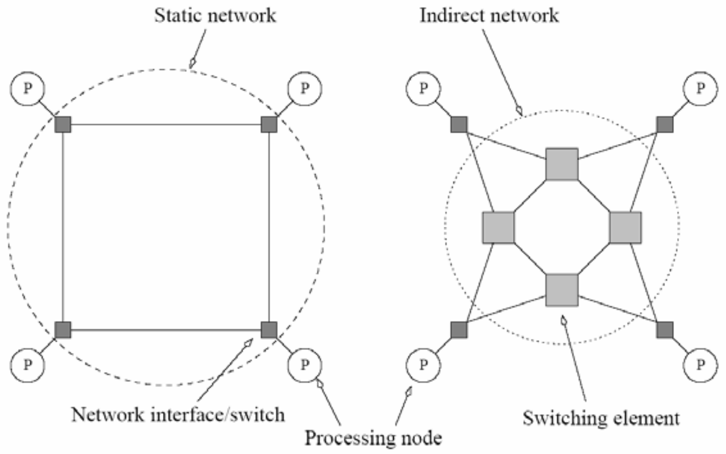
Physical Organization of Parallel Platforms

Alexandre David

1.2.05

Static vs. Dynamic Networks



Interconnection networks built using links and switches. How to connect:

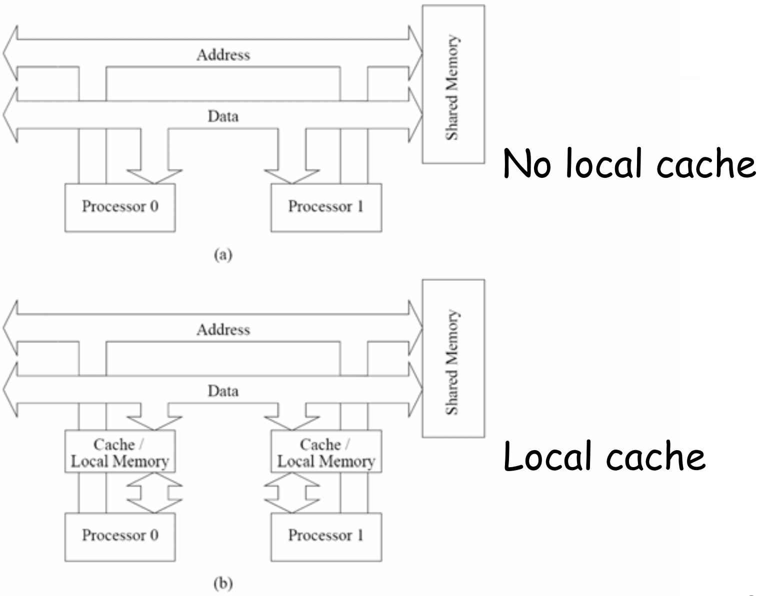
•Static networks, or direct networks, have p2p *static* communication links.

•Dynamic networks, or indirect networks, have switches to *route* the communication.

Number of (output) ports on a switch = degree. Mapping input -> output implemented by different technologies.

Static networks to connect processors <-> processors, dynamic networks to connect processors <-> memory.

Bus Based Networks



3

Good:

•Cost scales linearly with the number of nodes.

•The distance between all the nodes is constant.

•It is ideal for broadcasting.

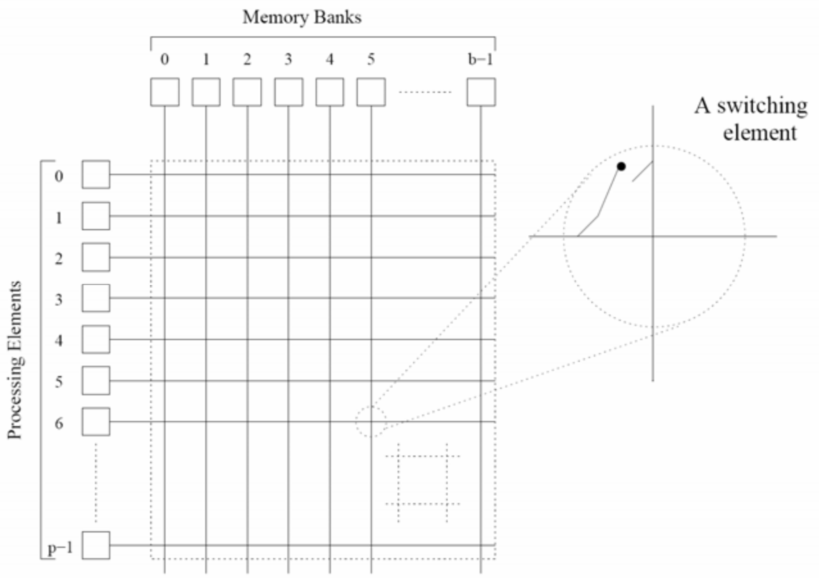
Bad:

•Shared bandwidth between all the nodes -> bottleneck in performance.

In practice bus based only for small SMP (Intel). Caches are only a trick to reduce bandwidth consumption on the bus (not to reduce bandwidth as stated in the book).

Both for processors & memory.

Crossbar Networks



4

Grid to connect *p* processors to *b* memory banks. Non blocking in the sense that a connection (routing) does not block the connection of any other processing node, in contrast to multistage networks.

Good: scalable in performance (non blocking).

Bad: number of switches = *p*b*, not scalable in cost.

Multistage Networks

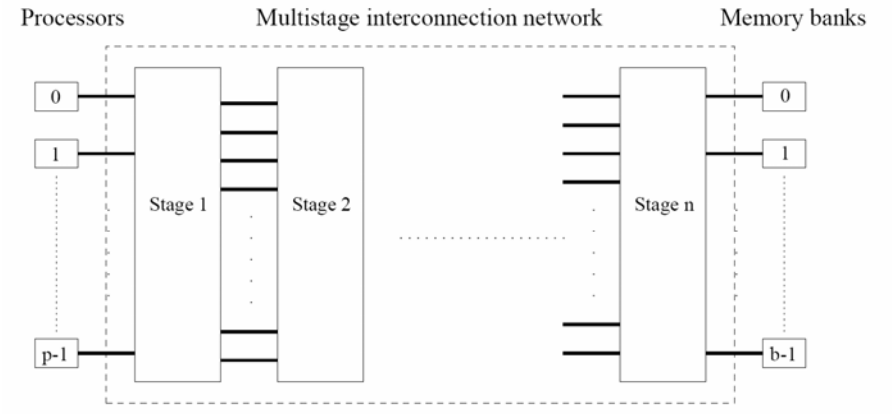


Figure 2.9 The schematic of a typical multistage interconnection network.

Intermediate network, between crossbar and bus.

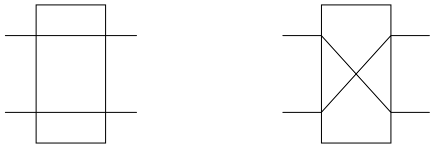
Again *p* processing nodes and *b* memory banks.

A common type is the **omega** network:

•It has log*p* stages (with matching number of inputs and outputs).

•It has a perfect shuffle interconnection pattern, easy with left rotate.

Switches in Omega Networks



Configurations: pass-through and cross-over.

p/2 * log p switching nodes:
log p stages, p/2 inputs & outputs.

Omega Network

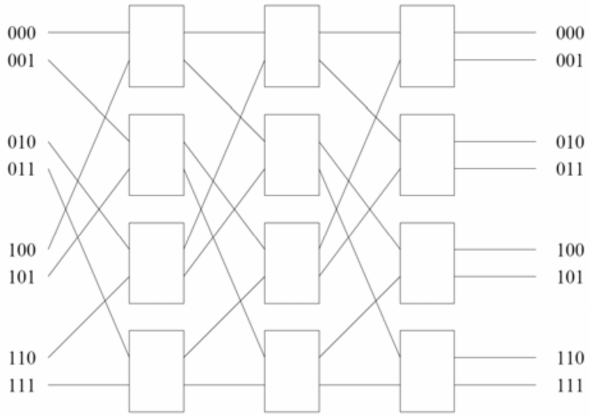


Figure 2.12 A complete omega network connecting eight inputs and eight outputs.

7

Blocking in Omega Networks

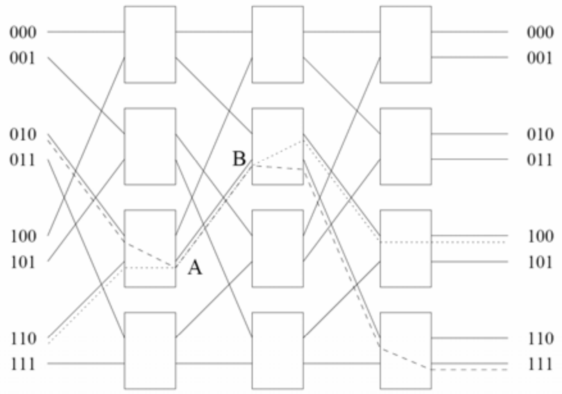


Figure 2.13 An example of blocking in omega network: one of the messages (010 to 111 or 110 to 100) is blocked at link AB.

Contention in the access, one is blocked. Such networks are called *blocking* networks.

So far processor <-> memory.

Processors <-> Processors Networks

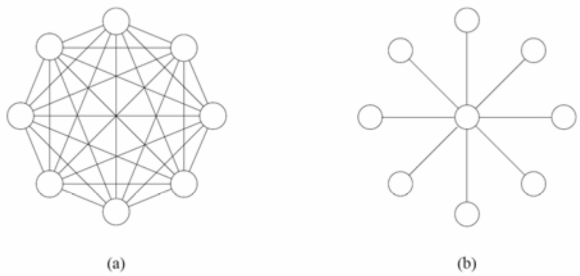


Figure 2.14 (a) A completely-connected network of eight nodes; (b) a Star connected network of nine nodes.

Performant, very expensive. Bottleneck, cheaper.

Number of edges: n(n-1)/2 vs. n-1.

Linear Arrays and Meshes

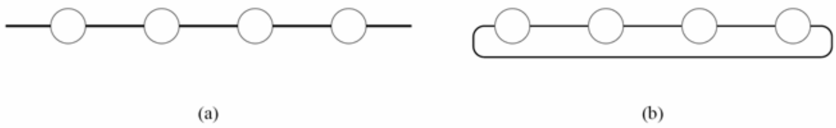


Figure 2.15 Linear arrays: (a) with no wraparound links; (b) with wraparound link.

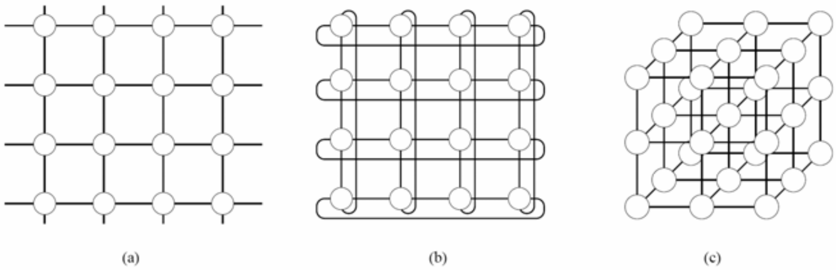


Figure 2.16 Two and three dimensional meshes: (a) 2-D mesh with no wraparound; (b) 2-D mesh with wraparound link (2-D torus); and (c) a 3-D mesh with no wraparound.

Wrap around changes the number of neighbors and distance for some nodes.

Linear array: each node has 2 neighbors (except start & end). It becomes a ring (or 1-D torus) with wraparound.

2-D mesh has ***p*** processors so the dimension is given by *sqrt(p)*. Every node (except on the border) has 4 neighbors. Attractive from a wiring point of view. Adding wraparound links gives a 2-D torus.

3-D, similarly. Every time we add a dimension, we add 2 neighbors. 3-D meshes good for physical simulations because they correspond to the modeled problem and the way processing is distributed.

Hypercubes

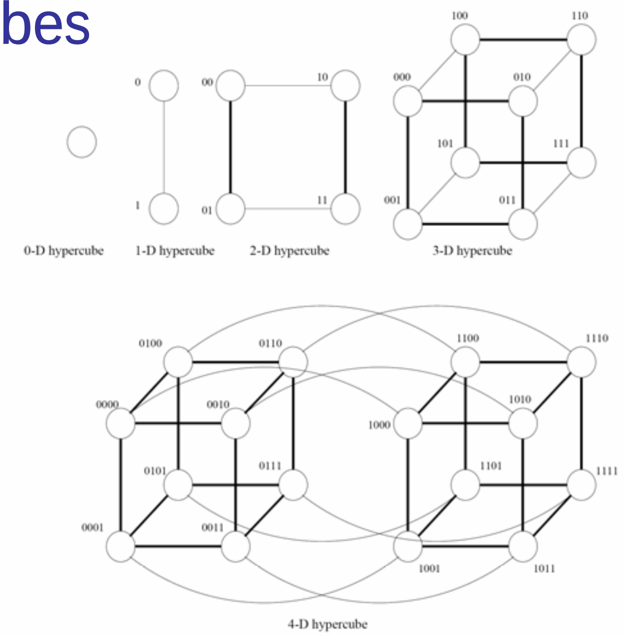


Figure 2.17 Construction of hypercubes from hypercubes of lower dimension.

13-02-2008 11

Hypercubes are the other extreme of linear meshes. 2 nodes per dimension and log *p* dimensions. Remember *p* processing nodes. Number of nodes for hypercube topology = $2^{\text{dimension}} = p$.

Very important: the clever way to distribute the indices. Remember this, it's useful to derive parallel algorithms running on hypercubes.

Property: minimum distance between two nodes = number of different bits in the two indices (how many dimensions we need to cross).

Tree Based Networks

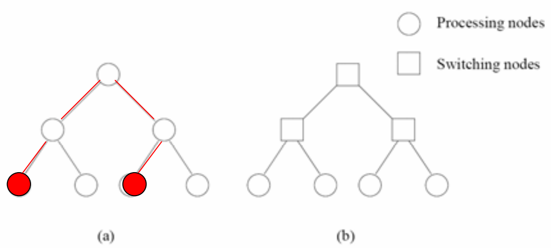


Figure 2.18 Complete binary tree networks: (a) a static tree network; and (b) a dynamic tree network.

(a) Some nodes share their connections for other nodes.

Routing for sending a message: Go up in the tree until it reaches a sub-tree that contains the destination, then go down. Performance in function of the height of the tree $O(\log p)$.

Issue with communication: Nodes (or switches) up in the tree may be bottlenecks w.r.t. bandwidth. Fat trees: alternative to give more bandwidth to shared routes.

Fat Trees

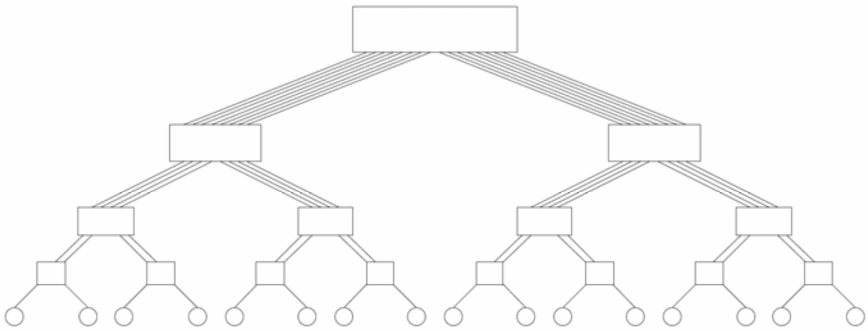


Figure 2.19 A fat tree network of 16 processing nodes.

More bandwidth where we need it.

Evaluating The Networks

- All the previous topologies have advantages and disadvantages.
- Important factors: cost and performance.
- Define criteria to characterize cost and performance.

Your turn: Give suggestions on measure criteria.

Criteria

- Diameter: maximum distance $p_a \leftrightarrow p_b$.
- Connectivity.
- Bisection width.
- Bisection bandwidth.
- Cost.

Distance = shortest path between 2 nodes.
Diameter: How far 2 nodes may be.

•Completely connected: 1.

•Star connected: 2.

•Ring: floor(p/2).

•2-D mesh without wraparound: 2(dim-1). With wraparound: 2*floor(dim/2).
Note: dim = sqrt(*p*).

•Hypercube: dim (=log *p*).

•Complete binary tree: height=h, $p=2^{h+1}-1$, h =log($(p+1)/2$), travel $2h$.

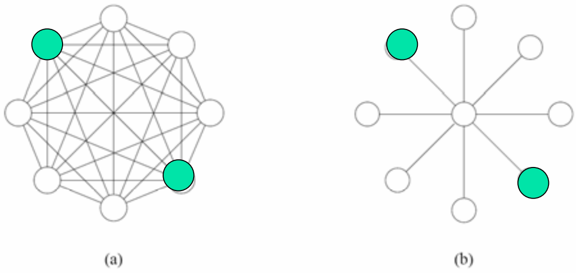


Figure 2.14 (a) A completely-connected network of eight nodes; (b) a Star connected network of nine nodes.

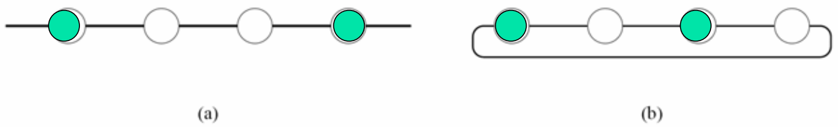


Figure 2.15 Linear arrays: (a) with no wraparound links; (b) with wraparound link.

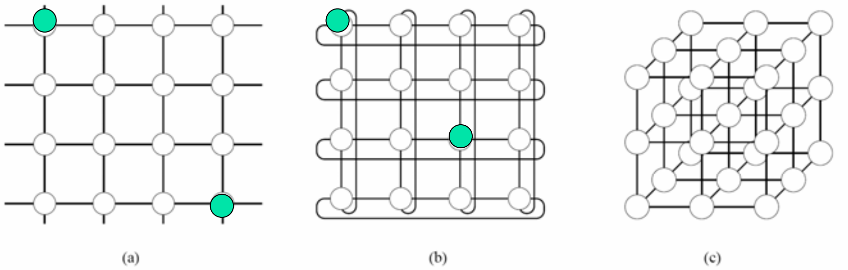


Figure 2.16 Two and three dimensional meshes: (a) 2-D mesh with no wraparound; (b) 2-D mesh with wraparound link (2-D torus); and (c) a 3-D mesh with no wraparound.

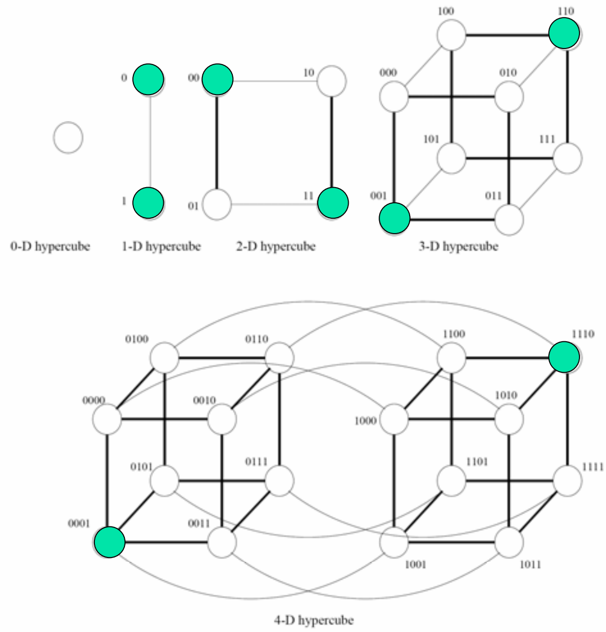


Figure 2.17 Construction of hypercubes from hypercubes of lower dimension.

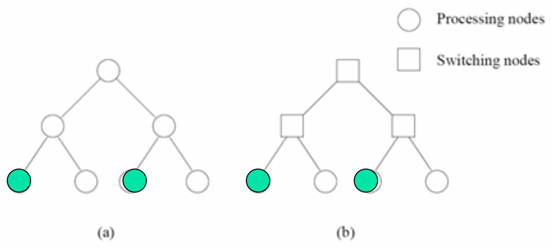


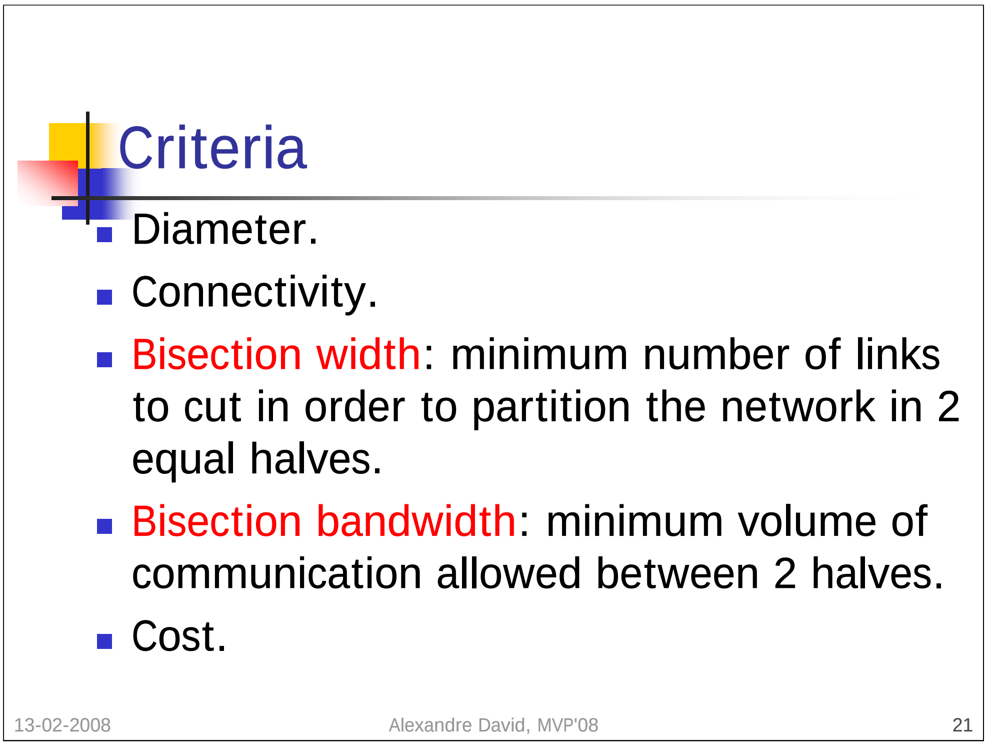
Figure 2.18 Complete binary tree networks: (a) a static tree network; and (b) a dynamic tree network.

Criteria

- Diameter.
- Connectivity: measure of multiplicity of paths.
- Bisection width.
- Bisection bandwidth.
- Cost.

High connectivity to lower contention and avoid congested networks.

With or without wraparound gives different results (consider min). Interesting to remember: Connectivity is *d* for *d*-dimensional hypercubes.



Bisection width measures the weakness/strength of the network, overall connectivity.

•Ring: 2.

•2-D mesh without wraparound: dim (=sqrt(p)); with wraparound: 2*dim.

•Completely connected network: needs to cut half of the edges = $p^2/2$.

•Hypercubes: how we construct… double nodes every time, so cut p/2 nodes.

Number of bits per link = channel bandwidth = channel rate (peak bit rate) * channel width (number of wires).

Bisection bandwidth also called cross-section bandwidth.

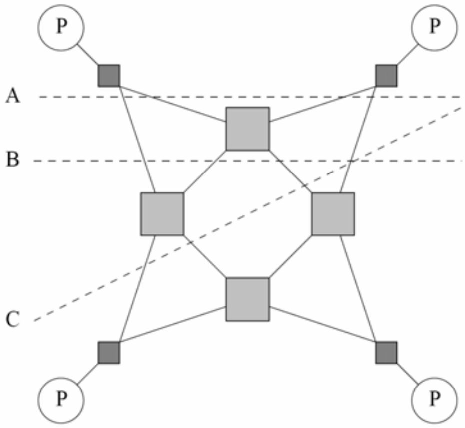


Figure 2.20 Bisection width of a dynamic network is computed by examining various equipartitions of the processing nodes and selecting the minimum number of edges crossing the partition. In this case, each partition yields an edge cut of four. Therefore, the bisection width of this graph is four.

Criteria

- Diameter.
- Connectivity.
- Bisection width.
- Bisection bandwidth.
- Cost: number of communication links, i.e., wires.

Number of wires:

•Linear arrays and trees: *p-1*.

•D-dimensional mesh: D**p*.

•Hypercube: *p**dim/2 (with dim = log *p*). Connectivity is dim.

Comparing The Topologies

Table 2.1 A summary of the characteristics of various static network topologies connecting p nodes.

Network	Diameter	Bisection Width	Arc Connectivity	Cost (No. of links)
Completely-connected	1	$p^2/4$	$p-1$	$p(p-1)/2$
Star	2	1	1	$p-1$
Complete binary tree	$2\log((p+1)/2)$	1	1	$p-1$
Linear array	$p-1$	1	1	$p-1$
2-D mesh, no wraparound	$2(\sqrt{p}-1)$	$\sqrt{p}$	2	$2(p-\sqrt{p})$
2-D wraparound mesh	$2\lfloor\sqrt{p}/2\rfloor$	$2\sqrt{p}$	4	$2p$
Hypercube	$\log p$	$p/2$	$\log p$	$(p\log p)/2$
Wraparound k-ary d-cube	$d\lfloor k/2\rfloor$	$2k^{d-1}$	$2d$	dp

Cache Coherence Protocols

- We need additional hardware to keep *multiple copies* of the same memory bank *consistent* with each other.
- We have seen that $$ is good but it does not come for free.
- Mechanism known as cache coherence protocol, usually described as state machines.

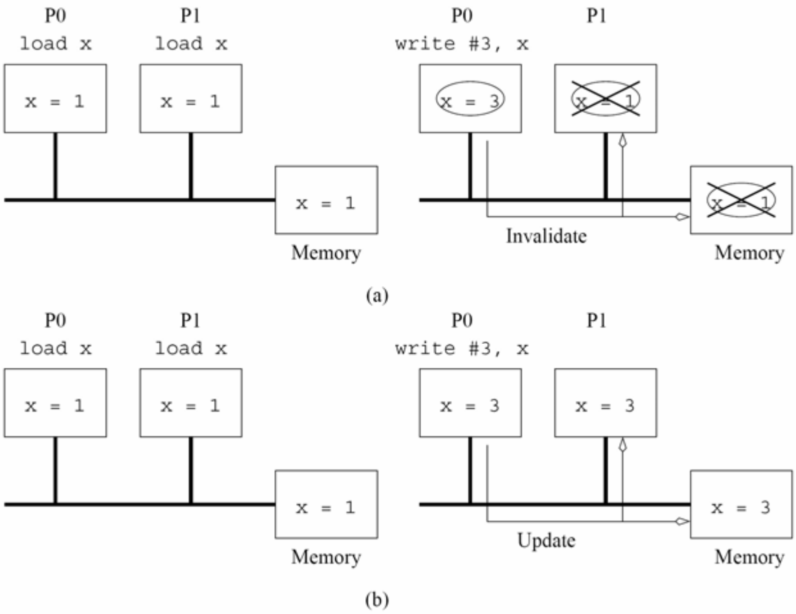


Figure 2.21 Cache coherence in multiprocessor systems: (a) Invalidate protocol; (b) Update protocol for shared variables.

26

2 principles: invalidate other copies or update other copies. 1st is cheap in terms of bandwidth.

Another factor that you don't see here: **false sharing**. Imagine y next to x on the same cache line. You will invalidate y as well. Degraded performance if different processors update different parts of the cache line: the cache line becomes shared although the variables are not.

Implementations of Cache Coherence Protocols

- Different ways to implement the protocol described by the state machine.
  - Snoopy cache: good on busses.
    Snoopy hardware that monitors states.
  - Directory based systems: states and presence bits for cache lines.
  - Distributed directory: physically distribute directory with memory.

Snoopy is popular.

Directory based is expensive.