Parallel Computing Platforms

**Ananth Grama, Anshul Gupta,
George Karypis, and Vipin Kumar**

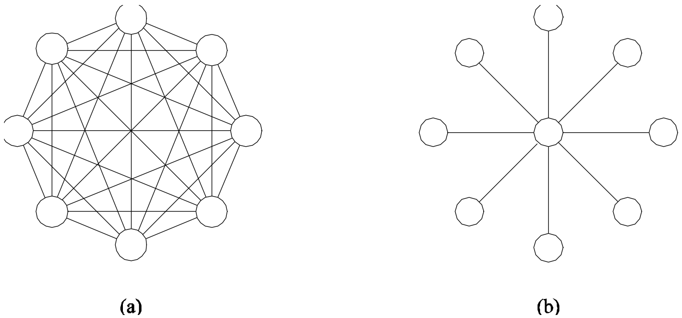
To accompany the text ``Introduction to Parallel Computing'',
Addison Wesley, 2003.

Network Topologies: Completely Connected Network

- Each processor is connected to every other processor.
- The number of links in the network scales as $O(p^2)$.
- While the performance scales very well, the hardware complexity is not realizable for large values of p.
- In this sense, these networks are static counterparts of crossbars.

Network Topologies: Completely Connected and Star Connected Networks

Example of an 8-node completely connected network.



(a) A completely-connected network of eight nodes;
(b) a star connected network of nine nodes.

Network Topologies: Star Connected Network

- Every node is connected only to a common node at the center.
- Distance between any pair of nodes is $O(1)$. However, the central node becomes a bottleneck.
- In this sense, star connected networks are static counterparts of buses.

Network Topologies: Linear Arrays, Meshes, and *k-d* Meshes

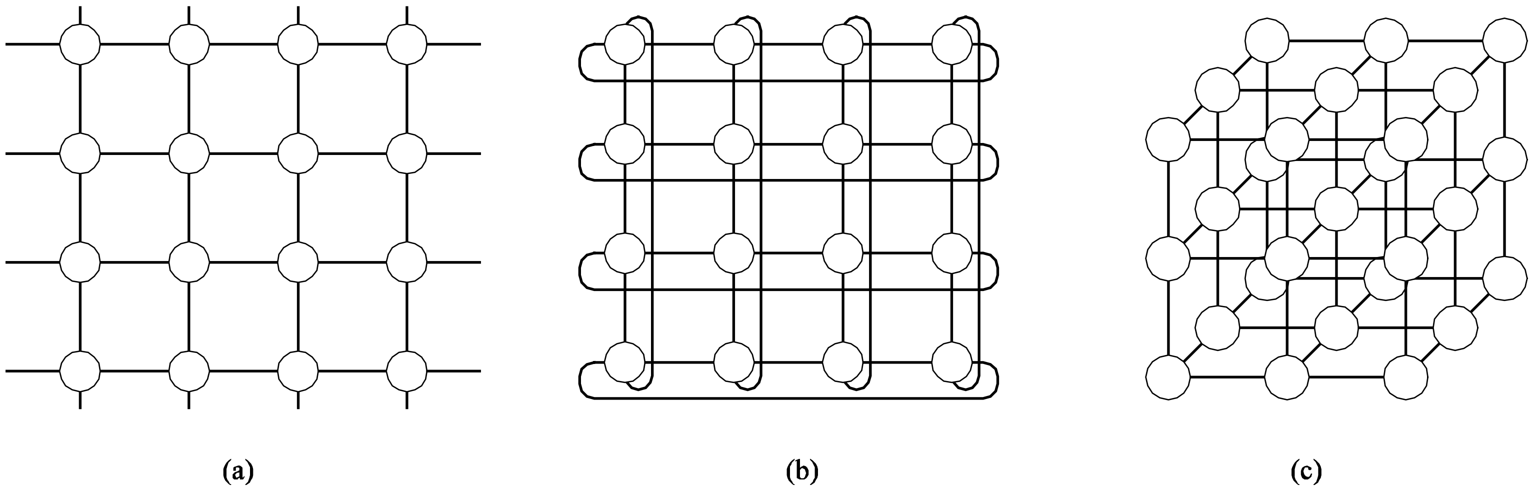
- In a linear array, each node has two neighbors, one to its left and one to its right. If the nodes at either end are connected, we refer to it as a 1-D torus or a ring.
- A generalization to 2 dimensions has nodes with 4 neighbors, to the north, south, east, and west.
- A further generalization to *d* dimensions has nodes with *2d* neighbors.
- A special case of a *d*-dimensional mesh is a hypercube. Here, $d = \log p$, where *p* is the total number of nodes.

Network Topologies: Linear Arrays

(a) (b)

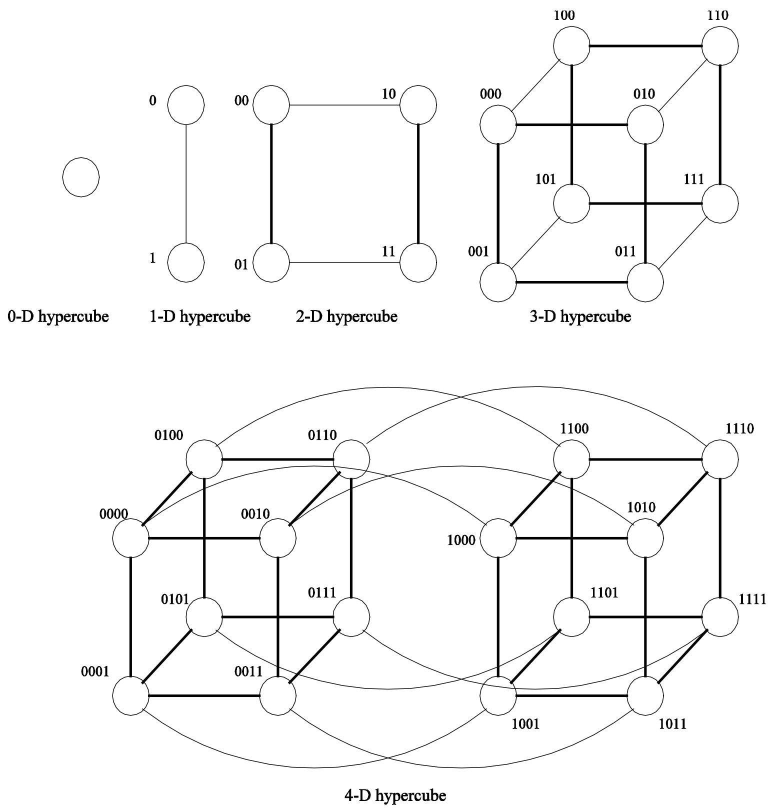
Linear arrays: (a) with no wraparound links; (b) with wraparound link.

Network Topologies: Two- and Three Dimensional Meshes



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.

Network Topologies: Hypercubes and their Construction

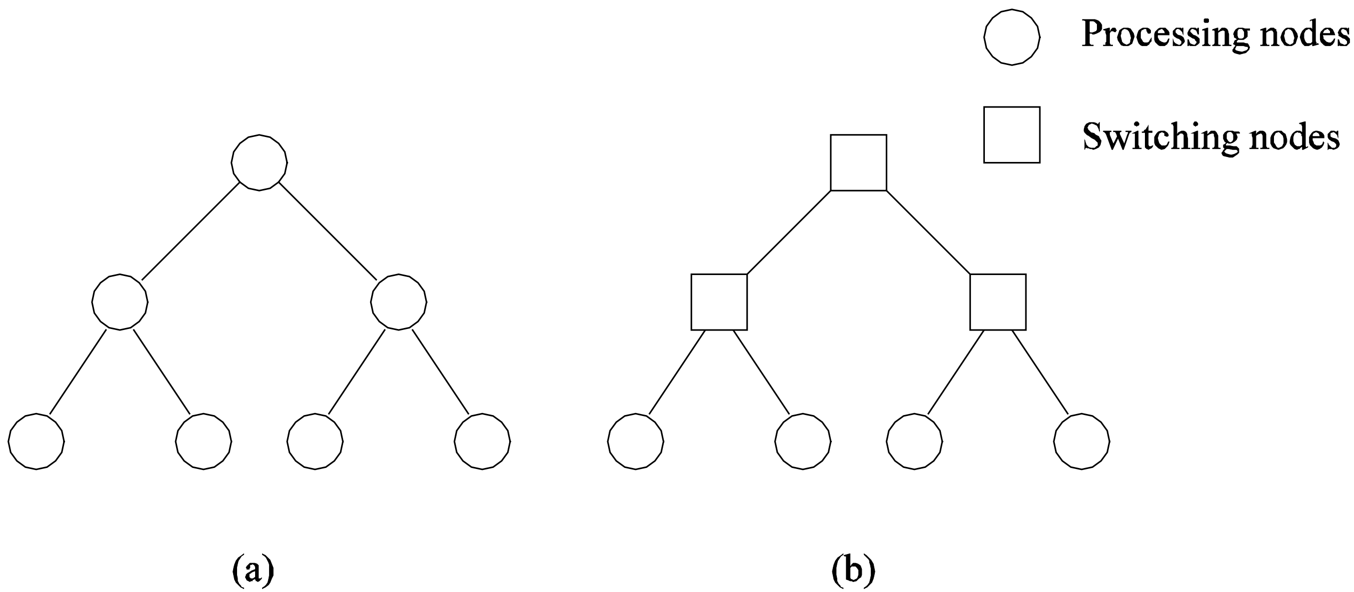


Construction of hypercubes from hypercubes of lower dimension.

Network Topologies: Properties of Hypercubes

- The distance between any two nodes is at most *log p*.
- Each node has *log p* neighbors.
- The distance between two nodes is given by the number of bit positions at which the two nodes differ.

Network Topologies: Tree-Based Networks

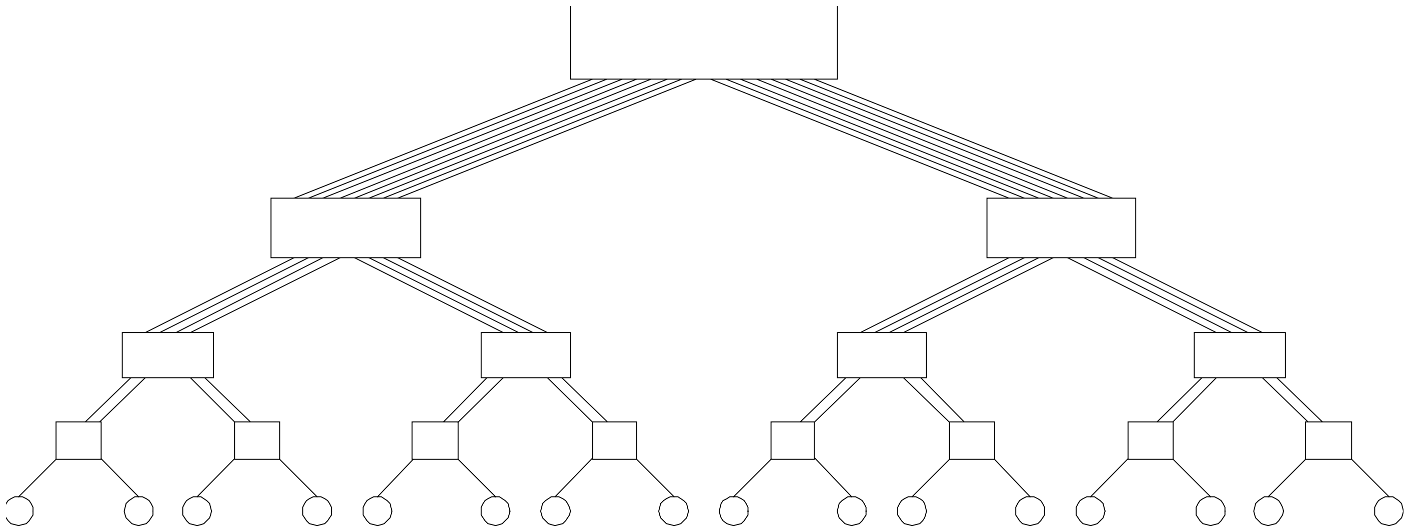


Complete binary tree networks: (a) a static tree network; and (b) a dynamic tree network.

Network Topologies: Tree Properties

- The distance between any two nodes is no more than *2logp*.
- Links higher up the tree potentially carry more traffic than those at the lower levels.
- For this reason, a variant called a fat-tree, fattens the links as we go up the tree.
- Trees can be laid out in 2D with no wire crossings. This is an attractive property of trees.

Network Topologies: Fat Trees



A fat tree network of 16 processing nodes.

Evaluating
Static Interconnection Networks

- *Diameter:* The distance between the farthest two nodes in the network. The diameter of a linear array is $p - 1$, that of a mesh is $2(\sqrt{p} - 1)$, that of a tree and hypercube is $\log p$, and that of a completely connected network is $O(1)$.
- *Bisection Width:* The minimum number of wires you must cut to divide the network into two equal parts. The bisection width of a linear array and tree is *1*, that of a mesh is $\sqrt{p}$, that of a hypercube is $p/2$ and that of a completely connected network is $p^2/4$.
- *Cost:* The number of links or switches (whichever is asymptotically higher) is a meaningful measure of the cost. However, a number of other factors, such as the ability to layout the network, the length of wires, etc., also factor in to the cost.

Evaluating Static Interconnection Networks

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

Communication Costs in Parallel Machines

- Along with idling and contention, communication is a major overhead in parallel programs.
- The cost of communication is dependent on a variety of features including the programming model semantics, the network topology, data handling and routing, and associated software protocols.

Message Passing Costs in Parallel Computers

- The total time to transfer a message over a network comprises of the following:
  - *Startup time* (t_s): Time spent at sending and receiving nodes (executing the routing algorithm, programming routers, etc.).
  - *Per-hop time* (t_h): This time is a function of number of hops and includes factors such as switch latencies, network delays, etc.
  - *Per-word transfer time* (t_w): This time includes all overheads that are determined by the length of the message. This includes bandwidth of links, error checking and correction, etc.

Store-and-Forward Routing

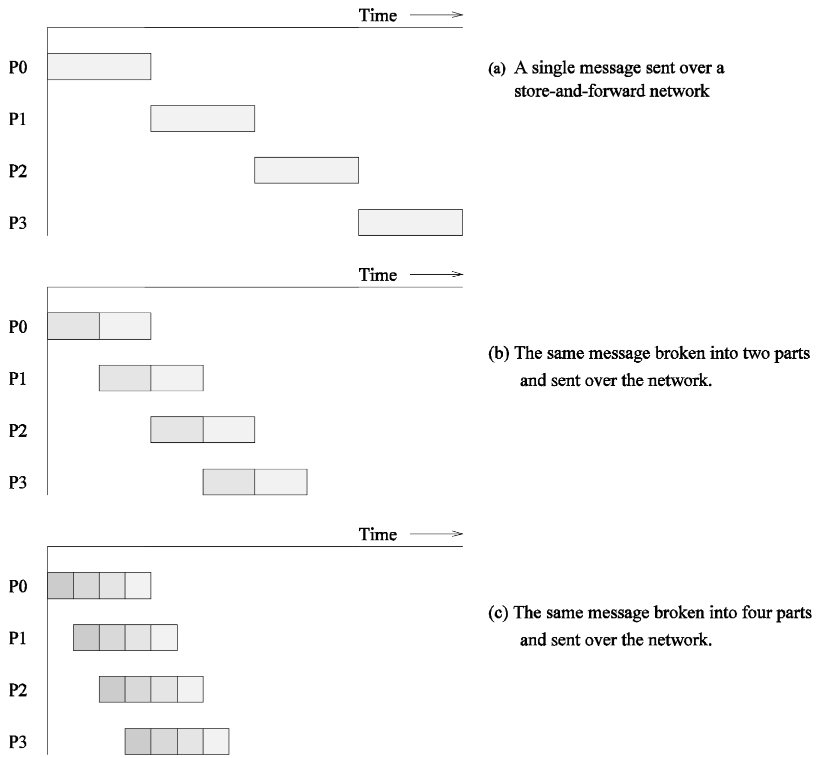
- A message traversing multiple hops is completely received at an intermediate hop before being forwarded to the next hop.
- The total communication cost for a message of size *m* words to traverse *l* communication links is

$$t_{comm} = t_s + (mt_w + t_h)l.$$

- In most platforms, t_h is small and the above expression can be approximated by

$$t_{comm} = t_s + mlt_w.$$

Routing Techniques



Passing a message from node P_0 to P_3 (a) through a store-and-forward communication network; (b) and (c) extending the concept to cut-through routing. The shaded regions represent the time that the message is in transit. The startup time associated with this message transfer is assumed to be zero.

Packet Routing

- Store-and-forward makes poor use of communication resources.
- Packet routing breaks messages into packets and pipelines them through the network.
- Since packets may take different paths, each packet must carry routing information, error checking, sequencing, and other related header information.
- The total communication time for packet routing is approximated by:

$$t_{comm} = t_s + t_h l + t_w m.$$

- The factor t_w accounts for overheads in packet headers.

Cut-Through Routing

- Takes the concept of packet routing to an extreme by further dividing messages into basic units called flits.
- Since flits are typically small, the header information must be minimized.
- This is done by forcing all flits to take the same path, in sequence.
- A tracer message first programs all intermediate routers. All flits then take the same route.
- Error checks are performed on the entire message, as opposed to flits.
- No sequence numbers are needed.

Cut-Through Routing

- The total communication time for cut-through routing is approximated by:

$$t_{comm} = t_s + t_h l + t_w m.$$

- This is identical to packet routing, however, t_w is typically much smaller.

Simplified Cost Model for Communicating Messages

- The cost of communicating a message between two nodes l hops away using cut-through routing is given by

$$t_{comm} = t_s + l t_h + t_w m.$$

- In this expression, t_h is typically smaller than t_s and t_w. For this reason, the second term in the RHS does not show, particularly, when m is large.
- Furthermore, it is often not possible to control routing and placement of tasks.
- For these reasons, we can approximate the cost of message transfer by

$$t_{comm} = t_s + t_w m.$$