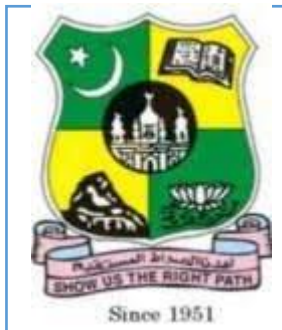


JAMAL MOHAMED COLLEGE (AUTONOMOUS)
Accredited (4th Cycle) with 'A++' Grade by NAAC
DBT Star College Scheme & DST-FIST Funded
(Affiliated to Bharathidasan University)
Tiruchirappalli , India.

DEPARTMENT OF COMPUTER SCIENCE & IT

DISTRIBUTED OPERATING SYSTEM

BY

Dr. S Vaaheedha Kfatheen
Assistant Professor
Department of Computer Science
Jamal Mohamed College(A)
Trichy

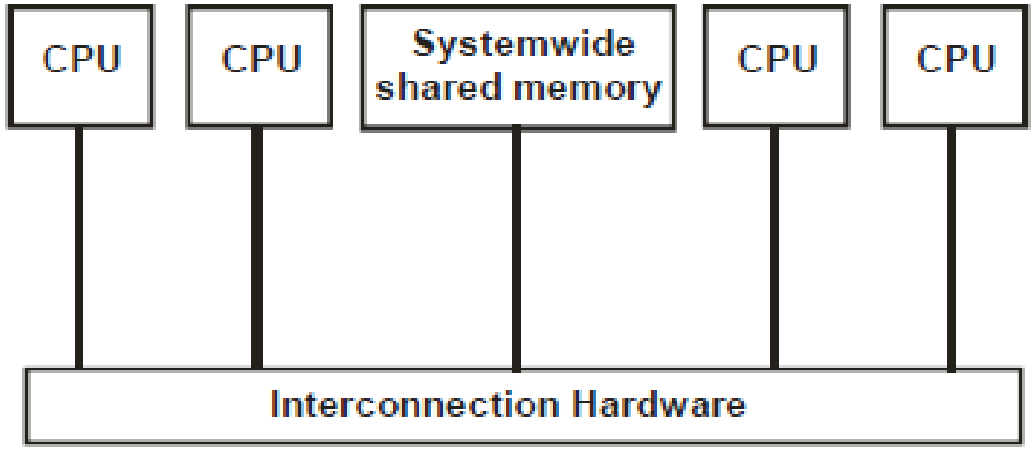
1.1 Fundamentals: What is a Distributed Operating System

Definition

Computer architectures consisting of interconnected, multiple processors are basically of two types:

Tightly coupled systems:

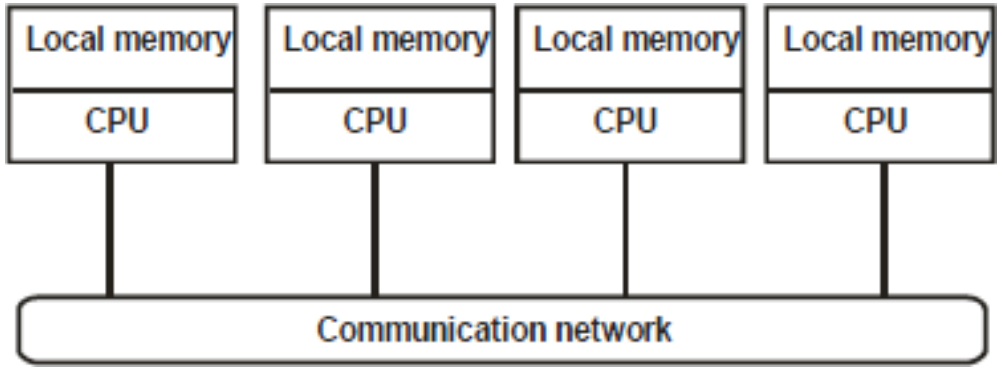
In these systems, there is a single system wide primary memory that is shared by all the processors. Any communication between the processors usually takes place through the shared memory.



Cont.,

Loosely coupled systems:

In these systems, the processors do not share memory, and each processor as its own local memory. All physical communication between the processors is done by passing messages across the network that interconnects the processors.



Cont.,

- Usually, tightly coupled systems are referred to as parallel processing systems, and loosely coupled systems are referred to as distributed computing systems or simply distributed systems.
- In short, distributed computing system is basically a collection of processors interconnected by a communication network in which each processor has its own local memory and other peripherals, and the communication between any two processors of the system takes place by message passing over the communication network.

1.2- Evolution of Distributed Computing Systems

- The job setup time was a real problem in early computers and wasted most the valuable CPU time.
- Several new concepts such as, batching, automatic job sequencing, offline processing and multiprogramming were introducted in the 1950s and 1960s.
- However, none of these concepts allowed multiple users to directly interact with a computer system and to share its resources simultaneously.
- It was in the early 1970s that the computers started to use the concept of time-sharing.

Cont.,

- 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
  - Sharing of computer resources simultaneously by many users
  - 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.

Cont.,

- Centralized time-sharing systems 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 the computer networking technology in the late 1960s and early 1970s that emerged as two key network technologies:
- LAN
- WAN

- The data rates continued to improve gradually in the 1980s providing data rates of up to 100 Mbps for LANs and data rates up to 64 Kbps for WANs .
- Recently, in 1990s, there has been another major advancement in networking technology – the ATM .
- The ATM is an emerging technology that will provide data transmission rates up to 1.2 Gbps in both LANs and WANs.
- The merging of computer and networking technologies gave birth to distributed computing systems in the late 1970s.

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 model
  - Workstation model
  - Workstation-server model
  - Processor-pool model
  - Hybrid model

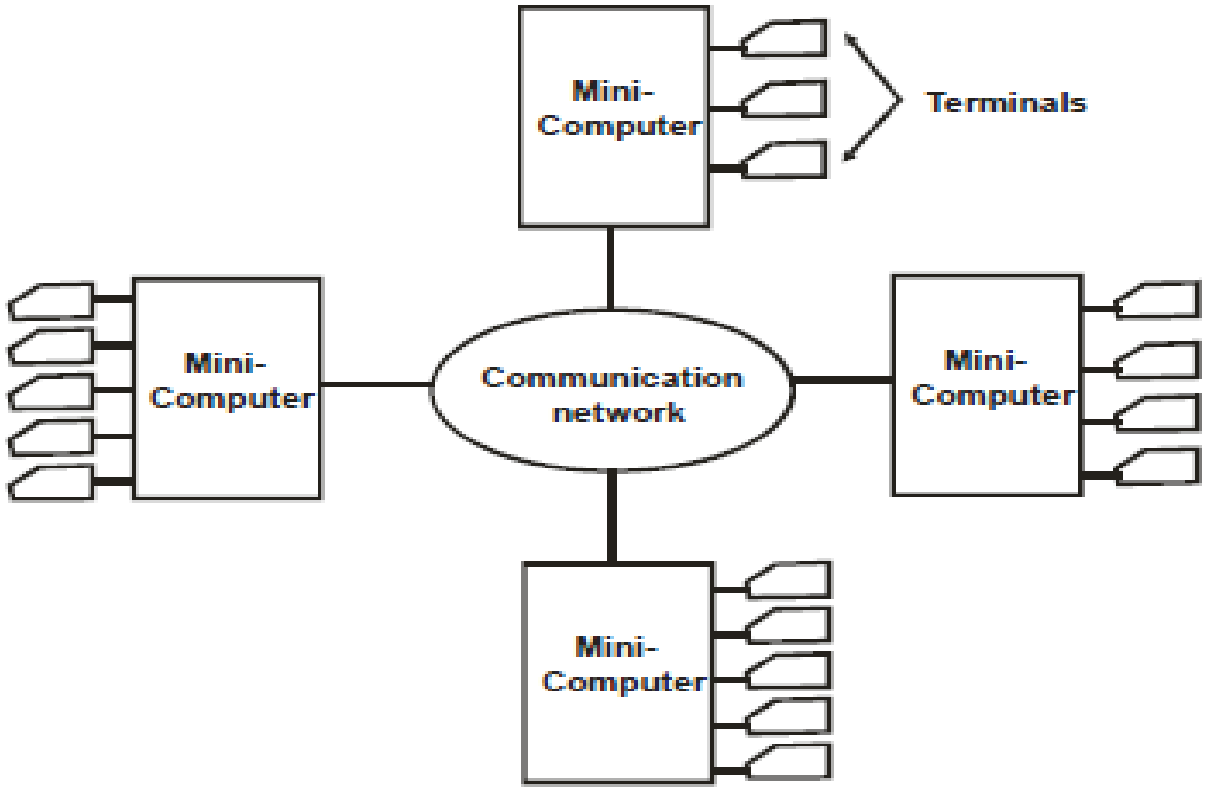
Minicomputer model

- The minicomputer 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 user is currently logged.

Cont.,

- The minicomputer model may be used when resource sharing with remote users is desired.
- The early ARPAnet is an example of a distributed computing systems based on the minicomputer model.

Cont.,


Mini-
Computer
Terminals
Mini-
Computer
Communication
network
Mini-
Computer
Mini-
Computer

Workstation model

- 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 on to other workstations and do not have sufficient processing power at their own workstations to get their jobs processed efficiently.
-

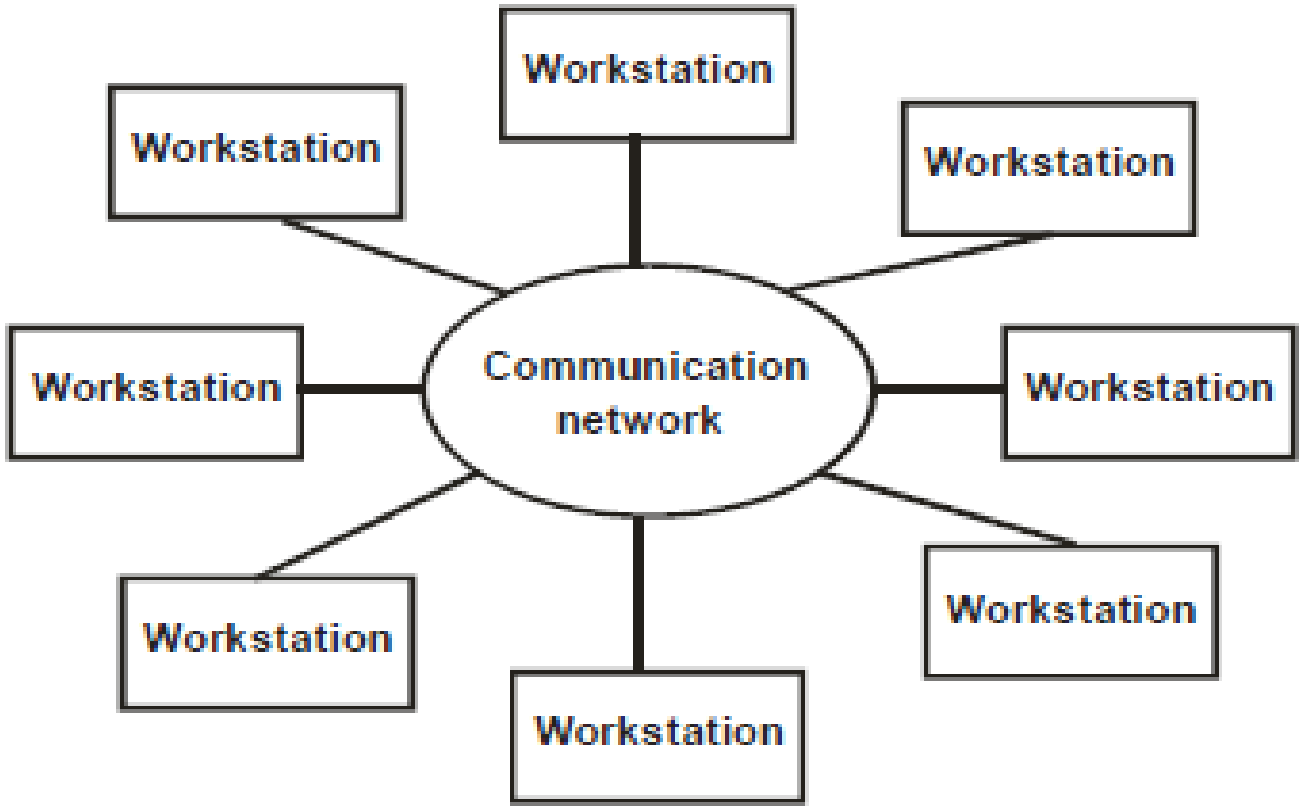
Cont.,

- 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.
- This model is not so simple to implement as it might appear at first sight because several issues must be resolved. These issues are
- How does the system find an idle workstation?
- How is a process transferred from one workstation to get it executed on another workstation?
- What happens to a remote process if a user logs on to a workstation that was idle until now and was being used to execute a process of another workstation?

Cont.,

- Three commonly used approaches for handling the third issue are as follows:
  - The first approach is to allow the remote process share the resources of the workstation along with its own logged-on user's processes. This method is easy to implement, but it defeats the main idea of workstations serving as personal computers.
  - The second approach is to kill the remote process. The main drawbacks of this method are that all processing done in the remote process gets lost and the file system may be left in an inconsistent state, making this method unattractive.
  - The third approach is to migrate the remote process back to its home workstation, so that its execution can be continued there. This method is difficult to implement because it requires the system to support preemptive process migration facility.
- 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.

Cont.,


Workstation
Workstation
Workstation
Workstation
Communication network
Workstation
Workstation
Workstation
Workstation

Workstation-Server Model

- 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, we need new model called workstation server model.
- In this model, a user logs on to a workstation called his or her home workstation. Normal computation activities required by the user's processes are preformed 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.

Cont.,

- 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 the virtual - memory management, and changing of remotely accessed data.

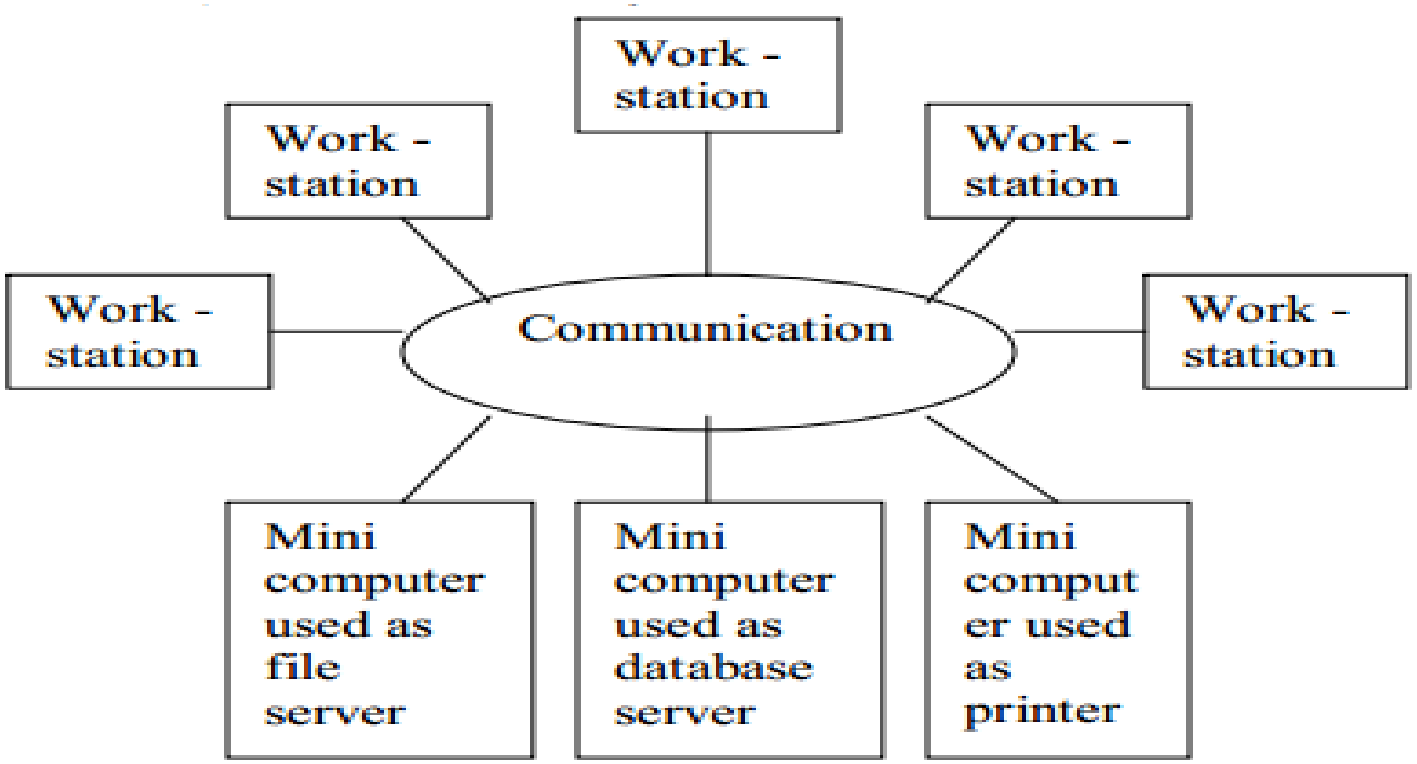
Cont.,

- As compared to the workstation model, the workstation server model has several advantages:
- 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.
- 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 larger disks than with many small disks scattered all over a building or campus. Furthermore, installing new releases of software is easier when the software is to be installed on a few file server machines than on every workstation.
- 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.
-

Cont.,

- 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 sends a request to a server process for getting some service such as 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.
- 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.

Cont.,


Work - station
Work - station
Work - station
Work - station
Communication
Work - station
Mini computer used as file server
Mini computer used as database server
Mini comput er used as printer

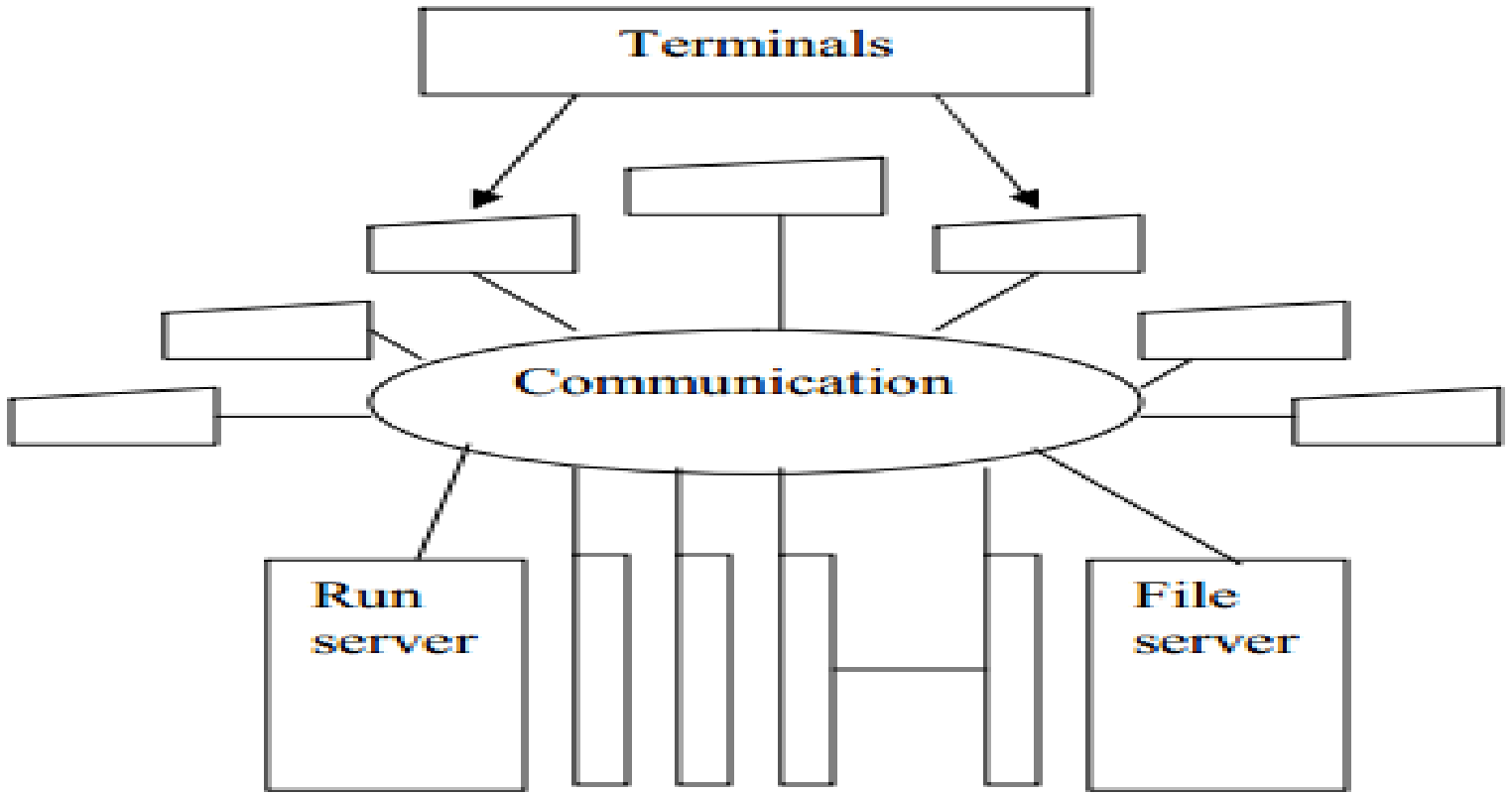
Processor-Pool Model

- 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.

Cont.,

- As shown in the figure, 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.
- 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 the system as a whole.

Cont.,



Hybrid Model

- Out of the four models described above, the workstation server model, is the most widely used models 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 continue the advantages of both the workstation server and processor pool models, a hybrid model may be used to build a distributed computing system.

Cont.,

- 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 requires 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?

- It is obvious that distributed computing systems are much more complex and difficult to build that traditional centralized systems.
- The increased complexity is mainly due to the fact that in addition of 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.
- Despite the increased complexity and the difficulty of building distributed computing systems are rapidly increasing.

Cont.,

- This is mainly because the advantages of distributed computing systems outweigh their disadvantages.
- The following are the reasons for the popularity of Distributed Computing Systems
  - Inherently Distributed Applications
  - Information Sharing among Distributed Users
  - Resource Sharing
  - Better Price Performance Ratio
  - Shorter Response Times and Higher Throughput
  - Higher Reliability
  - Extensibility and Incremental Growth
  - Better Flexibility in Meeting Users' Needs

Cont.,

- Inherently Distributed Applications

Distributed computing systems come into existence in some very natural ways.

(For example, several applications are inherently distributed in nature and require a distributed computing system)

Cont.,

Information Sharing among Distributed Users

In a distributed computing system, the users working at other nodes of a the system can easily and efficiently share information generated by one of the users.

(The use of distributed computing systems by a group of users to work cooperatively is known as groupware.)

Cont.,

- Resource Sharing

  Information is not the only thing that can be shared in a distributed computing system.

  (Sharing of software resources such as software libraries and databases as well as hardware resources such as printers, hard disks, and plotters can also be done in a very effective way among all the computers and the users of a single distributed computing system)

Cont.,

- Better Price Performance Ratio
  - With the rapidly increasing power and reduction in the price of microprocessors, combined with the increasing speed of communication networks, distributed computing systems potentially have a much better price-performance ratio than a single large centralized system.

Cont.,

- Shorter Response Times and Higher Throughput
  - Due to multiplicity of processors, distributed computing systems are expected to have better performance than single-processor centralized systems.
  - The two most commonly used performance metrics are response time and throughput of user processes.

Cont.,

- Higher Reliability
  - Reliability refers to the degree of tolerance against errors and component failures in a system.
  - A reliable system prevents loss of information even in the event of component failures.
  - The multiplicity of storage devices and processors in a distributed computing system allows the maintenance of multiple copies of critical information within the system and the execution of important computations redundantly to protect them against catastrophic failures.
  - An important aspect of reliability is availability, which refers to the fraction of time form which a system is available for use.

Cont.,

- Extensibility and Incremental Growth
  - Another major advantages of distributed computing systems is that they are capable of incremental growth.
  - (That is, it is possible to gradually extend the power and functionality of a distributed computing system by simply adding additional resources to the system as an when the need arises)

Cont.,

- Better Flexibility in Meeting Users' Needs
  - A distributed system may have a pool of different types of computers, in which case the most appropriate one can be selected for processing a user's job depending on the nature of the job.

1.5-WHAT IS A DISTRIBUTED OPERATING SYSTEM?

- An operating system is a program that controls the resources of a computer system and provides its users with an interface or virtual machine that is more convenient to use than the bare machine. According to this definition, the two primary tasks of an operating system are as follows
  - To present users with a virtual machine that is easier to program than the underlying hardware.
  - To manage the various resources of the system. This involves performing such tasks as keeping track of who is using which resource, granting resource requests, accounting for resource usage, and mediating conflicting requests from different programs and users.

Cont.,

The operating systems commonly used for distributed computing systems can be broadly classified into two types-network operating systems and distributed operating systems. The three most important features commonly used to differentiate between these two types of operating systems are

System image

Autonomy

Fault Tolerance capability.

Cont.,

- **System image:**
  - The most important feature used to differentiate between the two types of operating systems is the image of the distributed computing system from the point of view of its users.
  - In case of a network operating system, the users view the distributed computing system as a collection of distinct machines connected by a communication subsystem.
  - A distributed operating system hides the existence of multiple computers and provides a single- system image to its users. That is, it makes a collection of networked machines act as a virtual uniprocessor.

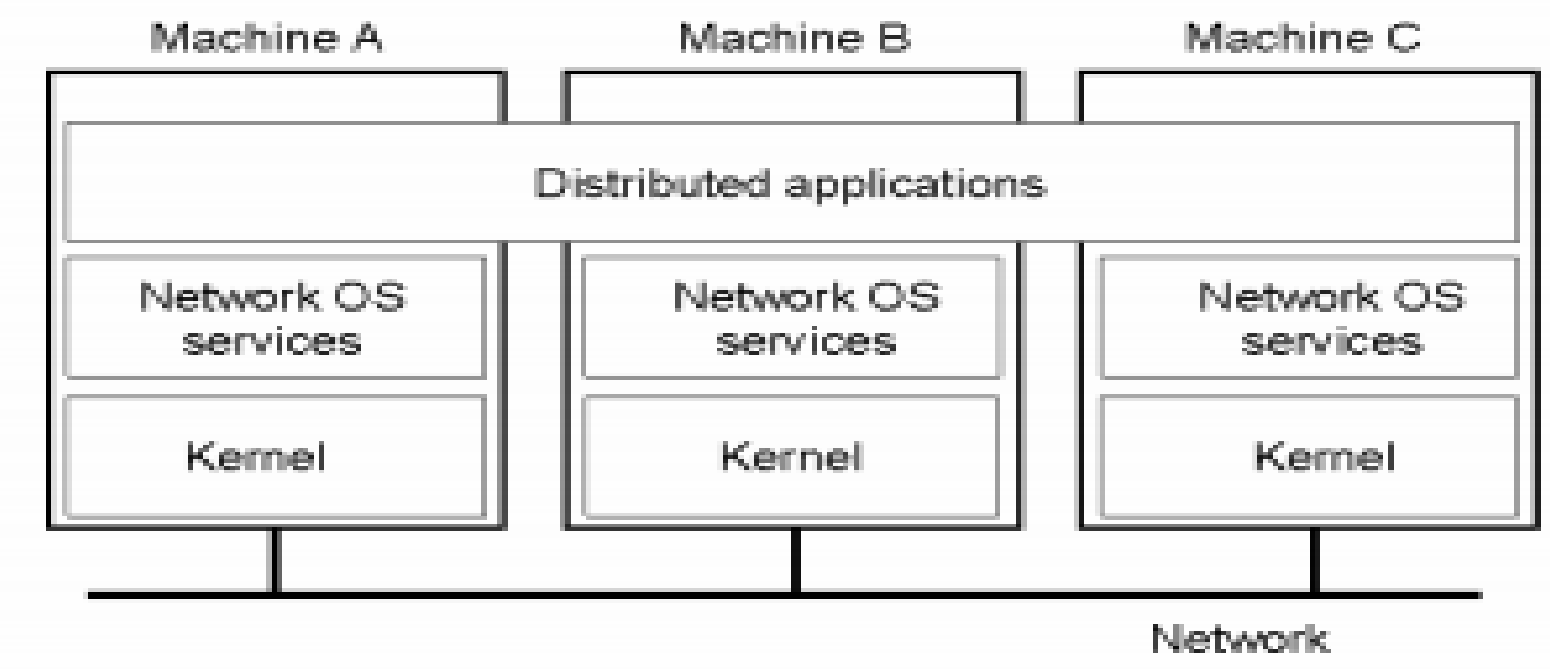
Cont.,

- **Autonomy:**
  - In the case of a network operating system, each computer of the distributed computing system has its own local operating system and there is essentially no coordination at all among the computers except for the rule that when two processes on different computers communicate with each other, they must use a mutually agreed on communication protocol.
  - With a distributed operating system, there is a single system wide operating system and each computer of the distributed computing system runs a part of this global operating system. The distributed operating system tightly interweaves all the computers of the distributed computing system in the sense that they work in close cooperation with each other for the efficient and effective utilization of the various resources of the system.

Cont.,

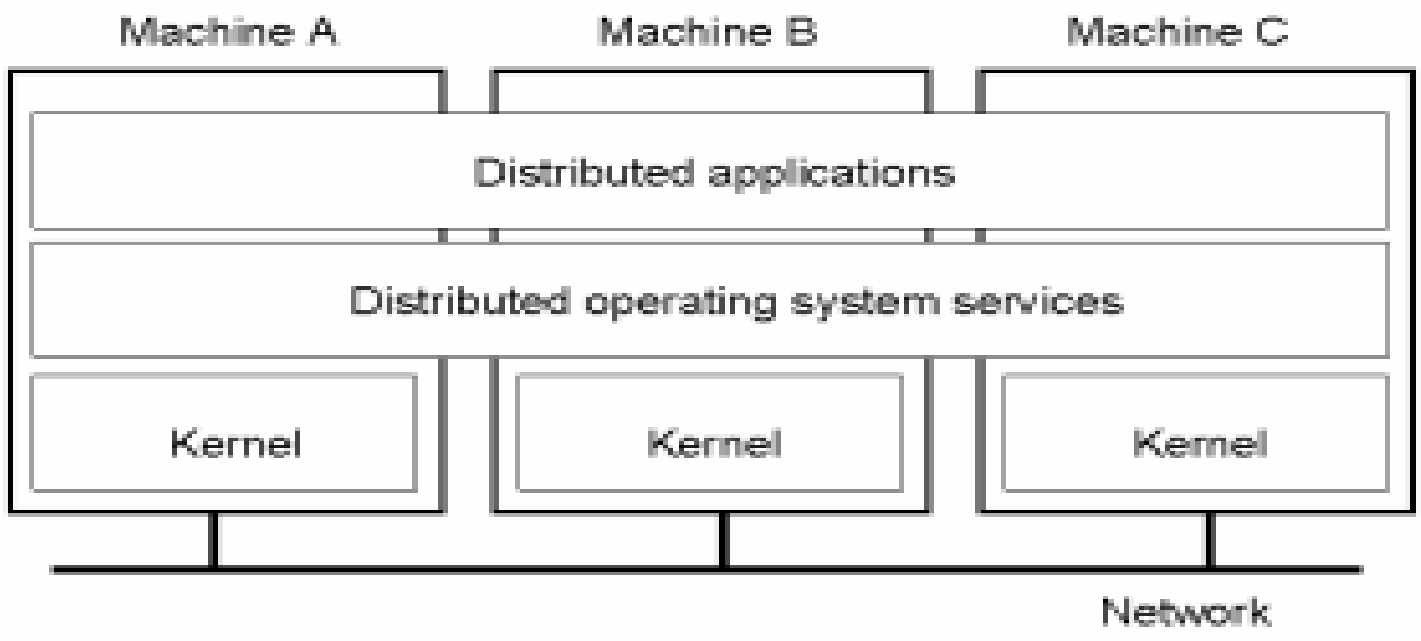
- **Fault tolerance capability:**
  - A network operating system provides little or no fault tolerance capability in the sense that if 10% of the machines of the entire distributed computing system are down at any moment, at least 10% of the users are unable to continue with their work.
  - n the other hand, with a distributed operating system, most of the users are normally unaffected by the failed machines and can continue to perform their work normally, with only a 10% loss in performance of the entire distributed computing system. Therefore, the fault tolerance capability of a distributed operating system is usually very high as compared to that of a network operating system.

Cont.,



Network Operating System

Cont.,



Distributed Operating System

1.6-ISSUES IN DESIGNING A DISTRIBUTED OPERATING SYSTEM

In general, designing a distributed operating system is more difficult than designing a centralized operating system for several reasons.

- The reasons are

- Transparency
- Reliability
- Flexibility
- Performance
- Scalability
- Heterogeneity
- Security
- Emulation of Existing Operating Systems

Cont.,

- **Transparency**
  - The main goals of a distributed operating system are to make the existence of multiple computers invisible (transparent) and provide a single system image to its users.
  - That is, a distributed operating system must be designed in such a way that a collection of distinct machines connected by a communication subsystem appears to its users as a virtual uniprocessor.
  - Achieving complete transparency is a difficult task and requires that several different aspects of transparency be supported by the distributed operating system.

Cont.,

The eight forms of transparency identified by the International Standards Organization's Reference Model for Open Distributed Processing [ISO 1992] are

- Access Transparency
- Location Transparency
- Replication Transparency
- Failure Transparency
- Migration Transparency
- Concurrency Transparency
- Performance Transparency
- Scaling Transparency.

Cont.,

- **Access Transparency:**
  - Access transparency means that users should not need or be able to recognize whether a resource (hardware or software) is remote or local. This implies that the distributed operating system should allow users to access remote resources in the same way as local resources.
- **Location Transparency**
  - The two main aspects of location transparency are as follows:
    - **Name transparency.** This refers to the fact that the name of a resource (hardware or software) should not reveal any hint as to the physical location of the resource.
    - **User mobility.** This refers to the fact that no matter which machine a user is logged onto, he or she should be able to access a resource with the same name.

Cont.,

- **Replication Transparency**
  - For better performance and reliability, almost all distributed operating systems have the provision to create replicas (additional copies) of files and other resources on different nodes of the distributed system. In these systems, both the existence of multiple copies of a replicated resource and the replication activity should be transparent to the users.

Cont.,

- **Failure Transparency**
  - Failure transparency deals with masking from the users' partial failures in the system, such as a communication link failure, a machine failure, or a storage device crash. A distributed operating system having failure transparency property will continue to function, perhaps in a degraded form, in the face of partial failures.

Migration Transparency

- For better performance, reliability, and security reasons, an object that is capable of being moved (such as a process or a file) often migrate from one node to another in a distributed system.
- The aim of migration transparency is to ensure that the movement of the object is handled automatically by the system in a user transparent manner.

Cont.,

- **Concurrency Transparency**
  - Concurrency transparency means that each user has a feeling that he or she is the sole user of the system and other users do not exist in the system.
  - For providing concurrency transparency, the following properties should be ensured
- Event ordering property
- Mutual exclusion property
- No starvation property
- No deadlock property

Cont.,

- **Performance Transparency**
  - The aim of performance transparency is to allow the system to be automatically reconfigured to improve performance, as loads vary dynamically in the system.
- **Scaling Transparency**
  - The aim of scaling transparency is to allow the system to expand in scale without disrupting the activities of the users.

Cont.,

- **Reliability**
  - In general, 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.

Cont.,

- A fault is a mechanical or algorithmic defect that may generate an error. A fault in a system causes system failure. Depending on the manner in which a failed system behaves, system failures are of two types: fail-stop failure and Byzantine failure.
- 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 fault avoidance and fault tolerance.

Cont.,

Fault Avoidance: Deals with designing the component of the system in such a way that the occurrence of faults is minimized.

Fault Tolerance: Ability of a system to continue functioning in the event of partial system failure. The following facts are used to improve fault tolerance ability

- Redundancy techniques
- Distributed control

Fault Detection and Recovery: Deals with the use of hardware and software mechanisms to determine the occurrence of the failure and then to correct the system to a state acceptable for continued operation. The commonly used techniques are

- Atomic Transactions
- Stateless Servers
- Acknowledgements and timeout-based retransmissions of messages.

Cont.,

- **Flexibility**
  - Another important issue in the design of distributed operating systems is flexibility.
  - Flexibility is the most important feature for open distributed systems. The design of a distributed operating system should be flexible due to the following reasons:
    - Ease of modification.
    - Ease of enhancement

Cont.,

- **Performance**
  - If a distributed system is to be used, its performance must be at least as good as a centralized system.
  - That is, when a particular application is run on a distributed system, its overall performance should be better than or at least equal to that of running the same application on a single- processor system.

Cont.,

- To achieve this goal, it is important that the various components of the operating system of a distributed system be designed properly; otherwise, the overall performance of the distributed system may turn out to be worse than a centralized system. Some design principles considered useful for better performance are as follows:
  - Batch if possible.
  - Cache whenever possible.
  - Minimize copying of data.
  - Minimize network traffic.
  - Take advantage of fine-grain parallelism for multiprocessing.

- **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 sub network 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. Some guiding principles for designing scalable distributed systems are as follows:
  - Avoid centralized entities.
  - Avoid centralized algorithms.
  - Perform most operations on client workstations.

- **Heterogeneity**
  - A heterogeneous distributed system consists of interconnected sets of dissimilar hardware or software systems.
  - Because of the diversity, designing heterogeneous distributed systems is far more difficult than designing homogeneous distributed systems in which each system is based on the same, or closely related, hardware and software.
  - However, as a consequence of large scale, heterogeneity is often inevitable in distributed systems. Furthermore, often heterogeneity is preferred by many users because heterogeneous distributed systems provide the flexibility to their users of different computer platforms for different applications.

- **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 more difficult than in a centralized system because of the lack of a single point of control and the use of insecure networks for data communication.

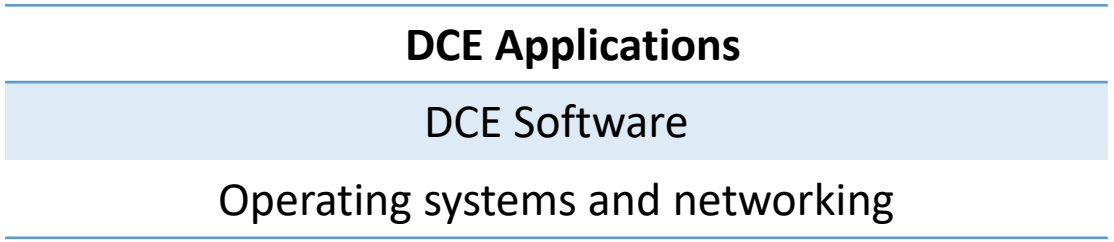
- Therefore, as compared to a centralized system, enforcement of security in a distributed system has the following additional requirements:
    - It should be possible for the sender of a message to know that the message was received by the intended receiver.
    - It should be possible for the receiver of a message to know that the message was sent by the genuine sender.
    - It should be possible for both the sender and receiver of a message to be guaranteed that the contents of the message were not changed while it was in transfer.
  - Cryptography is the only known practical method for dealing with these security aspects of a distributed system.

- **Emulation of Existing Operating System**
  - For commercial success, it is important that a newly designed distributed operating system be able to emulate existing popular operating systems such as UNIX.
  - With this property, new software can be written using the system call interface of the new operating system to take full advantage of its special features of distribution, but a vast amount of already existing old software can also be run on the same system without the need to rewrite them.
  - Therefore, moving to the new distributed operating system will allow both types of software to be run side by side.

1.7- Introduction to Distributed Computing Environment

- The Open Software Foundation, a consortium of computer manufacturers, including IBM, DEC, and Hewlett-Packard, defined DCE.
- It is not an operating system, nor is it an application.
- Rather, it is an integrated set of services and tools that can be installed as a coherent environment on top of existing operating systems and serve as a platform for building and running distributed applications.
- A primary goal of DCE is vendor independence.

- It runs on many different kinds of computers, operating systems and networks produced by different vendors.
- DCE is a middleware software layered between the DCE applications layer and the operating system and networking layer.
- The basic idea is to take a collection of existing machines. Interconnect them by a communication network add the DCE software platform on top of the native operating systems of the machines, and then be able to build and run distributed applications.



Position of DCE software in a DCE based distributed system

- DCE Components
- DCE is a blend of various technologies developed independently and nicely integrated by OSF.
- Each of these technologies forms a component of DCE.
- The main Components of DCE are as follows:

- References
- Pradeep K Sinha, “ Distributed Operating Systems – Concepts and Design”, PHI, 2013.


THANK
YOU