
# DISTRIBUTED OPERATING SYSTEM

## CHAPTER OUTLINE

**After comprehensive study of this chapter, you will be able to:**

- Introduction, Advantages of Distributed System over Centralized System
- Advantages of Distributed System Over independent PCs. Disadvantages of Distributed System
- Hardware and Software Concepts,
- Communication in Distributed Systems, Message Passing, Remote Procedure Call, Process in Distribution System
- Clock Synchronization

## INTRODUCTION

A distributed operating system is an operating system that runs on several machines whose purpose is to provide a useful set of services, generally to make the collection of machines behave more like a single machine. The distributed operating system plays the same role in making the collective resources of the machines more usable that a typical single-machine operating system plays in making that machine's resources more usable. Usually, the machines controlled by a distributed operating system are connected by a relatively high quality network, such as a high speed local area network. Most commonly, the participating nodes of the system are in a relatively small geographical area, something between an office and a campus.

Distributed operating systems typically run cooperatively on all machines whose resources they control. These machines might be capable of independent operation, or they might be usable merely as resources in the distributed system. In some architectures, each machine is an equally powerful peer as all the others. In other architectures, some machines are permanently designated as master or are given control of particular resources. In yet others, elections or other selection mechanisms are used to designate some machines as having special roles, often controlling roles.

Distributed Operating System is a model where distributed applications are running on multiple computers linked by communications. A distributed operating system is an extension of the network operating system that supports higher levels of communication and integration of the machines on the network.This system looks to its users like an ordinary centralized operating system but runs on multiple, independent central processing units (CPUs).

Examples of distributed operating systems are:Windows server 2003, Windows server 2008, Windows server 2012, Ubuntu, Linux (Apache Server) etc.

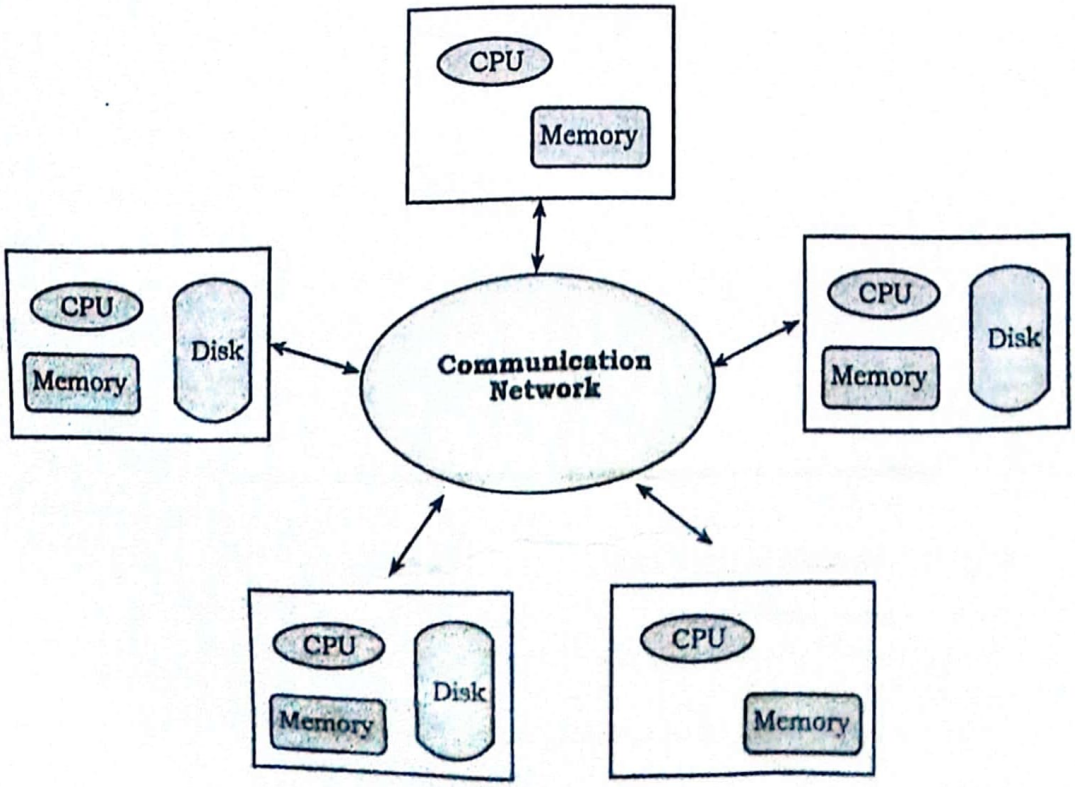



Fig 9.1: Distributed system

The reasons behind building such distributed systems are:

- **Resource sharing** – where several sites are connected with one another, shared resources provides different mechanisms for sharing files remotely, processing information in a distributed database, and performing many other operations.
- **Computation Speedup** – Where particular computations can be divided into sub-computations that can run concurrently among various sites; and load sharing are used to complete such tasks.
- **Reliability** – in a distributed system, if one site fails the remaining sites can continue operating to give the system the requested reliability. To achieve reliability in the distributed system, the failure of a site must be detected by the system, and actions must be implemented to recover from the failure.
- **Communication** – the distributed system communicate with one another via a communication network to provide users with the opportunity to exchange information.

**Advantages of distributed operating systems**

- Give more performance than single system
- If one pc in distributed system malfunction or corrupts then other node or pc will take care of the operation.
- More resources can be added easily
- Resources like printers can be shared on multiple pc's

**Disadvantages of distributed operating systems**

- Security problem due to sharing
- Some messages can be lost in the network system
- Bandwidth is another problem if there is large data then all network wires to be replaced which tends to become expensive
- Overloading is another problem in distributed operating systems
- If there is a database connected on local system and many users accessing that database through remote or distributed way then performance become slow
- The databases in network operating is difficult to administrate then single user system

**Advantages of Distributed System over independent PCs**

**Data sharing**

- Many users need to share data. For example, airline reservation clerks need access to the master data base of flights and existing reservations.Giving each clerk his own private copy of the entire database would not work, since nobody would know which seats of other clerks had already sold.
- Shared data are absolutely essential for this and many other applications, so the machines must be interconnected.

**Resource sharing**

- Expensive peripherals, such as color laser printers, phototypesetters and massive archival storage devices can also be shared.

**Communication**

- For many people, electronic mail has numerous attractions over paper mail, telephone and FAX.
- It is much faster than paper mail, does not require both parties to be available at the same time as does the telephone and unlike FAX, produces documents that can be edited, rearranged, stored in the computer and manipulated with text processing programs.

**Flexibility**

- A distributed system is more flexible than giving each user an isolated personal computer. One way is to give each person a personal computer and connect them all with a LAN, this is not the only possibility.
- Another one is to have a mixture of personal and shared computers, perhaps of different sizes, and let jobs run on the most appropriate one, rather than always on the owner's machine.
- In this way, the workload can be spread over the computers more effectively, and the loss of a few machines may be compensated for any letting people run their jobs elsewhere.

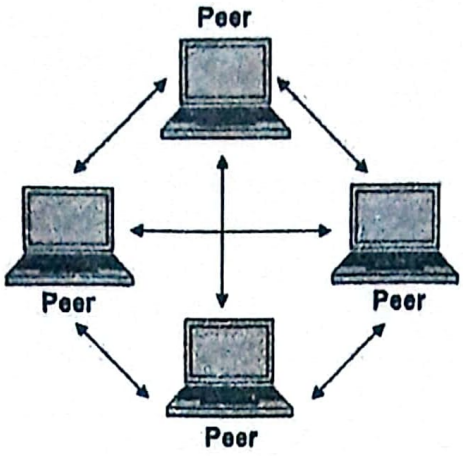
## NETWORK OPERATING SYSTEM

A network operating system (NOS) is a computer operating system (OS) that is designed primarily to support workstations, personal computers and, in some instances, older terminals that are connected on a local area network (LAN).A network operating system is an important category of the operating system that operates on a server using network devices like a switch, router, or firewall to handle data, applications and other network resources. It provides connectivity among the autonomous operating system, called as a network operating system. The network operating system is also useful to share data, files, hardware devices and printer resources among multiple computers to communicate with each other.Examples: MS Windows Server 2003, MS Windows Server 2008, NetWare, BSD etc.

**Types of network operating system**

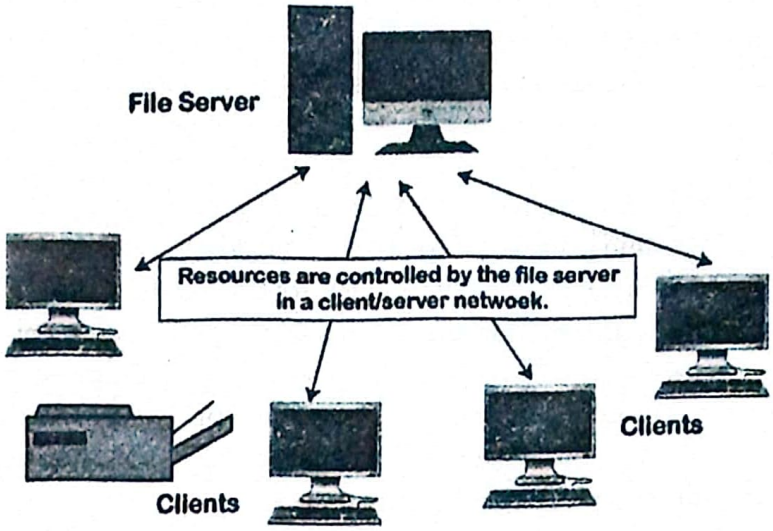
**Peer-to-peer network operating system**

The type of network operating system allows users to share files, resources between two ormore computer machines using a LAN.




**Client-Server network operating system**

It is the type of network operating system that allows the users to access resources, functions, and applications through a common server or center hub of the resources. The client workstation can access all resources that exist in the central hub of the network. Multiple clients can access and share different typesof the resource over the network from different locations.




**Advantages**

- The servers are highly stable and centralized
- Servers handle Security concerns
- New technologies and hardware up-gradation can be easily integrated
- Server access is possible remotely from various locations and different types of systems

**Disadvantages**

- Costly servers
- Depended on central location
- Regular updates and maintenance required

**Difference between network operating system and distributed operating system**

The main difference between these two operating systems (Network Operating System and Distributed Operating System) is that in network operating system each node or system can have its own operating system on the other hand in distribute operating system each node or system have same operating system which is opposite to the network operating system.

| Network operating system | Distributed operating system |
|---|---|
| Network Operating System's main objective is to provide the local services to remote client. | Distributed Operating System's main objective is to manage the hardware resources. |
| In Network Operating System, Communication takes place on the basis of files. | In Distributed Operating System, Communication takes place on the basis of messages and share memory. |
| Network Operating System is more sculable than Distributed Operating System. | Distributed Operating System is less scalable than Network Operating System. |
| In Network Operating System, fault tolerance is less. | While in Distributed Operating System, fault tolerance is high. |
| Rate of autonomy in Network Operating System is high. | While The rate of autonomy in Distributed Operating System is less. |
| Ease of implementation in Network Operating System is also high. | While in Distributed Operating System Ease of implementation is less. |
| In Network Operating System, All nodes can have different operating system. | While in Distributed Operating System, All nodes have same operating system. |

## HARDWARE AND SOFTWARE CONCEPTS

### Hardware Concept

All distributed systems consist of multiple CPUs. There are several different ways the hardware can be arranged. The important thing related to hardware is that how they are interconnected and how they communicate with each other. It is important to take a deep look at distributed system hardware, in particular, how the machines are connected together and how they interact. Many classification schemes for multiple CPU computer systems have been proposed over the years, but none of them have really implemented. Still, the most commonly used taxonomy is Flynn's, but it was in basic stage. In this scheme, Flynn took only two things to consider i.e. the number of **instruction streams** and **the number of data streams.**

**a. Single Instruction, Single Data Stream (SISD)**

A computer with a single instruction stream and a single data stream is called SISD. All traditional uni-processor computers (i.e., those having only one CPU) fall under this category. SISD flow concept is given in the figure below.

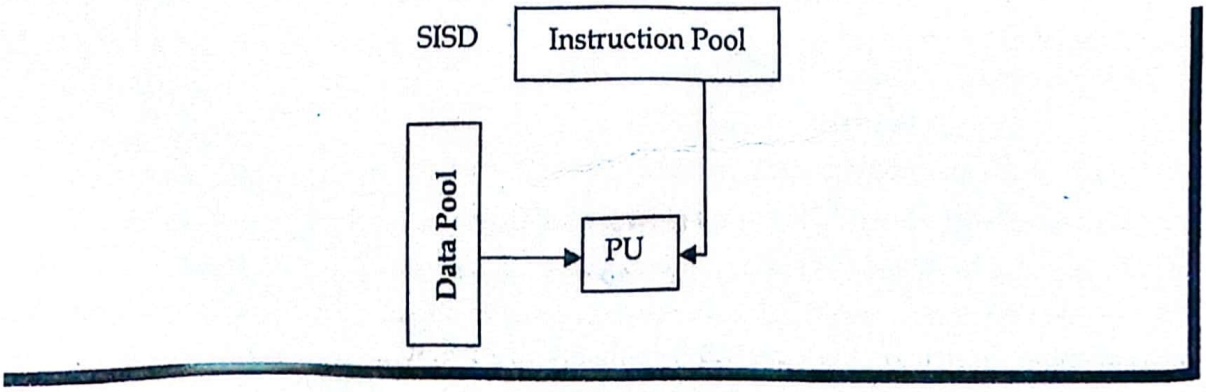



**Fig 9.2: SISD Flow structure**

### Single Instruction, Multiple Data Stream (SIMD)

The next category is SIMD, i.e. single instruction stream, multiple data stream. This type uses an array of processors with only one instruction unit that fetches an instruction, and multiple data units which work in parallel. These machines are used where there need to apply the same instruction for multiple data example, adding up all the elements of 64 independent vectors. Some supercomputers are SIMD. Figure below shows the SIMD flow structure.

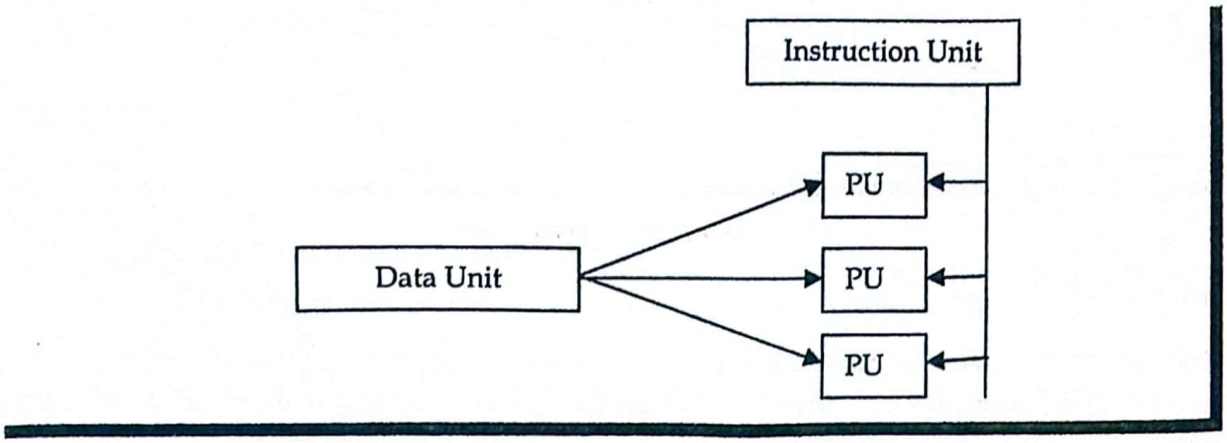



**Fig 9.3: SIMD Flow structure**

### Multiple Instructions, Single Data Stream (MISD)

The next category is MISD i.e. multiple instruction streams, single data stream. This structure was worked when there are multiple different instructions to operate on the same type of data. In general MISD architecture is not use more in practical.

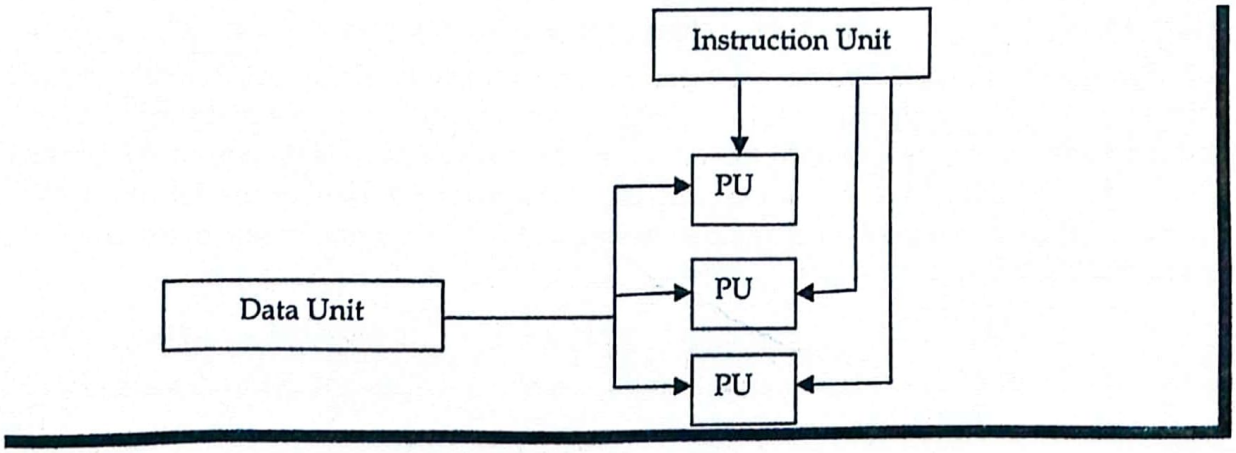



**Fig 9.4: MISD Flow structure**

### d. Multiple Instructions, Multiple Data Stream (MIMD)

The next category is MIMD, which has multiple instructions performances on multiple data units. This means a group of independent computers; each has its own program counter, program, and data. All distributed systems are MIMD, so this classification system is not more useful for simple purposes.

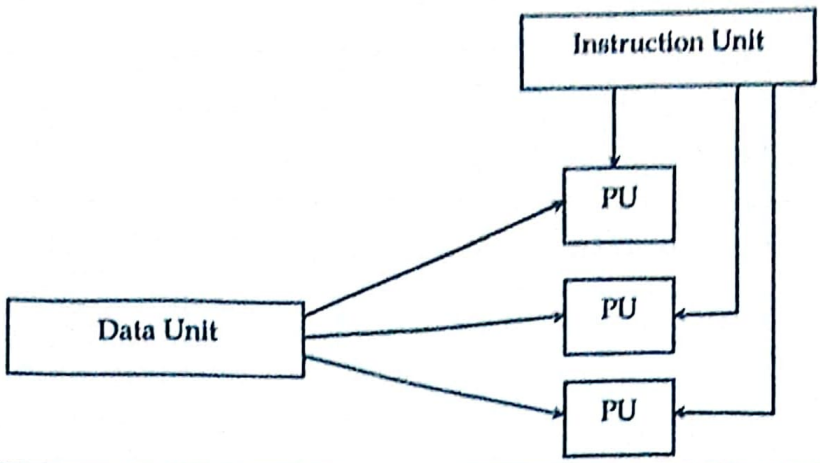



Fig 9.5: MIMD Flow structure

## Software Concept

Although the hardware is important, the software is even more important. The image that a system presents to its users and how they think about the system is largely determined by the operating system software, not the hardware.

There are basic two types of operating system namely **tightly coupled** operating system and **loosely couple** operating system; for multiprocessor and multicomputer. Loosely coupled software allows machines and users of a distributed system to be fundamentally independent of one another.Consider a group of personal computers, each of which has its own CPU, its own memory, its own hard disk, and its own operating system, but which share some resources, such as laser printers and databases over a LAN. This system is loosely coupled, since the individual machines are clearly distinguishable, each with its own job to do. If the network should go down for some reason, the individual machines can still continue to run to a considerable degree, although some functionality may be lost.

For tightly coupled system consider a multiprocessor dedicated to running a single chess program in parallel. Each CPU is assigned a board to evaluate and it spends its time examining that board and all the boards that can be generated from it. When the evaluation is finished, the CPU reports back the results and is given a new board to work on. The software for this system, both the application program and the operating system required to support it is clearly much more tightly coupled than in our previous example.

# COMMUNICATION IN DISTRIBUTED SYSTEMS

There are two approaches for communication in distributed system

- Shared memory and
- Message passing

**a. Shared memory**

Distributed shared memory (DSM) is a form of memory architecture where physically separated memories can be addressed as one logically shared address space. Here, the term shared does not mean that there is a single centralized memory; but that the address space is shared (same physical address on two processors refers to the same location in memory).

A distributed-memory system, often called a multicomputer, consists of multiple independent processing nodes with local memory modules which is connected by a general interconnection network. Software DSM systems can be implemented in an operating system, or as a programming library and can be thought of as extensions of the underlying virtual memory architecture. When implemented in the operating system, such systems are transparent to the developer; which means that the underlying distributed memory is completely hidden from the users. In contrast, software DSM systems implemented at the library or language level are not transparent and developers usually have to program them differently. However, these systems offer a more portable approach to DSM system implementations. A distributed shared memory system implements the shared-memory model on a physically distributed memory system.

The syntax used for DSM is the same as that of normal centralized memory multiprocessor systems.

```
read(shared_variable)
write(data, shared_variable)
```

The read() primitive requires the name of the variable to be read as its argument and the write() primitive requires the data and the name of the variable to which the data is to be written. The operating system locates the variable through its virtual address and, if necessary, moves the portion of memory containing the variable to the machine requiring it.

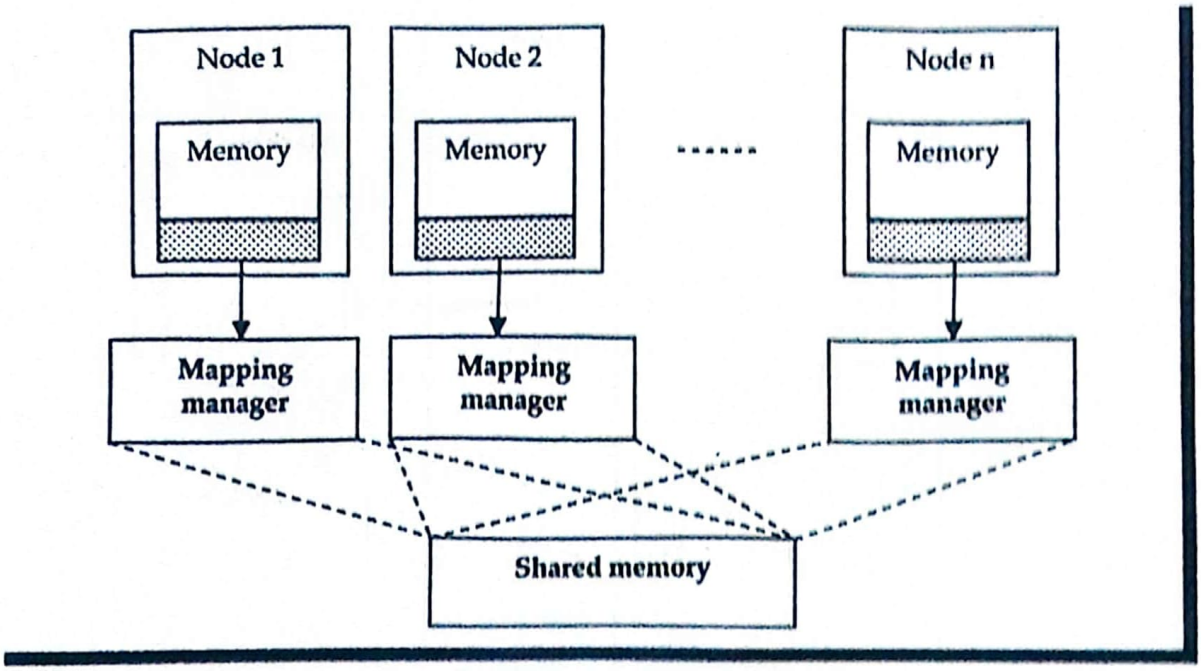



**Fig 9.6: Distributed shared memory**

**Advantages of Distributed Shared Memory**

- Hide data movement and provide a simpler abstraction for sharing data. Programmers don't need to worry about memory transfers between machines like when using the message passing model.
- Allows the passing of complex structures by reference, simplifying algorithm development for distributed applications.

- Takes advantage of "locality of reference" by moving the entire page containing the data referenced rather than just the piece of data.
- Cheaper to build than multiprocessor systems. Ideas can be implemented using normal hardware and do not require anything complex to connect the shared memory to the processors.
- Larger memory sizes are available to programs, by combining all physical memory of all nodes. This large memory will not incur disk latency due to swapping like in traditional distributed systems.
- Unlimited number of nodes can be used. Unlike multiprocessor systems where main memory is accessed via a common bus, thus limiting the size of the multiprocessor system.
- Programs written for shared memory multiprocessors can be run on DSM systems,

**b. Message Passing**

Message passing model allows multiple processes to read and write data to the message queue without being connected to each other. Messages are stored on the queue until their recipient retrieves them. Message queues are quite useful for inter-process communication and are used by most operating systems.

Message passing is the basis of most inter-process communication in distributed systems. It is at the lowest level of abstraction and requires the application programmer to be able to identify the destination process, the message, the source process and the data types expected from these processes.

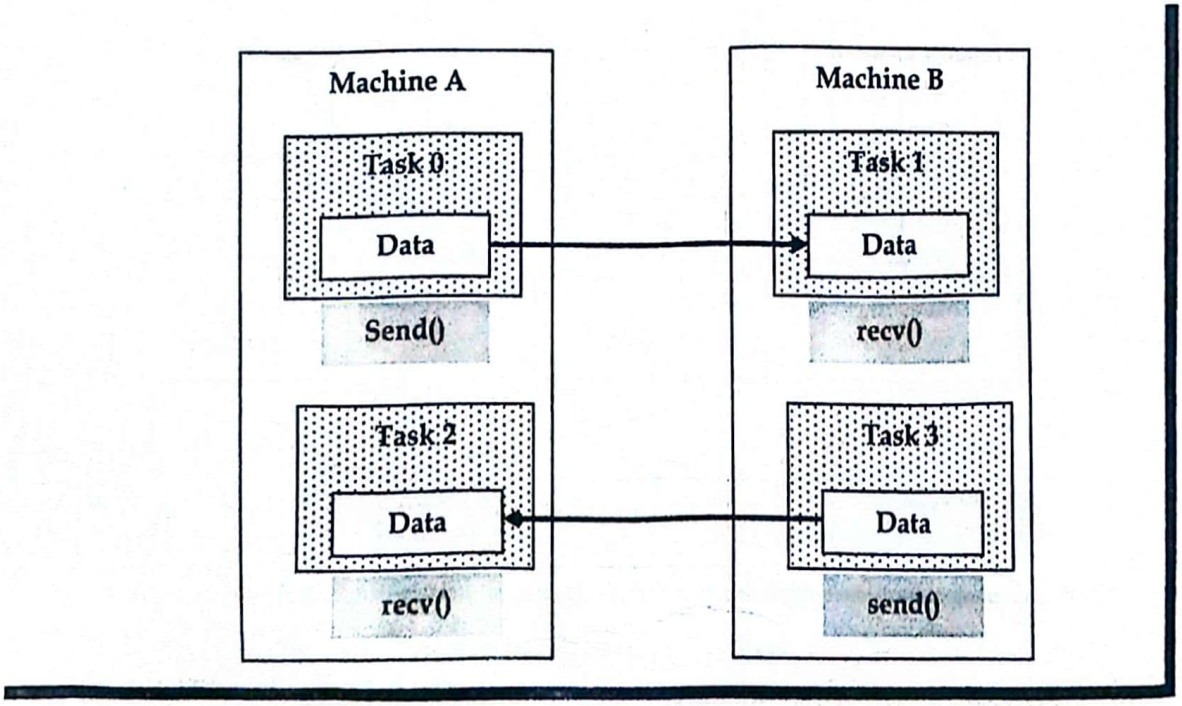



**Fig 9.7: Message Passing**

Communication in the message passing paradigm is performed using the send() and receive() primitives. The syntax is generally of the form:

send(receiver, message)

receive(sender, message)

The send() primitive requires the name of the destination process and the message data as parameters. The addition of the name of the sender as a parameter for the send() primitive would enable the receiver to acknowledge the message. The receive() primitive requires the name of the anticipated sender and should provide a storage buffer for the message.

c. **Remote Procedure Call**

Message passing leaves the programmer with the burden of the explicit control of the movement of data. Remote procedure calls (RPC) relieves this burden by increasing the level of abstraction and providing semantics similar to a local procedure call.

Remote Procedure Call (RPC) is a powerful technique for constructing distributed, client-server based applications. It is based on extending the conventional local procedure calling so that the called procedure need not exist in the same address space as the calling procedure. The two processes may be on the same system, or they may be on different systems with a network connecting them.

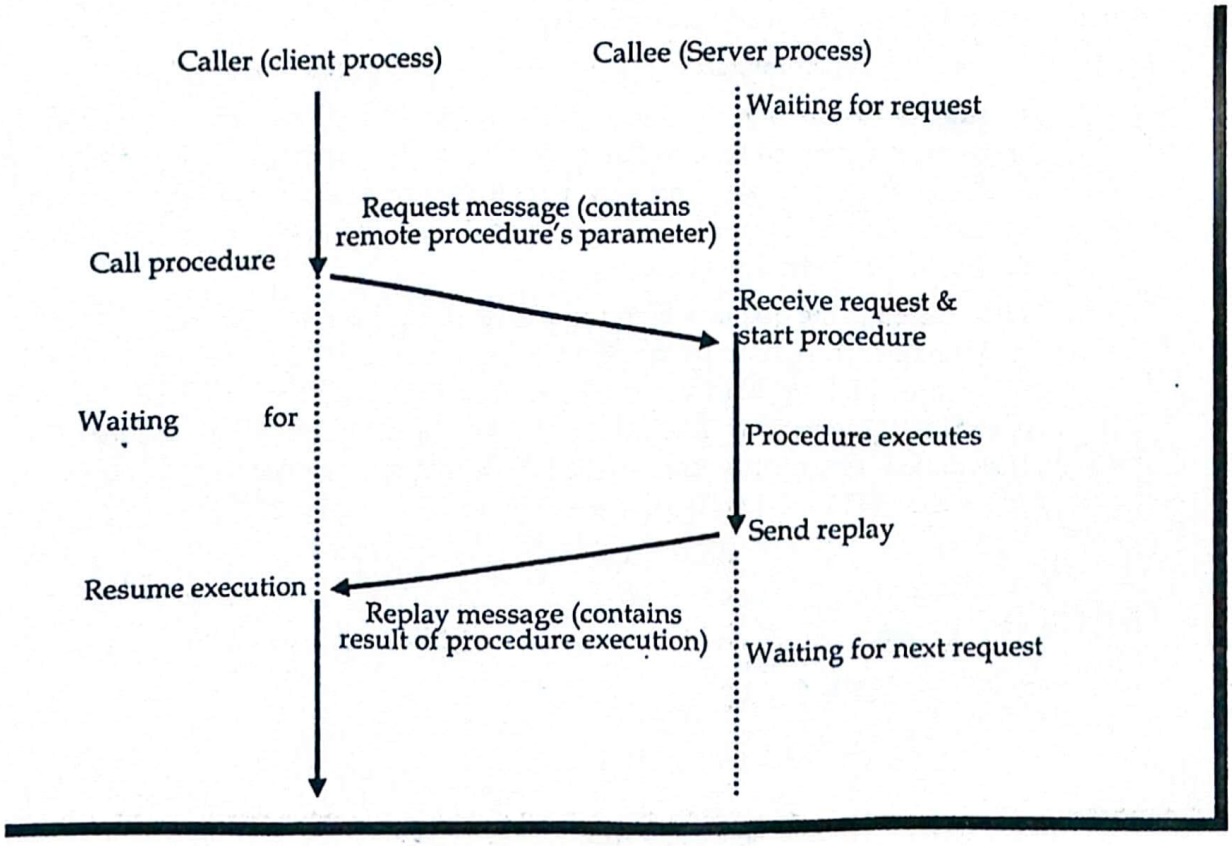



**Fig 9.8: Remote procedure call mode**

The syntax of a remote procedure call is generally of the form:

callprocedure_name(value_arguments; result_arguments)

The client process blocks at the call() until the reply is received. The remote procedure is the server processes which have already begun executing on a remote machine. It blocks at the receive() until it receives a message and parameters from the sender. The server then sends a reply() when it has finished its task. The syntax is as follows:

receiveprocedure_name(in value_parameters; out result_parameters)

reply(caller, result_parameters)

## CLOCK SYNCHRONIZATION

Distributed System is a collection of computers connected via the high speed communication network. In the distributed system, the hardware and software components communicate and coordinate their actions by message passing. Each node in distributed systems can share their resources with other nodes. So, there is need of proper allocation of resources to preserve the state of resources and help coordinate between the several processes. To resolve such conflicts, synchronization is used. Synchronization in distributed systems is achieved via clocks. The clock synchronization can be achieved by 2 ways:

- External and
- Internal Clock Synchronization

**External clocksynchronization** is the one in which an external reference clock is present. It is used as a reference and the nodes in the system can set and adjust their time accordingly.

**Internal clock synchronization** is the one in which each node shares its time with other nodes and all the nodes set and adjust their times accordingly.

There are 2 types of clock synchronization algorithms: Centralized and Distributed.

- **Centralized** is the one in which a time server is used as a reference. The single time server propagates it's time to the nodes and all the nodes adjust the time accordingly. It is dependent on single time server so if that node fails, the whole system will lose synchronization. Examples of centralized are- Berkeley Algorithm, Passive Time Server, Active Time Server etc.
- **Distributed** is the one in which there is no centralized time server present. Instead the nodes adjust their time by using their local time and then, taking the average of the differences of time with other nodes. Distributed algorithms overcome the issue of centralized algorithms like the scalability and single point failure. Examples of Distributed algorithms are – Global Averaging Algorithm, Localized Averaging Algorithm, NTP (Network time protocol) etc.

## EXERCISE

### Multiple Choice Questions

1. If one site fails in distributed system then _____
   a) the remaining sites can continue operating
   b) all the sites will stop working
   c) directly connected sites will stop working
   d) none of the mentioned
2. Network operating system runs on _____
   a) server
   b) every system in the network
   c) both server and every system in the network
   d) none of the mentioned

3. Which routing technique is used in a distributed system?
   a) fixed routing b) virtual routing
   c) dynamic routing d) all of the mentioned
4. In distributed systems, link and site failure is detected by _____
   a) polling b) handshaking
   c) token passing d) none of the mentioned
5. The capability of a system to adapt the increased service load is called _____
   a) scalability b) tolerance
   c) capacity d) none of the mentioned
6. Internet provides _____ for remote login.
   a) telnet b) http
   c) ftp d) rpc
7. In distributed systems, a logical clock is associated with _____
   a) each instruction b) each process
   c) each register d) none of the mentioned
8. For proper synchronization in distributed systems _____
   a) prevention from the deadlock is must
   b) prevention from the starvation is must
   c) prevention from the deadlock & starvation is must
   d) none of the mentioned
9. In distributed systems, what will the transaction coordinator do?
   a) starts the execution of transaction
   b) breaks the transaction into number of sub transactions
   c) coordinates the termination of the transaction
   d) all of the mentioned
10. Which technique is based on compile-time program transformation for accessing remote data in a distributed-memory parallel system?
   a) cache coherence scheme b) computation migration
   c) remote procedure call d) message passing

## Subjective Questions

1. Define distributed OS. Explain their features.
2. Describe advantages of Distributed System over Centralized System.
3. Describe advantages of Distributed System Over independent PCs.
4. Describe disadvantages of Distributed System.
5. Define Hardware and Software Concepts in distribute OS.
6. Define Communication in Distributed Systems,
7. What is Message Passing? How it is differ from Remote Procedure Call.
8. Define Process in Distribution System.

9. What is Clock Synchronization? Explain
10. Difference between network operating system and distributed operating system.
11. What is network operating system? Explain.
12. Differentiate between Shared memory and Message passing.
13. Describe software concept in distributed OS.
14. Write down the syntax for message passing and explain it.
15. Describe types of clock synchronizations with example.
16. Describe Single Instruction, Multiple Data Stream (SIMD).
17. Describe Multiple Instructions, Multiple Data Stream (MIMD).
18. Describe Single Instruction, Single Data Stream (SISD).
19. Explain Remote Procedure Call with suitable example.
20. Differentiate between hardware and software concept for distributed operating system with suitable example.

## ANSWERS KEY

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1. (a) | 2. (a) | 3. (a) | 4. (b) | 5. (a) | 6. (a) | 7. (b) | 8. (c) | 9. (d) | 10. (b) |

□□□