Crossbar Networks

Crossbar networks allow any processor in the system to connect to any other processor or memory unit so that many processors can communicate simultaneously without contention. A new connection can be established at any time as long as the requested input and output ports are free. Crossbar networks are used in the design of high-performance small-scale multiprocessors, in the design of routers for direct networks, and as basic components in the design of large-scale indirect networks. A crossbar can be defined as a switching network with N inputs and M outputs, which allows up to min{N, M} one-to-one interconnections without contention. Figure 1.9 shows an N × M crossbar network. Usually, M = N except for crossbars connecting processors and memory modules.

Cube Interconnection Network

Cube-Connected Cubes (CCC). A CCC is hypercube with each node represented as a cube. We consider four CCC network structures using different criteria for global connections. A comparison is made of their network connectivity and average node distance. We also propose a generalized routing algorithm and a generalized broadcasting algorithm that can be applied to different CCC structures. The results of the network topology study show the CCC still keeps desirable hypercube properties such as the existence of parallel node-disjoint paths. Moreover, the CCC has a lower node degree than the hypercube.

Fat tree Interconnection Network

The IP address assignment in a fat-tree topology follows a certain pattern. We will illustrate this using IPv4 private address block 10.0.0.0/8 for the entire 4-pod fat-tree topology. The basic addressing scheme is 10.pod.switch.1 for switches in pods. For example, switch 1 marked in Figure 12.3, which is located in pod-0, is assigned the IP address 10.0.0.1 while switch 2 is assigned 10.0.1.1, switch 9 is assigned 10.0.2.1, switch 10 is assigned 10.0.3.1, and so on. Thus, pod numbers go from left to right , and the switches are numbered left to right, and after that, from the bottom (edge) to the top (aggregation) within each pod. The core switches are assigned IP addresses of the form 10.k.x.y, where x starts with 1 and in increments of 1 every core switches, while y starts from 1 in each block of core switches, left to right. Thus, for the 4-pod fat-tree topology, the core switches are numbered 10.4.1.1, 10.4.1.2, 10.4.2.1, and 10.4.2.2, from left to right.

The end host numbering is of the form 10.pod.switch.ID. Here, the ID is numbered from left to

right starting with 2 as a child of the edge switch. Thus, in pod-0, the two hosts off of switch 1 with IP address 10.0.0.1 are numbered 10.0.0.2 and 10.0.0.3, from left to right. Therefore, we can think of switch 1 with its two child hosts to be in the subnet 10.0.0.0/24. Similarly, for switch 2 with IP address 10.0.1.1, hosts are numbered 10.0.1.2, 10.0.1.3, from left to right; i.e., they are in address block 10.0.1.0/24. Next, the address block is defined based on the location of each pod. For example, pod-0 has the address block 10.0.0.0/16. The pod-0 of the 4-pod fat-tree topology