In short, a distributed computing system is basically a collection of processor interconnected by a communication network in which each processor has its own loca memory and other peripherals, and the communication between any two processors of th system takes place by message passing over the communication network. For a particul processor, its own resources are *local*, whereas the other processors and their resources ar *remote*. Together, a processor and its resources are usually referred to as a *node* or *site* c *machine* of the distributed computing system.

1.2 EVOLUTION OF DISTRIBUTED COMPUTING SYSTEMS

Early computers were very expensive (they cost millions of dollars) and very large in siz (they occupied a big room). There were very few computers and were available only i research laboratories of universities and industries. These computers were run from console by an operator and were not accessible to ordinary users. The programmers wou write their programs and submit them to the computer center on some media, such punched cards, for processing. Before processing a job, the operator would set up th necessary environment (mounting tapes, loading punched cards in a card reader, etc.) f processing the job. The job was then executed and the result, in the form of printed outpu was later returned to the programmer.

The job setup time was a real problem in early computers and wasted most of t valuable central processing unit (CPU) time. Several new concepts were introduced in t 1950s and 1960s to increase CPU utilization of these computers. Notable among these a batching together of jobs with similar needs before processing them, automa sequencing of jobs, off-line processing by using the concepts of buffering and spoolin and multiprogramming. *Batching* similar jobs improved CPU utilization quite a l because now the operator had to change the execution environment only when a new bat of jobs had to be executed and not before starting the execution of every job. *Automa job sequencing* with the use of control cards to define the beginning and end of a j improved CPU utilization by eliminating the need for human job sequencing. *Off-li processing* improved CPU utilization by allowing overlap of CPU and input/output (I/ operations by executing those two actions on two independent machines (I/O devices a normally several orders of magnitude slower than the CPU). Finally, *multiprogrammi* improved CPU utilization by organizing jobs so that the CPU always had something execute.

However, none of these ideas allowed multiple users to directly interact with computer system and to share its resources simultaneously. Therefore, execution interactive jobs that are composed of many short actions in which the next action deper on the result of a previous action was a tedious and time-consuming activity. Developm and debugging of programs are examples of interactive jobs. It was not until the ea 1970s that computers started to use the concept of *time sharing* to overcome this hurc Early time-sharing systems had several dumb terminals attached to the main compu These terminals were placed in a room different from the main computer room. Us these terminals, multiple users could now simultaneously execute interactive jobs a share the resources of the computer system. In a time-sharing system, each user is giv

Scanned by CamScanner

the impression that he or she has his or her own computer because the system switches rapidly from one user's job to the next user's job, executing only a very small part of each job at a time. Although the idea of time sharing was demonstrated as early as 1960, time-sharing computer systems were not common until the early 1970s because they were difficult and expensive to build.

Parallel advancements in hardware technology allowed reduction in the size and increase in the processing speed of computers, causing large-sized computers to be gradually replaced by smaller and cheaper ones that had more processing capability than their predecessors. These systems were called *minicomputers*.

The advent of time-sharing systems was the first step toward distributed computing systems because it provided us with two important concepts used in distributed computing systems—the sharing of computer resources simultaneously by many users and the accessing of computers from a place different from the main computer room. Initially, the terminals of a time-sharing system were dumb terminals, and all processing was done by the main computer system. Advancements in microprocessor technology in the 1970s allowed the dumb terminals to be replaced by intelligent terminals so that the concepts of off-line processing and time sharing could be combined to have the advantages of both concepts in a single system. Microprocessor technology continued to advance rapidly, making available in the early 1980s single-user computers called *workstations* that had computing power almost equal to that of minicomputers but were available for only a small fraction of the price of a minicomputer. For example, the first workstation developed at Xerox PARC (called Alto) had a high-resolution monochrome display, a mouse, 128 kilobytes of main memory, a 2.5-megabyte hard disk, and a microprogrammed CPU that executed machine-level instructions at speeds of 2–6 μs. These workstations were then used as terminals in the time-sharing systems. In these time-sharing systems, most of the processing of a user's job could be done at the user's own computer, allowing the main computer to be simultaneously shared by a larger number of users. Shared resources such as files, databases, and software libraries were placed on the main computer.

Centralized time-sharing systems described above had a limitation in that the terminals could not be placed very far from the main computer room since ordinary cables were used to connect the terminals to the main computer. However, in parallel, there were advancements in computer networking technology in the late 1960s and early 1970s that emerged as two key networking technologies—*LAN* (*local area network*) and *WAN* (*wide-area network*). The LAN technology allowed several computers located within a building or a campus to be interconnected in such a way that these machines could exchange information with each other at data rates of about 10 megabits per second (Mbps). On the other hand, WAN technology allowed computers located far from each other (may be in different cities or countries or continents) to be interconnected in such a way that these machines could exchange information with each other at data rates of about 56 kilobits per second (Kbps). The first high-speed LAN was the Ethernet developed at Xerox PARC in 1973, and the first WAN was the ARPAnet (Advanced Research Projects Agency Network) developed by the U.S. Department of Defense in 1969. The data rates of networks continued to improve gradually in the 1980s, providing data rates of up to 100 Mbps for LANs and data rates of up to 64 Kbps for WANs. Recently (early 1990s) there has been another major advancement in networking technology—the *ATM* (*asynchronous*

Scanned by CamScanner

transfer mode) technology. The ATM technology is an emerging technology that is still not very well established. It will make very high speed networking possible, providing data transmission rates up to 1.2 gigabits per second (Gbps) in both LAN and WAN environments. The availability of such high-bandwidth networks will allow future distributed computing systems to support a completely new class of distributed applications, called *multimedia applications*, that deal with the handling of a mixture of information, including voice, video, and ordinary data. These applications were previously unthinkable with conventional LANs and WANs.

The merging of computer and networking technologies gave birth to distributed computing systems in the late 1970s. Although the hardware issues of building such systems were fairly well understood, the major stumbling block at that time was the availability of adequate software for making these systems easy to use and for fully exploiting their power. Therefore, starting from the late 1970s, a significant amount of research work was carried out in both universities and industries in the area of distributed operating systems. These research activities have provided us with the basic ideas of designing distributed operating systems. Although the field is still immature, with ongoing active research activities, commercial distributed operating systems have already started to emerge. These systems are based on already established basic concepts. This book deals with these basic concepts and their use in the design and implementation of distributed operating systems. Several of these concepts are equally applicable to the design of applications for distributed computing systems, making this book also suitable for use by the designers of distributed applications.

1.3 DISTRIBUTED COMPUTING SYSTEM MODELS

Various models are used for building distributed computing systems. These models can be broadly classified into five categories—minicomputer, workstation, workstation-server, processor-pool, and hybrid. They are briefly described below.

1.3.1 Minicomputer Model

The *minicomputer model* is a simple extension of the centralized time-sharing system. As shown in Figure 1.2, a distributed computing system based on this model consists of a few minicomputers (they may be large supercomputers as well) interconnected by a communication network. Each minicomputer usually has multiple users simultaneously logged on to it. For this, several interactive terminals are connected to each minicomputer. Each user is logged on to one specific minicomputer, with remote access to other minicomputers. The network allows a user to access remote resources that are available on some machine other than the one on to which the user is currently logged.

The minicomputer model may be used when resource sharing (such as sharing of information databases of different types, with each type of database located on a different machine) with remote users is desired.

The early ARPAnet is an example of a distributed computing system based on the minicomputer model.

Scanned by CamScanner

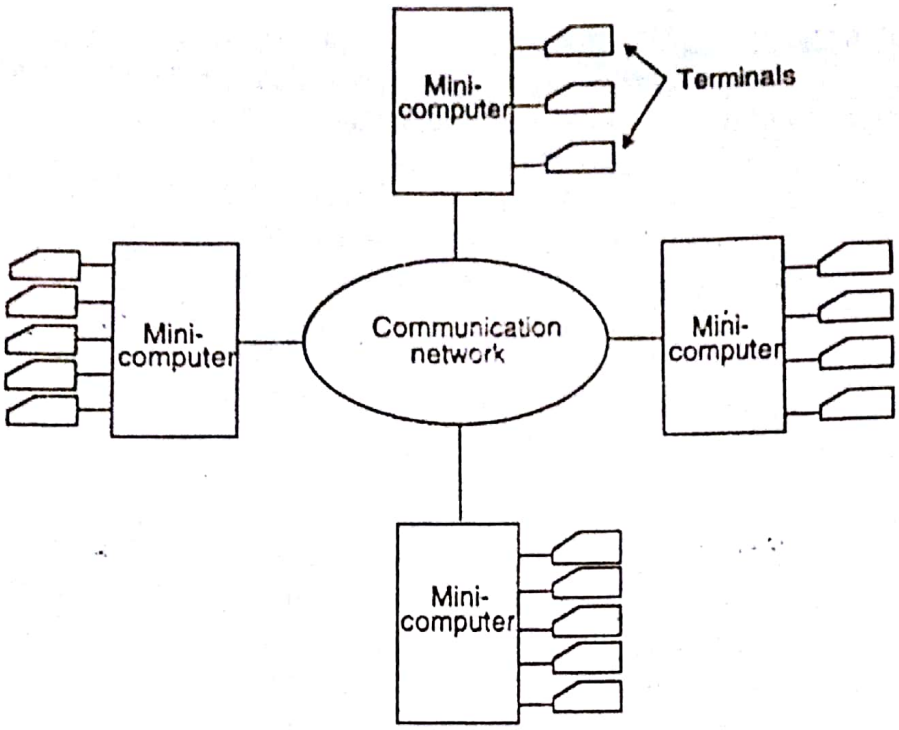

Fig. 1.2 A distributed computing system based on the minicomputer model.

1.3.2 Workstation Model

As shown in Figure 1.3, a distributed computing system based on the *workstation model* consists of several workstations interconnected by a communication network. A company's office or a university department may have several workstations scattered throughout a building or campus, each workstation equipped with its own disk and serving as a single-user computer. It has been often found that in such an environment, at any one time (especially at night), a significant proportion of the workstations are idle (not being used), resulting in the waste of large amounts of CPU time. Therefore, the idea of the workstation model is to interconnect all these workstations by a high-speed LAN so that idle workstations may be used to process jobs of users who are logged onto other workstations and do not have sufficient processing power at their own workstations to get their jobs processed efficiently.

In this model, a user logs onto one of the workstations called his or her "home" workstation and submits jobs for execution. When the system finds that the user's workstation does not have sufficient processing power for executing the processes of the submitted jobs efficiently, it transfers one or more of the processes from the user's workstation to some other workstation that is currently idle and gets the process executed there, and finally the result of execution is returned to the user's workstation.

This model is not so simple to implement as it might appear at first sight because several issues must be resolved. These issues are [Tanenbaum 1995] as follows:

Scanned by CamScanner

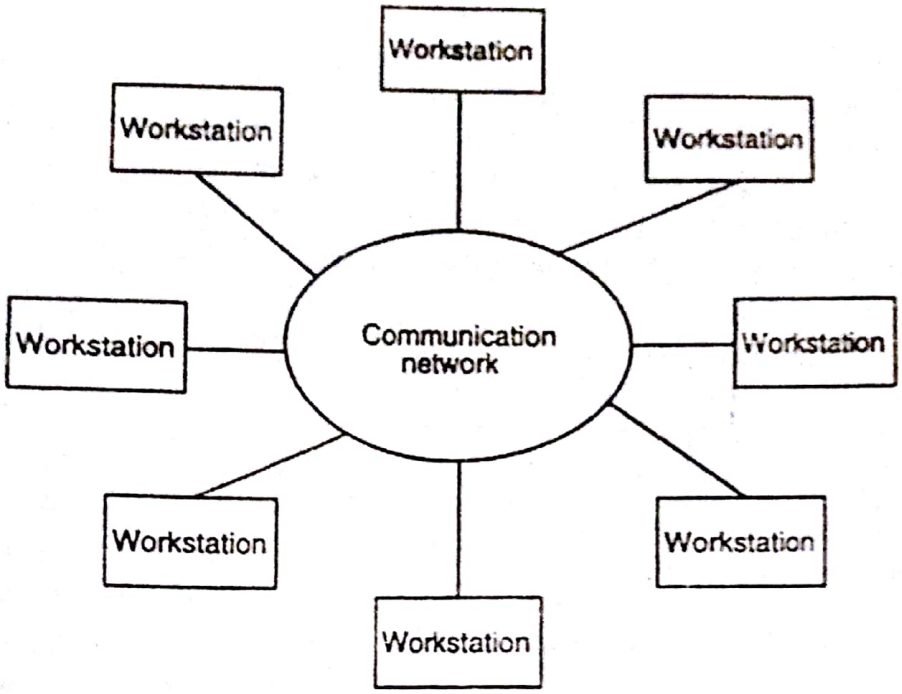

Fig. 1.3 A distributed computing system based on the workstation model.

1. How does the system find an idle workstation?
2. How is a process transferred from one workstation to get it executed on anoth workstation?
3. What happens to a remote process if a user logs onto a workstation that w idle until now and was being used to execute a process of anoth workstation?

Ways to handle the first two issues are described in Chapters 7 and 8, respectivel Three commonly used approaches for handling the third issue are as follows:

1. The first approach is to allow the remote process share the resources of t workstation along with its own logged-on user's processes. This method is easy implement, but it defeats the main idea of workstations serving as personal compute because if remote processes are allowed to execute simultaneously with the logged- user's own processes, the logged-on user does not get his or her guarante response.

2. The second approach is to kill the remote process. The main drawbacks of t method are that all processing done for the remote process gets lost and the file syste may be left in an inconsistent state, making this method unattractive.

3. The third approach is to migrate the remote process back to its ho workstation, so that its execution can be continued there. This method is difficult implement because it requires the system to support preemptive process migrati facility. The definition of preemptive process migration and the issues involved preemptive process migration are given in Chapter 8.

Scanned by CamScanner

The Sprite system [Ousterhout et al. 1988] and an experimental system developed at Xerox PARC [Shoch and Hupp 1982] are two examples of distributed computing systems based on the workstation model.

1.3.3 Workstation-Server Model

The workstation model is a network of personal workstations, each with its own disk and a local file system. A workstation with its own local disk is usually called a *diskful workstation* and a workstation without a local disk is called a *diskless workstation*. With the proliferation of high-speed networks, diskless workstations have become more popular in network environments than diskful workstations, making the workstation-server model more popular than the workstation model for building distributed computing systems.

As shown in Figure 1.4, a distributed computing system based on the *workstation-server model* consists of a few minicomputers and several workstations (most of which are diskless, but a few of which may be diskful) interconnected by a communication network.

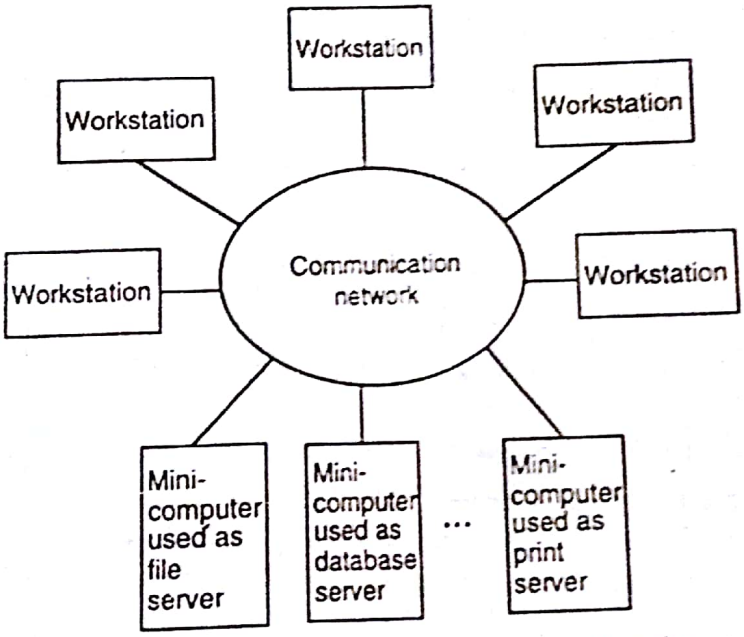

Fig. 1.4 A distributed computing system based on the workstation–server model.

Note that when diskless workstations are used on a network, the file system to be used by these workstations must be implemented either by a diskful workstation or by a minicomputer equipped with a disk for file storage. The minicomputers are used for this purpose. One or more of the minicomputers are used for implementing the file system. Other minicomputers may be used for providing other types of services, such as database service and print service. Therefore, each minicomputer is used as a server machine to provide one or more types of services. Hence in the workstation-server model, in addition to the workstations, there are specialized machines (may be specialized workstations) for running server processes (called *servers*) for managing and providing access to shared resources.

Scanned by CamScanner

For a number of reasons, such as higher reliability and better scalability, multiple servers are often used for managing the resources of a particular type in a distributed computing system. For example, there may be multiple file servers, each running on a separate minicomputer and cooperating via the network, for managing the files of all the users in the system. Due to this reason, a distinction is often made between the services that are provided to clients and the servers that provide them. That is, a *service* is an abstract entity that is provided by one or more servers. For example, one or more file servers may be used in a distributed computing system to provide file service to the users.

In this model, a user logs onto a workstation called his or her home workstation. Normal computation activities required by the user's processes are performed at the user's home workstation, but requests for services provided by special servers (such as a file server or a database server) are sent to a server providing that type of service that performs the user's requested activity and returns the result of request processing to the user's workstation. Therefore, in this model, the user's processes need not be migrated to the server machines for getting the work done by those machines.

For better overall system performance, the local disk of a diskful workstation is normally used for such purposes as storage of temporary files, storage of unshared files, storage of shared files that are rarely changed, paging activity in virtual-memory management, and caching of remotely accessed data.

As compared to the workstation model, the workstation-server model has several advantages:

1. In general, it is much cheaper to use a few minicomputers equipped with large, fast disks that are accessed over the network than a large number of diskful workstations, with each workstation having a small, slow disk.

2. Diskless workstations are also preferred to diskful workstations from a system maintenance point of view. Backup and hardware maintenance are easier to perform with a few large disks than with many small disks scattered all over a building or campus. Furthermore, installing new releases of software (such as a file server with new functionalities) is easier when the software is to be installed on a few file server machines than on every workstation.

3. In the workstation-server model, since all files are managed by the file servers, users have the flexibility to use any workstation and access the files in the same manner irrespective of which workstation the user is currently logged on. Note that this is not true with the workstation model, in which each workstation has its local file system, because different mechanisms are needed to access local and remote files.

4. In the workstation-server model, the request-response protocol described above is mainly used to access the services of the server machines. Therefore, unlike the workstation model, this model does not need a process migration facility, which is difficult to implement.

The request-response protocol is known as the *client-server model* of communication. In this model, a client process (which in this case resides on a workstation) sends a

Scanned by CamScanner

request to a server process (which in this case resides on a minicomputer) for getting some service such as reading a block of a file. The server executes the request and sends back a reply to the client that contains the result of request processing.

The client-server model provides an effective general-purpose approach to the sharing of information and resources in distributed computing systems. It is not only meant for use with the workstation-server model but also can be implemented in a variety of hardware and software environments. The computers used to run the client and server processes need not necessarily be workstations and minicomputers. They can be of many types and there is no need to distinguish between them. It is even possible for both the client and server processes to be run on the same computer. Moreover, some processes are both client and server processes. That is, a server process may use the services of another server, appearing as a client to the latter.

5. A user has guaranteed response time because workstations are not used for executing remote processes. However, the model does not utilize the processing capability of idle workstations.

The V-System [Cheriton 1988] is an example of a distributed computing system that is based on the workstation-server model.

1.3.4 Processor-Pool Model

The *processor-pool model* is based on the observation that most of the time a user does not need any computing power but once in a while he or she may need a very large amount of computing power for a short time (e.g., when recompiling a program consisting of a large number of files after changing a basic shared declaration). Therefore, unlike the workstation-server model in which a processor is allocated to each user, in the processor-pool model the processors are pooled together to be shared by the users as needed. The pool of processors consists of a large number of microcomputers and minicomputers attached to the network. Each processor in the pool has its own memory to load and run a system program or an application program of the distributed computing system.

As shown in Figure 1.5, in the pure processor-pool model, the processors in the pool have no terminals attached directly to them, and users access the system from terminals that are attached to the network via special devices. These terminals are either small diskless workstations or graphic terminals, such as X terminals. A special server (called a *run server*) manages and allocates the processors in the pool to different users on a demand basis. When a user submits a job for computation, an appropriate number of processors are temporarily assigned to his or her job by the run server. For example, if the user's computation job is the compilation of a program having n segments, in which each of the segments can be compiled independently to produce separate relocatable object files, n processors from the pool can be allocated to this job to compile all the n segments in parallel. When the computation is completed, the processors are returned to the pool for use by other users.

In the processor-pool model there is no concept of a home machine. That is, a user does not log onto a particular machine but to the system as a whole. This is in contrast

Scanned by CamScanner

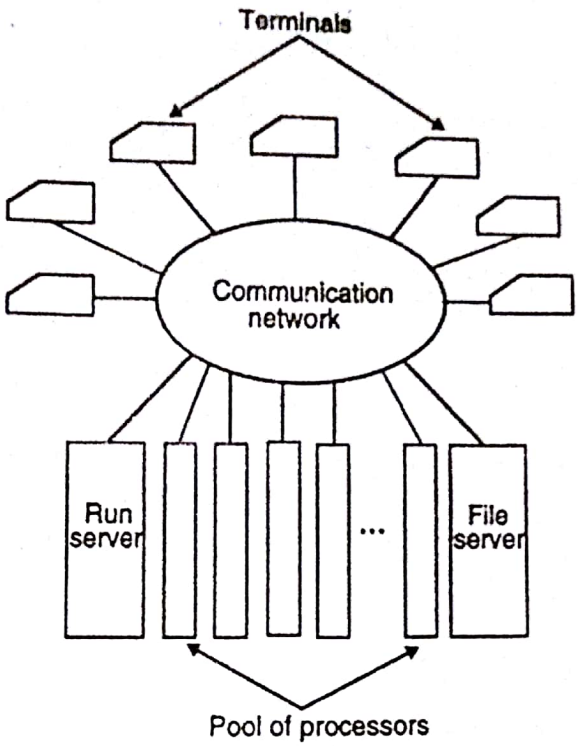

Fig. 1.5 A distributed computing system based on the processor-pool model.

to other models in which each user has a home machine (e.g., a workstation or minicomputer) onto which he or she logs and runs most of his or her programs there by default.

As compared to the workstation-server model, the processor-pool model allows better utilization of the available processing power of a distributed computing system. This is because in the processor-pool model, the entire processing power of the system is available for use by the currently logged-on users, whereas this is not true for the workstation-server model in which several workstations may be idle at a particular time but they cannot be used for processing the jobs of other users. Furthermore, the processor-pool model provides greater flexibility than the workstation-server model in the sense that the system's services can be easily expanded without the need to install any more computers; the processors in the pool can be allocated to act as extra servers to carry any additional load arising from an increased user population or to provide new services. However, the processor-pool model is usually considered to be unsuitable for high-performance interactive applications, especially those using graphics or window systems. This is mainly because of the slow speed of communication between the computer on which the application program of a user is being executed and the terminal via which the user is interacting with the system. The workstation-server model is generally considered to be more suitable for such applications.

Amoeba [Mullender et al. 1990], Plan 9 [Pike et al. 1990], and the Cambridge Distributed Computing System [Needham and Herbert 1982] are examples of distributed computing systems based on the processor-pool model.

Scanned by CamScanner

1.3.5 Hybrid Model

Out of the four models described above, the workstation-server model, is the most widely used model for building distributed computing systems. This is because a large number of computer users only perform simple interactive tasks such as editing jobs, sending electronic mails, and executing small programs. The workstation-server model is ideal for such simple usage. However, in a working environment that has groups of users who often perform jobs needing massive computation, the processor-pool model is more attractive and suitable.

To combine the advantages of both the workstation-server and processor-pool models, a hybrid model may be used to build a distributed computing system. The hybrid model is based on the workstation-server model but with the addition of a pool of processors. The processors in the pool can be allocated dynamically for computations that are too large for workstations or that require several computers concurrently for efficient execution. In addition to efficient execution of computation-intensive jobs, the hybrid model gives guaranteed response to interactive jobs by allowing them to be processed on local workstations of the users. However, the hybrid model is more expensive to implement than the workstation-server model or the processor-pool model.

1.4 WHY ARE DISTRIBUTED COMPUTING SYSTEMS GAINING POPULARITY?

From the models of distributed computing systems presented above, it is obvious that distributed computing systems are much more complex and difficult to build than traditional *centralized systems* (those consisting of a single CPU, its memory, peripherals, and one or more terminals). The increased complexity is mainly due to the fact that in addition to being capable of effectively using and managing a very large number of distributed resources, the system software of a distributed computing system should also be capable of handling the communication and security problems that are very different from those of centralized systems. For example, the performance and reliability of a distributed computing system depends to a great extent on the performance and reliability of the underlying communication network. Special software is usually needed to handle loss of messages during transmission across the network or to prevent overloading of the network, which degrades the performance and responsiveness to the users. Similarly, special software security measures are needed to protect the widely distributed shared resources and services against intentional or accidental violation of access control and privacy constraints.

Despite the increased complexity and the difficulty of building distributed computing systems, the installation and use of distributed computing systems are rapidly increasing. This is mainly because the advantages of distributed computing systems outweigh their disadvantages. The technical needs, the economic pressures, and the major advantages that have led to the emergence and popularity of distributed computing systems are described next.

Scanned by CamScanner