Date :

Page No. :

Name : Khan Nausheen Ayub

Class : TY MCA - Sem V

Subject : Distributed Computing

Application ID : 370510

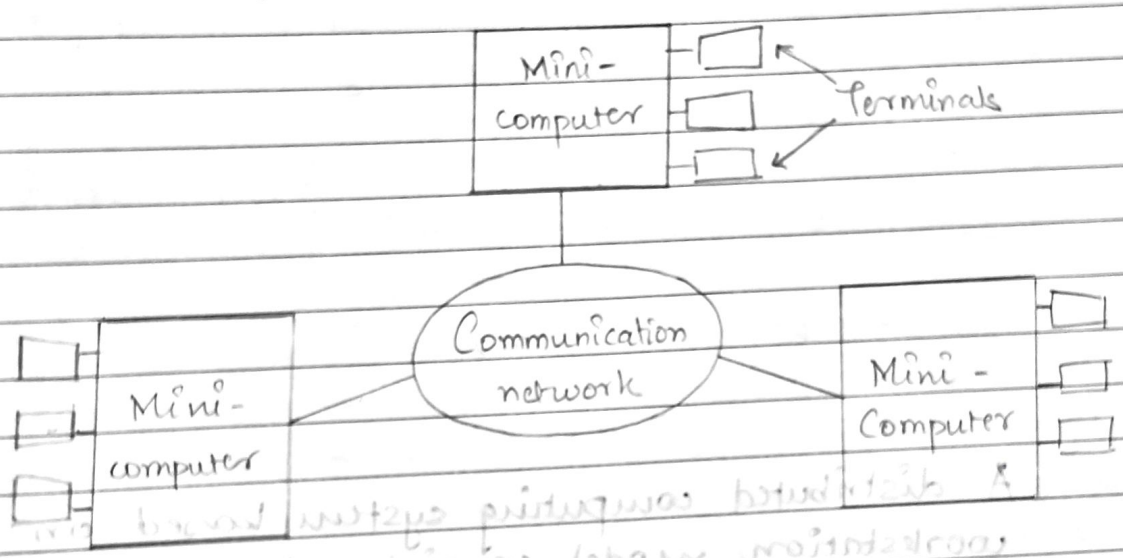
PRN No. : 20170279003233883

Distributed Computing

Q. 1. Explain different distributed computing model with detail.

Ans: The various models that are used for building distributed computing systems can be classified into 5 categories :

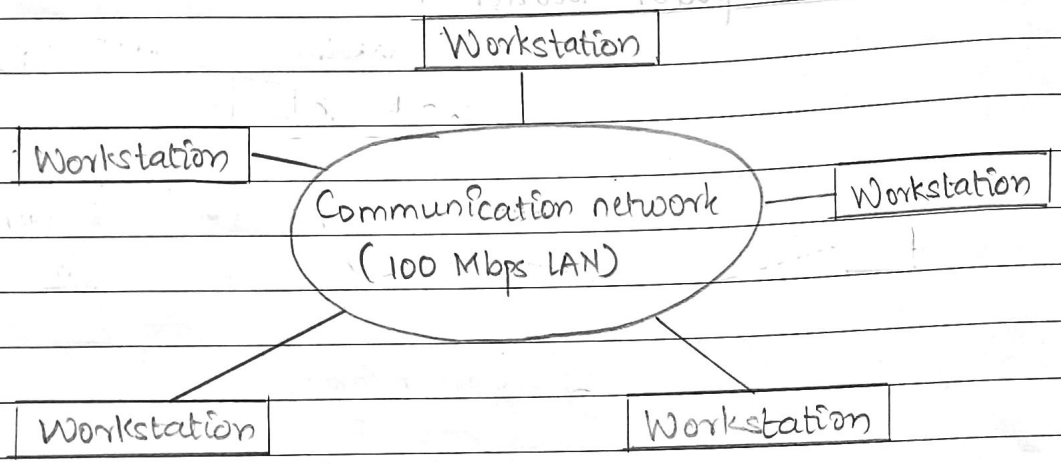
1. Minicomputer Model



The minicomputer model is a simple extension of the centralized time-sharing system. A distributed computing system based on this model consists of a few minicomputers interconnected by a communication network where each minicomputer usually has multiple users simultaneously logged on to it. Several interactive terminals are connected to each minicomputer. Each user logged on to one specific minicomputer has 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 resource sharing with remote users is desired. The early ARPA net is an example of a distributed computing system based on the minicomputer model.

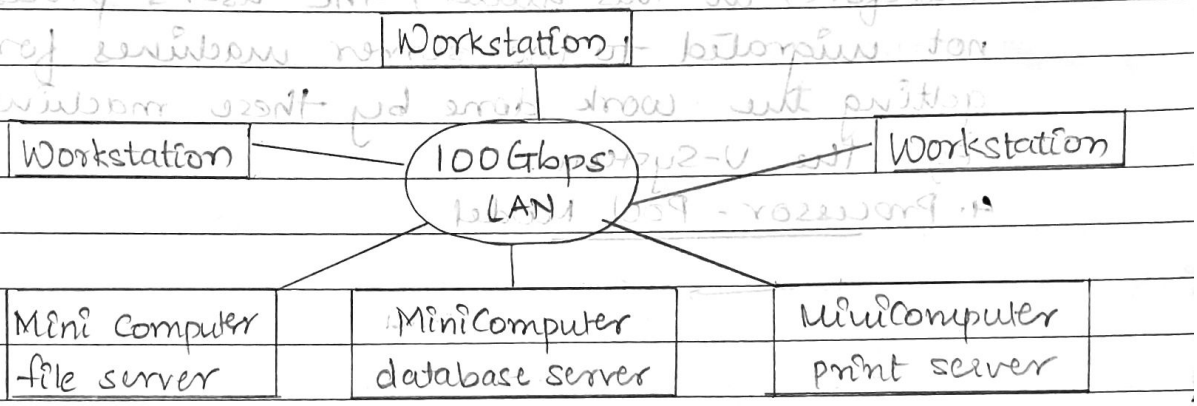
2. Workstation Model



A distributed computing system based on the workstation model consists of several workstations interconnected by a communication network. An organization may have several workstations located throughout an infrastructure where each workstation is equipped with its own disk and serves as a single-user computer. In such an environment, at any one time a significant proportion of the workstations are idle which results in the waste of large amounts of CPU time. Therefore, the idea of the workstation model is to interconnect all these stations by a high-speed LAN so that idle workstations made be used to process jobs of users who are logged onto other workstations

& donot have sufficient processing power at their own workstations to get their jobs processed efficiently. Eg. Sprite system & Xerox PARC.

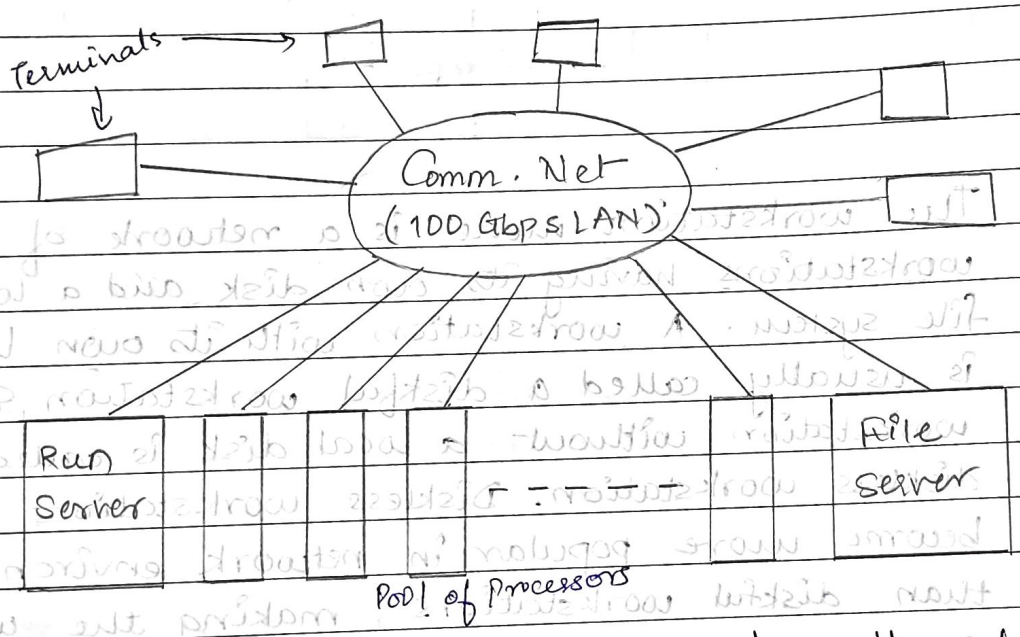
3. Workstation - Server Model



The workstation model is a network of personal workstations having its own disk and 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 and 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 these machines.
Eg: The V-System.

4. Processor - Pool Model



The processor-pool model is based on the observation that most of the time a user does not need any computing power but once in a while 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 process-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. 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.
Eg : Amoeba and the Cambridge Distributed Computing System.

5. Hybrid Model

The workstation-server model has a large number of computer users only performing simple interactive tasks and executing small programs. In a working environment that has progn groups of users who often perform jobs needing massive computation, the processor-pool model is more attractive and suitable. To combine advantages of workstation-server and 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.

Q.2. What is Service Oriented Architecture (SOA)? Explain in detail.

Ans: SOA is a structure that allows services to communicate with each other across different platforms and languages by implementing what is known as a "loose coupling" system. While the concept of SOA has been around for many years, it is only within the past decade that it has risen to the forefront of software-related technologies. The term "loose coupling" refers to the dient of a service, and its ability to remain independent of the service that it requires. The most important part of this concept is the client, which in itself can be a service, can communicate with the service even if they are not closely related. This facilitated communication is achieved through the implementation of a specified interface that is able to perform the necessary actions to allow for the transmission of data. A common example of this increased ability to communicate without service constraints involves coding languages used by these services. There is an array of different languages from which software platforms are created and not all of these languages can interact fluently, without encountering communication issues. By using an SOA, it is not necessary for the client to understand the language that is being used by

the service, but instead, it relies on a structured interface that is able to process the transmission between the service and the client.

There are a variety of ways that implementing an SOA structure can benefit a business, particularly, those that are based around web services.

Some of these advantages include -

i) Creates reusable code

The primary motivator for companies to switch to an SOA is the ability to reuse code for different applications. By reusing code that already exists within a service, enterprises can significantly reduce the time that is spent during the development process. It also lowers costs that are incurred during the development of applications. Since SOA allows varying languages to communicate through a central interface, this means that application engineers do not need to be concerned with the type of environment in which these services will be run. Instead, they only need to focus on the public interface that is being used.

ii) Promotes interaction

A major advantage in using SOA is the level of interoperability that can be achieved when properly implemented. With SOA, no longer will communication between platforms be hindered in operation by the languages on

which they are built. Once a standardized protocol has been put in place, the platform and the languages can remain independent of each other, while still being able to transmit data between clients and services.

3. Allows for scalability

When developing applications for web services, one issue that is of concern is the ability to increase the scale of the service to meet the needs of the client. By using an SOA where there is a standard communication protocol in place, enterprises can drastically reduce the level of interaction that is required between clients and services, and this reduction means that applications be scaled without putting added pressure on the application.

4. Reduced costs

In business, the ability to reduce costs while still maintaining a desired level of output is vital to success, and this concept holds true with customized service solutions as well. By switching to an SOA based system, businesses can limit the level of analysis that is often required when developing customized solutions for specific applications. The cost reduction is facilitated by the fact that loosely coupled systems are easier to maintain and do not necessitate the need for costly development and analysis. Furthermore, the increasing popularity in SOA drives cost lower.

Q.8 What is Stub? Explain the implementation of stub in RPC Mechanism.

Ans: A stub, in the context of distributed computing, is a piece of code that is used to convert parameters during a remote procedure call (RPC). An RPC allows a client computer to remotely call procedures on a server computer. The parameters used in a function call have to be converted because the client and server computers use different address spaces. Stubs perform this conversion so that the remote server computer perceives the RPC as a local function call.

RPC mechanism uses the concept of stubs to achieve the goal of semantic transparency. Stubs provide a local procedure call abstraction by concealing the underlying RPC mechanism. A separate stub procedure is associated with both the client and server processes.

RPC communication package known as RPC Runtime is used on both the sides to hide existence and functionalities of a network.

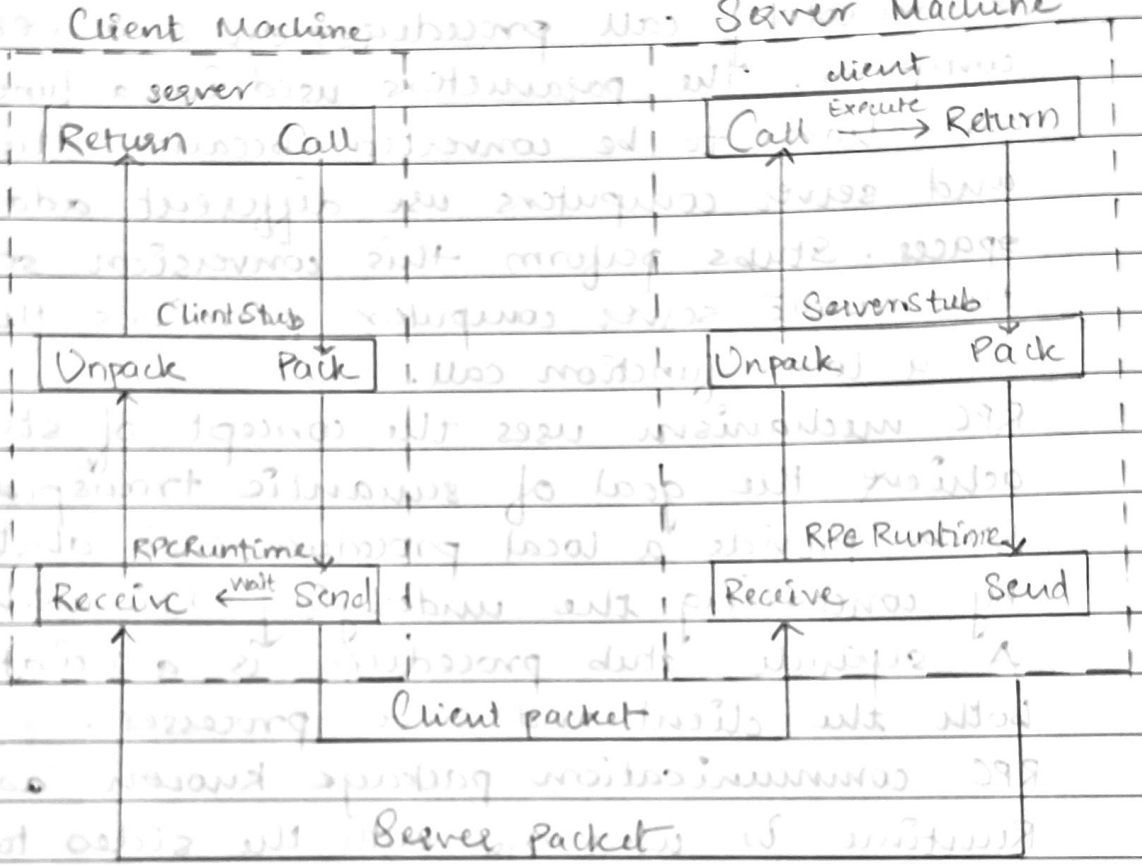
Thus implementation of RPC involves the 5 elements of program:

i.) Client
ii.) Client Stub
iii.) RPC Runtime
iv.) Server Stub
v.) Server

The client, the client stub, and one instance

virtual memory is accessed. Data moves

of RPC Runtime execute on the client machine. the server, the server stub, and one instance of RPC Run time execute on the server machine.
Remote services are accessed by the user by making ordinary LPC



1. Client
A client is a user process which initiates a RPC.
The client makes a normal call that will invoke a corresponding procedure in the client stub.

2. Client Stub
Client stub is responsible for the following two

tasks :
i.) On receipt of a call request from the client, it packs specifications of the target procedure and arguments into a message and asks the local RPC Runtime to sent it to the server stub.
ii.) On receipt of the result of procedure execution, it unpacks the result and passes it to the client.

8. RPCRuntime
Transmission of messages between client and the server machine across the network is handled by RPCRuntime.
It performs Retransmission, Acknowledgement, Routing and Encryption.
RPCRuntime on client machine receives messages containing result of procedure execution from server and sends it to client stub as well as the RPCRuntime on server machine receives the same message from the server stub and passes it to client machine.
It also receives call request messages from client machine and sends it to server stub.

4. Server Stub
Server stub is similar to client stub and is responsible for the following two tasks :
i.) On receipt of a call request message from the local RPCRuntime, it unpacks and makes a normal call to invoke the required procedure in the server.
ii.) On receipt of the result of procedure

execution from the server, it unpacks the result into a message and then asks the local RPCRuntime to send it to the client stub.

5. Server

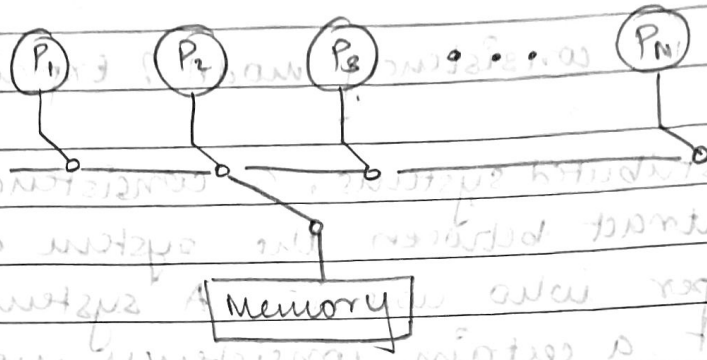
When a call request is received from the server stub, the server executes the required procedure and returns the result to the server stub.

Q.4. What is consistency model? Explain in detail.

Ans: In distributed systems, a consistency model is a contract between the system and the developer who uses it. A system is said to support a certain consistency model if operations on memory respect the rules defined by the model.
They are a powerful abstraction which helps to describe a system in terms of its observable properties. From a practical point of view, a consistency model answers to questions like: If client 1 adds an object to a distributed data store at 1:00 pm and client 2 tries to read the same object at 1:01 pm, will client-2 be able to get the new object? Or will it get a 404 because the resource was not found?
The answer is that... it depends.
There are many factors to consider to answer such a question. One of them is the consistency model adopted by the distributed data source.

Sequential Consistency

The result of any execution is the same as if the operations of all processors were executed in some sequential order and the operations of each individual processor appear in this sequence in the order specified by its program.



Conceptually there is one global memory and a switch that connects an arbitrary processor to the memory at any time. Each processor to the memory at any time. Each processor issues memory operations in program order and the switch provides the global serialization among all the ~~places~~ processors.

Sequential consistency is not deterministic because multiple execution of the distributed program might lead to a different order of operations.

Strict Consistency

It is also called Strong consistency or Linearizability.

Like sequential consistency, but the execution order of programs between processors must be the order in which those operations were issued. As analogy, this is pretty much what we would get from a multithreaded program in which there is no cache between the cores and the main memory, and neither the compiler nor the CPU is allowed to reorder instructions.

Therefore, if each program in each processor is

deterministic, then the distributed program is deterministic.

Casual Consistency

Relax sequential consistency allowing also reording of the operations in the same processors when they are not 'casually related'.

PRAM Consistency

The write operations (and only the writes) of each processor appear in the order specified in the program.

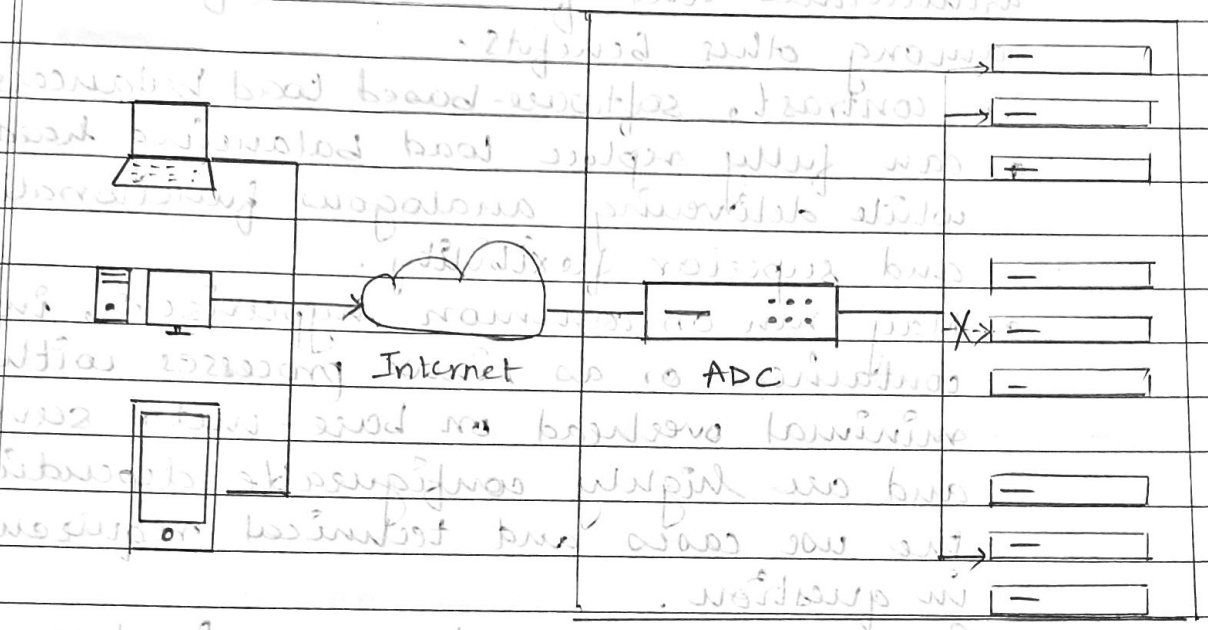
Q.5. Explain the Load balancing model in detail.

Ans: Load balancing refers to efficiently distributing incoming network traffic across a group of backend servers, also known as a server farm or server pool.

Modern Load balancing is defined as the methodical and efficient distribution of network or application traffic across multiple servers in a server farm. Each load balancer sits between client devices and backend servers, receiving and then distributing incoming requests to any available server capable of fulfilling them.

A load balancer may be :

- A physical device, a virtualized instance of running on specialized hardware or a software process.
- Incorporated into application delivery controllers (ADCs) designed to more broadly improve the performance and security of three-tier web and microservice-based applications, regardless of where they're hosted.
- Able to leverage many possible load balancing algorithms, including round robin, server response time and the least connection method to distribute traffic in line with current requirements.



Regardless of whether its hardware or software, or what algorithm(s) it uses, a load balancer distributes traffic to different web servers in the resource pool to ensure that no single server becomes overworked and subsequently unreliable. Load balancers effectively minimize server response time and maximize throughput.

Hardware-based load balancers work as follows:

- They are ~~capal~~ typically high-performance appliances, capable of securely processing multiple gigabits of traffic from various types of applications.
- These appliances may also contain built-in virtualization capabilities, which consolidate numerous virtual load balancer instances on the same hardware.
- That allows for more flexible multi-tenant

architectures and full isolation of tenants, among other benefits.

In contrast, software-based load balancers:

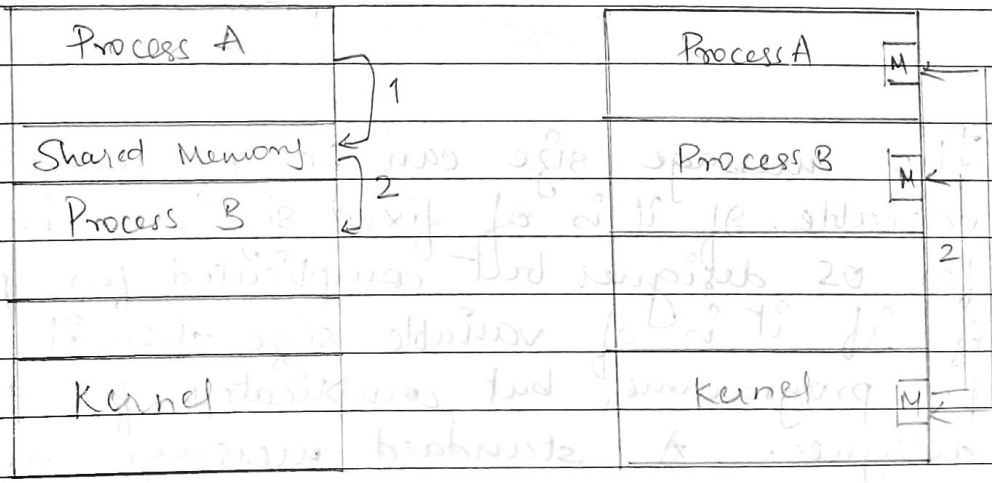
- can fully replace load balancing hardware while delivering analogous functionality and superior flexibility.
- May run on common hypervisors, in containers or as Linux processes with minimal overhead on bare-metal servers and are highly configureable depending on the use cases and technical requirements in question.
- Can save space and reduce hardware expenditures.

An ADC with load balancing capabilities helps IT departments ensure scalability and availability of services. Its advanced traffic management functionality can help a business steer requests more efficiently to the correct resources for each end user. An ADC offers many other functions (for example, encryption, authentication and web application firewalling) that can provide a single point of control for securing, managing and monitoring the many applications and services across environments and ensuring the best end-user experience.

Q.6. Explain the Message Passing Mechanism in the IPC.

Ans: A process can be of two types:
i) Independent process
ii) Co-operating process.

An independent process is not affected by the execution of other processes while a co-operating process can be affected by other executing processes. IPC is a mechanism which allows processes to communicate each other and synchronize their actions. The communication between these processes can be seen as a method of co-operation between them.
Processes can communicate with each other using 2 ways :-
i) Shared Memory
ii) Message passing.
The below figure shows a basic structure of communication between processes via shared memory method & via message passing.

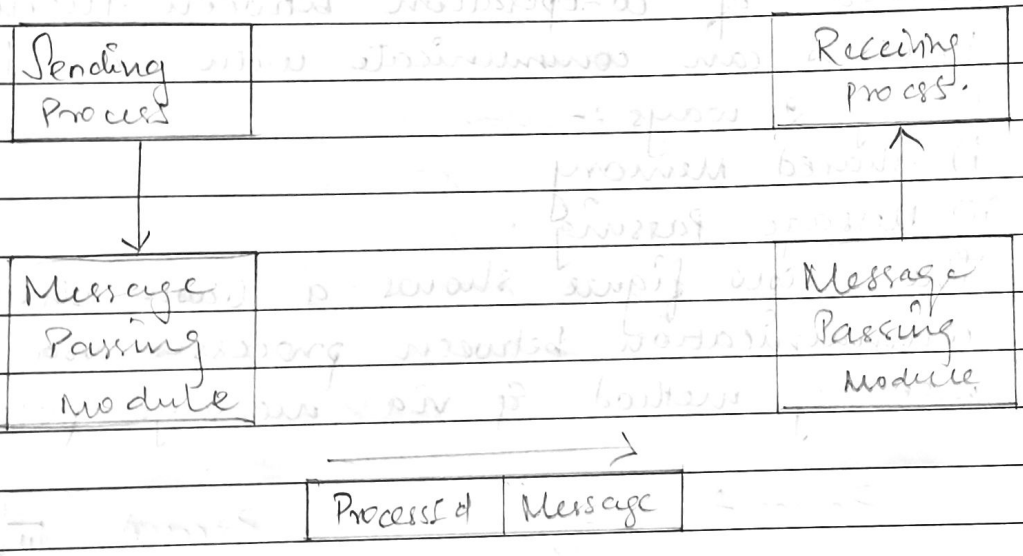


In Message Passing mechanism, processes communicate with each other without using any kind of shared memory. If two processes P1 and P2 want to communicate with each other, they proceed as follows -

- Establish a communication link (if a link already exists, no need to establish it again).
- Start exchanging messages using basic primitives.

We need at least two primitives :-

- send (message, destination) or send (message)
- receive (message, host) or receive (message)



The message size can be of fixed or variable. If it is of fixed size, it is easy for OS designer but complicated for programmer Eg if it is of variable size then it is easy for programmer but complicated for the OS designer. A standard message can have

two parts : header and body.
The header part is used for storing Message type, destination id, source id, message length and control information.
The control information contains information like what to do if runs out of buffer space, sequence no., priority. Generally, message is sent using FIFO.
A process that is blocked is the one that is waiting for some event, such as a resource becoming available for the completion of an I/O operation. IPC is possible between the processes on same computer as well as on the processes running on different computer i.e. in network/ distributed system. In both cases, the process may or may not be blocked while sending a message or attempting to receive a message. Message passing may be blocking or unblocking. Blocking is considered synchronous and blocking send means the sender will be blocked until the message is received by the receiver. Similarly, blocking receive has the receiver block until a message is available. Non-blocking is considered asynchronous and Non-blocking send has the sender send the message and continue. Similarly, non-blocking receive has the receiver receive a valid message or null.
There are basically 3 most preferred combinations —

i) Blocking send and blocking receive.
ii) Non-Blocking send and Non-blocking receive.
iii) Non-blocking send and Non-blocking receive (Mostly used.

In direct message passing, the process which want to communicate must explicitly name the recipient or sender of communication.

In indirect message passing, processes uses mailboxes (also referred to as ports) for sending and receiving messages. Each mailbox has a unique Id and processes can communicate only if they share a mailbox.

Q.7. Explain the difference between load balancing and load sharing.

Ans:

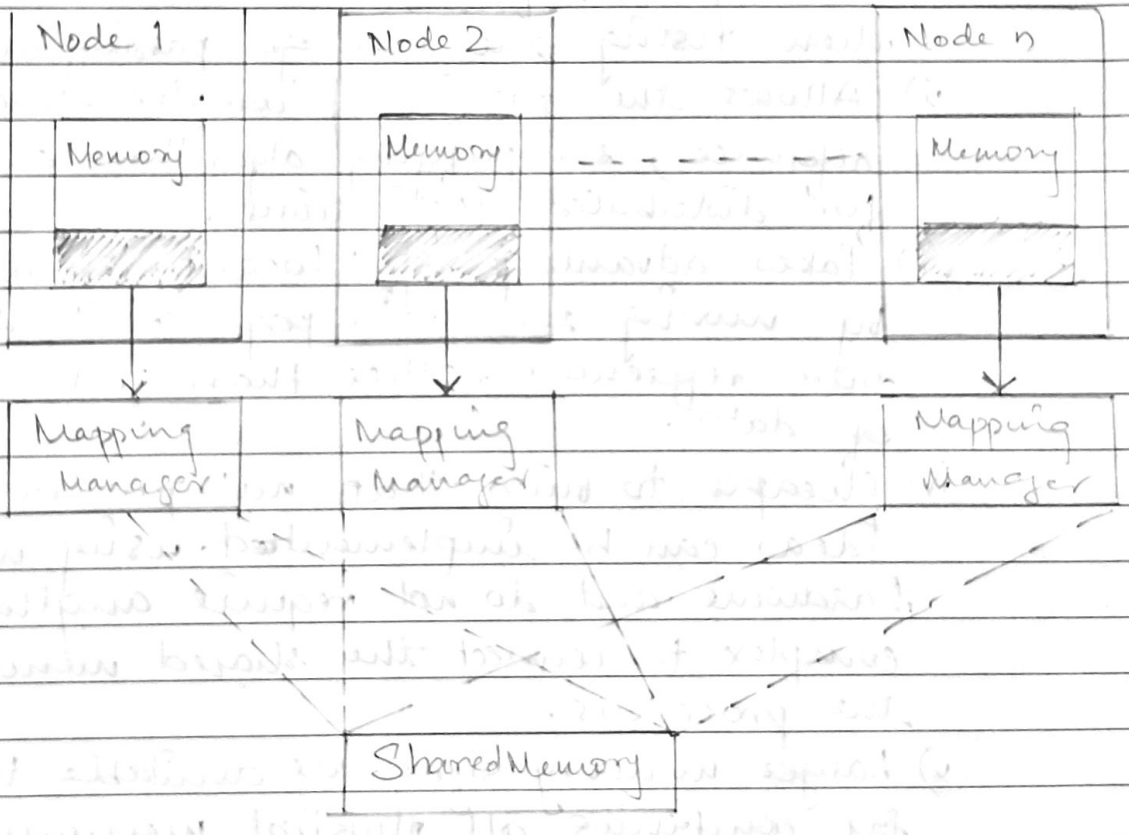
	Load Balancing	Load Sharing
Example protocols utilizing the technique	IGP protocols, per packet process switching, per packet CEF switching, PFR	MHRP, eBGP, PBR
Technical approach	More "dynamic" concept where load is balanced across 2 or more active links.	More of "static" concept like in case of eBGP where we can manipulate traffic flows on a static basis, by doing local preference, weight for outbound decisions
Distribution/ Splitting of traffic	Load balancing works by distributing the traffic evenly and dynamically across multiple links.	Load sharing works by splitting the traffic to perform its function.
Per Packet load spreading	Can be performed on a packet-by-packet basis in a round-robin fashion	Cannot be performed on per packet basis.

→

Load distribution across links.	More efficiently handles load across both links. Even amount of traffic traverses across both the links.	Less efficiently handles load across both links. Uneven amount of traffic traverses across both the links. Needs manual intervention every time load distribution needs to be applied across multiple links.
Principle of working	Utilizes algorithm like round robin, maximum connections, minimal link utilization, etc. to perform load balancing.	Utilizes source or destination IP or mac address based decision making while performing load sharing.
Define	Traffic is balanced across different links, equal or closely equal amounts	Traffic is only shared across different links without being necessarily balanced.

Q.8. What is DSM ? Explain the advantages of DSM.

Ans: Distributed Shared Memory (DSM) is a resource management component of a distributed operating system that implements the shared memory model in distributed systems, which have no physically shared memory. The shared memory model provides a virtual address space that is shared among all computers in a distributed system.



In DSM, data is accessed from a shared address space similar to the way that virtual memory is accessed. Data moves

between secondary and main memory, as well as, between the distributed main memories of different nodes. Ownership of pages in memory starts out in some pre-defined state but changes during the cause of normal operation. Ownership changes take place when data moves from one node to another due to an access by a particular process.

Advantages of Distributed Shared Memory :-

i) 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.

ii) Allows the passing of complex structures by reference, simplifying algorithm development for distributed applications.

iii) Takes advantage of "locality of reference" by moving the entire page containing the data referenced rather than just the piece of data.

iv) 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.

v) Larger memory sizes are available to programs, by combining all physical memory to the processors of all nodes. Plus large memory will not incur disk latency due to swapping like in traditional distributed systems.

vi) 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.

vii) Programs written for shared memory multiprocessors can be run on DSM systems.

Q.9. What are the desirable features of a good naming system?

Ans: The naming facility of a distributed system enables users and programs to assign character-string names to objects and subsequently use these names to refer to those objects.
The naming system plays a very important role in achieving the goal of
- location transparency
- facilitating transparent migration and replication of objects.
- object sharing.

The desirable features of a good naming system are -

i) Location transparency.
Location transparency means that the name of an object should not reveal any hint as to the physical location of the object. That is, an object's name should be independent of the physical connectivity or topology of the system, or the current location of the object.

ii) Location independency.
For performance, reliability, availability, and security reasons, distributed systems provide the facility of object migration that allows the among the various nodes of a system.

Location independency means that the name of an object need not be changed when the object's location changes.
Furthermore, a user should be able to access an object by its same name irrespective of the node from where he or she accesses it.
Therefore, the requirement of location independency calls for a global naming facility with the following two features :-
a) An object at any node can be accessed without the knowledge of its physical location (location independency of request-receiving objects).
b) An object at any node can issue an access request without the knowledge of its own physical location. This property is also known as user mobility.

(ii) Scalability
Distributed systems vary in size ranging from one with a few nodes to one with many nodes. Moreover, distributed systems are normally open systems, and their size changes dynamically.
Therefore, it is impossible to have an a priori idea about how large the set of names to be dealt with is liable to get.
Hence, a naming system must be capable of adapting to the dynamically changing scale of a distributed system that-

normally leads to a change in the size of the name space. That is, a change in the system scale should not require any change in the naming or locating mechanisms.

iv) Uniform naming convention
In many existing systems, different ways of naming objects, called naming conventions, are used for naming different types of objects. For example, file names typically differ from user names and process names. Instead of using such nonuniform naming conventions, a good naming system should use the same naming convention for all types of objects in the system.

v) Multiple user-defined names for the same object
For a shared object, it is desirable that different users of the object can use their own convenient names for accessing it. Therefore, a naming system must provide the flexibility to assign multiple user-defined names to the same object. In this case, it should be possible for a user to change or delete his or her name for the object without affecting those of other users.

vi) Group naming
A naming system should allow many different

objects to be identified by the same name. Such a facility is useful to support broadcast facility or to group objects for conferencing or other applications.

vii) Meaningful names

A name can be simple any character string identifying some object. However, for users, meaningful names are preferred to lower level identifiers such as memory pointers, disk block numbers, or network addresses. This is because meaningful names typically indicate something about the contents or function of their referents, are easily transmitted between users, and are easy to remember and use. Therefore, a good naming system should support atleast two level of object identifiers, one convenient for human users and one convenient for machines.

viii) Performance

The most important performance measurement of a naming system is the amount of time needed to map an object's name to its attributes, such as its location. In a distributed environment, this performance is dominated by the number of messages exchanged during the name-mapping operation. Therefore, a naming system should be efficient in the sense that the number of messages exchanged in a name-mapping operation should be small

as possible

ix) fault tolerance

A naming system should be capable of tolerating, to some extent, faults that occur due to the failure of a node or a communication link in a distributed system network.

x) Replication transparency

In a distributed system, replicas of an object are generally created to improve performance and reliability. A naming system should support the use of multiple copies of the same object in a user-transparent manner ie. if not necessary, a user should not be aware that multiple copies of an object are in use.

xi) Locating the nearest replica

When a naming system supports the use of multiple copies of the same object, it is important that the object-locating mechanism of the system should always supply the location of the nearest replica of desired object. This is because the efficiency of the object accessing operation will be affected if the object-locating mechanism does not take this point into consideration.

Q.10. What are the desirable features of a good message system?

Ans: Message System of Distributed system provides a set of message based IPC protocols. It hides the details of complex network protocols and multiplies heterogeneous platforms from programmers.
A typical message passing system uses two primitives for communication :
SEND (Data, Receivers Address), RECEIVE (Data)
Features of Message Passing :-
i) Simplicity
A MPS should be simple and easy to use. It must have simple and clear semantics of IPC Protocols which makes message passing easier.
ii) Uniform Semantics
There are two types of communication : local and Remote semantics of Remote should be same or atleast as close as possible to local communication for ease of use.
iii) Efficiency
A message passing system should be efficient and it can be made efficient by reducing the total number of message exchange. Avoiding cost of setting and terminating connections between the same pair. Minimizing the cost of maintaining connections. Piggybacking the acknowledgement with next message to same receiver.

iv) Reliability

Distributed system is prone to node crashes or communication link failure resulting into loss of data. MPS can be made reliable by designing the algorithms which deal with the problems like : Handling of lost message, Duplicate messages. A MPS should be capable of detecting and handling duplicates, Generating and assigning appropriate sequences.

v) Atomicity

The atomicity property tells that message in the distributed system must be sent to all intended receivers or none. A good MPS delivers the message in a proper order - It also offers the Survivability : Means ability of message delivery despite of partial failure.

vi) Flexibility

User may choose types and level of reliability. synchronous / Asynchronous, Send / Receive choice.

vii) Security

A good MPS must provide secure end to end communication - To achieve security the possible ways are : Authentication of receiver by sender, Authentication of sender by receiver, encryption & decryption.

viii) Portability :

A good MPS should be portable with respect to two aspects : Message passing should itself be portable and applications written should be portable

Q.11. Explain RPC mechanism

Ans. RPC mechanism uses the concepts of stubs to achieve the goal of semantic transparency. Stubs provide a local procedure call abstraction by concealing the underlying mechanism. A separate stub procedure is associated with both the client and server processes. RPC communication package known as RPC Runtime is used on both the sides to hide existence and functionalities of a network. Thus implementation of RPC involves the 5 elements of program :

i) Client
ii) Client Stub
iii) RPC Runtime
iv) Server stub
v) Server

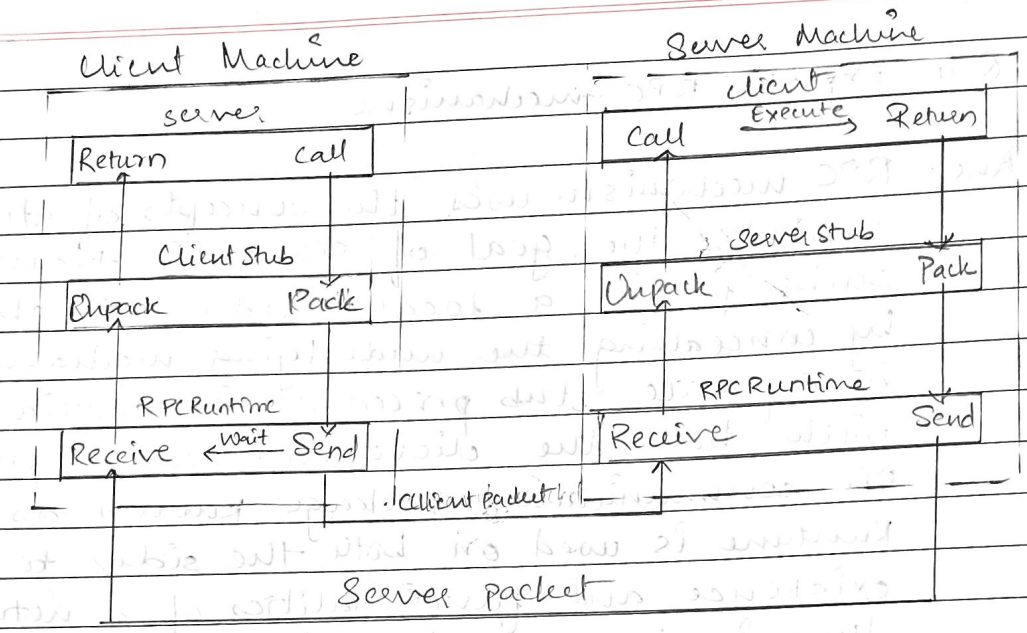
The client, the client stub, and one instance of RPC Runtime execute on the client machine. The server, the server stub, and one instance of RPC Runtime execute on the server machine.

Remote services are accessed by the user by making ordinary LPC.

i) Client

A client is a user process which initiates a RPC.

The client makes a normal call that will invoke a corresponding procedure in the client stub



ii) Client stub

Client stub is responsible for the following two tasks :

a) On receipt of a call request from the client, it packs specifications of the target procedure and arguments into a message and asks the local RPC runtime to send it to the server stub.

b) On receipt of the result of procedure execution, it unpacks the result and passes it to the client.

iii) RPC Runtime

Transmission of messages between client and the server machine across the network is handled by RPC Runtime. It performs Retransmission, Acknowledgement, Routing and Encryption.

RPC Runtime on client machine receives

containing result of procedure execution from server and sends it client stub as well as the RPC Runtime on server machine receives the same message for server stub and passes it to the client machine.
It also receives call request messages from client machine and sends it to server stub.

iv) Server Stub

Server stub is similar to client stub and is responsible for the following two tasks :

a) On receipt of a call request message from the local RPC Runtime, it unpacks and makes a normal call to invoke the required procedure in the server.

b) On receipt of the result of procedure execution from the server, it unpacks the result into a message and then ask the local RPC Runtime to send it to the client stub.

v) Server

When a call request is received from the server stub, the server executes the required procedure and returns the result to the server stub.

Q.12. What are election algorithms ? Explain with the help of a diagram.

Ans: Election algorithms choose a process from a group of processors to act as a coordinator. If the coordinator process crashes due to some reasons, then a new coordinator is elected on other processor. Election algorithm basically determines whether a new copy of coordinator should be restarted.
Election algorithm assumes that every active process in the system has a unique priority number. The process with highest priority will be chosen as a new coordinator. Hence, when a coordinator fails, this algorithm elects that active process which has highest priority number. Then this number is sent to every active process in the distributed system.
We have two election algorithms for two different configurations of distributed system.

① The Bully Algorithm
This algorithm applies to system where every process can send a message to every other process in the system.
Algorithm - Suppose process P sends a message to the coordinator.
a) If coordinator does not respond to it within a time interval T, then it is assumed that coordinator has failed.

b) Now process P sends elect message to every process with high priority number.
c) It waits for responses if no one respond for time interval T then process P elects itself as a coordinator.
d) Then it sends a message to all lower priority no. processes that it is elected as the coordinator.
e) However, if an answer is received within time T from any other process Q,
i) process P again waits for time T' to receive another message from Q that it has been elected as coordinator
ii) If Q doesn't respond within T' then it is assumed to have failed and algorithm is restarted.

(ii) The Ring Algorithm -
This algorithm applies to systems organized as a ring (logically or physically). In this algorithm we assume that the link between the process are unidirectional and every process can message to the process on its right only.
Data structure that this algorithm uses is active list, a list that has priority number of all active processes in the system.
Algorithm —
a) If process P1 detects a coordinator failure, it creates new active list which is empty initially. It sends election

message to its neighbour on right and adds number 1 to its active list.

b) If process P2 receives message elect from processes on left, it responds in 3 ways -

a i) If message received does not contain 1 in active list then P1 adds 2 to its active list and forwards the message.

ii) If this is the first election message it received or sent, P1 creates new active list with numbers 1 and 2. It then sends election message 1 followed by 2.

iii) If process P1 receives its own election message 1 then active list for P1 now contains numbers of all the active processes in the system. Now process P1 detects highest priority number from list and elects it as the new coordinator.

Q. 13. Explain group communication in message passing.

Ans: A group is a collection of processes that act together in some system or user specified way. The key property that all groups have is that when a message is sent to the group, itself all the members of the group receive it. It is a form of one to many communication.

One - to - many

a) Group management
b) Group Addressing
c) Message delivery to receiving process
d) Buffered and unbuffered multicast
e) Flexible reliability in multicast communication
f) Atomic multicast
g) Group communication primitives

Single sender and multiple receiver is also known as multicast communication. Broadcast is a special case of multicast communication.

a) Group management

In case of group communication the communicating processes forms a group. Such a group may of either of 2 types.

i) Closed group - is the one in which only member can send message, outside procds cannot send message to the group as whole but can send to single member of a group.

ii) Open group - is one in which any process

In the system can send the message to group as a whole.
Message passing provides flexibility and create groups dynamically and allow process join or leave group dynamically.
Centralized group server is used for this purpose but suffers from poor reliability and scalability.

b) Group Addressing
Two level naming scheme is used for group addressing.
Multicast address is a special n/w address, packet is delivered automatically hence to all m/cs listening to address.
It broadcast address the all m/cs has to check if packet is intended for it or else simply discard so broadcast is less efficient.
If network doesn't support any addressing among two then one to one communication is used. First two methods send single packet but one to many sends many, creating much traffic.

c) Message delivery to receiving process
User application uses high level group names in program.
Centralized group server maintains a mapping of high level group names to their low level names, when sender sends message to the group. With high level name, kernel of server m/c asks the group servers low level name and list of process identifiers.

When packet reaches m/c kernel, that m/c extract list of process IDs and forwards message to those processes belonging to its own m/c. If none of ID's is matching, packet is discarded.

d) Buffered and unbuffered multicast

Send to all semantics : A copy of message is sent to each process of multicast group and message is buffered until it is accepted by process.

Bulletin board semantics : Message to be multicast is addressed to channel instead of each process.

e) ~~In~~ multicast ~~mec~~

e) Flexible Reliability in Multicast communication

In multicast mechanism there is flexibility for user definable reliability.

Degree of reliability is expressed in 4 forms.

i) 0 - Reliable :- No response is expected at all

ii) 1 - Reliable :- Sender expects response from any one of the receiver.

iii) ~~multi~~ m of n Reliable :- Sender expects response from m of n receiver.

iv) All reliable :- Sender expects response from all.

f) Atomic multicast

All or nothing property all reliable form requires atomic multicast facility. It should provide flexibility to sender to specify whether atomicity is required or not. Message passing system should support both atomic & non-atomic facility.

It is difficult to implement atomic

multicast if sendee or receivee fails.
Solution to this is fault tolerated atomic multicast protocol. In this protocol, each message has message identifier field to distinguish message and one field to indicate data message in atomic multicast message.

g) Group communication primitives
In one to one and one to many send or has to specify destination address and pointer to data so some send primitives can be used for both. But some systems use send group primitives because —
i) It simplifies design and implementation.
ii) It provides greater flexibility.

Q.14 Discuss the issues in design of Distributed Systems.

Ans: The issues in designing distributed systems are -

i) Heterogeneity

The Internet enable users to access services and run applications of a heterogenous collection of computers and networks. 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.

ii) Openness

The openness of a computer system is the characteristic that determines whether the system can be extended & re-implemented in various ways. The openness of distributed systems is determined primarily by the degree to which new resource-sharing services can be added and be made available for use by a variety of client programs.

iii) Security

Many of the information resources that are made available and maintained in distributed systems have a high intrinsic value to their users. Their security is therefore of considerable importance. Security for information resource has three components : confidentiality ,

integrity, and availability.

iv) Scalability

Distributed systems operate effectively and efficiently at many different scales, ranging from a small intranet to the Internet. A system is described as scalable if it will remain effective when there is significant increase in the number of resources and the number of users.

v) Failure Handling

Computer systems sometimes fail. When faults occur in hardware or software, programs may produce incorrect results or may stop before they have completed the intended computation. Failures in a distributed system are partial - that is, some components fail while others continue to function - therefore the handling of failures is particularly difficult.

vi) Concurrency

Both services and application provide resource that can be shared by clients in a distributed system. Therefore any programmer who takes an implementation of an object that was not intended for use in a distributed system must do whatever is necessary to make it safe in a concurrent environment.

vii) Transparancy

It can be achieved at two different levels: Easiest is to hided the distribution from the users. The concept of transparency can be applied to

several aspects of a distributed system.
a) Location transparency
b) Migration transparency
c) Replication transparency
d) Concurrency transparency
e) Parallelism transparency

viii) Quality of service
Once users are provided with the functionality that they require of a service, such as the file service in a distributed system, we can go on to ask about the quality of the service provided. Resource availability is also important aspect of service quality.

ix) Reliability
One of the original goals of building distributed systems was to make them more reliable than single-processor systems. In general, the more copies that are kept, the better the availability. But the greater the chance that they will be inconsistent, especially if updates are frequent.

x) Performance
Always the hidden data in the background is the issue of performance. Building a transparent, flexible, reliable distributed system, more important lies in its performance. In particular, when running a particular application on a distributed system, it should not be appreciably worse than running the same application on a single processor.

Q.15. Explain different file access ~~methods~~ models.

Ans: The specific client's request for accessing a particular file is serviced on the basis of the file accessing model used by the distributed file system.
The file accessing model basically depends on
i) the unit of data access
ii) the method used for accessing remote files.
On the basis of the unit of data access, following file access models might be used in order to access the specific file.
a) File - level transfer model
b) Block - level transfer model
c) Byte - level transfer model
d) Record - level transfer model

a) File - level transfer model
In this model, the complete file is moved while a particular operation necessitates the file data to be transmitted all the way through the distributed computing network amongst client and server. This model has better scalability and is efficient.

b) Block - level transfer model
In this model, file data transfers through the network amongst client and a server is accomplished in units of file blocks. In short, the unit of data - transfer in

In block-level transfer model is file block. The block-level transfer model might be used in distributed computing environment comprising several diskless workstations.

iii) Byte level transfer model
In this model, file data transfers the network amongst client and a server is accomplished in units of bytes. In short, the unit of data transfer in byte-level transfer model is bytes. The model offers more flexibility in comparison to the other file transfer models since, it allows retrieval and storage of an arbitrary sequential subrange of a file. The major disadvantage of this model is the trouble in cache management because of the variable-length data for different access requests.

iv) Record-level transfer model
This model might be used in the file models where the file contents are structured in the form of records. In record-level transfer model, file data transfers through the network amongst client and a server is accomplished in units of records. The unit of data transfer in record-level transfer model is record.

Q.16. Discuss the various algorithms to carry out clock synchronization?

Ans: Clock synchronization algorithms can be broadly classified as Centralized & Distributed.

Centralized Algorithms :-

In centralized clock synchronization algorithms one node has a real-time receiver - this node, called the time server node whose clock time is regarded as the reference time. The goal of these algorithms is to keep the clocks of all other nodes synchronized with the clock time of the time server node. Depending on the role of the time server node, centralized clock synchronization algo. are again of 2 types —

i) Passive time server ii) Active time server

i) Passive time server Algorithm

In this method, each node periodically sends a message to the time server. When the time server receives the message, it quickly responds with a message ("time = T"), where T is the current time in the clock of time server node.

Assume that when the client node sends the "time = ?" message, its clock time is T_0, and when it receives the "time = T" message, its clock time is T_1. Since T_0 and T_1 are measured using the same clock, in the absence of any other information, the best estimate of the time required for propagation

of the message "time = T" from server node to client's node is $(T_1 - T_0/2)$. Therefore, when the reply is received at the client's node, its clock is readjusted to $T + (T_1 - T_0/2)$.

ii) Active Time Server Algorithm

In this approach, the time server periodically broadcasts its clock time ("time = T"). The other nodes receive the broadcast message and use the clock time in the message for correcting their own clocks. Each node has a prior knowledge of the approx. time (T_a) required for the message propagation from server to itself. Therefore, when a broadcast message is received at a node, the node's clock is readjusted to the time $T + T_a$.

Distributed Algorithms

We know that externally synchronized clocks are also internally synchronized. ie., if each node's clock is independently synchronize with real time, all the clocks of the system remain mutually synchronized. Therefore, a simple method for clock synchronization may be to equip each node of the system with a real time receiver so that each node's clock can be independently synchronized with real time. Theoretically, internal synch. is not required. However, in practice, due to inherent inaccuracy of real-time clocks, different real-time clocks produce different time. Therefore it is performed for better accuracy. One of the two approaches used include —

1) Global Averaging Algorithm

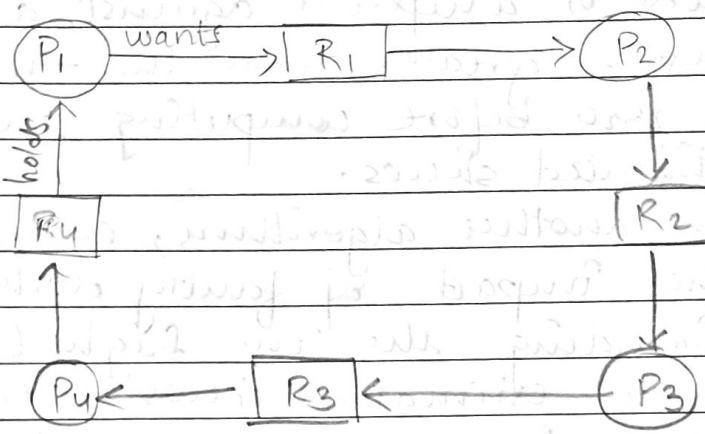
In this approach, the clock process at each node broadcasts its local time in the form of a special "resync" message when its local time equals $T_0 + iR$ for some integer i, where T_0 is a fixed time in the past agreed upon by all nodes and R is a system parameter that depends on such factors as the total number of nodes in the system, maximum allowable drift rate, and so on. i.e. - resync message is broadcast from each node at the beginning of every fixed length resynchronization interval. However, since the clocks of different nodes run slightly different rates, these broadcasts will not happen simultaneously from all nodes. After broadcasting the clock value, the clock process of a node waits for time T, where T is a parameter to be determined by the algorithm - During this waiting period, the clock process records the time, according to its own clock, when the message was received. At the end of the waiting period, the clock process estimates the skew of its clock w.r.to each of the other nodes on the basis of the times at which it received resync messages. It then computes a fault-tolerant average of the next resynchronization interval

→

ii) The global averaging algorithms differ mainly in the manner in which the fault-tolerant average of the estimated skews is calculated.
Two commonly used algorithms are
a) The simplest algorithm is to take the average of the estimated skews and use it as the correction for the local clock. However, to limit the impact of faulty clocks on the average value, the estimated skew with respect to each node is compared against a threshold, and skews greater than the threshold are set to zero before computing the average of the estimated skews.
b) In another algorithm, each node limits the impact of faulty clocks by first discarding the m highest and m lowest estimated skews and then calculating the average of the remaining skews, which is then used as the correction for the local clock. The value of m is usually decided based on the total number of clocks (nodes).

Q.17. What is a deadlock in a distributed system?

Ans: A deadlock is a condition in a system where a set of processes (or threads) have requests for resources that can never be satisfied.
Essentially a process cannot proceed because it needs to obtain a resource held by another process but it itself is holding a resource that the other process needs.



Deadlock

More formally, Coffman defined 4 conditions to have to be met for a deadlock to occur in a system:
i) Mutual exclusion - A resource can be held by at most one process.
ii) Hold & wait - Processes that already hold resources wait for another resource
iii) Non-preemption - A resource once granted cannot be taken away.
iv) Circular wait - Two or more processes are

waiting for resources held by one of the other processes.
A directed graph model used to record the resource allocation state of a system. This state consists of n processes, $P_1 \ldots P_n$ and m resources, $R_1 \ldots R_m$.
In such a graph:
$P_1 \rightarrow R_1$ means that resource R_1 is allocated to process P_1.
$P_1 \leftarrow R_1$ means that resource R_1 is requested by process P_1.
Deadlock is present when the graph has directed cycles.
The same conditions for deadlock in uniprocessor apply to distributed systems. Unfortunately, as in many other aspects of distributed systems, they are harder to detect, avoid and prevent.
Four strategies can be used to handle deadlocks -
a) ignorance - assume that deadlock will never occur.
b) detection - let a deadlock occur, detect it and deal with it.
c) prevention - make a deadlock impossible by granting requests.
d) avoidance - choose resource allocation carefully so that deadlock will not occur.

Q.18. What is critical section? Explain mutual exclusion.

Ans: Process synchronization is the task of coordinating the execution of processes in a way that no two processes can have access to the same shared data and resources.

It is specially needed in a multiprocess system when multiple processes are running together, and more than one processes try to gain access to the same share resource or data at the same time. This can lead to the inconsistency of shared data. So the change made by one process not necessarily reflected when other processes accessed the same shared data. To avoid this type of inconsistency of data, the processes need to be synchronized with each other.

For example, process A changing the data in a memory location while another process B is trying to read the data from the same memory location. There is a high probability that data read by the second process will be erroneous.

Here, are four essential elements of the critical section:

i.) Entry Section: It is part of the process which decides the entry of a particular process

ii) Critical Section : This part allows one process to enter and modify the shared variable.

iii) Exit Section : Exit section allows the other process that are waiting in the Entry section, to enter into the critical section.

iv) Remainder Section : All other parts of the code, which is not in critical, Entry, and Exit section, are known as the Remainder Section.

A critical section is a segment of code which can be accessed by a signal process at a specific point of time. The section consists of shared data resources that required to be accessed by other processes.

- The entry to the critical section is handled by the wait() function, and it is represented as P()
- The exit from a critical section is handled by the signal() function represented as V().

In the critical section, only a single process can be executed. Other processes, waiting to execute their critical section, need to wait until the current process completes its execution.

Mutual exclusion is a special type of binary semaphore which is used for controlling access to the shared resource.

It includes a priority inheritance mechanism to avoid extended priority inversion problems. Not more than one process can execute in its critical section at one time
Mutual exclusion implies that only one process can be inside the critical section at any time. If any other processes require the critical section, they must wait until it is free.

Q.19. What is cloud computing? Explain the principles of cloud computing.

Ans: Cloud computing is the delivery of different services through the Internet. These resources include tools and applications like data storage, servers, databases, networking and software.
Cloud computing is a popular option for people and businesses for a number of reasons including cost savings, increased productivity, speed and efficiency, performance and security.
Cloud computing is named as such because the information being accessed is found remotely in the cloud or a virtual space. The companies that provide cloud services enable users to store files and applications on remote servers and then access all the data via the Internet. This means the user is not required to be in a specific place to gain access to it, allowing the user to work remotely.
Cloud computing can be both public and private. Public cloud services provide their services over the Internet for a fee. Private cloud services to a certain number of people. These services are a system of people. These services are a system of networks that supply hosted services. There is also a hybrid option, which combines elements

of both the public and private services. Cloud computing is different from the traditional web service because of its principles. These principles are :-

i) Resource pooling:
Cloud computing provides harness large economies of scale through resource pooling. They put together a vast network of servers and hard drives and apply the same set of configurations, protections and works for them.

ii) Virtualization:
Users donot have to ~~create~~ care about the physical states to their hardware nor worry about hardware compatibility.

iii) Elasticity:
Addition of more hard disk space or server bandwidth can be done with just a few clicks of the mouse on-demand. Geographical scalibility is also available in cloud computing.

iv) Automatic / easy resource deployment:
The user only needs to choose the types and specifications of the resources he require and the cloud computing provider will configure and set them up accordingly.

v) Metered billing:
Users are charged only for what they use.

Q. 20 What is a deadlock in a distributed system?

Ans. A deadlock is a condition in a system where a set of processes (or threads) have requests for resources that can never be satisfied. Essentially, a process cannot proceed because it needs to obtain a resource held by another process but it itself is holding a resource that the other process needs. More formally, Coffman defined four conditions have to be met for a deadlock to occur in a system:

i) Mutual exclusion
A resource can be held by at most one process.

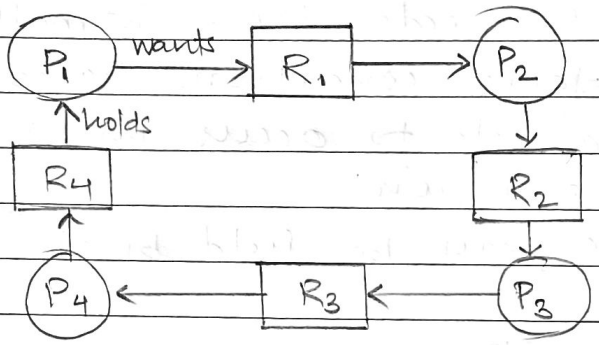
ii) Hold & wait
Processes that already hold resources can wait for another resource.

iii) Non-preemption.
A resource, once granted, cannot be taken away.

iv) Circular wait
Two or more processes are waiting for resources held by one of the other processes.

A directed graph model is used to record the resource allocation state of a system. This state consists of n processes $P_1 \dots P_n$

and m resources, $R_1 \ldots R_m$. In such a graph:
$P_1 \rightarrow R_2$ means that resource R_1 is allocated to process P_1.
$P_1 \leftarrow R_1$ means that resource R_1 is requested by process P_1.
Deadlock is present when the graph has a directed cycles. An example is shown in figure below. Such a graph is called a Wait-For Graph (WFG).



The same conditions for deadlock in uniprocessors apply to distributed systems. Unfortunately, as in many other aspects of distributed systems, they are harder to detect, avoid and prevent. Four strategies can be used to handle deadlock:

i) Ignorance
Ignore the problem; assume that a deadlock will never occur. This is a surprisingly common approach.

ii) Detection
Let a deadlock occur; detect it and then deal with it by aborting and later

restarting a process that causes deadlock.

iii) Prevention

Make a deadlock impossible by granting requests so that one of the necessary conditions of deadlock doesnot hold.

iv) Avoidance

Choose resource allocation carefully so that deadlock will not occur. Resources request can be honored as long as the system remains in a safe (non-deadlock) state after resources are allocated.