➢ Distributed System Models

❖ Physical Model

A physical model is basically a representation of the underlying hardware elements of a distributed system. It encompasses the hardware composition of a distributed system in terms of computers and other devices and their interconnections. It is primarily used to design, manage, implement and determine the performance of a distributed system.

A physical model majorly consists of the following components:

- Nodes
- Links
- Middleware
- Network topology
- Communication Protocols

Nodes

Nodes are the end devices that have the ability of processing data, executing tasks and communicating with the other nodes.

These end devices are generally the computers at the user end or can be servers, workstations etc.

Nodes provision the distributed system with an interface in the presentation layer that enables the user to interact with other back-end devices, or nodes, that can be used for storage and database services, or processing, web browsing etc.

Each node has an Operating System, execution environment and different middleware requirements that facilitate communication and other vital tasks.

Links

Links are the communication channels between different nodes and intermediate devices.

These may be wired or wireless.

Wired links or physical media are implemented using copper wires, fiber optic cables etc.

The choice of the medium depends on the environmental conditions and the requirements.

Generally, physical links are required for high performance and real-time computing.

Different connection types that can be implemented are as follows:

Point-to-point links – It establishes a connection and allows data transfer between only two nodes.

Broadcast links – It enables a single node to transmit data to multiple nodes simultaneously.

Multi-Access links – Multiple nodes share the same communication channel to transfer data. Requires protocols to avoid interference while transmission.

Middleware

These are the softwares installed and executed on the nodes.

By running middleware on each node, the distributed computing system achieves a decentralized control and decision-making.

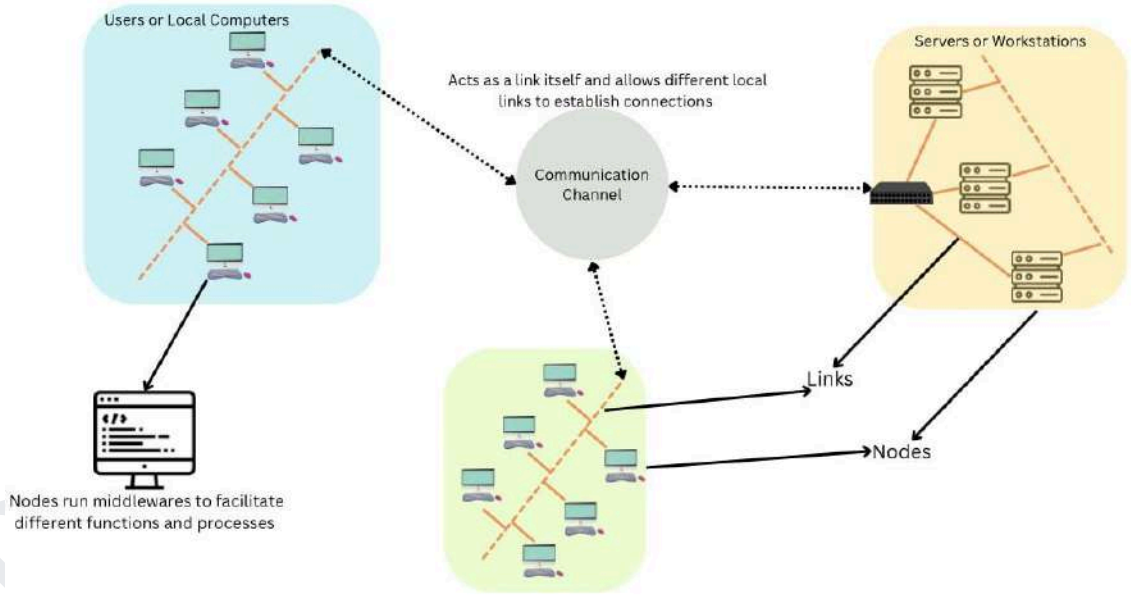
It handles various tasks like communication with other nodes, resource management, fault tolerance, synchronization of different nodes and security to prevent malicious and unauthorized access.

Network Topology

This defines the arrangement of nodes and links in the distributed computing system. The most common network topologies that are implemented are bus, star, mesh, ring or hybrid. Choice of topology is done by determining the exact use cases and the requirements.

Communication Protocols

Communication protocols are the set rules and procedures for transmitting data from in the links. Examples of these protocols include TCP, UDP, HTTPS, MQTT etc. These allow the nodes to communicate and interpret the data.



❖ Architectural Model

Architectural model in a distributed computing system is the overall design and structure of the system, and how its different components are organized to interact with each other and provide the desired functionalities.

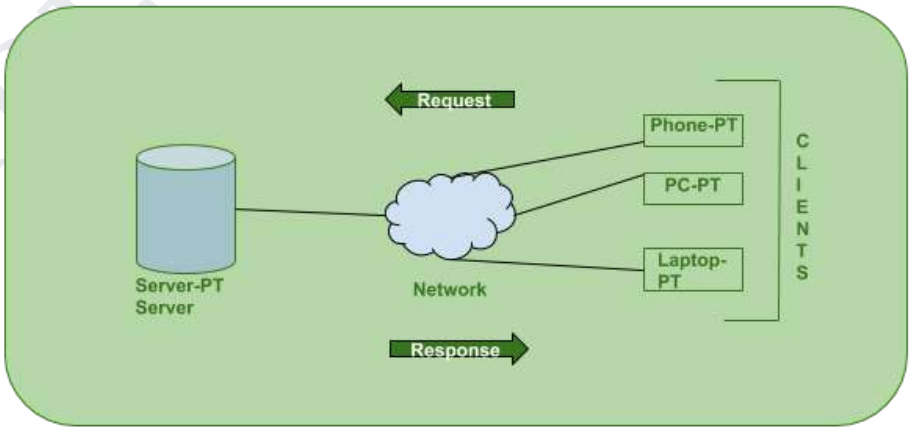
It is an overview of the system, on how the development, deployment and operations take place.

Construction of a good architectural model is required for efficient cost usage, and highly improved scalability of the applications.

The key aspects of architectural model are

- Client server model
- Peer to peer model
- Layered model
- Micro services model

Client Server Model



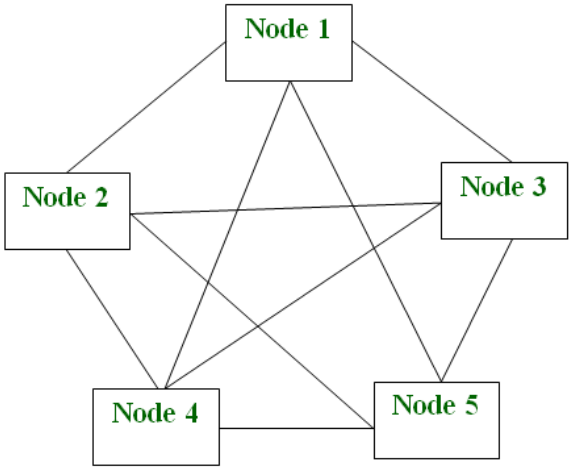
It is a centralized approach in which the clients initiate requests for services and servers respond by providing those services.

It mainly works on the request-response model where the client sends a request to the server and the server processes it, and responds to the client accordingly.

It can be achieved by using TCP/IP, HTTP protocols on the transport layer.

This is mainly used in web services, cloud computing, database management systems etc.

Peer to Peer Model



P2P Architecture

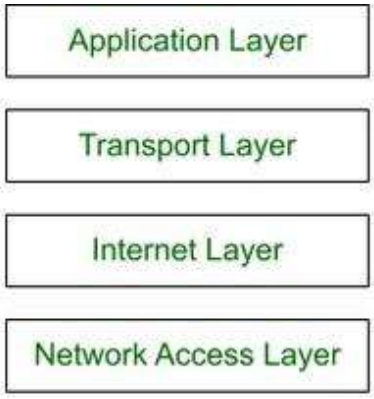
It is a decentralized approach in which all the distributed computing nodes, known as peers, are all the same in terms of computing capabilities and can both request as well as provide services to other peers.

It is a highly scalable model because the peers can join and leave the system dynamically, which makes it an ad-hoc form of network.

The resources are distributed and the peers need to look out for the required resources as and when required.

The communication is directly done amongst the peers without any intermediaries according to some set rules and procedures defined in the P2P networks.

Layered Model



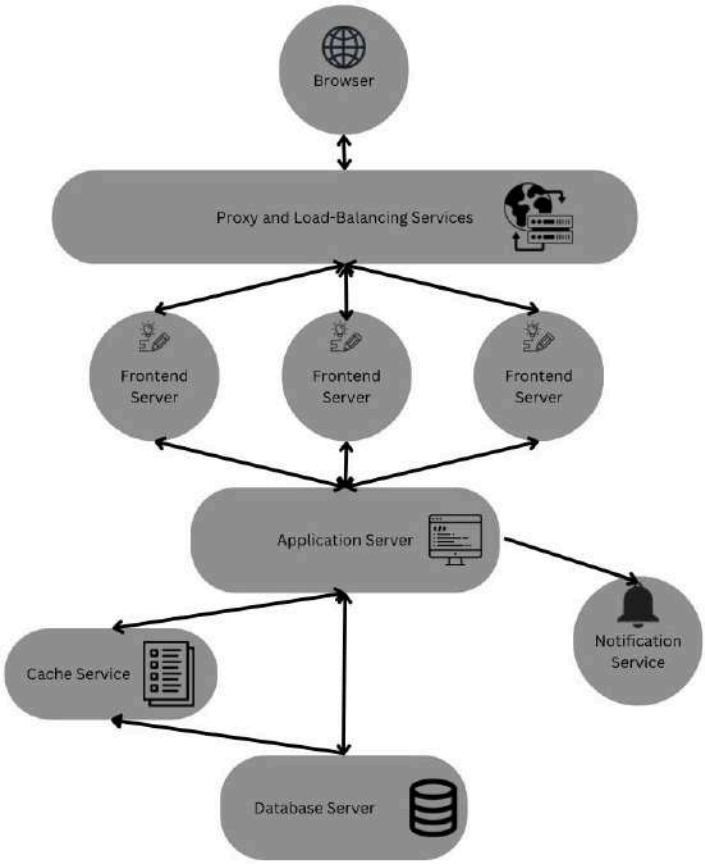
Various Layers of the TCP/ IP Model

It involves organizing the system into multiple layers, where each layer will provision a specific service.

Each layer communicated with the adjacent layers using certain well-defined protocols without affecting the integrity of the system.

A hierarchical structure is obtained where each layer abstracts the underlying complexity of lower layers.

Micro Services Model



In this system, a complex application or task, is decomposed into multiple independent tasks and these services running on different servers.

Each service performs only a single function and is focussed on a specific business-capability.

This makes the overall system more maintainable, scalable and easier to understand.

Services can be independently developed, deployed and scaled without affecting the ongoing services.

❖ Fundamental Model

The fundamental model in a distributed computing system is a broad conceptual framework that helps in understanding the key aspects of the distributed systems.

These are concerned with more formal descriptions of properties that are generally common in all architectural models.

It represents the essential components that are required to understand a distributed system's behavior. Three fundamental models are as follows:

- Interaction Model
- Remote Procedure Call
- Failure Model
- Security Model

Interaction Model

Distributed computing systems are full of many processes interacting with each other in highly complex ways.

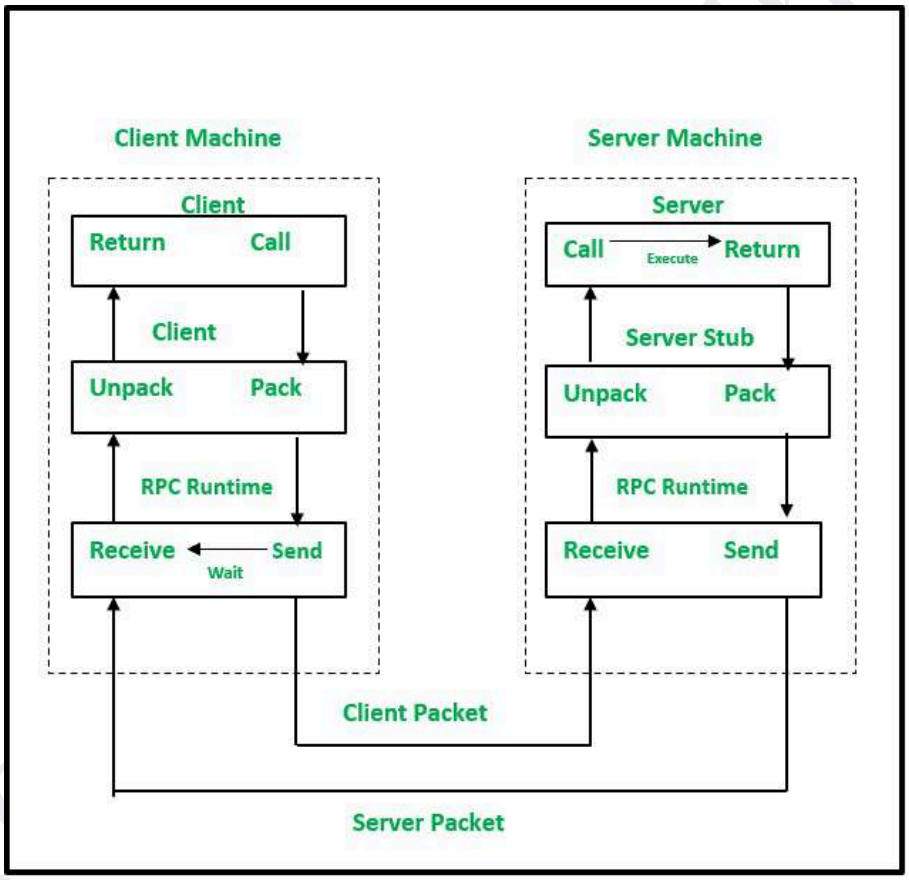
Interaction model provides a framework to understand the mechanisms and patterns that are used for communication and coordination among various processes.

Different components that are important in this model are –

Message Passing – It deals with passing messages that may contain data, instructions, a service request, or process synchronization between different computing nodes. It may be synchronous or asynchronous depending on the types of tasks and processes.

Publish/Subscribe Systems – In this the publishing process can publish a message over a topic and the processes that are subscribed to that topic can take it up and execute the process for themselves. It is more important in an event-driven architecture.

RPC



It is a communication paradigm that has an ability to invoke a new process or a method on a remote process as if it were a local procedure call.

The client process makes a procedure call using RPC and then the message is passed to the required server process using communication protocols.

These message passing protocols are abstracted and the result once obtained from the server process, is sent back to the client process to continue execution.

Failure Model

This model addresses the faults and failures that occur in the distributed computing system.

It provides a framework to identify and rectify the faults that occur or may occur in the system.

Fault tolerance mechanisms are implemented so as to handle failures by replication and error detection and recovery methods.

Different failures that may occur are:

- Crash failures – A process or node unexpectedly stops functioning.
- Omission failures – It involves a loss of message, resulting in absence of required communication.
- Timing failures – The process deviates from its expected time quantum and may lead to delays or unsynchronised response times.
- Byzantine failures – The process may send malicious or unexpected messages that conflict with the set protocols.

Security Model

Distributed computing systems may suffer malicious attacks, unauthorised access and data breaches.

Security model provides a framework for understanding the security requirements, threats, vulnerabilities, and mechanisms to safeguard the system and its resources.

Various aspects that are vital in the security model are –

Authentication – It verifies the identity of the users accessing the system. It ensures that only the authorized and trusted entities get access. It involves –

Password-based authentication – Users provide a unique password to prove their identity.

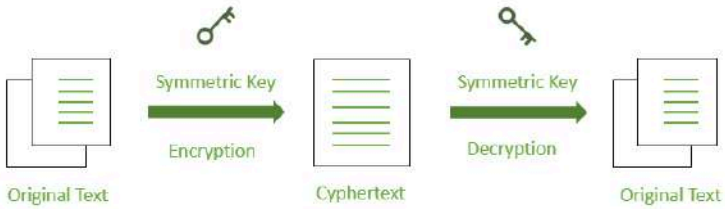
Public-key cryptography – Entities possess a private key and a corresponding public key, allowing verification of their authenticity.

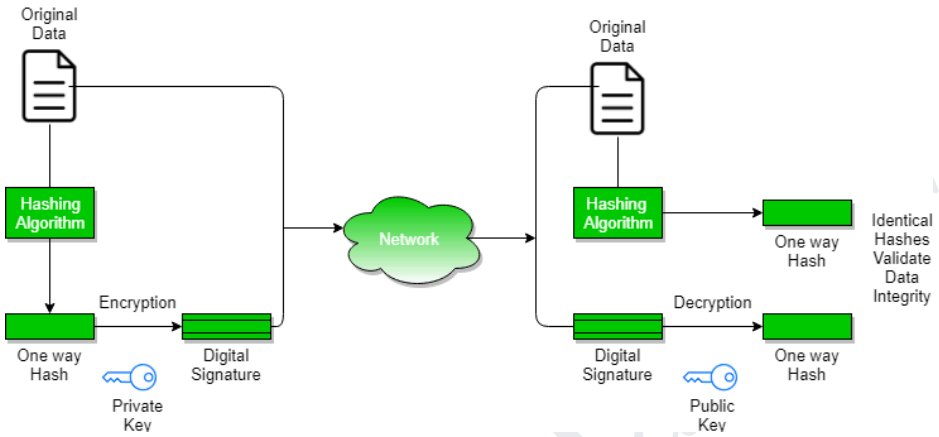
Multi-factor authentication – Multiple factors, such as passwords, biometrics, or security tokens, are used to validate identity.

Encryption – It is the process of transforming data into a format that is unreadable without a decryption key. It protects sensitive information from unauthorized access or disclosure.

Data Integrity – Data integrity mechanisms protect against unauthorized modifications or tampering of data. They ensure that data remains unchanged during storage, transmission, or processing. Data integrity mechanisms include:

- Hash functions – Generating a hash value or checksum from data to verify its integrity.
- Digital signatures – Using cryptographic techniques to sign data and verify its authenticity and integrity.



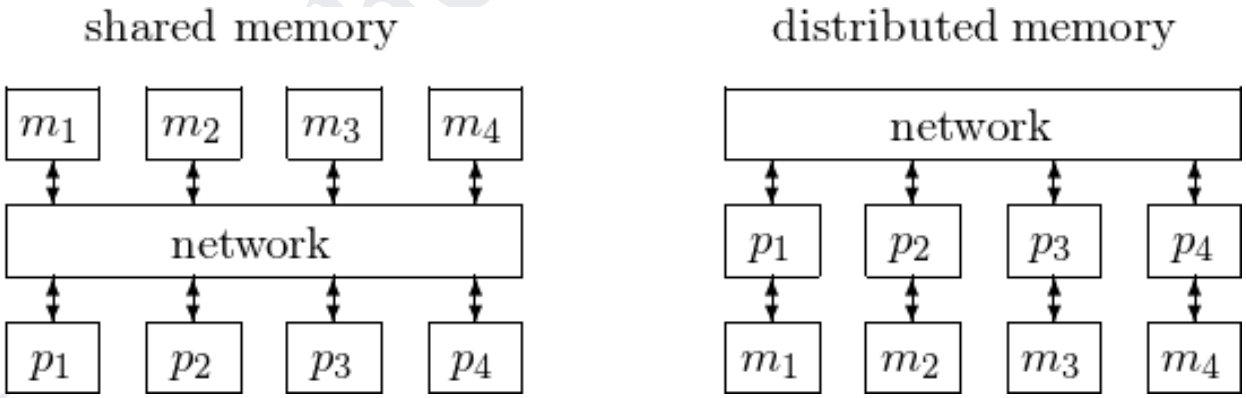


➢ Hardware concepts

Types:

1. Multiprocessor
2. Multicomputer

Multiprocessor: A Multiprocessor is a computer system with two or more central processing units (CPUs) sharing full access to a common RAM. The main objective of using a multiprocessor is to boost the system's execution speed, with other objectives being fault tolerance and application matching. There are two types of multiprocessors, one is called shared memory multiprocessor and another is distributed memory multiprocessor. In shared memory multiprocessors, all the CPUs share the common memory but in a distributed memory multiprocessor, every CPU has its own private memory.



Applications of Multiprocessor –

As a uniprocessor, such as single instruction, single data stream (SISD).

As a multiprocessor, such as single instruction, multiple data stream (SIMD), which is usually used for vector processing.

Multiple series of instructions in a single perspective, such as multiple instruction, single data stream (MISD), which is used for describing hyper-threading or pipelined processors.

Inside a single system for executing multiple, individual series of instructions in multiple perspectives, such as multiple instruction, multiple data stream (MIMD).

Benefits of using a Multiprocessor –

Enhanced performance.

Multiple applications.

Multi-tasking inside an application.

High throughput and responsiveness.

Hardware sharing among CPUs.

Advantages:

Improved performance: Multiprocessor systems can execute tasks faster than single-processor systems, as the workload can be distributed across multiple processors.

Better scalability: Multiprocessor systems can be scaled more easily than single-processor systems, as additional processors can be added to the system to handle increased workloads.

Increased reliability: Multiprocessor systems can continue to operate even if one processor fails, as the remaining processors can continue to execute tasks.

Reduced cost: Multiprocessor systems can be more cost-effective than building multiple single-processor systems to handle the same workload.

Enhanced parallelism: Multiprocessor systems allow for greater parallelism, as different processors can execute different tasks simultaneously.

Disadvantages:

Increased complexity: Multiprocessor systems are more complex than single-processor systems, and they require additional hardware, software, and management resources.

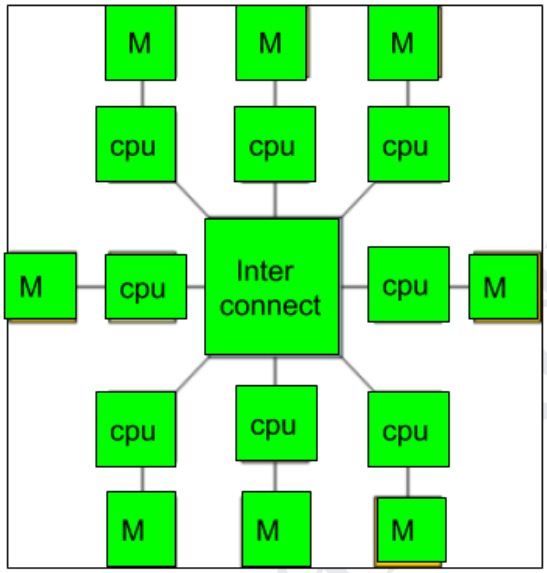
Higher power consumption: Multiprocessor systems require more power to operate than single-processor systems, which can increase the cost of operating and maintaining the system.

Difficult programming: Developing software that can effectively utilize multiple processors can be challenging, and it requires specialized programming skills.

Synchronization issues: Multiprocessor systems require synchronization between processors to ensure that tasks are executed correctly and efficiently, which can add complexity and overhead to the system.

Limited performance gains: Not all applications can benefit from multiprocessor systems, and some applications may only see limited performance gains when running on a multiprocessor system.

Multicomputer: A multicomputer system is a computer system with multiple processors that are connected together to solve a problem. Each processor has its own memory and it is accessible by that particular processor and those processors can communicate with each other via an interconnection network.



As the multicomputer is capable of messages passing between the processors, it is possible to divide the task between the processors to complete the task. Hence, a multicomputer can be used for distributed computing. It is cost effective and easier to build a multicomputer than a multiprocessor.

Difference between multiprocessor and Multicomputer:

Multiprocessor is a system with two or more central processing units (CPUs) that is capable of performing multiple tasks whereas a multicomputer is a system with multiple processors that are attached via an interconnection network to perform a computation task.

A multiprocessor system is a single computer that operates with multiple CPUs whereas a multicomputer system is a cluster of computers that operate as a singular computer.

Construction of a multicomputer is easier and cost effective than a multiprocessor.

In a multiprocessor system, the program tends to be easier whereas in a multicomputer system, the program tends to be more difficult.

Multiprocessor supports parallel computing, Multicomputer supports distributed computing.

Advantages:

Improved performance: Multicomputer systems can execute tasks faster than single-computer systems, as the workload can be distributed across multiple computers.

Better scalability: Multicomputer systems can be scaled more easily than single-computer systems, as additional computers can be added to the system to handle increased workloads.

Increased reliability: Multicomputer systems can continue to operate even if one computer fails, as the remaining computers can continue to execute tasks.

Reduced cost: Multicomputer systems can be more cost-effective than building a single large computer system to handle the same workload.

Enhanced parallelism: Multicomputer systems allow for greater parallelism, as different computers can execute different tasks simultaneously.

Disadvantages:

Increased complexity: Multicomputer systems are more complex than single-computer systems, and they require additional hardware, software, and management resources

Higher power consumption: Multicomputer systems require more power to operate than single-computer systems, which can increase the cost of operating and maintaining the system.

Difficult programming: Developing software that can effectively utilize multiple computers can be challenging, and it requires specialized programming skills.

Synchronization issues: Multicomputer systems require synchronization between computers to ensure that tasks are executed correctly and efficiently, which can add complexity and overhead to the system.

Network latency: Multicomputer systems rely on a network to communicate between computers, and network latency can impact system performance.

➢ Software Concept

Used to provide interaction between a user and the hardware

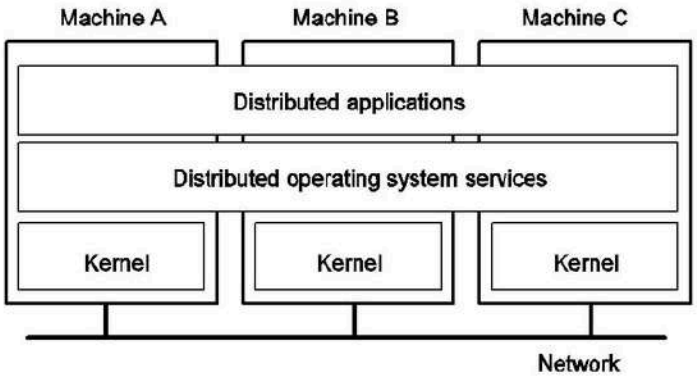
Types:

1. Distributed operating system
2. Network operating system
3. middleware

Distributed operating system

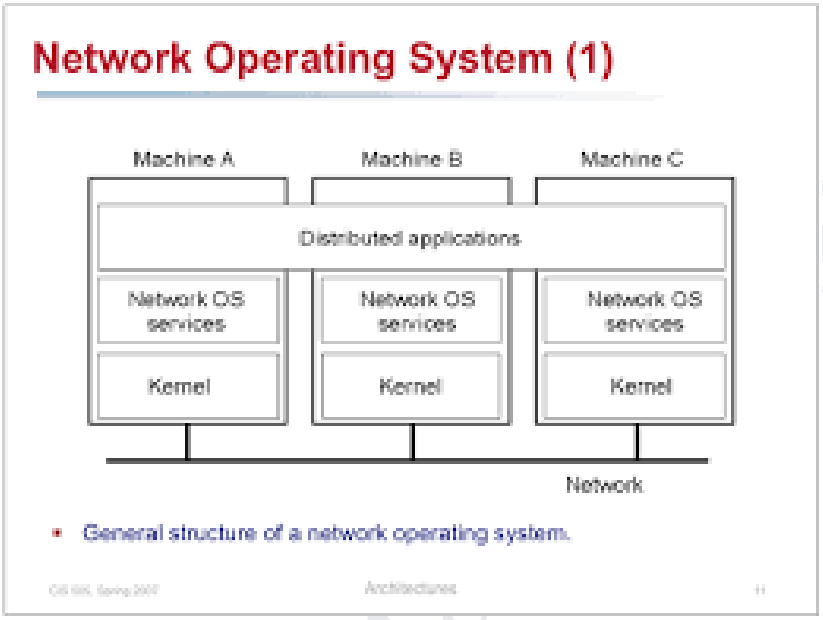
Basic type of OS. It is DS that abstracts resources, such as memory or CPUs and exposes common services and primitives that in turn are used by applications. Through a single communication channel it links several computers. Each of the system has its own processor and memory CPUs may communicate through high speed buses It is also called a tightly coupled system.

Distributed Operating Systems (DOS)



Network Operating System

It is created specifically to handle workstations, database sharing, application sharing, file and printer access sharing and other network wide computer functions. Designed for heterogeneous computer systems. It has multiple OSes that are running on different hardware platforms. Also called as loosely coupled system



Middleware

It refers to software that offers additional services above and beyond those offered by the OS to allow data management and communication across the various distributed system components.

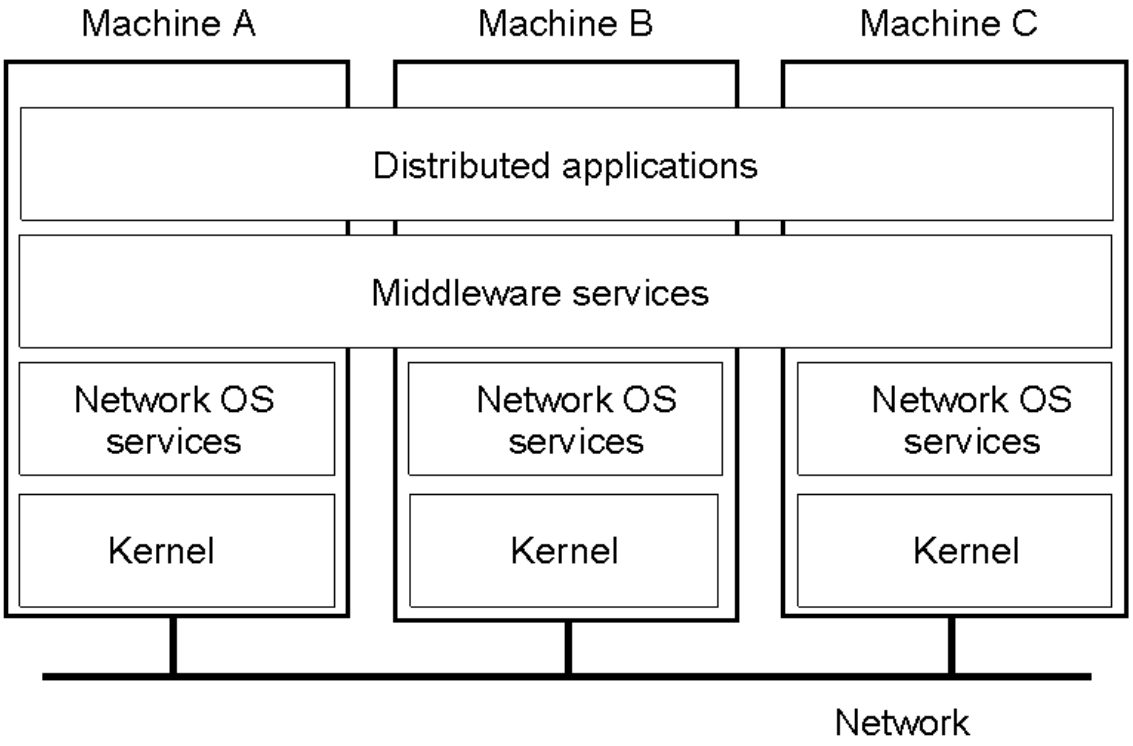
Complex distributed applications are supported and made easier by middleware

It enables interoperability between applications that run on different OS, by supplying services so that the application can exchange data in standard format.

It lies in between the application software that may be working on different OS

It comes in many forms, including database middleware, transactional middleware, intelligent middleware, message oriented middleware.

Provides a variety of services such as control services, communication services and security services.



It refers to software that offers additional services above and beyond those offered by the OS to allow data management and communication across the various distributed system components.

Complex distributed applications are supported and made easier by middleware

It enables interoperability between applications that run on different OS, by supplying services so that the application can exchange data in standard format.

It lies in between the application software that may be working on different OS

It comes in many forms, including database middleware, transactional middleware, intelligent middleware, message oriented middleware.

Provides a variety of services such as control services, communication services and security services.