CMP 432 DISTRIBUTED COMPUTING SYSTEM

Department of mathematics and Computer science

Benue State University

Dr. Adekunle Adeyelu

Dr. Harold Kpojime

LECTURE NOTES: PART A

Textbooks:

Distributed Systems: Principles and Paradigms by Andrew Tanenbaum and Marteen V.S. Chapters: 1,2,4,5,6,7,8,9 and 11

Distributed Systems: Concepts and design by George Coulouris and Jean D

Any other textbook on distributed systems

A distributed system is a collection of independent computers that appear to its users as a single coherent system. It consists of several computers who do not share a memory or clock. The computers communicate with each other by exchanging messages over a communication network and each computer has its own memory and runs its own operating system. The resources owned and controlled by a computer are said to be local to it while the resources owned and controlled by other computers and those that can be only accessed through the network are said to be remote.

Distributed systems are ubiquitous today throughout business, academia, government and the home. Typically the goal of distributed system is to provide means to share resources, for instance, special purpose equipment such as colour printers or scanners, and to share data, crucial for our information based economy. More ambitious distributed systems attempt to provide improved performance by attacking sub-problems in parallel, and to provide improved availability in case of failures of some components.

A typical distributed system is shown in Fig. 1. Each computer has a memory-processing unit and the computers are connected by a communication network. Fig. 1 shows the relationships of the software components that run on each of the computers and use the local operating system and network protocol stack for functioning. The distributed software is also termed as middleware. A distributed execution is the execution of processes across the distributed system to collaboratively achieve a common goal. An execution is also sometimes termed a computation or a run.

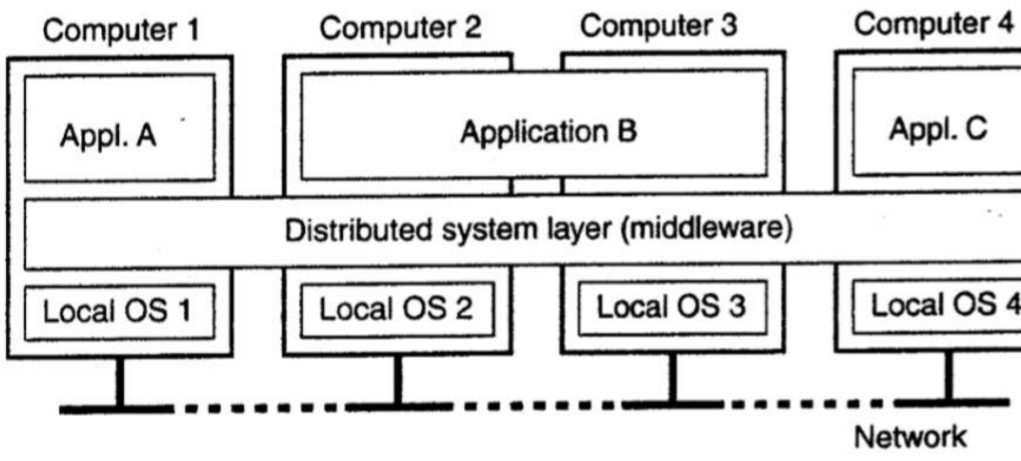

Fig.1: A distributed system with middleware

FEATURES OF A DISTRIBUTED SYSTEM

i. **No common physical clock**

This plays an important role to introduce the element of "distribution" in a system and takes the responsibility to provide inherent asynchrony amongst the processors. In distributed network the nodes do not share common physical clock

ii. **No shared memory**

This is an important aspect of for message-passing communication among the nodes present in a network. There is no common physical clock concept in this memory architecture. But it is still possible to provide the abstraction of a common address space via the distributed shared memory abstraction.

iii. **Geographical separation**

In distributed computing system the processors are geographically distributed even over the globe. However, it is not essential for the processors to be present on a wide-area network (WAN). It is possible to make a network/cluster of workstations present on a LAN can be considered as a small distributed system.

iv. **Autonomy and heterogeneity**

The processors are autonomous in nature because they have independent memories, different configurations and are usually not part of a dedicated system connected through any network, but cooperate with one another by offering services or solving a problem together.

ADVANTAGES OF DISTRIBUTED SYSTEMS

Reliability

A distributed system has the inherent potential to provide increased reliability because of the possibility of replicating resources and executions, as well as the reality that geographically distributed resources are not likely to crash/malfunction at the same time under normal circumstances. By distributing the workload over many machines, a single chip failure will bring down at most one machine while leaving the rest intact.

Speed

By resource sharing and accessing geographically remote data and resources, the performance/cost ratio is increased. Although higher throughput has not necessarily been the main objective behind using a distributed system, nevertheless, any task can be partitioned across the various computers in the distributed system. Such a configuration provides a better performance/cost ratio than using special parallel machines.

Inherent Distribution

Many institutions have applications which are more suitable for distributed computation. Assume that there is a large company buying and selling goods from different countries which makes it geographically diverse. If the company wishes for a unified computing system, it should implement a distributed computing system. Also, consider the global employee database of such a company. The branch offices can create a local database and link them up for global viewing.

Data sharing

In many scenarios, the data cannot be replicated at every site participating in the distributed execution because it may be too large or too sensitive to be replicated. For example, payroll data within a multinational corporation is both too large and too sensitive to be replicated at every branch office/site. It is therefore stored at a central server which can be queried by branch offices. Similarly, special resources such as supercomputers exist only in certain locations, and to access such supercomputers, users need to log in remotely.

Resource sharing

Resources such as peripherals (printers and scanners), complete data sets in databases, special libraries, as well as data (variable/files) cannot be fully replicated at all the sites because it is often neither practical nor cost-effective. Further, they cannot be placed at a single site because access to that site might prove to be a bottleneck. Therefore, such resources are typically distributed across the system.

TYPES OF DISTRIBUTED SYSTEMS

Before starting to discuss the principles of distributed systems, let us first take a closer look at the various types of distributed systems. In the following we make a distinction between distributed computing systems, distributed information systems, and distributed embedded systems.

- **Distributed computing systems**
- **Distributed information systems**

- **Distributed embedded systems**

DISTRIBUTED COMPUTING SYSTEMS

Distributed computing System (DCS) is an important class of distributed systems which is used for **high-performance computing** tasks. Distributed computing systems are divided into two sub groups namely Cluster computing and Grid Computing.

Cluster Computing

Cluster computing systems became popular when the price/performance ratio of personal computers and workstations improved. At a certain point, it became financially and technically attractive to build a supercomputer using off-the-shelf technology by simply hooking up a collection of relatively simple computers in a high-speed network. In virtually all cases, cluster computing is used for parallel programming in which a single (compute intensive) program is run in parallel on multiple machines.

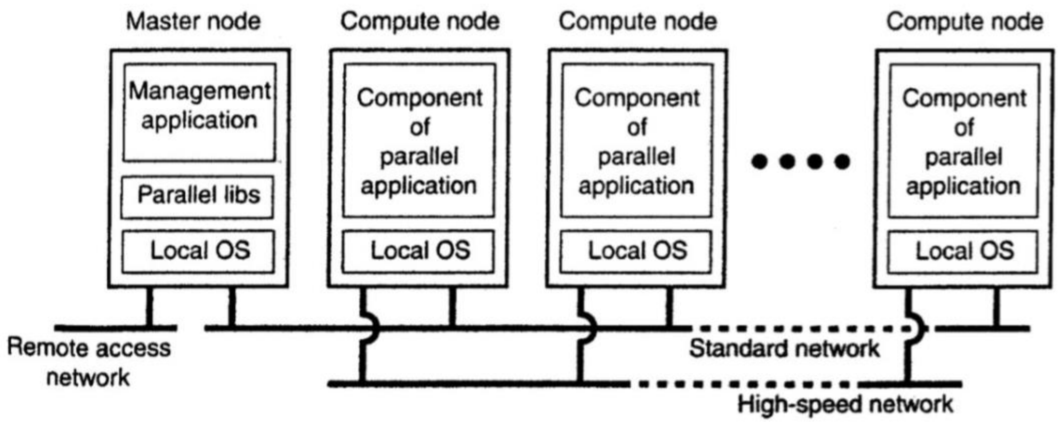

Fig 2: Cluster computing System

Characteristics of cluster computing:

- The underlying hardware consists of a collection of similar workstations or PCs (compute nodes)
- Connected by means of a high speed local-area network
- Each node runs the same operating system
- Compute nodes are controlled and accessed by means of a single master node.
- The master typically **handles the allocation of nodes to a particular parallel program, maintains a batch queue of submitted jobs, and provides an interface** for the users of the system.
- The **master runs the middleware** needed for the execution of programs and management of the cluster,
- An important part of this middleware is formed by the libraries for executing parallel programs.

One well-known example of a cluster computer is formed by Linux-based Beowulf clusters

Grid Computing Systems

The situation becomes quite different in the case of grid computing. This sub group consists of distributed systems that are often constructed as a federation of computer systems, where each system may fall under a different administrative domain, and may be very different when it comes to hardware, software, and deployed network technology.

Grid computing enables the sharing, selection, and aggregation by users of a wide variety of geographically distributed resources owned by different organizations and is well-suited for solving IT resource intensive problems in science, engineering and commerce. Grids are very large-scale virtualized, distributed computing systems. They cover multiple administrative domains and enable virtual organizations. Such organizations can share their resources collectively to create an even larger grid.

For instance, 80,000 CPU cores are shared within EGEE (Enabling Grids for E-sciencE), one of the largest multi-disciplinary grid infrastructure in the world. This brings together more than 10,000 users in 140 institutions (300 sites in 50 countries) to produce a reliable and scalable computing resource available to the European and global research community.

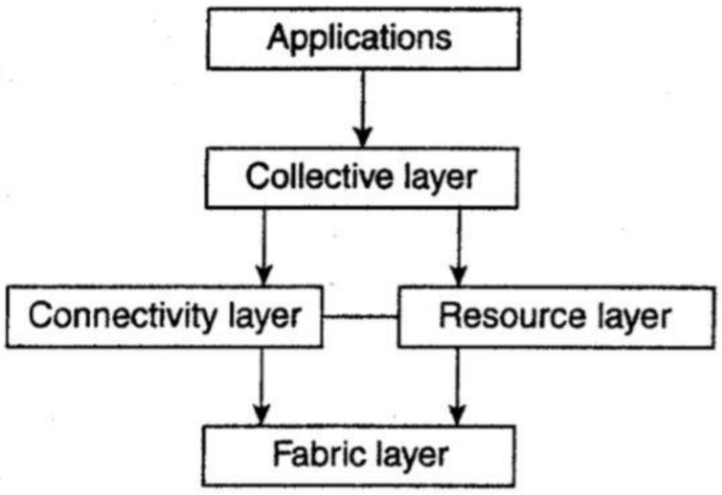

Fig 3: Grid Computing System Architecture

Characteristics of Grid computing:

- It is its **heterogeneous** which means no assumptions are made concerning hardware, operating systems, networks, administrative domains, security policies, etc unlike clusters where they systems are homogeneous.
- Form a **virtual organisation** – Resources from different organizations are brought together to allow the collaboration of a group of people or institutions.
- The people belonging to the same virtual organization have **access rights to the resources** that are provided to that organization.

- Typically, resources consist of computer servers (including supercomputers, possibly implemented as cluster computers), storage facilities, and databases.
- In addition, special networked devices such as telescopes, sensors, etc., can be provided as well.

CLASSIFICATION OF DISTRIBUTED COMPUTING SYSTEMS

Distributed computing systems can be classified into 5 categories namely;

- Minicomputer
- Workstation
- Workstation-server
- Processor pool
- Hybrid

Minicomputer

- The mini computer model consists of a few minicomputers 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.

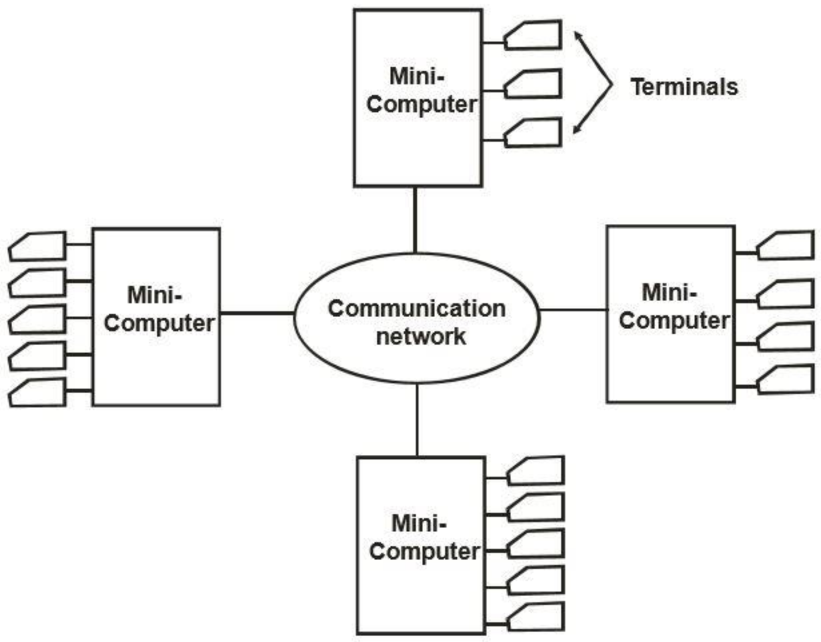

Fig. 4: Minicomputer Computing System

Workstation Model

As shown in Fig. 5, a distributed computing system based on the workstation model consists of several workstations interconnected by a communication network.

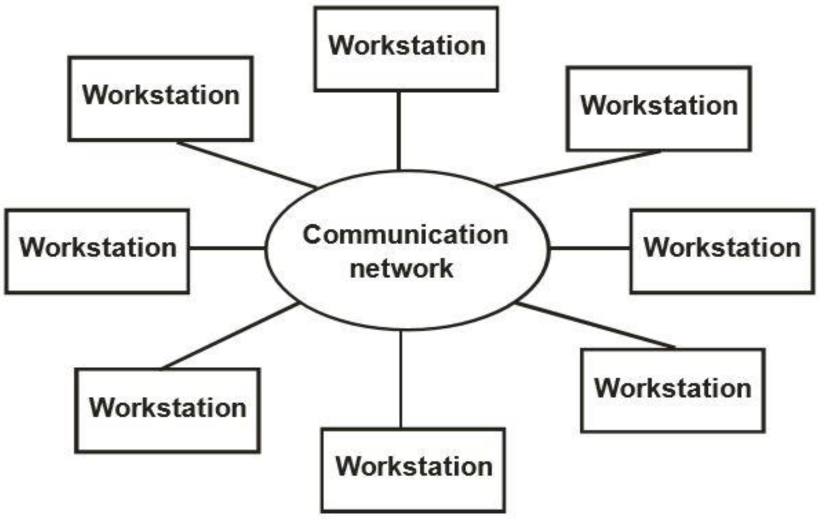

Fig. 5: Work station model

Scenario where this applies;

A university department such as ours 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 process 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.

Workstation–Server Model

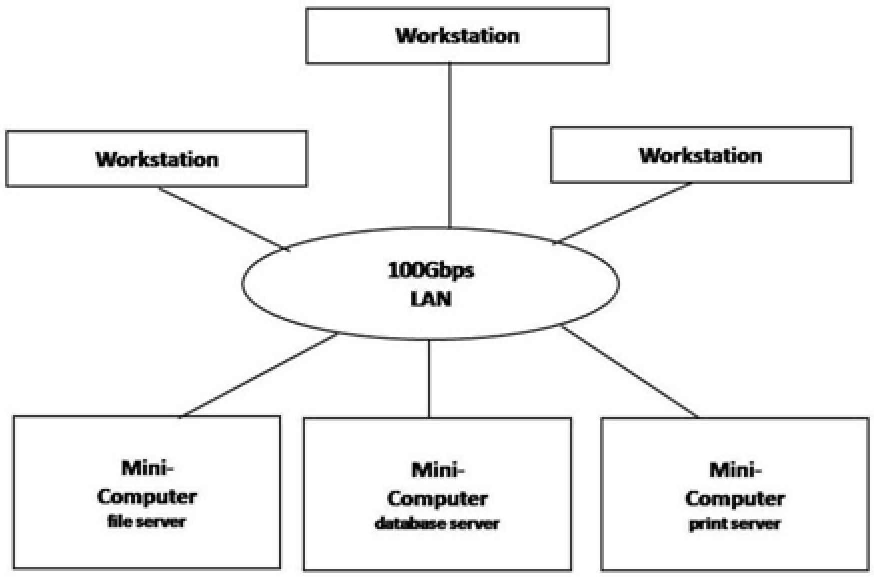
Workstation
Workstation
Workstation
100Gbps LAN
Mini-Computer file server
Mini-Computer database server
Mini-Computer print server

Fig 6: Workstation Server model

- The workstation model is a network of personal workstations having its own disk & a local file system.
- A workstation with its own local disk is usually called a diskful workstation & a workstation without a local disk is called a diskless workstation. 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.

- A distributed computing system based on the workstation-server model consists of a few minicomputers & several workstations interconnected by a communication network.
- 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 are sent to a server providing that type of service that performs the user's requested activity & returns the result of request processing to the user's workstation.
- Therefore, in this model, the user's processes need not migrated to the server machines for getting the work done by those machines.
- Example: The V-System.

Processor–Pool Model:

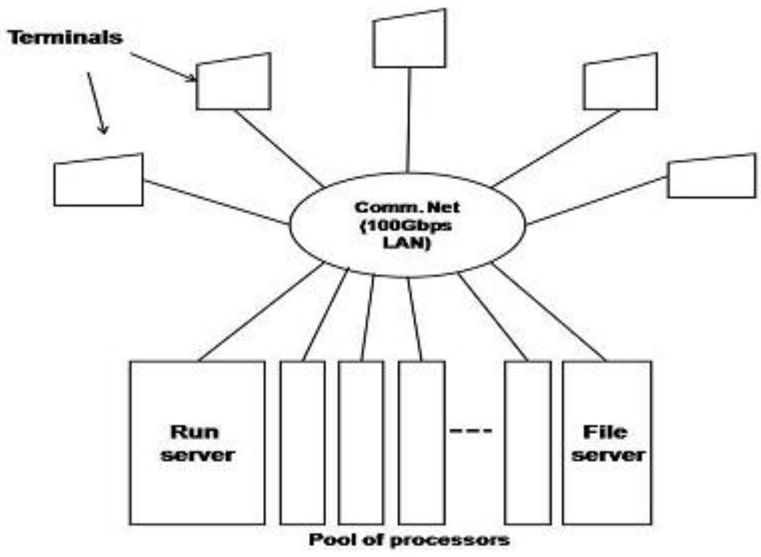

Fig 7: Processor pool Model

- 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 the user may need a very large amount of computing power for a short time.
- Therefore, unlike the workstation-server model in which a processor is allocated to each user, in 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 & minicomputers attached to the network.
- Each processor in the pool has its own memory to load & run a system program or an application program of the distributed computing system.
- In this model no home machine is present & the user does not log onto any machine.
- This model has better utilization of processing power & greater flexibility.
- Example: Amoeba & the Cambridge Distributed Computing System.

Hybrid Model:

- The workstation-server model has a large number of computer users only performing simple interactive tasks &-executing small programs.
- In a working environment that has groups of users who often perform jobs needing massive computation, the processor-pool model is more attractive & suitable.
- To combine Advantages of workstation-server & processor-pool models, a hybrid model can be used to build a distributed system.
- The processors in the pool can be allocated dynamically for computations that are too large or require several computers for execution.
- The hybrid model gives guaranteed response to interactive jobs allowing them to be more processed in local workstations of the users

CHALLENGES IN DESIGNING A DISTRIBUTED SYSTEM

Heterogeneity

Heterogeneity simply means variety and it applies to all of the following:

- networks
- computer hardware
- operating systems
- programming languages
- implementations by different developers

Although the Internet consists of many different sorts of network, their differences are masked by the fact that all of the computers attached to them use the Internet protocols to communicate with one another. For example, a computer attached to an Ethernet has an implementation of the Internet protocols over the Ethernet, whereas a computer on a different sort of network will need an implementation of the Internet protocols for that network.

Data types such as integers may be represented in different ways on different sorts of hardware – for example there are two alternatives for the byte ordering of integers. These differences in representation must be dealt with if messages are to be exchanged between programs running on different hardware. Although the operating systems of all computers on the Internet need to include an implementation of the Internet protocols, they do not necessarily all provide the same application programming interface to these protocols. For example, the calls for exchanging messages in UNIX are different from the calls in Windows.

Different programming languages use different representations for characters and data structures such as arrays and records. These differences must be addressed if programs written in different languages are to be able to communicate with one another. Programs written by different developers cannot communicate with one another unless they use common standards, for example, for network communication and the representation of primitive data items and data structures in messages. For this to happen, standards need to be agreed and adopted – as have the Internet protocols.

Transparency

A distributed system that is able to present itself to user and application as if it were only a single computer system is said to be transparent. There are eight types of transparencies in a distributed system:

a. **Access Transparency**: It hides differences in data representation and how a resource is accessed by a user. Example, a distributed system may have a computer system that runs different operating systems, each having their own file naming conventions. Differences in naming conventions as well as how files can be manipulated should be hidden from the users and applications.
b. **Location Transparency**: Hides where exactly the resource is located physically. Example, by assigning logical names to resources like yahoo.com, one cannot get an idea of the location of the web page's main server.
c. **Migration Transparency**: Distributed system in which resources can be moved without affecting how the resource can be accessed are said to provide migration transparency. It hides that the resource may move from one location to another.
d. **Relocation Transparency**: this transparency deals with the fact that resources can be relocated while it is being accessed without the user who is using the

application to know anything. Example: using a Wi-Fi system on laptops while moving from place to place without getting disconnected.

e. **Replication Transparency**: Hides the fact that multiple copies of a resource could exist simultaneously. To hide replication, it is essential that the replicas have the same name. Consequently, as system that supports replication should also support location transparency.

f. **Concurrency Transparency**: It hides the fact that the resource may be shared by several competitive users. Example, two independent users may each have stored their file on the same server and may be accessing the same table in a shared database. In such cases, it is important that each user doesn't notice that the others are making use of the same resource.

g. **Failure Transparency**: Hides failure and recovery of the resources. It is the most difficult task of a distributed system and is even impossible when certain apparently realistic assumptions are made. Example: A user cannot distinguish between a very slow or dead resource. Same error message come when a server is down or when the network is overloaded of when the connection from the client side is lost. So here, the user is unable to understand what has to be done, either the user should wait for the network to clear up, or try again later when the server is working again.

h. **Persistence Transparency**: It hides if the resource is in memory or disk. Example, Object oriented database provides facilities for directly invoking methods on storage objects. First the database server copies the object states from the disk i.e. main memory performs the operation and writes the state back to the disk. The user does not know that the server is moving between primary and secondary memory.

Reliability

Distributed systems are expected to be more reliable than centralized systems due to the existence of multiple instances of resources. However, the existence of multiple instances of the resources alone cannot increase the system’s reliability. Rather, the distributed operating system, which manages these resources, must be designed properly to increase the system’s reliability by taking full advantage of this characteristic feature of a distributed system.

For higher reliability, the fault-handling mechanisms of a distributed operating system must be designed properly to avoid faults, to tolerate faults, and to detect and recover from faults. Commonly used methods for dealing with these issues are briefly described text.

Fault avoidance

Fault avoidance deals with designing the components of the system in such a way that the occurrence of faults in minimized. Conservative design practice such as using high reliability components are often employed for improving the system’s reliability based on the idea of fault avoidance. Although a distributed operating system often has little or no role to play in improving the fault avoidance capability of a hardware component, the designers of the various software components of the distributed operating system must test them thoroughly to make these components highly reliable.

Fault tolerance is the ability of a system to continue functioning in the event of partial system failure. The performance of the system might be degraded due to partial failure, but otherwise the system functions properly.

Scalability

Scalability refers to the capability of a system to adapt to increased service load. It is inevitable that a distributed system will grow with time since it is very common to add new machines or an entire subnetwork to the system to take care of increased workload or organizational changes in a company. Therefore, a distributed operating system should be designed to easily cope with the growth of nodes and users in the system. That is, such growth should not cause serious disruption of service or significant loss of performance to users.

Security

In order that the users can trust the system and rely on it, the various resources of a computer system must be protected against destruction and unauthorized access. Enforcing security in a distributed system is difficult because of the lack of a single point of control and the use of insecure networks for data communication. Cryptography is the only known practical method for dealing with these security aspects of a distributed system. In this method comprehension of private information is prevented by encrypting the information, which can then be decrypted only by authorized users.

COMMUNICATION IN DISTRIBUTED COMPUTING SYSTEMS

To make it easier to deal with the numerous levels and issues involved in communication, the International Standards Organization (ISO) developed a reference model that clearly identifies the various levels involved, gives them standard names, and points out which level should do which job. This model is called the Open Systems Interconnection (OSI) Reference Model.

The OSI model is designed to allow open systems to communicate. An open system is one that is prepared to communicate with any other open system by using standard rules that govern the format, contents, and meaning of the messages sent and received. These rules are formalized in what are called protocols. To allow a group of computers to communicate over a network, they must all agree on the protocols to be used. A distinction is made between two general types of protocols. With connection oriented protocols, before exchanging data the sender and receiver first explicitly establish a connection, and possibly negotiate the protocol they will use. When they are done, they must release (terminate) the connection. The telephone is a connection-oriented communication system. With connectionless protocols, no setup in advance is needed