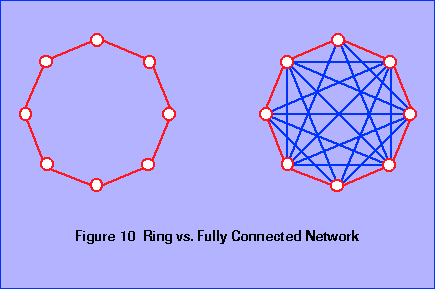
**Topology**

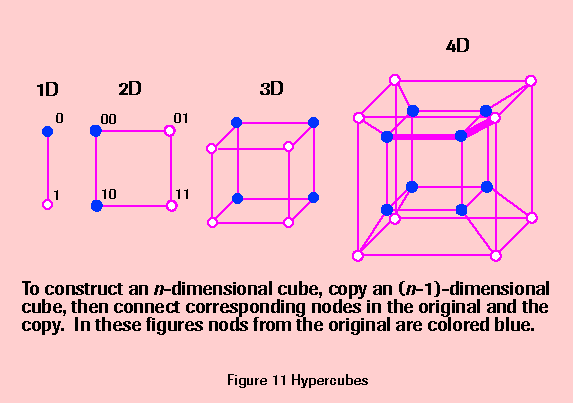
A major consideration in the design of parallel systems is the set of pathways over which the processors, memories, and switches communicate with each other. These connections define the *interconnection network*, or *topology*, of the machine. Attributes of the topology determine how processors will share data and at what cost.



The following discussion of the properties of interconnection networks is based on a collection of nodes that communicate via links. In an actual system the nodes can be either processors, memories, or switches. Unless otherwise noted the links will always be point-to-point data paths, i.e. not buses that are shared by several nodes. The properties discussed here apply equally to MIMD and SIMD machines, or to shared memory or distributed memory architectures. Examples of most of the topologies will be given in the survey of high performance systems.

Two nodes are neighbors if there is a link connecting them. The degree of a node is defined to be the number of its neighbors. Figure 10 shows two common topologies, a ring and a fully connected network, each with eight nodes. Each node in the ring is connected to only two other nodes, while each node in the fully connected network is linked to every other node. In practice the degree of a topology has an effect on cost; since the more links a node has the more logic it takes to implement the connections.

When a node is not connected to every other node, messages may have to go through intervening nodes to reach their final destination. The diameter of a network is the longest path between any two nodes. Again the ring and fully connected network show two extremes. A ring of http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2153.gif nodes has diameterhttp://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2147.gif, but a fully connected network has a fixed diameter (1) no matter how many nodes there are.



The diameter of a ring grows as more nodes are added, but the diameter of a fully connected network remains the same. On the other hand, a ring can expand indefinitely without changing the degree, but each time a new node is added to a fully connected network a link has to be added to each existing node. Scalability refers to the increase in the complexity of communication as more nodes are added. In a highly scalable topology more nodes can be added without severely increasing the amount of logic required to implement the topology and without increasing the diameter.

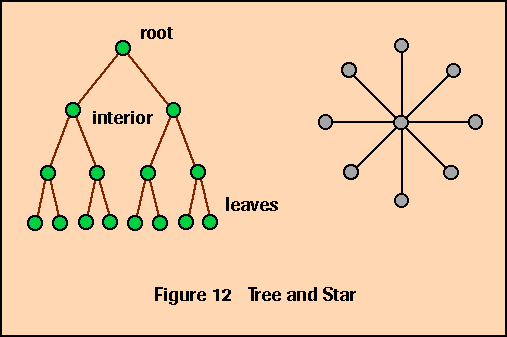
A scalable topology that has been used in several parallel processors is the hypercube, shown in Figure 11. A line connecting two nodes defines a 1-dimensional ``cube.'' A square with four nodes is a 2-dimensional cube, and a 3D cube has eight nodes. This pattern reveals a rule for constructing an http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2153.gif-dimensional cube: begin with an (http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2007.gif)-dimensional cube, make an identical copy, and add links from each node in the original to the corresponding node in the copy. Doubling the number of nodes in a hypercube increases the degree by only 1 link per node, and likewise increases the diameter by only 1 path. It is left as an exercise to prove that an http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2153.gif-dimensional hypercube has http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap1857.gif nodes, diameter http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2153.gif, and degree http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2153.gif.

Communication in a hypercube is based on the binary representation of node IDs. The nodes are numbered so that two nodes are adjacent if and only if the binary representations of their IDs differ by one bit. For example, nodes 0110 and 0100 are immediate neighbors but 0110 and 0101 are not. An easy way to label nodes is to assign node IDs as the cube is constructed. When you copy an (http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2007.gif)-dimensional cube, make sure the corresponding nodes in the two copies have the same IDs. Then extend all the IDs by one bit. Append a 0 to the IDs of nodes in the original cube, and append a 1 to the IDs of nodes in the copy. As an example the nodes in the 1D and 2D cubes in Figure 11 are labeled according to this scheme; the labeling of the 3D and 4D cubes is left for an exercise.

Node IDs are the basis for a simple algorithm for routing information in a hypercube. An http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2153.gif-dimensional cube will have http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2153.gif-bit node IDs. Sending a message from node A to node B can be done in http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2153.gif cycles, where on each cycle a node will either hold a message or forward it along one of its links. On cycle http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2387.gif the node that currently holds the message will compare bit http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2387.gif of its own ID with bit http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2387.gif of the destination ID. If the bits match, the node holds the message. If they don't match, it forwards the message along dimensionhttp://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2387.gif, where dimension http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2387.gif is the dimension that was added in the http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2179.gif step of the construction of the cube (i.e. it is the same ``direction'' at all nodes). As an example, the path from node 2 to node 7 in a 4D cube is marked with a heavy gray line in **Figure** **11**.

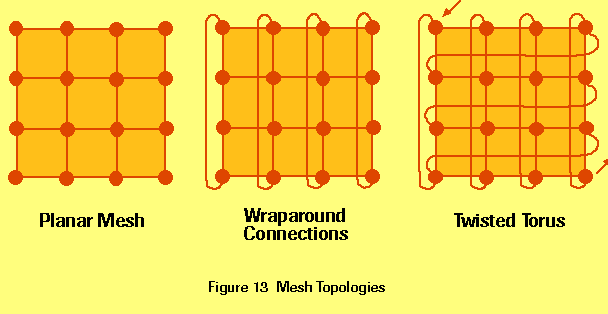
Another desirable property of interconnection networks is node symmetry. A node symmetric network has no distinguished node, that is, the ``view'' of the rest of the network is the same from any node. Rings, fully connected networks, and hypercubes are all node symmetric. Trees and stars, shown in Figure 12, are not. A tree has three different types of nodes, namely a root node, interior nodes, and leaf nodes, each with a different degree. A star has a distinguished node in the center which is connected to every other node. When a topology is node asymmetric a distinguished node can become a communications bottleneck.

A more formal definition of a communication bottleneck is based on a property known as the bisection width, which is the minimum



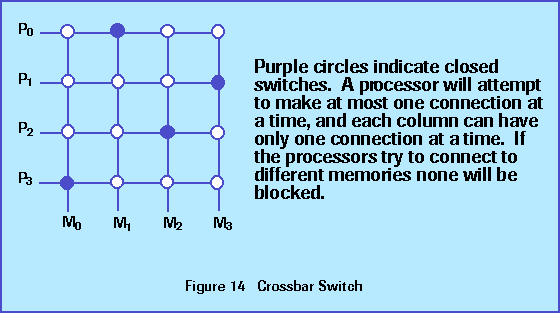
number of links that must be cut in order to divide the topology into two independent networks of the same size (plus or minus one node). The bisection width of a tree is 1, since if either link connected to the root is removed the tree is split into two subtrees. The bisection bandwidth of a parallel system is the communication bandwidth across the links that are cut in defining the bisection width. This bandwidth is useful in defining worst- case performance of algorithms on a particular network, since it is related to the cost of moving data from one side of the system to the other.

Another common topology is a planar (2D) mesh, shown in Figure 13. This network is basically a matrix



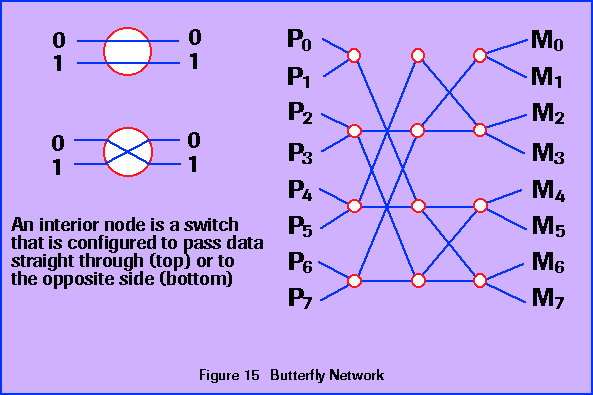
of nodes, each with connections to its nearest neighbors. Meshes usually have ``wraparound'' connections, e.g. the node at the top of the grid has an ``up'' link that connects to the node at the bottom of the grid. If you visualize only north-south links in a rectangular mesh, you can see these links turn the 2D mesh into a 3D cylinder. Now if the east-west links are added, it connects the ends of the cylinder to form a toroidal solid. Thus a mesh topology with wraparound connections is often referred to as a torus. In many systems the wraparound connections are skewed by one or more rows (or columns, or both); in this case the topology is known as a twisted torus. Note that a path that starts in the northwest corner of a twisted torus and heads continually east will visit every node exactly once before returning to the northwest corner.

The two final interconnection networks introduced in this section are examples of multistage networks. Systems built with these topologies have processors on one edge of the network, memories or processors on another edge, and a series of switching elements at the interior nodes. In order to send information from one edge to another, the interior switches are configured to form a path that connects nodes on the edges. The information then goes from the sending node, through one or more switches, and out to the receiving node. The size and number of interior nodes contributes to the path length for each communication, and there is often a ``setup time'' involved when a message arrives at an interior node and the switch decides how to configure itself in order to pass the message through.



The first example of a multistage network is the crossbar switch Figure 14. In a typical application there will be a column of processors on the left edge and a row of memories on the bottom edge. The switch configures itself dynamically to connect a processor to a memory module. As long as each processor wants to communicate with a different memory there will be no contention. If two or more processors need to access the same memory, however, one will be blocked until the switch reconfigures itself. A crossbar has a short diameter - information needs to pass through only one switching element on a path from one edge to another - but poor scalability. If there are http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2153.gif processors and a like number of memories there are http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2003.gif interior switches. Adding another processor and memory means adding another http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2185.gif interior nodes.

A banyan network is a multistage switching network that has the same number of inputs as outputs and interior nodes that are http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2187.gif switches. Examples of banyan networks are butterfly networks and omega networks, which are both built from http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2197.gif switches. The diameter of a butterfly is http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2191.gif, where http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2153.gif is the number of inputs and outputs, and there are http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2195.gif switches, so these networks scale more efficiently than a crossbar (Figure 15). The http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2197.gif switch in a butterfly can be configured in one of two states (Figure 15). One configuration connects input 0 to output 0 and input 1 to output 1. The other configuration flips the outputs, so input 0 connects to output 1 and input 1 connects to output 0. The switching network uses the binary representation of the destination address in order to construct a path from input to output. The switch at stage http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2387.gif in the network uses bit http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2387.gif to determine how to configure itself: if the bit is 0, the request should go through the top output, and if it is 1 it should go through the bottom output. For example, suppose a processor needs to fetch information from memory Mhttp://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2203.gif. The binary representation of 5 is 101. The first switch will pass the request out its bottom output, the second switch will pass the request out its top output, and the last switch will pass the request out its bottom output. Note that this pattern of connections (top-bottom-top) works no matter which processor generates the request. Whether a switch configures itself in the straight-through or flipped configuration depends on which input the request comes from. For example, if the request comes from the top input and should be routed out the top output, then the switch will go into the straight-through configuration, but if the request comes from the top input and should go out the bottom the switch will use the flipped configuration.



As is the case with the crossbar switch, there are configurations of the butterfly that will allow each processor to connect to a different memory so all processors can be active and no requests are blocked. However, the butterfly is not as flexible as the crossbar, since combinations of requests that are non blocking in the crossbar are blocking in the butterfly. For example, if the first switch in the first column is in the straight-through configuration because processor Phttp://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2227.gif is making a request to memory Mhttp://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2207.gif, processor Phttp://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2209.gif is constrained to communicate with memories 4 through 7 (http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2211.gif  through http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2213.gif). With a crossbar P1 would be allowed to connect to Mhttp://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2227.gif, Mhttp://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2209.gif, or Mhttp://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2219.gif without blocking.

Crossbar and butterfly switches have both been used to implement shared memory multiprocessors. Even though there are independent memory modules, there is a single memory space, i.e. an address http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2387.gif generated by one processor refers to the same cell as an address http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2387.gif generated by any other processor. Addresses are not interleaved, though. Instead the memory space is divided into contiguous blocks of equal size. For example, suppose there are 4 memory units and the address space has http://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2225.gif words. Mhttp://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2227.gif would hold addresses 0 to 255; Mhttp://www.phy.ornl.gov/csep/ca/_10165_tex2html_wrap2209.gif would have 256 to 511, and so on.

Three important attributes of an interconnection network are the timing strategy, control strategy, and switching strategy. The two alternatives for control are a single central controller or a distributed control system in which routing strategies are implemented in each node. Message routing based on node IDs in hypercubes and butterfly switches are examples of distributed control, since each node decides for itself how to reroute incoming messages. A centralized strategy would work well in a star network: messages from outer nodes must pass through the center, which would then decide how to forward the message. Synchronous control techniques are characterized by a global clock that broadcasts clock signals to all devices in a system so that the entire system operates in a lock-step fashion. Asynchronous techniques do not utilize a single global clock, but rather distribute the control function throughout the system, often utilizing many individual clocks for timing. Control and coordination of the various parts of the system are accomplished via some form of communication or ``hand shaking.'' Thus the interconnection network can operate synchronously off of a global clock or it may have distributed control down to the level of the individual switches. The advantage of a single global clock for control is simplicity in both the hardware and the software; the advantage of distributed control is expandability and flexibility. Synchronous and asynchronous timing strategies are a fundamental characteristic of computing systems in general. The SIMD systems discussed previously normally operate synchronously with a global clock while the MIMD systems function asynchronously with a clock in each PE.

Switching strategy is the other important characteristic of interconnection networks. The two most popular techniques are packet switching and circuit switching. In packet switching, a message is broken into small packets which are transmitted through the network in a ``store and forward'' mode. A packet traverses one link, where the receiving node will examine it and decide what to do. It may have to store the packet for a while before forwarding it toward its final destination, e.g. there may be other packets waiting to go out on that link. It is also possible that packets will traverse different sets of links on their route from source to destination. Packets may experience delays at each switching point depending on the traffic in the network. The circuit switching technique establishes a complete path between the source and the destination and then starts transferring information along the path. The circuit is kept open until the entire message has been transmitted. We will see examples of both strategies in the section on MIMD systems.