flushed via special broadcast that aborts in-progress action. NI had saved operation params (until completed), so easy to restart.

Current Directions

NOWs (Networks of Workstations)

- A cluster of commodity machines, running commodity OS's, using commodity LANs (e.g. ATM).

+ Less expensive than CM-5 (no design, implementation of a custom network).

— Not optimized for short messages!

— Fast broadcast, synchronization a question mark.

— Scalability a question mark.

— Ability to avoid OS a question mark.

What Have we Learned?

- Supercomputer networks should be optimized for short messages.
  - Use cut-through or wormhole routing.
- Should handle uneven communication distribution.
- Fat-trees are a quick, scalable, guaranteed-efficient choice.
- Avoid OS overhead (requires guarantee that running programs have a dedicated portion of the network).
- Many established, modern supercomputers (e.g. Cray T3D) have terrible networks.
- Current directions (NOWs) are certainly more cost-efficient, but may sacrifice important performance features.