CHAPTER 7

Network Computing

The proliferation of the Internet has stimulated rapid growth of interest in large-scale network computing that may span the entire globe. The Internet is the most widely used distributed system in the world. Nodes in the Internet may be single-processor workstations, shared-memory MIMD machines, massively parallel SIMD machines, or other types. Links are TCP/IP packet-switched connections and the bandwidth varies with load, number of hops, and underlying communication technology. When one node connects with another, the packets of data may be sent through a wireless link, fiber optical cable, coaxial cable, digital telephone line, and so on. These physical layers introduce delays and may be errors, which must be corrected by retransmission and dynamic reconfiguration of the Internet's links.

In previous chapters, we studied computing systems consisting of multiple processing units connected via some interconnection network. There are two major factors that differentiate such systems: the processing units and the interconnection network that ties them together. We learned that the processing units could communicate and interact with each other using either shared memory or message passing methods. In this chapter we discuss network computing, in which the nodes are stand-alone computers that could be connected via a switch, local area network, or the Internet. The main idea is to divide the application into semi-independent parts according to the kind of processing needed. Different nodes on the network can be assigned different parts of the application. This form of network computing takes advantage of the unique capabilities of diverse system architectures. For example, the fine-grained SIMD part of the application would be shipped off to the SIMD machine and the graphical presentation and I/O portions to one or more single-processor workstations. The overall coordination may be done by a PC on someone's desk. It also maximally leverages potentially idle resources within a large organization. Therefore, unused CPU cycles may be utilized during short periods of time resulting in bursts of activity followed by periods of inactivity. In what follows, we discuss the utilization of network technology in order to create a computing infrastructure using commodity computers.

Advanced Computer Architecture and Parallel Processing, by H. El-Rewini and M. Abd-El-Barr
ISBN 0-471-46740-5

7.1 COMPUTER NETWORKS BASICS

Networks can be divided into the following four categories based on their sizes and the geographic distances they cover:

- Wide area network (WAN);
- Metropolitan area network (MAN);
- Local area network (LAN);
- System or storage area network (SAN).

A WAN connects a large number of computers that are spread over large geographic distances. It can span sites in multiple cities, countries, and continents. A LAN connects a small number of computers in a small area within a building or campus. The MAN is an intermediate level between the LAN and WAN and can perhaps span a single city. A SAN connects computers or storage devices to make a single system. The major factor that distinguishes WAN from other network types is the scalability factor. A WAN must be able to grow as long as there are more computers to be added to the network. A message sent over WAN uses intermediate nodes in its route from the source to the destination. Computers hooked to a LAN often communicate using a shared medium. Also, LAN technologies provide higher speed connections compared to WAN because they cover short distances and hence offer lower delay than WANs.

Network routing schemes can be classified as connection-oriented and connectionless. In connection-oriented, the entire message follows the same path from source to destination. Only the first packet holds routing information such as the destination address. In connectionless schemes, a message is divided into packets. The packets of a given message may take different routes from source to destination. Therefore, the header of every packet holds routing information. Using a serial number, the message can be reassembled in the correct order at the destination as packets may arrive in a different order.

7.1.1 Network Performance

The following are two popular laws that predict the advances in network technologies.

Gilder's Law George Gilder projected that the total bandwidth of communication systems triples every 12 months.

Metcalfe's Law Robert Metcalfe projected that the value of a network is proportional to the square of the number of nodes.

Gilder's law tells us that networking speed is increasing faster than processing power. While this remains true for the backbone network, end-to-end performance is likely to be limited by bottlenecks. For example, over about 15 years, LAN technology has increased in speed from 10 Megabits per second (10 Mbps) to 10 Giga-

bits per second (10 Gbps), which is a factor of 1000 increase. Over a similar time period, advances in silicon technology, driven by Moore's Law, have allowed the CPU clock frequency in an average PC to increase from roughly 25 MHz to 2.5 GHz (a factor of about 100 increase in processing power). Metcalfe's law also explains the prolific growth of the Internet. As a network grows, the value of being connected to it grows exponentially, while the cost per user remains the same or even reduces.

7.1.2 Internet

Internet is the collection of networks and routers that form a single cooperative virtual network, which spans the entire globe. The Internet relies on the combination of the Transmission Control Protocol and the Internet Protocol or TCP/IP. The majority of Internet traffic is carried using TCP/IP packets. Internet has evolved from a research prototype to become the largest communication media in the world. The explosive usage of the Internet and the World Wide Web (WWW) has stimulated rapid growth of interest in electronic publishing, browsing, and distributed computing. Table 7.1

TABLE 7.1 Top Ten Countries with Highest Number of Internet Users

Country	Internet Users Latest Data	Population (2004 Est.)	% of Population	Source of Latest Data	% of World Usage/Users
United States	209,518,183	294,540,100	71.1	Nielsen//NR Mar/04	27.7
China	79,500,000	1,327,976,227	6.0	CNNIC Dec/03	10.5
Japan	63,884,205	127,944,200	49.9	Nielsen//NR Mar/04	8.4
Germany	45,315,166	82,633,200	54.8	Nielsen//NR Mar/04	6.0
United Kingdom	35,089,470	59,157,400	59.3	Nielsen//NR Mar/04	4.6
South Korea	29,220,000	47,135,500	62.0	KRNIC Dec/03	3.9
France	22,534,967	59,494,800	37.9	Nielsen//NR Mar/04	3.0
Brazil	20,551,168	183,199,600	11.2	Nielsen//NR Mar/04	2.7
Italy	19,900,000	56,153,700	35.4	ITU Dec/02	2.6
Canada	16,841,811	32,026,600	52.6	Nielsen//NR May/02	2.2
Top ten countries	542,354,970	2,270,261,327	23.9	IWS–Apr.6/04	71.6
Rest of the world	215,175,767	4,183,049,740	5.1	IWS–Mar.19/04	28.4
Totals	757,530,737	6,453,311,067	11.7	IWS–Apr.6/04	100.0

shows some statistics by Internet World Stats (www.internetworldstats.com) about the top 10 countries with the highest number of Internet users. As you can see from the table, in the United States alone, 71% of the population use the Internet. Also, close to 12% of the entire world population use the Internet. With the projections of Gilder and Metcalfe, the number of users is expected to grow even more.

7.1.3 Other Network Technologies

In addition to the popular TCP/IP protocol, many more protocols and combinations of protocols exist. Some of these protocols are briefly mentioned below. Figure 7.1 shows different network technologies and their speed in relation to the network taxonomy provided above.

Fast Ethernet and Gigabit Ethernet Fast Ethernet (100Base-T) is a high-speed LAN that allows a computer to transmit or receive data at 100 Megabits per second (100 Mbps). The demand for a bandwidth that is even higher than 100 Mbps has motivated the extension of Ethernet to a bit rate of 1 Gbps. Gigabit Ethernet (1000Base-T) has become an attractive choice for corporate backbone networks and high-performance clusters of workstations.

The Fiber Distributed Data Interface (FDDI) The FDDI specifies a 100 Mbps token-passing, dual-ring LAN using fiber-optic cable. The FDDI is frequently used as high-speed backbone technology because of its support for high bandwidth and greater distances than copper.

High-Performance Parallel Interface (HiPPI) The HiPPI is a point-to-point communication channel and it does not support multidrop configurations. HiPPI

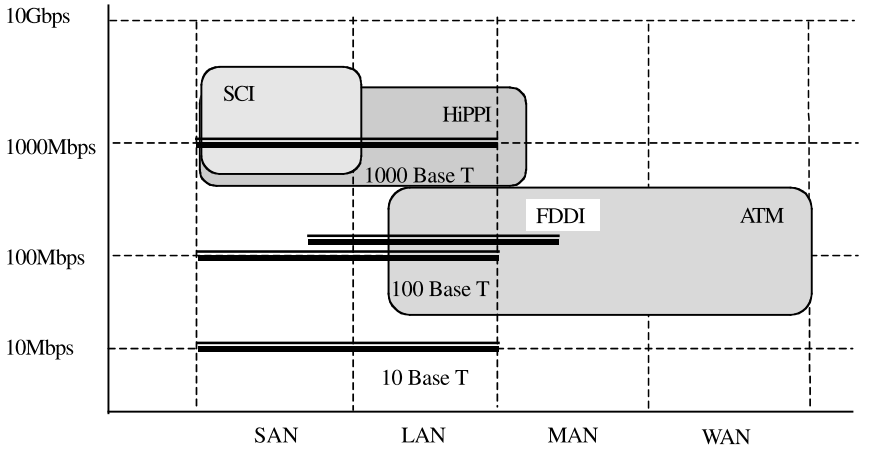

Figure 7.1 Representation of network technologies.

is capable of transferring data at 800 Mbps using 32 parallel line or 1.6 Gbps over 64 parallel lines.

Asynchronous Transfer Mode (ATM) The ATM is a connection-oriented scheme that is suitable for both LANs and WANs. It transfers data in small fixed-size packets called cells. It can handle multimedia in an integrated way. Cells are allowed to transfer using several different media such as both copper and fiber-optic cables. It is designed to permit high-speed data. The fastest ATM hardware can switch data at a gigabit rate.

Scalable Coherent Interface (SCI) The SCI is an IEEE standard that is quite popular for PC clusters. It represents a point-to-point architecture with directory-based cache coherence. It provides a cluster-wide shared memory system. A remote communication in SCI takes place as just part of a simple load or store process in a processor.

7.2 CLIENT/SERVER SYSTEMS

A Client/Server is a distributed system whereby the application is divided into at least two parts: one or more servers perform one part and the other part is performed by one or more clients. Furthermore, the clients are connected to the servers by some kind of network. A client computer may do very little more than simply display data accessed from the server, or a more sophisticated client may run a full application, which uses data provided by the server. Client/Server systems are often categorized as two-tier or three-tier. A two-tier system separates clients from servers: all clients are on one tier, and all servers are on the second tier. For example, client PCs may access a database on one or more servers. A three-tier system separates the clients from the servers as does a two-tier system, but in addition, servers are divided into two more tiers. The application servers fit into a middle level, called the second tier, and the database servers fit into a third level called tier 3. For example, client PCs might be connected to a web server (tier 2) that in turn accesses a database server (tier 3) to handle storage.

Modern programming languages provide constructs for building client/server-based distributed applications. These applications are divided into clients and servers, which are allocated to different computers in a network. A client sends a request to the server and waits for a response. At the other end, when the server receives a request, it processes it and sends the results back to the client. In a traditional client/server environment, a powerful machine acts as a server, serving requests from multiple clients. For example, in a database system, several clients send queries to the server that has access to the database. The server executes the queries on behalf of the clients and sends each client its respective result. A multi-threaded process is considered an efficient way to provide server applications. A server process can service a number of clients as shown in Figure 7.2. Each client request triggers the creation of a new thread in the server.

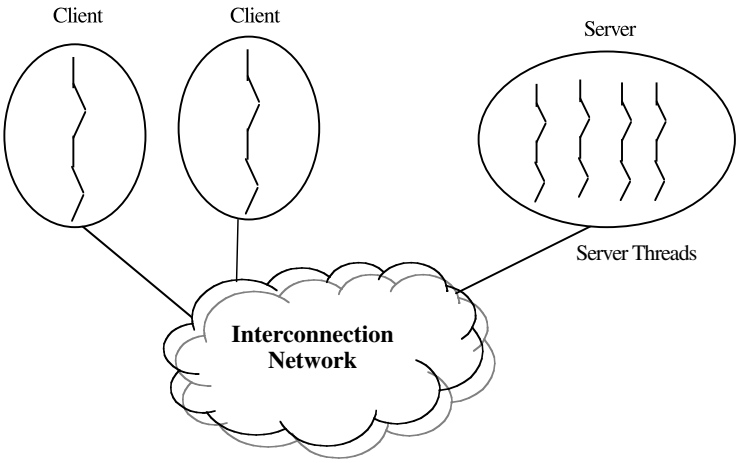

Figure 7.2 A multithreaded server in a client server system.

7.2.1 Sockets

Sockets are used to provide the capability of making connections from one application running on one machine to another running on a different machine. A socket abstraction consists of the data structure that holds the information needed for communication, and the system calls that manipulate the socket structure. Once a socket is created, it can be used to wait for an incoming connection (passive socket), or can be used to initiate connection (active socket).

A client can establish an active connection to a remote server by creating an instance of a socket. To establish a server connection and bind it to a particular port number, we should create an instance of a server socket. A server socket listens on a TCP port for a connection from a client (passive socket). When a client connects to that port, the server accepts the connection. Figure 7.3 visualizes the socket connection. Once the connection is established, the client and server can read from and write to the socket using input and output streams. Streams are ordered sequences of data that have a source (input stream), or destination (output stream). Once the client or server finishes using the socket, the socket structure is de-allocated.

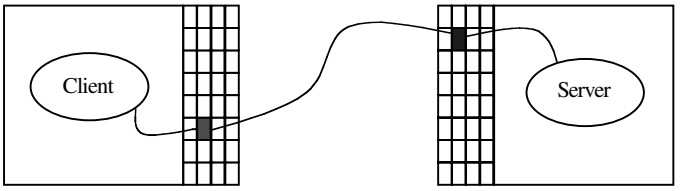

Figure 7.3 A socket connection.

Example 1 In this example, we write a small segment of a simple Client/Server Java program, in which the client reads from the server. This simple example does not require prior knowledge of Java as we will explain the main Java constructs needed in this example.

Java programs, like programs written in other object-oriented languages, are built from classes. You can create any number of objects from a class. These objects are known as instances of that class. Think of a class as a template or blueprint for the creation of objects. When a house is constructed from a blueprint, it is one of many possible instances of the same blueprint. Thus, an object is the actual thing, but a class is only a blueprint. A class contains functions called methods or behavior and states called fields or instance variables. Fields are data belonging to either a class or the objects that belong to a class. Methods are program statements that operate on the field to manipulate the state.

CLIENT/SERVER JAVA STATEMENT

The `Socket` class provides a client-side socket interface. A client can establish an active connection to a remote server by creating an instance of `Socket` as follows:

```
Socket <connection> = new Socket(<hostname>,<portnumber>);
```

The variable `<hostname>` gives the name of the server to connect to; and `<portnumber>` should match the port number at the server end.

The `ServerSocket` class provides a server-side socket interface. To establish a server connection and bind it to a particular port number, we should create an instance of `ServerSocket` as follows:

```
ServerSocket <connection> = new ServerSocket(<portnumber>);
```

Once the connection is established, the client and server can read from and write to the socket using input and output streams. Streams in Java are used for Input/Output. They are ordered sequences of data that have a source (input stream), or destination (output stream). For example, the `DataInputStream` and `DataOutputStream` are classes that provide the implementations for methods to transmit Java primitive types across a stream. Input and output streams can be used to read from and write to a socket connection as follows:

```
DataInputStream in = new DataInputStream(<connection>.
  getInputStream());
DataOutputStream out = new
DataOutputStream(<connection>.getOutputStream());
```

CLIENT STEPS

1. The client tries to establish a connection with the server.
2. When the connection is established, the client receives a string from the server.
3. The client displays the string received.

JAVA MAIN STATEMENTS

```
.....
Socket connect = new Socket (host, 8888);
.....
DataInputStream in = new DataInputStream(connect.getInput
  Stream());
.....
msg = in.readByte();
connect.close();
```

SERVER STEPS

1. The server waits for a connection from a client.
2. When the connection is established, the server sends a string to the client.
3. The server closes the connection.

JAVA MAIN STATEMENTS

```
.....
Socket connect = null;
.....
ServerSocket sconnect = new ServerSocket(8888);
connect = sconnect.accept();
DataOutputStream out = new DataOutputStream(connect.get
  OutputStream());
out.writeByte(message);
sconnect.close();
.....
```

7.2.2 Remote Procedure Call (RPC)

Remote procedure call (RPC) is the basis of most client/server systems. Think of RPC as a procedure call where the procedure is located on a different computer than the caller. Thus, when the procedure is called, its parameters are passed (sent) via the network to the remote computer, and then the remote computer executes the procedure, returns the result(s), and continues on its way.

The RPC can be constructed on top of sockets. That is, the socket mechanism can be used to pass parameters and the name of the procedure to be activated on the remote computer, and so on. The remote procedure call mechanism is simple to use because it looks much like any other procedure call familiar to programmers. However, it covers up many complexities.

The RPC can be blocking or nonblocking. A *blocking RPC* means that the program that places the call is stopped in its tracks while waiting for a reply. The nonblocking RPC, however, allows the calling program to continue without waiting for a reply. In this case, the caller must explicitly ask for the reply at some later time, or else the return value will never get back to the caller.

7.2.3 Middleware

Middleware is an important part of client/server systems because it solves many interoperability problems, opens the door for multiple servers, and in general provides great flexibility. Middleware is an important intermediate layer of software, for the following reasons:

- It makes it possible for new systems to coexist with legacy systems, which means we can use it to glue together new clients with old mainframe databases;
- It solves a number of interoperability problems because it can simultaneously convert formats and gain access without code rewriting;
- It isolates system components so that changes in one component have little effect on other components; and
- It lowers effort and time to develop and deploy systems because programmers do not need to know network and distributed programming details.

7.2.4 A Client Server Framework for Parallel Applications

Parallel applications can be designed using the client/server model. A client may divide a big application into several smaller problems that can be processed by multiple servers simultaneously. All the servers compute the solution to their respective problems and send their results to the client. The client assembles the results from each server and outputs the final result to the user. The client acts as the master (supervisor) while the servers act as the slaves (workers) in the master–slave (supervisor–workers) model as shown in Figure 7.4. The steps taken at the client and each server are summarized as follows.

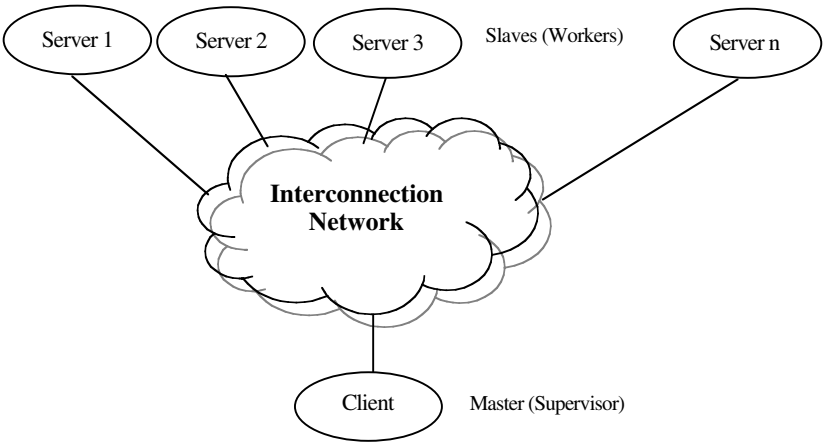

Figure 7.4 Supervisor workers model in client server.

Client (Supervisor)

1. Client creates an array of sockets and input/output data streams with all the servers. Optimally, the client should spawn a thread for each available server, which would then make connections with an individual server.
2. Client passes control to the client body, which contains the code specific to the application being executed in parallel. Mainly, it divides the main task into smaller portions and passes one small portion of the task to each server. It then waits for all the servers to send back the result of their smaller computations. Finally it merges the results of each server and computes the final solution to the big problem.
3. Client closes all the streams and sockets with all the servers.

Server (Worker)

1. Server creates a server socket on an unused port number.
2. Server waits for connections on that port. Once it gets a request from a client, it accepts that connection.
3. Server creates input and output data streams for that socket. This establishes the foundation for communication between the server socket and the client.
4. Server passes control to the server body, which contains the code specific to the application executed in parallel. The main server would accept the connection from the client, create a socket, and invoke the server body thread to handle that client.
5. Server goes back and waits for another connection from a client.

7.3 CLUSTERS

The 1990s have witnessed a significant shift from expensive and specialized parallel machines to the more cost-effective clusters of PCs and workstations. Advances in network technology and the availability of low-cost and high-performance commodity workstations have driven this shift. Clusters provide an economical way of achieving high performance. Departments that could not afford the expensive proprietary supercomputers have found an affordable alternative in clusters.

A cluster is a collection of stand-alone computers connected using some interconnection network. Each node in a cluster could be a workstation, personal computer, or even a multiprocessor system. A node is an autonomous computer that may be engaged in its own private activities while at the same time cooperating with other units in the context of some computational task. Each node has its own input/output systems and its own operating system. When all nodes in a cluster have the same architecture and run the same operating system, the cluster is called *homogeneous*, otherwise, it is *heterogeneous*. The interconnection network could be a fast LAN or a switch. To achieve high-performance computing, the interconnection network must provide high-bandwidth and low-latency communication. The nodes of a cluster may be dedicated to the cluster all the time; hence

computation can be performed on the entire cluster. Dedicated clusters are normally packaged compactly in a single room. With the exception of the front-end node, all nodes are headless with no keyboard, mouse, or monitor. Dedicated clusters usually use high-speed networks such as fast Ethernet and Myrinet. Alternatively, nodes owned by different individuals on the Internet could participate in a cluster only part of the time. In this case, the cluster can utilize the idle CPU cycles of each participating node if the owner's permission is granted.

Figure 7.5 shows the architecture of a homogeneous cluster made of similar nodes, where each node is a single-processor workstation. The middleware layer in the architecture makes the cluster appears to the user as a single parallel machine, which is referred to as the single system image (SSI). The SSI infrastructure offers unified access to system resources by supporting a number of features including:

- Single entry point: A user can connect to the cluster instead of to a particular node.
- Single file system: A user sees a single hierarchy of directories and files.
- Single image for administration: The whole cluster is administered from a single window.
- Coordinated resource management: A job can transparently compete for the resources in the entire cluster.

In addition to providing high-performance computing, clusters can also be used to provide high-availability environment. High availability can be achieved when

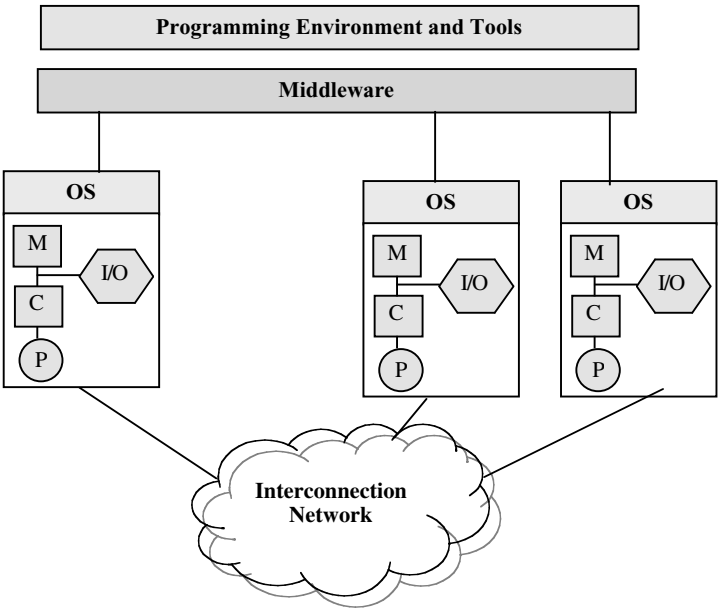

Figure 7.5 A cluster made of homogenous single-processor computers.

only a subset of the nodes is used in the computation and the rest is used as a backup in case of failure. In cases when one of the main objectives of the cluster is high availability, the middleware will also support features that enable the cluster services for recovery from failure and fault tolerance among all nodes of the cluster. For example, the middleware should offer the necessary infrastructure for checkpointing. A checkpointing scheme makes sure that the process state is saved periodically. In the case of node failure, processes on the failed node can be restarted on another working node.

The programming environment and tools layer provide the programmer with portable tools and libraries for the development of parallel applications. Examples of such tools and libraries are Threads, Parallel Virtual Machine (PVM), and Message Passing Interface (MPI). Note that PVM and MPI will be covered in detail in Chapters 8 and 9, respectively.

7.3.1 Threads

An important aspect of modern operating systems is their support for threads within processes. A thread, sometimes called a lightweight process, is a basic unit of processor utilization. It runs sequentially on a processor and is interruptible so that the processor can switch to other threads. A process does nothing if it has no threads in it, and a thread must be in exactly one process. A thread is different from the traditional or heavy-weight process, which is equivalent to a task with one thread. Context switching among peer threads is relatively inexpensive, compared with context switching among heavy-weight processes.

Concurrency among processes can be exploited because threads in different processes may execute concurrently. Moreover, multiple threads within the same process can be assigned to different processors. Threads can be used to send and receive messages while other operations within a task continue. For example, a task might have many threads waiting to receive and process request messages. When a message is received by one thread, it is processed by this thread concurrently with other threads processing other messages. Java has support for multithreading built in the language. Java threads are usually mapped to real operating system threads if the underlying operating system supports multithreads. Thus, applications written in Java can be executed in parallel on a multiprocessor environment.

Example 2: Creating Threads in Java Here is a simple example of how Java creates multiple threads. This program is the multithreaded version of the program *HelloWorld*, which is normally the first program in most programming languages books. Two threads are created, each of which prints a message of the form "Hello World from thread *i*", where *i* is in {1, 2}, a number of times (three times in this example). Again, this simple example does not require prior knowledge of Java. We will explain the main Java constructs needed in this example.

The class `Thread` is one of the classes in the standard Java libraries. This class supports a number of methods to handle threads. For example, the method `start` spawns a new thread of control, which can be stopped or suspended by invoking the

methods stop or suspend. The class Thread also supports the method run, which is invoked by start to make the thread active. The standard implementation of Thread.run does nothing, but the class Thread can be extended to provide a new run method. The Java code is shown below:

```
class HelloWorld extends Thread {
  int myId;          // a thread id
  int count;         // how many times
  int sleepTime;     // how long to pause
  HelloWorld (int id, int cnt, int slp){
    myId = id;
    count = cnt;
    sleepTime = slp;
  }
  public void run() {
    while (count--> 0) {
    System.out.println('Hello World from thread' + id);
    try {
      Thread.sleep(sleepTime);
    catch (Exception e) {
      return;
    }
  }
  public static void main (String[] args) {
    new HelloWorld(1,3,100).start();
    new HelloWorld(2,3,300).start();
  }
}
```

This example illustrates a number of features in Java. The program first creates a class called HelloWorld, which extends the class Thread. The input for this class is an identifier for the thread, how many times the message will be printed, and a time interval to sleep between the times the message is printed. Its run method loops the required number of times, printing its message, and waiting the specified amount of sleep time.

Finally, the main function creates two HelloWorld objects, each with its own identifier and sleep time, and invokes each object's start method. A possible result (remember that the order of the messages is unpredictable) when you run this program is:

```
Hello World from thread 1
Hello World from thread 2
Hello World from thread 1
Hello World from thread 1
Hello World from thread 2
Hello World from thread 2
```

7.4 INTERCONNECTION NETWORKS

The overall performance of a cluster system can be determined by the speed of its processors and the interconnection network. Many researchers argue that the interconnection network is the most important factor that affects cluster performance. Regardless of how fast the processors are, communication among processors, and hence scalability of applications, will always be bounded by the network bandwidth and latency. The bandwidth is an indication of how fast a data transfer may occur from a sender to a receiver. Latency is the time needed to send a minimal size message from a sender to a receiver. In the early days of clusters, Ethernet was the main interconnection network used to connect nodes. Many solutions have been introduced to achieve high-speed networks. Key solutions in high-speed interconnects include Gigabit Ethernet, Myrinet, and Quadrics. While Ethernet resides at the low end of the performance spectrum, it is considered a low-cost solution. Other solutions add communication processors on the network interface cards, which provide programmability and performance. Table 7.2 shows the relative performance and other features of different high-speed networks.

In this section, we will cover the evolution of Ethernet and discuss a sample of switches used in connecting computers to form a cluster. The most important distinguishing factor among the different switches will be the bandwidth and the latency time.

7.4.1 Ethernet

Ethernet is a packet-switched LAN technology introduced by Xerox PARC in the early 1970s. Ethernet was designed to be a shared bus technology where multiple hosts are connected to a shared communication medium. All hosts connected to an Ethernet receive every transmission, making it possible to broadcast a packet to all hosts at the same time. Ethernet uses a distributed access control scheme called *Carrier Sense Multiple Access* with *Collision Detect* (CSMA/CD). Multiple machines can access an Ethernet at the same time. Each machine senses whether a carrier wave is present to determine whether the network is idle before it sends a packet. Only when the network is not busy sending another message can transmission start. Each transmission is limited in duration and there is a minimum

TABLE 7.2 Data Rate, Switching Method, and Routing Scheme for Interconnection Networks

Interconnection Network	Data Rate	Switching	Routing
Ethernet	10 Mbit/s	Packet	Table-based
Fast Ethernet	100 Mbit/s	Packet	Table-based
Gigabit Ethernet	1 Gbit/s	Packet	Table-based
Myrinet	1.28 Gbit/s	Wormhole	Source-path
Quadrics	7.2 Gbyte/s	Wormhole	Source-path

idle time between two consecutive transmissions by the same sender. In the early days of LAN technology, an Ethernet speed of 10 million bits per second (10 Mbps) (*10Base-T*) was quite sufficient. However, with the dramatic increase in CPU speed and advances in network technology, a 10 Mbps Ethernet became an obvious bottleneck. Fast Ethernet, which uses twisted-pair wiring, was later introduced with speed of 100 Mbps (*100Base-T*). Dual-speed Ethernet (10/100 Ethernet) was also introduced to accommodate both 10 or 100 Mbps connections. By the late 1990, demand has increased for even higher speed Ethernet. Therefore Gigabit Ethernet was introduced. Gigabit Ethernet (*1000Base-T*) extended the Ethernet technology to a bit rate of 1 gigabit per second (1 Gbps).

Each computer connected to an Ethernet network is assigned a unique 48-bit address known as its Ethernet address. Ethernet manufacturers assign unique Ethernet addresses as they produce hardware interfaces. The Ethernet address, which is also called a media access (MAC) address, is fixed in a machine-readable form on the host interface hardware. The address can specify the physical address of one network interface, the network broadcast address, or a multicast. In addition to its MAC address, an interface must recognize a broadcast address (all 1s) and the group addresses in the case of multicast. The Ethernet interface hardware is usually given the set of addresses to recognize by the operating system during boot time. The host interface hardware, which receives a copy of every packet that passes by, will use the destination address to determine the packets that should be passed to the host. Other packets addressed to other hosts will be ignored.

In order to achieve an acceptable level of performance and to eliminate any potential bottleneck, there must be some balance between the Ethernet speed and the processor speed. The initial Beowulf prototype cluster in 1994 was built with DX4 processors and 10 Mbit/s Ethernet. The processors were too fast for this kind of Ethernet. In late 1997, a good choice for a cluster system was sixteen 200 MHz P6 processors connected by Fast Ethernet. The network configuration of a high-performance cluster is dependent on the size of the cluster, the relationship between processor speed and network bandwidth and the current price list for each of the components.

7.4.2 Switches

An $n_1 \times n_2$ switch consists of n_1 input ports, n_2 output ports, links connecting each input to every output, control logic to select a specific connection, and internal buffers. Although n_1 and n_2 do not have to be equal, in practice and in most cases they have the same value, which is usually power of two. A switch is used to establish connections from the input ports to the output ports. These connections may be one-to-one, which represent point-to-point connections, or one-to-many, which represent multicast or broadcast. The case of many-to-one should cause conflicts at the output ports and therefore needs arbitration to resolve conflicts if allowed. When only one-to-one connections are allowed, the switch is called crossbar. An $n \times n$ crossbar switch can establish $n!$ connections. The proof is quite simple. To allow

only one-to-one connections, the first input port should have n choices of output ports, the second input port will have $(n-1)$ choices, the third input port will have $(n-2)$ choices, and so on. Thus, the number of one-to-one connections is $n*(n-1)*(n-2)\cdots*2*1=n!$ If we allow both one-to-one as well as one-to-many in an $n \times n$ switch, the number of connections that can be established is n^n (see Problem 3). For example, a binary switch has two input ports and two output ports. The number of one-to-one connections in a binary switch is two (straight and crossed), while the number of all allowed connections is four (straight, crosses, lower broadcast, and upper broadcast).

Routing can be achieved using two mechanisms: source-path and table-based. In source-path, the entire path to the destination is stored in the packet header at the source location. When a packet enters the switch, the outgoing port is determined from the header. Used routing data is stripped from the header and routing information for the next switch is now in the front. In table-based routing, the switch must have a complete routing table that determines the corresponding port for each destination. When a packet enters the switch, a table lookup will determine the outgoing port. Figure 7.6 illustrates the difference between source-path routing and table-based routing in the case when a packet enters an 8-port switch at port 0. In the source-path case, the header contains the entire path and the next port is port 6. In the table-based case, the destination address dest-id is looked up in the routing table and port 6 is followed.

7.4.3 Myrinet Clos Network

Myrinet is a high-performance, packet-communication and switching technology. It was produced by Myricom as a high-performance alternative to conventional Ethernet networks. Myrinet switches are multiple-port components that route a packet entering on an input channel of a port to the output channel of the port selected by the packet. Myrinet switches have 4, 8, 12, 16 ports. For an n-port switch, the ports are addressed $0, 1, 2, \ldots, n-1$. For any switching permutation, there may be as many packets traversing a switch concurrently as the switch has ports. These switches are implemented using two types of VLSI chips: crossbar-switch chips and the Myrinet-interface chip.

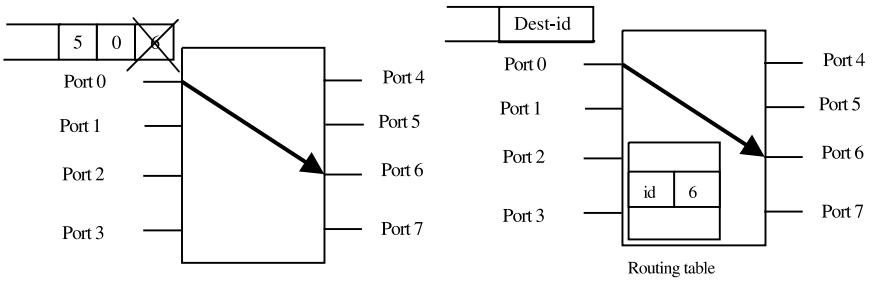

Figure 7.6 Source-path routing vs. table-based routing.

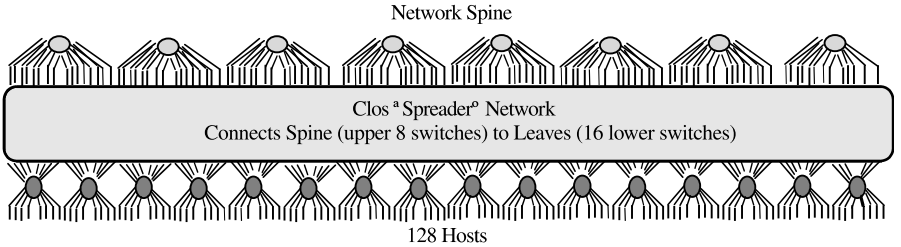

Figure 7.7 A 128-host Clos network using 16-port Myrinet switch.

The basic building block of the Myrinet-2000 network is a 16-port Myrinet crossbar switch, implemented on a single chip designated as Xbar16. It can be interconnected to build various topologies of varying sizes. The most common topology is the Clos network. Figure 7.7 shows a 128-host Clos network, which includes 24 Xbar16s. Each Xbar16 switch is represented using a circle. The eight switches forming the upper row is the Clos network *spine*, which is connected through a Clos spreader network to the 16 *leaf* switches forming the lower row. The Clos network provides routes from any host to any other host. There is a unique shortest route between hosts connected to the same Xbar16. Routes between hosts connected to different Xbar16s traverse three Xbar16 switches.

Figure 7.8 shows a 64-host Clos network. Please note that a network of 64 hosts or fewer would require eight-port switches for the spine. In the figure, an Xbar16 switch can serve the purpose of two 8-port switches. The thick line connecting a spine switch to a leaf switch represents two links. Similarly, Figure 7.9 shows a 32-host Clos network. Each thick line connecting a spine switch to a leaf switch represents four links.

The routing of Myrinet packets is based on the source routing approach. Each Myrinet packet has a variable length header with complete routing information. When a packet enters a switch, the leading byte of the header determines the

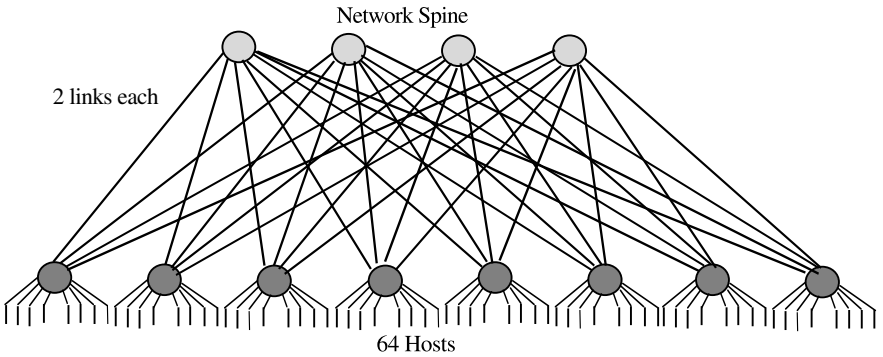

Figure 7.8 A 64-host Clos network using 16-port Myrinet switch (each line represents two links).

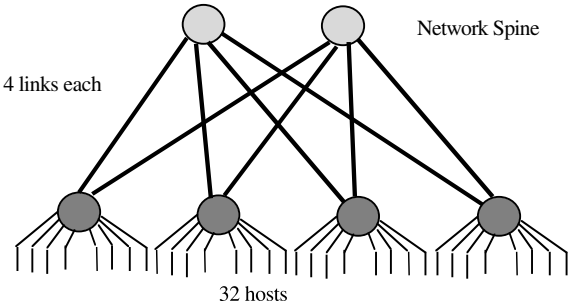

Figure 7.9 A 32-host Clos network using 16-port Myrinet switch (each line represents four links).

outgoing port before being stripped off the packet header. At the host interface, a control program is executed to perform source-route translation.

7.4.4 The Quadrics Network

According to Petrini et al. (2002), the Quadrics network (QsNet) consists of two hardware building blocks: a programmable network interface called Elan and a high-bandwidth, low-latency communication switch called Elite. The Elan network interface connects the Quadrics network to a processing node containing one or more CPUs. In addition to generating and accepting packets to and from the network, Elan provides substantial local processing power to implement high-level message passing protocols such as the Message Passing Interface (MPI), which we will study in Chapter 9.

QsNet connects Elite switches in a quaternary fat-tree topology. A quaternary fat tree of dimension n is composed of 4^n processing nodes and $n \times 4^{n-1}$ switches interconnected as a delta network. It can be recursively built by connecting four quaternary fat trees of dimension $n - 1$. Figures 7.10 and 7.11 show quaternary fat trees of dimensions 1 and 2, respectively. When $n = 1$, the network consists of one switch and four processing nodes. When $n = 2$, the network consists of eight switches and 16 processing nodes.

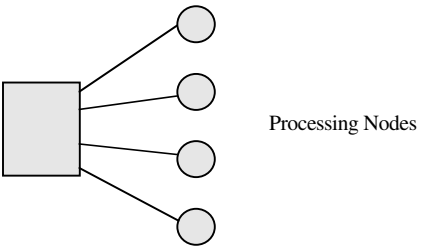

Figure 7.10 Quaternary fat tree of dimension 1.

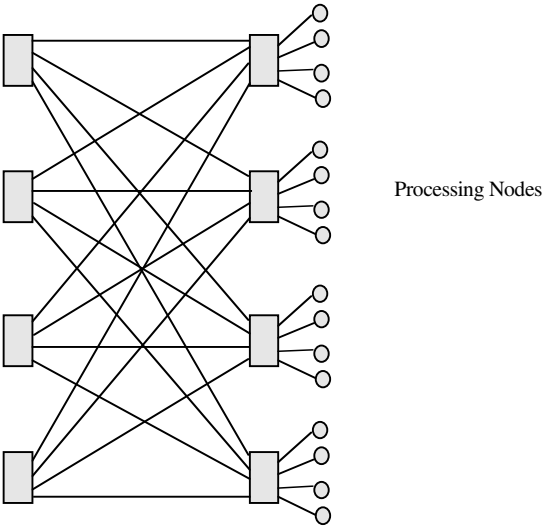

Figure 7.11 Elite switch of Quadrics networks.

Elite networks are source routed. The Elan network interface attaches route information to the packet header before transmitting the packet into the network. The route information is a sequence of Elite link tags. As the packet moves inside the network, each switch removes the first route tag from the header and forwards the packet to the next switch in the route or the final destination. The routing tag can identify either a single output link or a group of links. Packets are routed using wormhole routing flow control. As discussed in Chapter 5, each packet is divided into flow control digits (flits). In QsNet, the size of each flit is 16 bits. Network nodes can send packets to multiple destinations using the network's broadcast capability.

7.5 CLUSTER EXAMPLES

7.5.1 Berkeley Network of Workstations (NOW)

The Berkeley Network of Workstations (NOW) is an important representative of cluster systems. In 1997, the NOW project achieved over 10 Gflops on the Linpack benchmark, which made it one of the top 200 fastest supercomputers in the world. The hardware/software infrastructure for the project included 100 SUN Ultrasparcs and 40 SUN Sparcstations running Solaris, 35 Intel PCs running Windows NT or a PC Unix variant, and between 500 and 1000 disks, all connected by a Myrinet switched network. The programming environments used in NOW are sockets, MPI, and a parallel version of C, called Split C. Active Messages is the basic communication primitive in Berkeley NOW. The Active Messages communication is a simplified remote procedure call that can be implemented efficiently on a wide range of hardware. Figure 7.12 shows the different components of NOW.

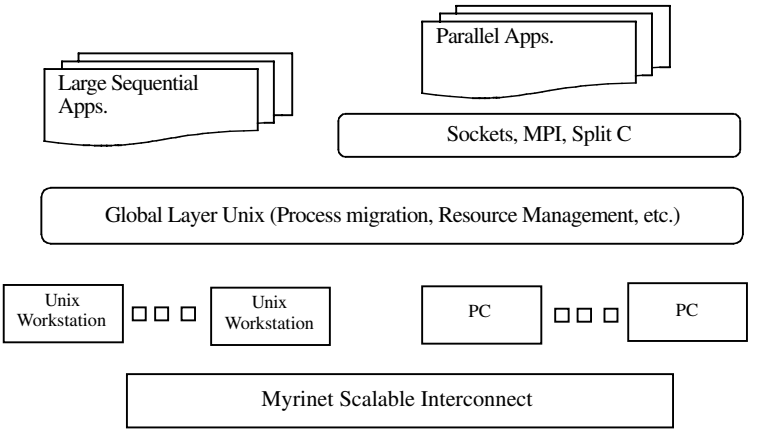

Figure 7.12 Berkeley Network of Workstations (NOW).

7.5.2 The Beowulf Cluster

The idea of the Beowulf cluster project was to achieve supercomputer processing power using off-the-shelf commodity machines. One of the earliest Beowulf clusters contained sixteen 100 MHz DX4 processors that were connected using 10 Mbps Ethernet. The second Beowulf cluster, built in 1995, used 100 MHz Pentium processors connected by 100 Mbps Ethernet. The third generation of Beowulf clusters was built by different research laboratories. JPL and Los Alamos National Laboratory each built a 16-processor machine incorporating Pentium Pro processors. These machines were combined to run a large N-body problem, which won the 1997 Gordon Bell Prize for high performance.

The communication between processors in Beowulf has been done through TCP/IP over the Ethernet internal to the cluster. Multiple Ethernets were also used to satisfy higher bandwidth requirements. Channel bonding is a technique to connect multiple Ethernets in order to distribute the communication traffic. Channel bonding was able to increase the sustained network throughput by 75% when dual networks were used.

Two of the early successful Beowulf clusters are Loki and Avalon. In 1997, Loki was built using 16 Pentium Pro Processors connected using Fast Ethernet switches. It achieved 1.2 Gflops. In 1998, the Avalon was built using one hundred and forty 533 MHz Alpha Microprocessors connected. Avalon achieved 47.7 Gflops.

7.5.3 FlashMob I

In April 2004, the University of San Francisco hosted the first Flash Mob Computing computer; FlashMob I, with the purpose of creating one of the fastest supercomputers on the planet. A FlashMob supercomputer was created by connecting a large number of computers via a high-speed LAN, to work together as a single supercomputer. A FlashMob computer, unlike an ordinary cluster, is temporary and organized ad hoc

for the purpose of working on a single problem. It used volunteers and ordinary laptop PCs, and was designed to allow anyone to create a supercomputer in a matter of hours.

Over 700 computers came into the gym and they were able to hook up 669 to the network. The best Linpack result was a peak rate of 180 Gflops using 256 computers; however, a node failed 75% through the computation. The best completed result was 77 Gflops using 150 computers. The biggest challenge was identifying flaky computers and determining the best configuration for running the benchmark. Each of the 669 computers ran Linpack at some point in the day.

7.6 GRID COMPUTING

While clusters are collections of computers tied together as a single system, grids consist of multiple systems that work together while maintaining their distinct identities. In their article "The grid grows up", Fred Douglis and Ian Foster (2003) defined the term Grid to denote middleware infrastructure, tools, and applications concerned with integrating geographically distributed computational resources. Owing to the decentralized and heterogeneous nature of the grid, the middleware that glues the different components is more complicated compared with that of clusters. Resembling an electric power grid, the computing grid is expected to become a pervasive computing infrastructure that supports large-scale and resource-intensive applications. Grid resources, which span the entire globe, include hardware, software, data, and instruments. The significant increase in application complexity and the need for collaboration have made grids an attractive computing infrastructure. Applications will continue to be complex, multidisciplinary, and multidimensional, and collaboration will become the default mode of operation. Thus, the need for the distributed grid infrastructure will continue to be an important resource.

An important concept in grids is the virtual organization, which offers a unified view of resources. Although the resources in a grid might be in separate administrative domains, they are made available as virtual local resources to any node on the grid. A user signing on at one location would view computers at other remote locations as if they were part of the local system. Grid computing works by polling the resources available, and then allocating them to individual tasks as the need arise. Resources are returned to the pool upon completion of the task. Grid gives an illusion of a big virtual computer capable of carrying out enormous tasks. The challenge is to allow meaningful sharing of resources without compromising local autonomy. Support of grids requires innovative solutions to a number of challenging issues including: resource management, resource monitoring, interoperability, security, billing and accounting, communication, and performance.

There are several examples of grid platforms and tools such as Globus and TeraGrid. The Globus Toolkit is an enabling technology for the grid. It allows users to share computing power, databases, and other tools securely on line across corporate, institutional, and geographic boundaries, without sacrificing local autonomy. The toolkit includes software services and libraries for resource monitoring, discovery, and management, plus security and file management. It also includes software for

communication, fault detection, and portability. The Globus Toolkit has grown through an open-source strategy. Version 1.0 was introduced in 1998 followed by the 2.0 release in 2002. The latest 3.0 version is based on new open-standard Grid services.

TeraGrid is a large high-performance computing project headed by the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign. The TeraGrid uses thousands of Intel Itanium 2 processors located at four sites in the United States. The TeraGrid is an effort to build and deploy the world's largest, fastest distributed infrastructure for open scientific research. The TeraGrid is expected to include 20 teraflops of computing power, facilities capable of managing and storing nearly 1 petabyte of data, high-resolution visualization environments, and toolkits for grid computing. These components will be tightly integrated and connected through a network that will operate at 40 gigabits per second.

7.7 CHAPTER SUMMARY

The recent migrations to distributed platforms have increased the need for a better understanding of network computing. Distributed platforms may be connected in a variety of ways ranging from geographically dispersed networks to architecture-specific interconnection structures. A processing unit in such systems is an autonomous computer that may be engaged in its own private activities while at the same time cooperating with other units in the context of some computational task. Network computing is concerned with how to use multiple computers to solve single or multiple problems, more or less simultaneously. The infrastructure includes desktop machines connected by a WAN, workstation clusters and Ethernet-connected or switch-connected workstations. A number of models exist to aggregate the resources of multiple compute engines for large-scale processing tasks. Multiprocessor systems incorporate multiple processors into a single machine, whether it is a desktop workstation, a mainframe, or something in between. Clusters aggregate many machines into a large, centrally managed entity. Grid computing allows each node to access resources on other nodes as if they were local. Whatever the choice, it is clear that as processor power becomes less expensive, capable stand-alone commodity processors connected via some type of high-speed network will become a standard part of computing in the future.

PROBLEMS

1. Read the Ethernet Standard and find the details including:
   (a) CSMA/CD.
   (b) What happens when multiple machines access an Ethernet at the same time?
   (c) The minimum idle time between two consecutive transmissions.

2. Explain the following:
   (a) Two-tier client/server.
   (b) Three-tier client/server.
   (c) Thin client.
   (d) Fat client.
   (e) Sockets.
   (f) Port.
3. Prove that the number of one-to-one and one-to-many connections in an $n \times n$ switch is n^n.
4. How many switches are needed in a Clos network in each of the following cases?
   (a) Number of hosts is 256.
   (b) Number of hosts is 192.
   (c) Number of hosts is 1042.
5. Draw a QsNet of dimension 3. How many switches? How many processing units?
6. Find a lower bound on the time it takes to transfer a 100 MB file across a network that operates at 1.5 Mbps, 10 Mbps, 100 Mbps, 1 Gbps, 2.4 Gbps.
7. Which of the following applications are better suited for clusters and which are better for a grid? Justify your answer.
   (a) Parallel computation with minimal interprocess communication and workflow dependencies.
   (b) Interactive.
   (c) Noninteractive (batch jobs).
   (d) Simulation.
   (e) Games.
8. Compare shared memory systems, distributed memory systems, clusters, and grids in the following aspects:
   (a) Advantages and disadvantages.
   (b) Cost and benefits.
   (c) Applications.
   (d) Performance.
9. Draw a block diagram that shows the most important modules in basic grid architecture. Show the main function of each module and the relationship between the different modules.
10. Conduct a literature search to find five grid platforms. Construct a table that shows the similarities and differences among the platforms you found. What is the most distinguishing feature of each platform?

REFERENCES

Becker, D. J., Sterling, T., Savarese, D., Dorband, J. E., Ranawake, U. A. and Packer, C. V. BEOWULF: A parallel workstation for scientific computation. *Proceedings of the 1995 International Conference on Parallel Processing (ICPP)*, 1995, pp. 11–14.

Boden, N. J., Cohen, D., Felderman, R. E., Kulawik, A. E., Seitz, C. L., Seizovic, J. N. and Su, W.-K. Myrinet – A gigabit-per-second local-area network. *IEEE Micro* (1995).

Comer, D. E. *Internetworking with TCP/IP: Principles, Protocols, and Architectures*, 4th edition, Prentice-Hall, 2000.

Douglis, F. and Foster, I. The Grid Grows Up. *IEEE Internet Computing* (2003).

Foster, I. The Grid: A new infrastructure for 21st century science. *Physics Today*, 55 (2), 42–47 (2002).

Foster, I., Kesselman, C. and Tuecke, S. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15 (3) (2001).

Hwang, K. et al. Designing SSI clusters with hierarchical checkpoints and single I/O space. *IEEE Concurrency*, 7 (1), (1999).

Petrini, F., Feng, W.-C., Hoisie, A., Coll, S. and Frachtenberg, E. The Quadrics network: High performance clustering technology. *IEEE Micro* (2002).

Warren, M. S., Germann, T. C., Lomdahl, P. S., Beazley, D. M. and Salmon, J. K. Avalon: An Alpha/Linux cluster achieves 10 Gflops for $150k. In *Supercomputing '98*, Los Alamitos, 1998, IEEE Comp. Soc.

Websites

http://www.flashmobcomputing.org/
http://www.beowulf.org/
http://www.ieeetfcc.org/
http://www.myricom.com/myrinet/overview/
http://www.internetworldstats.com
http://now.cs.berkeley.edu/
http://www.teragrid.org
http://www.globus.org/