Chapter 2 Parallel Architectures and Interconnection Networks

> *"The interconnection network is the heart of parallel architecture."* - Chuan-Lin and Tse-Yun Feng [1]

2.1 Introduction

You cannot really design parallel algorithms or programs without an understanding of some of the key properties of various types of parallel architectures and the means by which components can be connected to each other. Parallelism has existed in computers since their inception, at various levels of design. For example, bit-parallel memory has been around since the early 1970s, and simultaneous I/O processing (using channels) has been used since the 1960s. Other forms of parallelism include bit-parallel arithmetic in the arithmetic-logic unit (ALU), instruction look-ahead in the control unit, direct memory access (DMA), data pipe-lining, and instruction pipe-lining. However, the parallelism that we will discuss is at a higher level; in particular we will look at processor arrays, multiprocessors, and multicomputers. We begin, however, by exploring the mathematical concept of a topology

2.2 Network Topologies

We will use the term ***network topology*** to refer to the way in which a set of nodes are connected to each other. In this context, a network topology is essentially a discrete graph – a set of nodes connected by edges. Distance does not exist and there is no notion of the length of an edge[1]. You can think of each edge as being of unit length.

Network topologies arise in the context of parallel architectures as well as in parallel algorithms. In the domain of parallel architectures, network topologies describe the interconnections among multiple processors and memory modules. You will see in subsequent chapters that a network topology can also describe the communication patterns among a set of parallel processes. Because they can be used in these two different ways, we will first examine them purely as mathematical entities, divorced from any particular application.

Formally, a network topology $< S, E >$ is a finite set S of ***nodes*** together with an adjacency relation $E \subseteq S \times S$ on the set. If v and w are nodes such that $(v, w) \in E$, we say that there is a ***directed edge*** from v to w.[2] Sometimes all of the edges in a particular topology will be undirected, meaning that both $(v, w) \in E$ and $(w, v) \in E$. *Unless we state otherwise, we will treat all edges as undirected.* When two nodes are connected by an edge, we say they are ***adjacent***. An example of a network topology that should be quite familiar to the reader is the binary tree shown in Figure 2.1. Notice that the nodes in that figure are ***labeled***. A label is an arbitrary symbol used to refer to the node, nothing more.

We will examine the following topologies in these notes:

- Binary tree
- Fully-connected (also called completely-connected)
- Mesh and torus

[1] Network topologies are a kind of mathematical topological space, for those familiar with this concept, but this is of no importance to us.

[2] The reader familiar with graphs will notice that a network topology is essentially a discrete graph.

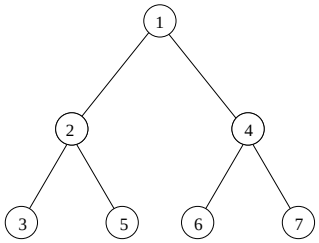

Figure 2.1: Binary tree topology with 7 nodes.

- Hypercube (also called a binary n-cube)
- Butterfly

2.2.1 Network Topology Properties

Network topologies have properties that determine their usefulness for particular applications, which we now define.

Definition 1. A ***path*** from node n_1 to node n_k is a sequence of nodes $n_1, n_2, \ldots, n_k$ such that, for $1 \leq i < k$, n_i is adjacent to n_{i+1}. The length of a path is the number of *edges* in the path, not the number of nodes.

Definition 2. The ***distance*** between a pair of nodes is the length of the *shortest* path between the nodes.

For example, in Figure 2.1, the distance between nodes 3 and 7 is 6, whereas the distance between nodes 3 and 5 is 2.

Definition 3. The ***diameter*** of a network topology is the largest distance between any pair of nodes in the network.

The diameter of the network in Figure 2.1 is 4, since the distance between nodes 3 and 7 is 4, and there is no pair of nodes whose distance is greater than 4. Diameter is important because, if nodes represent processors that must communicate via the edges, which represent communication links, then the diameter determines a lower bound on the communication time. (Note that it is a lower bound and not an upper bound; if a particular algorithm requires, for example, that all pairs of nodes send each other data before the next step of a computation, then the diameter determines how much time will elapse before that step can begin.)

Definition 4. The ***bisection width*** of a network topology is the smallest number of edges that must be deleted to sever the set of nodes into two sets of equal size, or size differing by at most one node.

In Figure 2.1, edge (1,2) can be deleted to split the set of nodes into two sets {2,3,5} and {1,4,6,7}. Therefore, the bisection width of this network is 1. Bisection width is important because it can determine the total communication time. Low bisection width is bad, and high is good. Consider the extreme case, in which a network can be split by removing one edge. This means that all data that flows from one half to the other must pass through this edge. This edge is a bottleneck through which all data must pass sequentially, like a one-lane bridge in the middle of a four-lane highway. In contrast, if the bisection width is high, then many edges must be removed to split the node set. This means that there are many paths from one side of the set to the other, and data can flow in a high degree of parallelism from any one half of the nodes to the other.

Definition 5. The ***degree*** of the network topology is the maximum number of edges that are incident to a node in the topology.

The maximum number of edges per node can affect how well the network scales as the number of processors increases, because of physical limitations on how the network is constructed. A binary tree, for example, has the property that the maximum number of edges per node is 3, regardless of how many nodes are in

the tree. This is good, because the physical design need not change to accommodate the increase in number of processors. Not all topologies have a constant degree. If the degree increases with network size, this generally means that more connections need to be made to each node. Nodes might represent switches, or processors, and in either case they have a fixed pin-out, implying that the connections between processors must be implemented by a complex fan-out of the wires, a very expensive and potentially slow mechanism.

Although the edges in a network topology do not have length, we assume that nodes cannot be infinitely small. As a consequence, the definition of the topology itself can imply that, as the number of nodes increases, the physical distance between them must increase. ***Maximum edge length*** is a measure of this property. It is important because the communication time is a function of how long the signals must travel. It is best if the network can be laid out in three-dimensional space so that the maximum edge length is a constant, independent of network size. If not, and the edge length increases with the number of processors, then communication time increases as the network grows. This implies that expanding the network to accommodate more processors can slow down communication time. The binary tree in Figure 2.1 does not have a constant maximum edge length, because as the size of the tree gets larger, the leaf nodes must be placed further apart, which in turn implies that eventually the edges that leave the root of the tree must get longer.

2.2.2 Binary Tree Network Topology

In a binary tree network, the $2^k - 1$ nodes are arranged in a complete binary tree of depth $k-1$, as in Figure 2.1. The ***depth*** of a binary tree is the length of a path from the root to a leaf node. Each interior node is connected to two children, and each node other than the root is connected to its parent. Thus the degree is 3. The diameter of a binary tree network with $2^k - 1$ nodes is $2(k-1)$, because the longest path in the tree is any path from a leaf node that must go up to the root of the tree and then down to a different leaf node. If we let $n = 2^k - 1$ then $2(k-1)$ is approximately $2\log_2 n$; i.e., the diameter of a binary tree network with n nodes is a logarithmic function of network size, which is very low.

The bisection width is low, which means it is poor. It is possible to split the tree into two sets differing by at most one node in size by deleting either edge incident to the root; the bisection width is 1. As discussed above, maximum edge length is an increasing function of the number of nodes.

2.2.3 Fully-Connected Network Topology

In a ***fully-connected network***, every node is connected to every other node, as in Figure 2.2. If there are n nodes, there will be $n(n-1)/2$ edges. Suppose n is even. Then there are $n/2$ even numbered nodes and $n/2$ odd numbered nodes. If we remove every edge that connects an even node to an odd node, then the even nodes will form a fully-connected network and so will the odd nodes, but the two sets will be disjoint. There are $(n/2)$ edges from each even node to every odd node, so there are $(n/2)^2$ edges that connect these two sets. Not removing any one of them fails to disconnect the two sets, so this is the minimum number. Therefore, the bisection width is $(n/2)^2$. The diameter is 1, since there is a direct link from any node to every other node. The degree is proportional to n, so this network does not scale well. Lastly, the maximum edge length will increase as the network grows, because nodes are not arbitrarily small. (Think of the nodes as lying on the surface of a sphere, and the edges as chords connecting them.)

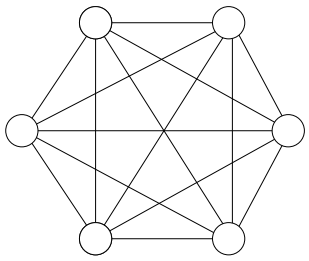

Figure 2.2: Fully-connected network with 6 nodes.

2.2.4 Mesh Network Topology

In a ***mesh network***, nodes are arranged in a q-dimensional lattice. A 2-dimensional lattice with 36 nodes is illustrated in Figure 2.3. The mesh in that figure is square. Unless stated otherwise, meshes are usually square. In general, there are k^2 nodes in a 2-dimensional mesh. A 3-dimensional mesh is the logical extension of a 2-dimensional one. It is not hard to imagine a 3-dimensional mesh. It consists of the lattice points in a 3-dimensional grid, with edges connecting adjacent points. A 3-dimensional mesh, assuming the same number of nodes in all dimensions, must have k^3 nodes. While we cannot visually depict q-dimensional mesh networks when q $>$ 3, we can describe their properties. A q-dimensional mesh network has k^q nodes. k is the number of nodes in a single dimension of the mesh. Henceforth we let q denote the dimension of the mesh.

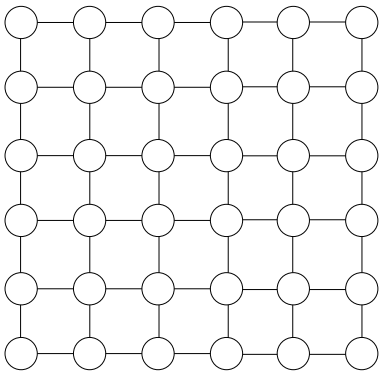

Figure 2.3: A two-dimensional square mesh with 36 nodes.

The diameter of a q-dimensional mesh network with k^q nodes is $q(k-1)$. To see this, note that the farthest distance between nodes is from one corner to the diagonally opposite one. An inductive argument is as follows. In a 2-dimensional lattice with k^2 nodes, you have to travel $(k-1)$ edges horizontally, and $(k-1)$ edges vertically to get to the opposite corner, in any order. Thus you must traverse $2(k-1)$ edges. Suppose we have a mesh of dimension $q-1$, $q > 3$. By assumption its diameter is $(q-1)(k-1)$. A mesh of one higher dimension has $(k-1)$ copies of the $(q-1)$-dimensional mesh, side by side. To get from one corner to the opposite one, you have to travel to the corner of the $(q-1)$-dimensional mesh. That requires crossing $(q-1)(k-1)$ edges, by hypothesis. Then we have to get to the k^{th} copy of the mesh in the new dimension. We have to cross $(k-1)$ more edges to do this. Thus we travel a total of $(q-1)(k-1)+(k-1) = q(k-1)$ edges. This is not rigorous, but this is the idea of the proof.

If k is an even number, the bisection width of a q-dimensional mesh network with k^q nodes is k^{q-1}. Consider the 2D mesh of Figure 2.3. To split it into two halves, you can delete $6 = 6^1$ edges. Imagine the 3D mesh with 216 nodes. To split it into two halves, you can delete the $36 = 6^2$ vertical edges connecting the 36 nodes in the third and fourth planes. In general, one can delete the edges that connect adjacent copies of the $(q-1)$-dimensional lattices in the middle of the q-dimensional lattice. There are k^{q-1} such edges. This is a very high bisection width. One can prove by an induction argument that the bisection width when k is odd is $(k^q-1)/(k-1)$. Thus, whether k is even or odd, the bisection width is $\Theta(k^{q-1})$. As there are $n = k^q$ nodes in the mesh, this is roughly $\sqrt[q]{n}$, which is a very high bisection width. (When $q = 2$, it is $\sqrt{n}$.)

The degree in a mesh is fixed for each given q: it is always 2^q. The maximum edge length is also a constant, independent of the mesh size, for two- and three-dimensional meshes. For higher dimensional meshes, it is not constant.

An extension of a mesh is a ***torus***. A torus, the 2-dimensional version of which is illustrated in Figure 2.4, is an extension of a mesh by the inclusion of edges between the exterior nodes in each row and those in each column. In higher dimensions, it includes edges between the exterior nodes in each dimension. It is called a torus because the surface that would be formed if it were wrapped around the nodes and edges with a thin film would be a mathematical torus, i.e., a doughnut. A torus, or toroidal mesh, has lower diameter than a non-toroidal mesh, by a factor of 2.

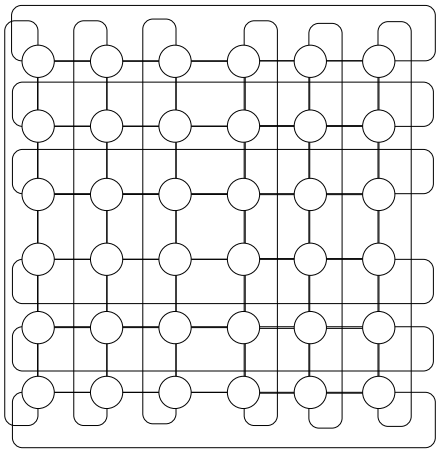

Figure 2.4: Two-dimensional mesh with toroidal connections.

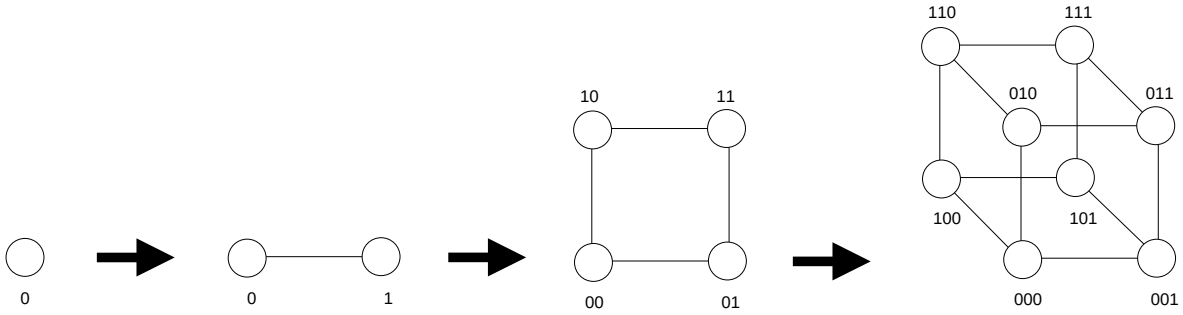

Figure 2.5: Hypercubes of dimensions 0, 1, 2, and 3.

2.2.5 Hypercube (Binary n-Cube)

A ***binary n-cube*** or ***hypercube*** network is a network with 2^n nodes arranged as the vertices of a n-dimensional cube. A hypercube is simply a generalization of an ordinary cube, the three-dimensional shape which you know. Although you probably think of a cube as a rectangular prism whose edges are all equal length, that is not the only way to think about it.

To start, a single point can be thought of as a 0-cube. Suppose its label is 0. Now suppose that we replicate this 0-cube, putting the copy at a distance of one unit away, and connecting the original and the copy by a line segment of length 1, as shown in Figure 2.5. We will give the duplicate node the label, 1.

We extend this idea one step further. We will replicate the 1-cube, putting the copy parallel to the original at a distance of one unit away in an orthogonal direction, and connect corresponding nodes in the copy to those in the original. We will use binary numbers to label the nodes, instead of decimal. The nodes in the copy will be labeled with the same labels as those of the original except for one change: the most significant bit in the original will be changed from 0 to 1 in the copy, as shown in Figure 2.5. Now we repeat this procedure to create a 3-cube: we replicate the 2-cube, putting the copy parallel to the original at a distance of 1 unit away in the orthogonal direction, connect nodes in the copy to the corresponding nodes in the original, and relabel all nodes by adding another significant bit, 0 in the original and 1 in the copy.

It is now not hard to see how we can create hypercubes of arbitrary dimension, though drawing them becomes a bit cumbersome. A 4-cube is illustrated in Figure 2.6 though.

The node labels will play an important role in our understanding of the hypercube. Observe that

- The labels of two nodes differ by exactly one bit change if and only if they are connected by an edge.
- In an n-dimensional hypercube, each node label is represented by n bits. Each of these bits can be

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.