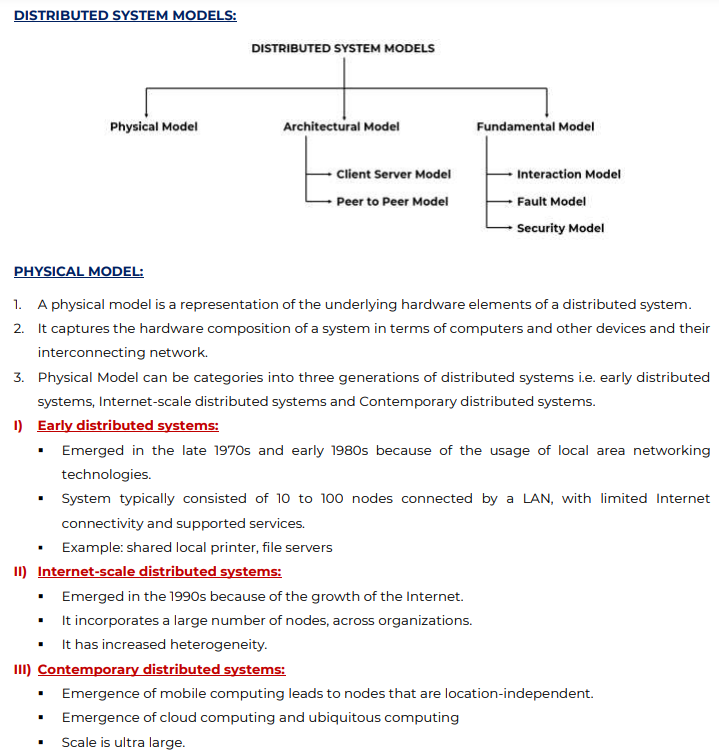
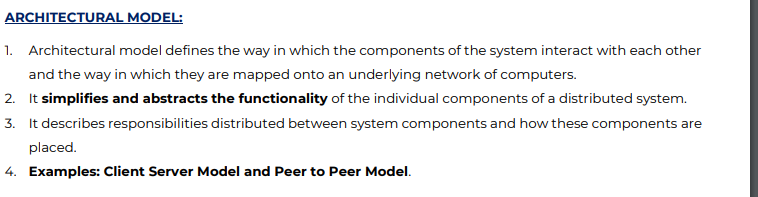
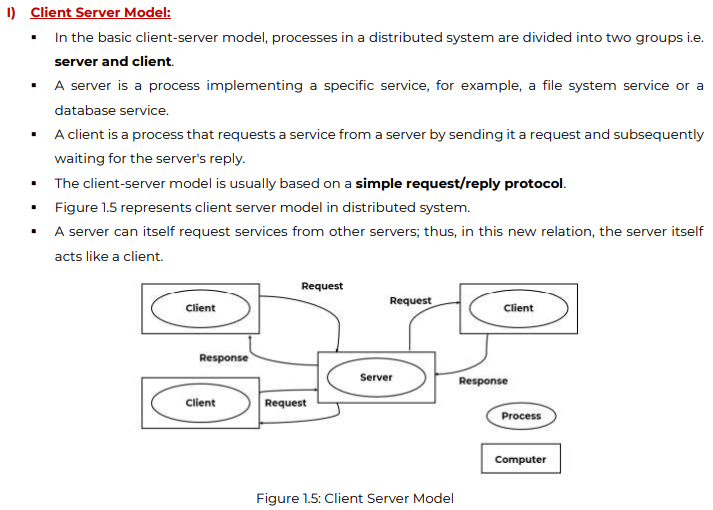
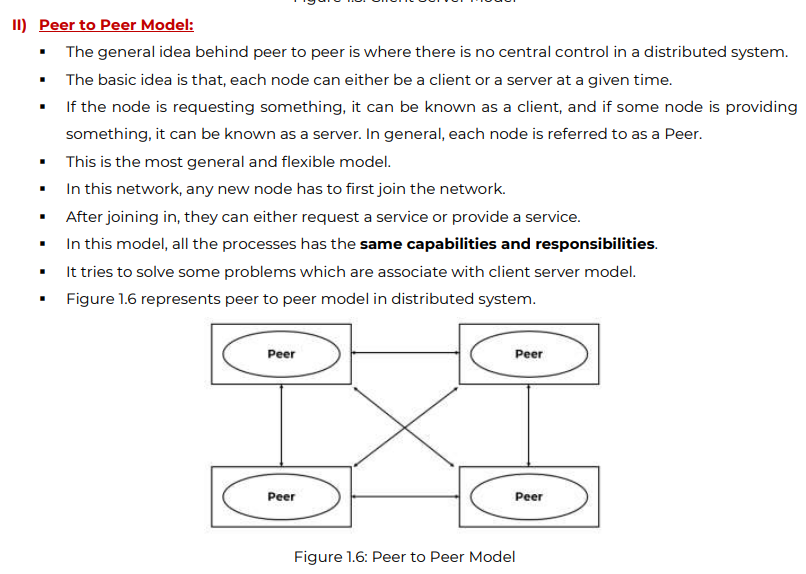
**Module 01** : Introduction to Distributed Systems

What are various system models of distributed system? Architecture Model Types? Explain with Suitable Diagram.

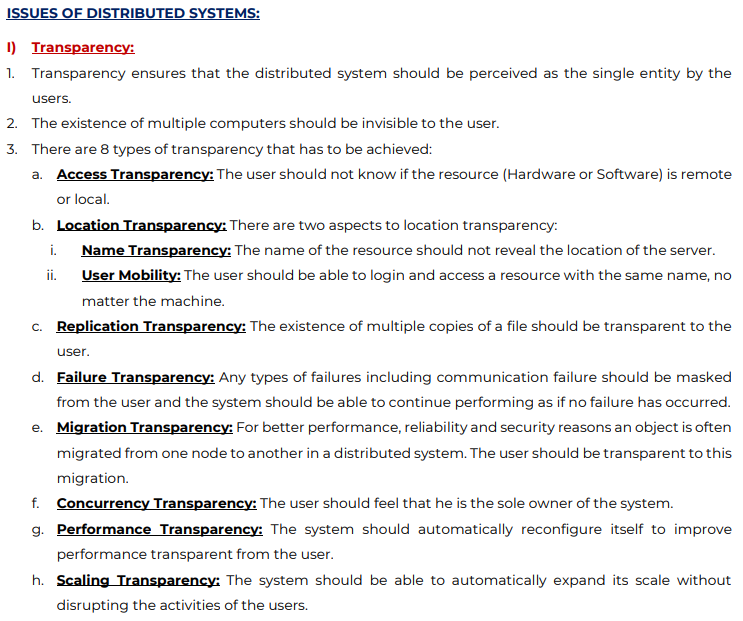


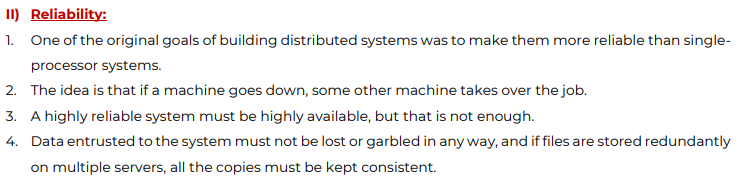


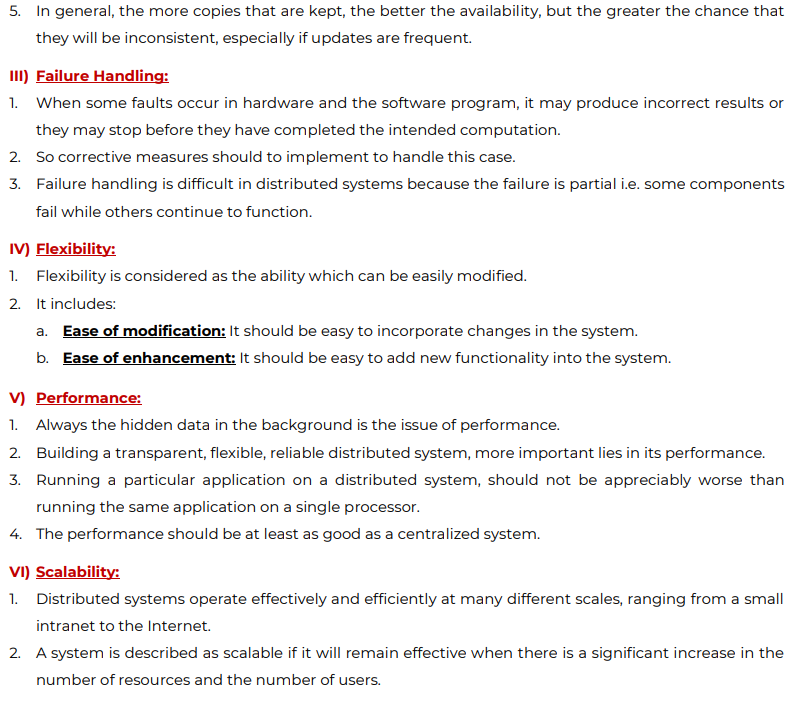


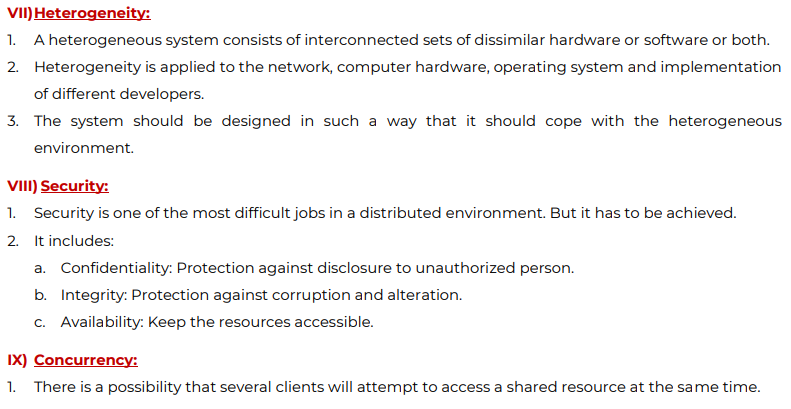


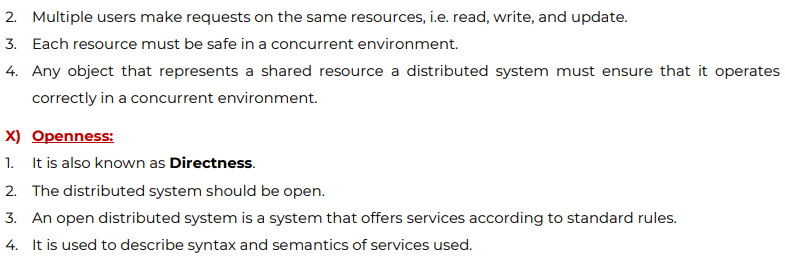
What are the various issues of distributed system



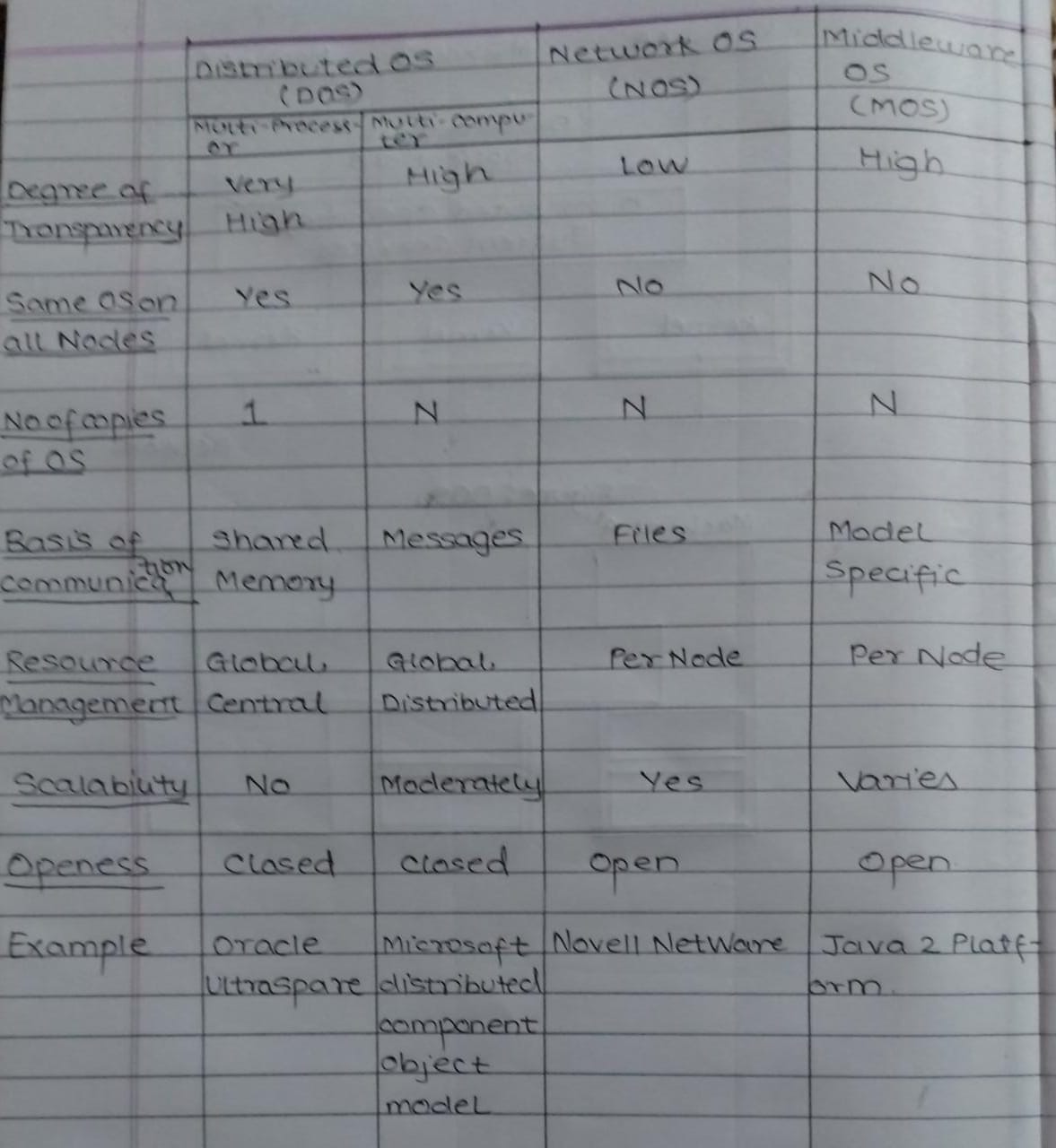








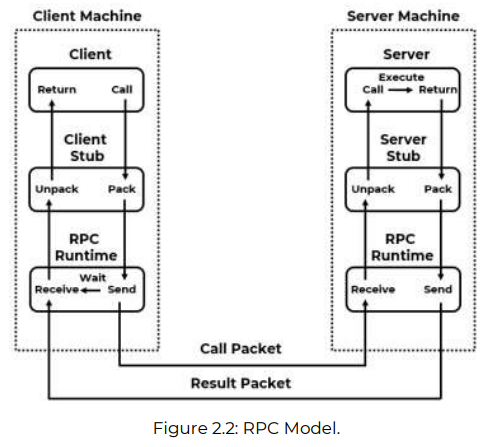
Differentiate between DOS, NOS, Middleware OS.



**Module 02** : **Communication**

***RPC***

* Remote Procedure Call is an inter-process communication used in client-server applications.
* It is also called Remote Function/Subroutine Call.
* The basic idea of RPC is to make a rpc call look transparent. The calling procedure should not be aware that the called procedure is executing on a different machine.
* The caller or the client sends a request. The request is passed to the client stub which packs the request with parameters and then passed to the clients kernel which is then sent to the server to receive.
* The message travels through the network. Once it is received by the servers kernel machine it is then unpacked by the server stub which reads the parameters of the message and sends it to the server to execute the request.
* The server executes and returns the result back to the server stub which packs the response with parameters and back to the server kernel to send it back to the client machine.
* It is received at the clients machine and unpacked at the client stub and finally the response is received at the clients machine.
* This process happens remotely without the user being aware of it, hence achieving transparency.



The task of the stub is to pack the specification of the target procedure and the arguments into a message. When it receives this, it unpacks the result and passes it to the client.

RPC Runtime : It handles transmission of messages across the network between client and server machine including encryption, routing acknowledgement.

Advantages :

* RPCs support process and thread oriented models
* It achieves transparency by appearing as a single coherent system to the user.
* The code is minimum in case if one needs to re-develop the application.
* It can be used in distributed as well as local environments.

Disadvantages :

* There is no flexibility when it comes to hardware architecture.
* There is an increase in cost because of RPC

**Explain stream-oriented communication with suitable example**

* Communication is sharing or exchanging independent and complete units of information.
* A stream-oriented communication is a form of communication in which timing plays a crucial role
* It is also referred to as continuous streams of data.
* A medium is a means by which the data is conveyed. Example : Monitor Screen allows users to view Images, Videos etc.
* A data stream is a continuous representation of a stream of data.
* Streams are unidirectional and generally has a single source.

Transmission Modes :

1. Asynchronous Modes : The data items in a stream are transmitted one after the other, but there are no timing constraints on when the transmission items should take place.

Example : A file transferred as a data stream.

2. Synchronous Modes : There is a maximum end-to-end delay defined for each unit in a data stream

Example : Data sent from sensor to the operator.

3. Isochronous Modes : There is a maximum and minimum end-to-end delay.

Example : Representing Audio and Video.

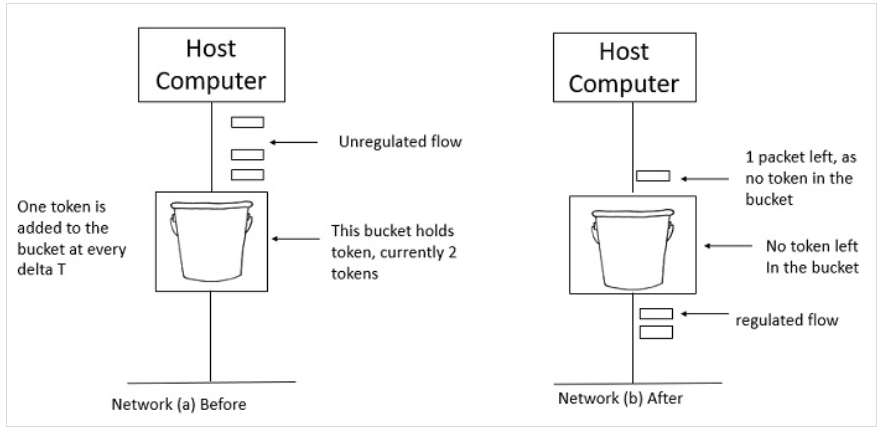
Stream Types :

* Simple : It consists of a single flow (e.g. audio or video)
* Complex : It consists of multiple data flows ( e.g. stereo audio)

The relation between the sub-streams in a complex stream is often also time independent. For example : For transmitting a movie, the stream could consist of a single video stream, along with two streams for transmitting the audio and another stream which could show subtitles.

Example : Token Bucket Algorithm

* It is easily implemented with a counter.
* The token is initialized to zero.
* Each time a token is added, the counter is incremented by 1.
* Each time a data is dispatched, the counter is decremented by 1.
* If the counter contains zero, the host cannot send data.

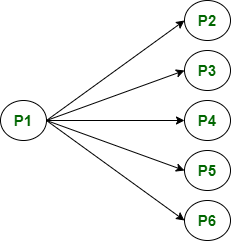


Short Note : Group Communication.

A group communication happens when a certain process sends a message to other processes in the system and can also communicate and share information with the other processes. There are three types of group communication in a distributed system : Broadcast – MultiCast and Unicast Communication.

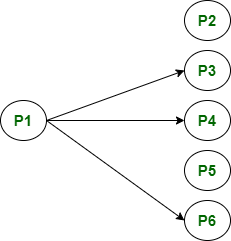
* Broadcast Communication

When a process wants to communicate with every other process in a distributed system is called as a broadcast C. The information or data shared by this process is seen by every other process. It is a useful method when a single information needs to be shared to every other node. It is very efficient and communication is very fast. However, it cannot treat a specific process individually.



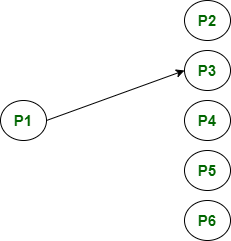
* Multicast Communication :

When a single process tries to communicate to a particular group of process from various processes in a distributed system. It is a useful method when a certain category or only few processes are required to share information. It is a selective method in which a list of processes are selected as a group. It is very effective method since not all processes need to be selected.



* Unicast Communication :

When a single process wants to communicate with only one process is called as unicast communication. Here the information shared is between two processes and no other process is involved. It increases overhead as it has to find exact process and then exchange data with it.



Mod 03 :

***What is Clock Synchronization? Explain Centralized Clock Synchronization Algorithms i.e Cristian and Berkeley.***

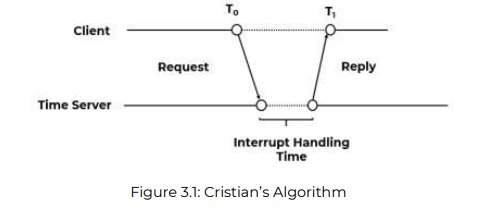
Clock Synchronization is a problem in computer science which deals with the idea that internal clocks of several computers may differ. Every computer needs a timer mechanism because during times of failure, when certain files are lost, time happens to be the only reference to the type of file that was lost. Having clock mechanism also helps to keep the system in order of events. The variance of time is called as clock skew. One major clock synchronization issue in a distributed system is that multiple clocks have to be synchronized from time to time. Physical clock synchronization can be classified as Centralized and Distributed.

Centralized Clock Synchronization :

* In a centralized system, a node called as a time server node has the real time and the other nodes in the distributed system need to update their clock time with the time server node.
* The goal is to maintain clock time with the clock time of time server node
* The drawback is that if the time server node fails or crashes, the nodes cannot update.
* Second drawback is that heavy burden is put on a single node.

It includes two algorithms : Cristian’s Algorithm and Berkeley Algorithm.

Cristian’s Algorithm :



* In Cristians algorithm, the node sends a request to the time server node for synchronizing clock time from a client process
* It works well with low-latency networks and where round trip time is short as compared to accuracy.
* Round Trip Time is the time duration between start of a request and end of the reply
* The nodes periodically send request to time server node.
* Now when the client sends a request for clock time to the client server, the time before it makes a request is T0, and T1 is the time which is a synchronized time given by the time server node.
* Tclient = Tserver + (T1-T0)/2

Where tclient is the current time of the client node, Tserver is the time of the server node which is the actual time of the server node.

T1-T0 is the combined time taken by the system to send a request, process it, update the time and send the reply back the client node. The time at the client side differs from the actual time by at least (T1-T0)/2. Using the above statement we can conclude that the time is at most at (T1-T0)/2 seconds. The total time is then calculated and updated at the client side.

Advantages :

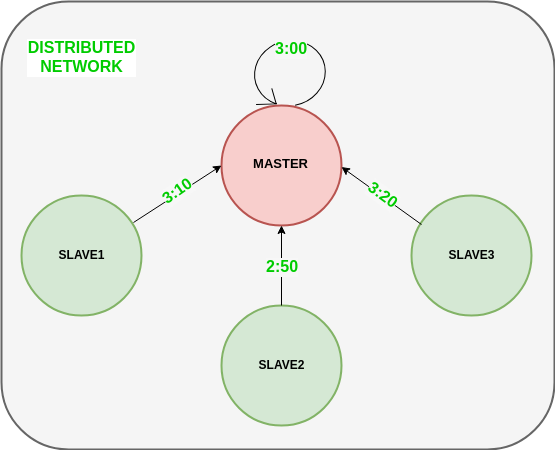
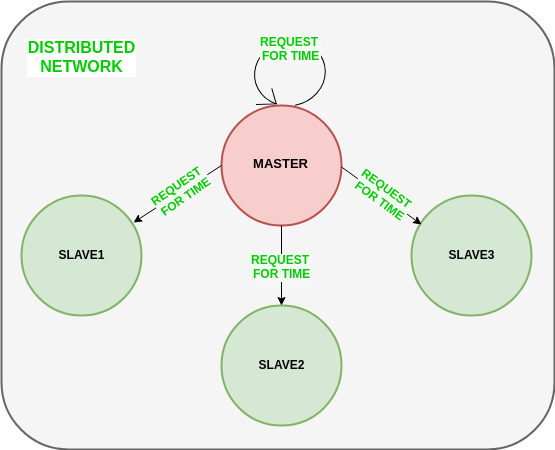
* No other information is sent to the server other than the request, which keeps the network away from congestion.
* The accuracy is very good.

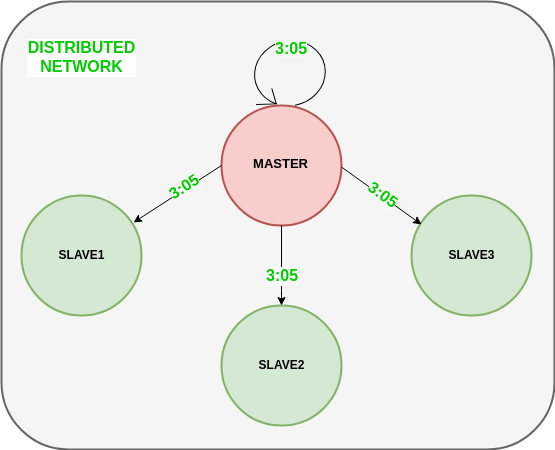
Disadvantages :

* It restricts the number of measurements for estimating the value.
* If the time serve node fails, no other node is able to take the task.

Berkley Algorithm :

* Berkely Algorithm is a clock synchronization algorithm which assumes that there is no node present in the system that has an accurate clock time.
* It was developed by Gussella and Zati in University of California, Berkeley in 1989.
* A server node is present which is the master node and the rest of the nodes are slave nodes.
* The server node sends a message to all the nodes present in the system to send their clock times
* After each node in the system receives this message, it sends its respective clock time to the server node.
* The server node then computes the average of these clock times and the resulting clock time becomes the time which all the nodes in the system need to be synchronized to.
* The server node changes its own time with this time and instead of sending the clock time to the other nodes it sends the clock time that needs to be adjusted to this averaged clock time.
* It can be a positive or a negative value.



* 

Advantages :

* The time at all the nodes in the system is adjusted at the same time
* It is periodic

Disadvantages :

* Only one node is responsible to calculate the time, so in case if this node crashes it affects the system time
* It ignores significant outliers in the calculation of average time difference.

***What is a logic clock? Why are logic clocks required in distributed systems? How does Lamport Synchronize logical clocks? Which events are said to be concurrent in Lamport timestamps.***

A logical clocks refers to implementing a protocol on all machines in distributed system so that machines are able to order consistent ordering of events within some virtual timestamp. In logical clock systems each process has two data structures : logical local time and logical global time. Logical local time is used by the process to mark its own events, and logical global time is the information about the global time.

Lamport’s Logical Clocks :

* Clocks need not be synchronized correctly.
* If two processes do not interact, it is not necessary that their clocks be synchronized because the lack of synchronization would not be observable and thus it does not cause problem.
* It is not important that all processes agree on what the actual time is, but that they agree on the order in which events occur.
* Lamport clocks are a simple technique used for determining the order of events in a distributed system.
* Rules of Lamport’s Logical Clocks :

a. Happened Before Relation :

- If a and b are events in the same process and a occurs before b, a -> b

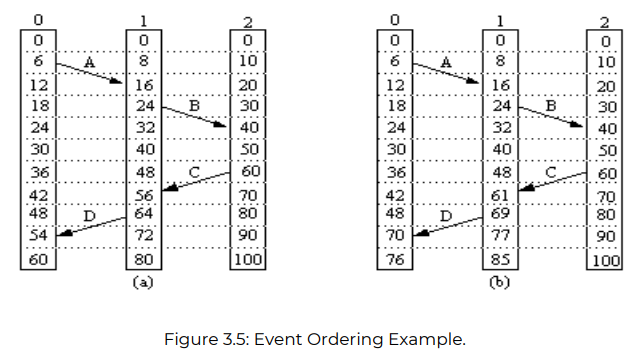
- If a and b belong to two different processes and a sends message to b, then a -> b

- If a-> b and b->c then a-> c

b. Logical Clocks Concept

- Maintaining a proper clock or a set of perfectly synchronized clock is difficult.

- So, local timestamp is given for happened before relation.



Consider three processes 0, 1 and 2. For process 0 the time ticked is 6, for 1 its 8 and for 2 its 10. The clocks run at constant rate but different due to difference in the crystals. At time 6, process 0 sends message A to process 1 and is received at time 16. Time taken for the transfer is 10. Similarly process 1 sends message B to process 2 and is received at time 40. Time taken for transfer is 16. Now message C is send from process 2 to process 1 it ticks 16 to reach and similarly message D takes 10 ticks to reach. This value is impossible and such a situation must be prevented. Solution for this is Lamport happen before relation. Since message C left at 60 so it must arrive at 61 or later and similarly message D left at 69 and should reach from 70 or later.

***Election Algorithm***

It chooses a process from a group of processors and appoints one process as the coordinator. If the coordinator process crashes for some reason, it appoints a new process or elects a new process as coordinator.

It assumes that every active process in the system has a unique priority number. The process with the highest priority number is chosen as the coordinator. If the coordinator fails or crashes, then it chooses the process which has highest priority from the available processes.

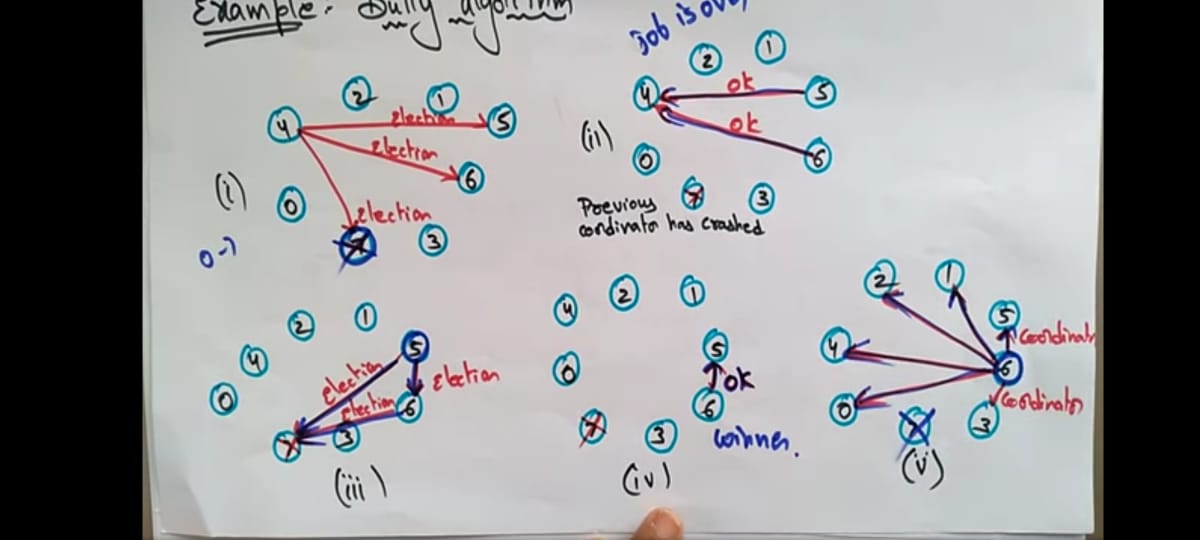
After the selection of the coordinator, every other process is informed about the new elected coordinator.

Election Algorithms : Bully Algorithm and Ring Algorithm

Bully Algorithm : It is an election algorithm which is used to elect or appoint a new coordinator. It is an algorithm applied where all the processes are able to interact with every other process. A coordinator is responsible to perform functions needed by other processes.

Let’s assume that P is a process that wants to send a message to the coordinator

* If the process P does not receive any message from the coordinator, it assumes that the coordinator node has crashed
* An election message will be sent by the process P to all other processes that have higher priority from process P.
* If the process P does not receive any messages from these nodes, then it chooses itself as the coordinator. Because it assumes that other nodes have failed or crashed
* After which, it sends a message to all its lowers processes that it has become the new coordinator.
* But, if the process P receives a message from a process say Q. Then Q sends an acknowledgement message to P. Process Q sends an election message to processes having higher priority than Q and so on. Until the process with the highest priority is reached, this continues
* When a process with the highest priority is selected as the coordinator. It sends message to all the other processes informing about the new coordinator



Advantages :

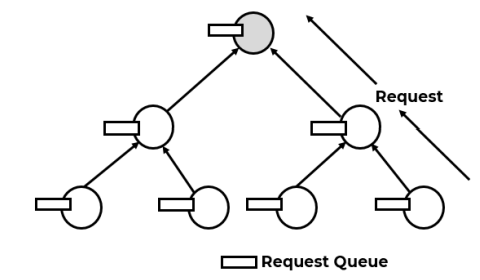
* If any nodes fail, other processes take over as the position of coordinator
* It is simple to understand and implement

Disadvantages :

* When communication a lot of messages is sent to other processes, which can congest the network.

***Raymonds Tree Algo.***

* It is a lock based algorithm for mutual exclusion in which a site is allowed to enter the critical section if it has the token.
* It forms a tree data structure to achieve this in which the tree is directed.
* Every node has a parent pointer and all the nodes are pointing to the parent pointer.
* The parent pointer is the one holding the token.
* The holder of the taken is allowed to enter the critical section
* Each node also has a FIFO Queue of requests.



* As the token moves across node, the parent pointer changes and the nodes directing to the nodes also change depending on the position of the parent node in the tree.

Requesting a Token:

* The node adds “self” in its request queue.
* The request is the forwarded to the parent of that node.
* If the parent node doesn’t hold the token and has not sent any requests to get the token, it sends a request to its parent node and so on.
* The process continues until a parent node with a token is found.

Releasing a Token:

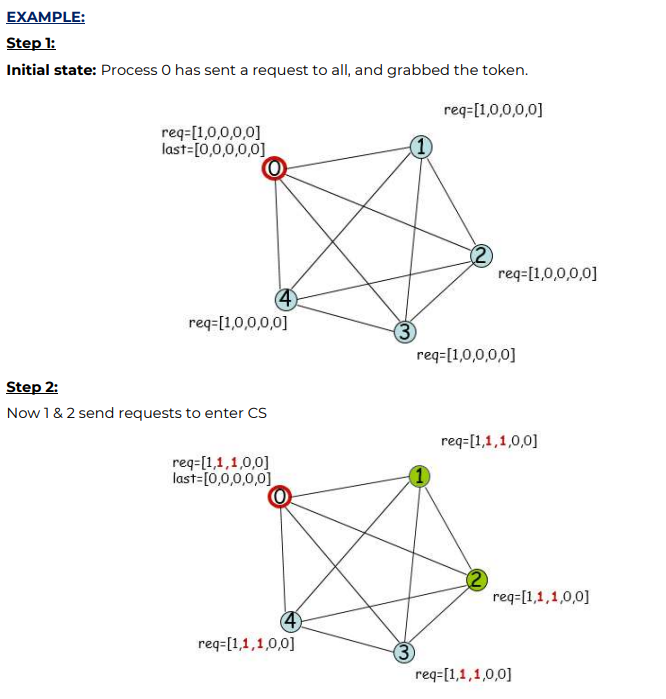
* Once the request is reached to the token holder, the parent node waits until it is done with the critical section.
* Once done, it forwards the token to the node in its request queue.
* It will dequeue the element from the queue and once the token is passed, it will change its parent pointer pointing to that node.
* If in the request queue, a “self” node is found, then that node will enter the critical section otherwise the same process of dequeue and changing the parent pointer will continue until the “self” in the request queue is found.
* Once it is found all the tree is directed to that node following the same principle as mentioned.

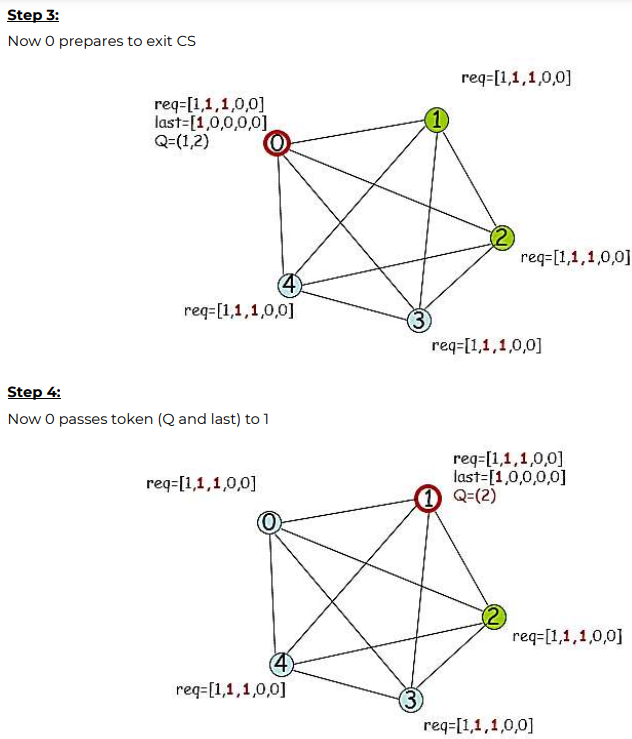
Suzuki-Kasami’s Broadcast Algorithm. Explain with example.

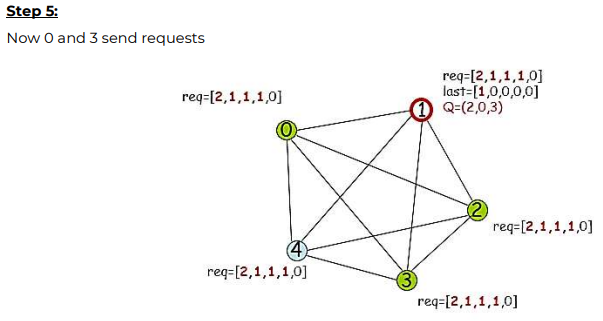
* It is a token based algorithm
* It is used to achieve mutual exclusion
* It consists of a request queue which consists of processes that have requested for critical section and last queue which indicates the processes that had critical section.
* The processes holding the token is the only process able to enter critical section.

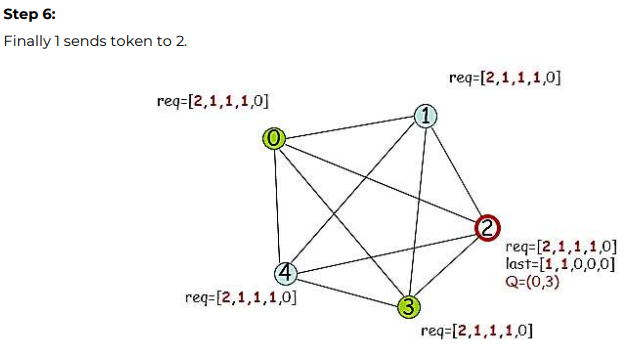
Algorithm :

* Lets consider there are many processes in the system.
* A single process is holding a token among these processes say Process A.
* Now process B and C want to enter the critical section and send request to Process A.
* All these processes hold a request queue in which the ith position of A B and C are updated by one.
* And Q which is the queue maintained for processes requesting critical section. Is updated at A as B and C.
* Now that A has finished it will check Q and pass A the token. Before passing the token it updates the last queue and sends the last, Q and token to the user. All other processes holding the request queue have the same updates
* The same process continues.









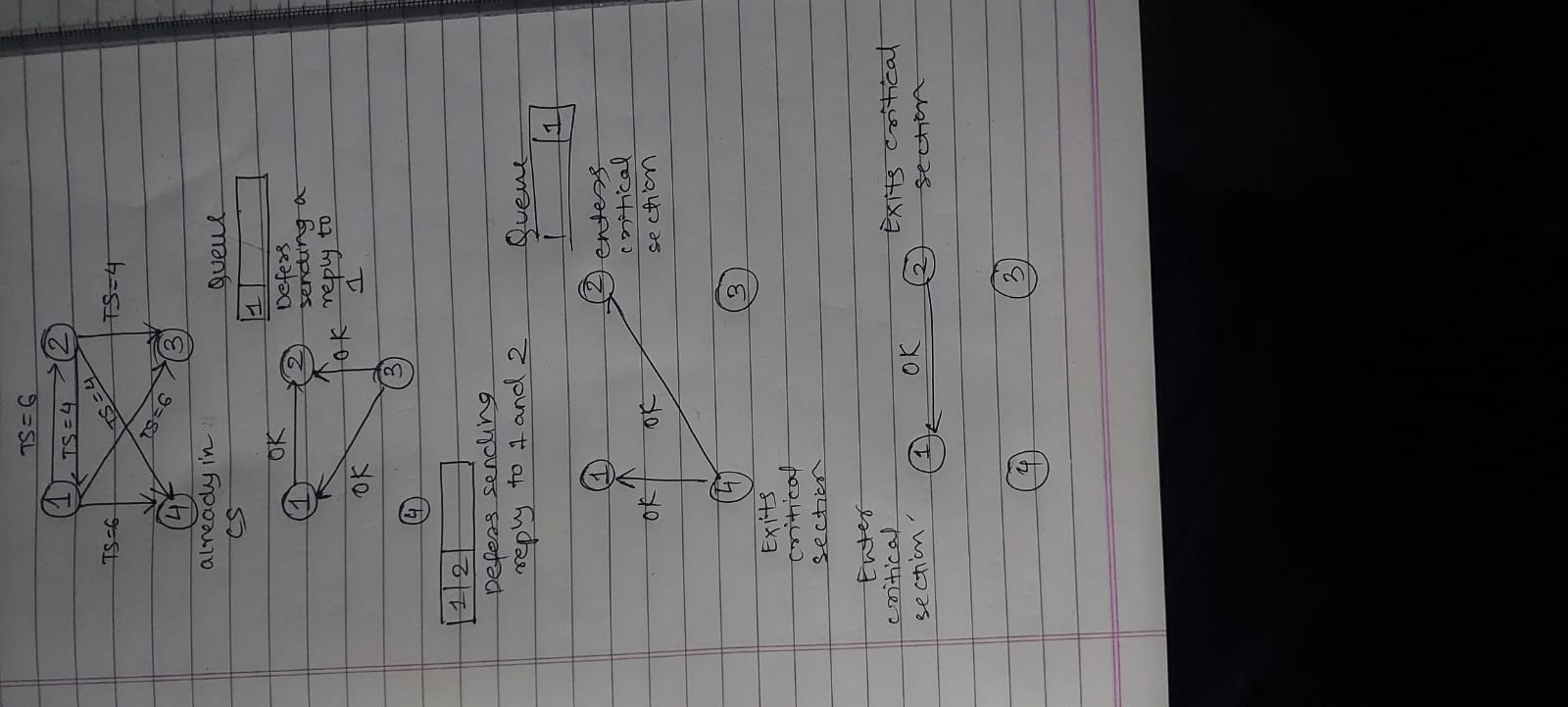
***Ricart-Agrawala’s Algorithm. Or Distributed Algorithm for Mutual Exclusion ? Adv and Disadv over Centralized Algorithms?***

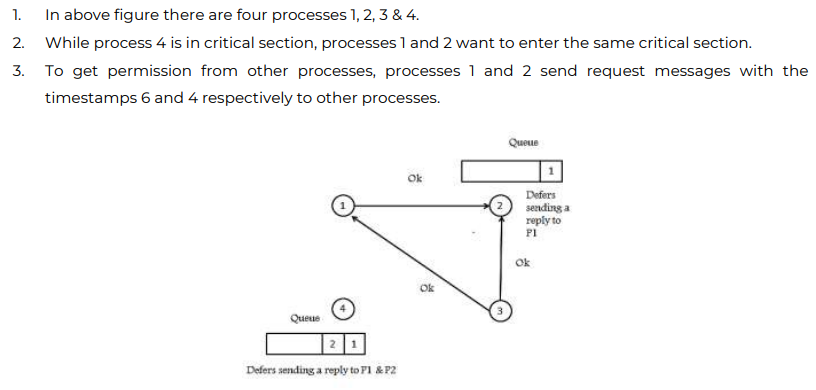
Whenever a process enters a critical section it broadcasts a message that it is entering the critical section. This message includes

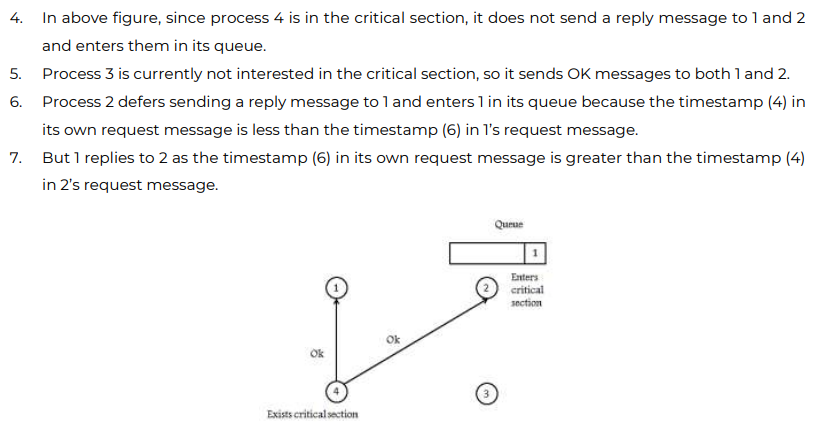
* Process Identifier
* Name of CS
* A unique timestamp

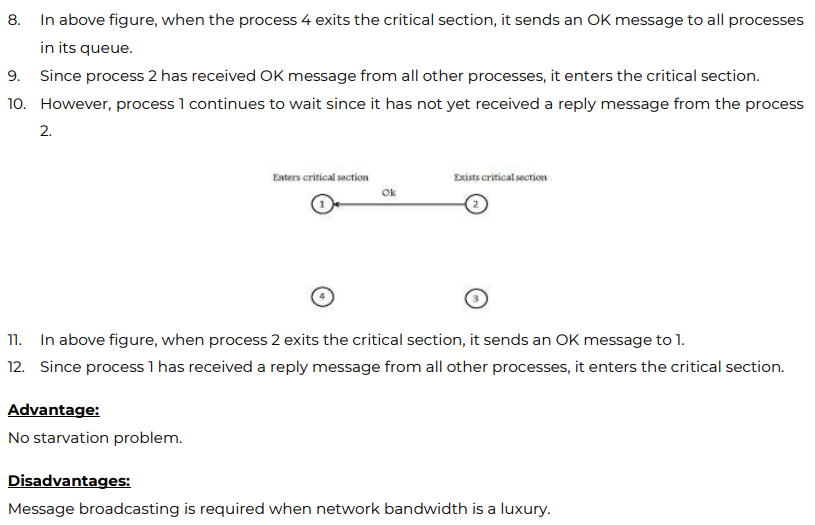
On receiving a request message for CS the process will do one of the following :

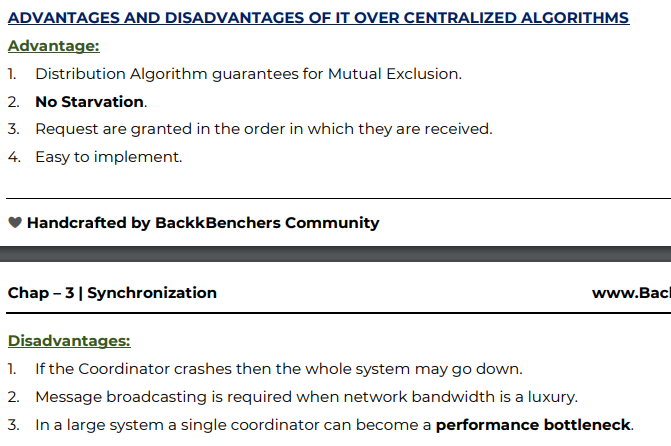
* If it is currently executing in CS it will not respond
* If the process is waiting for CS it will compare the timestamp of its own process to the requesting message. If the timestamp of the process is greater than the timestamp of the requesting message it will reply back, otherwise it won’t reply
* In any other case, the process will reply back.

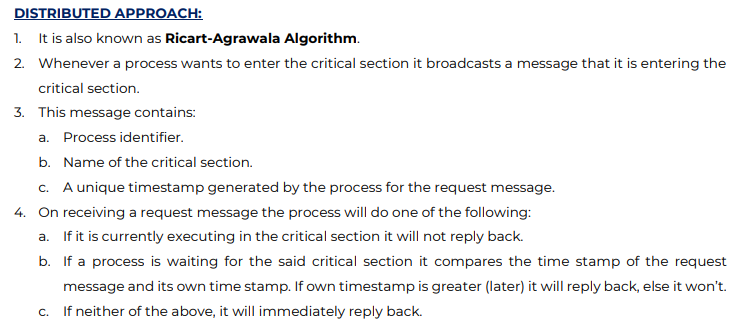












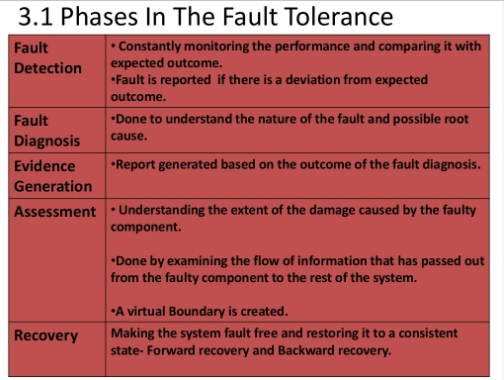
Mod 5 :

Define Fault Tolerance. Describe the different types of faults ?

Fault tolerance is the ability of the system to function even after it has crashed or its components fail to work or during times of partial failure. It tells how good it is able to handle the system once it fails.

Fault tolerance is needed to provide three main features to DS ;

* Reliability : A DS should be reliable not only in terms of data but also the services it provides
* Availability : A DS should be available at all times in case if any other nodes stop functioning.
* Security : A DS should have protection measures to protect itself of malicious attacks from hackers.
* Detection – Diagnosis- EvidenceGeneration – Assessment – Recovery



Types of Failures :

COT-TRAP

1. Crash Failure : It occurs when certain nodes in the system crash due to hardware problems or any software that has stopped functioning that might have an hardware issue as its underlying cause.

2. Omission Failure : It occurs when a particular node does not respond to client messages or sends an messages to the client whenever in need.

3. Timing Failure : It occurs when a server fails to respond in the given time.

4. Response Failure : It occurs when a server sends incorrect message in response to the clients message

5. Arbitrary Failure : It occurs when a server sends any arbitrary message.

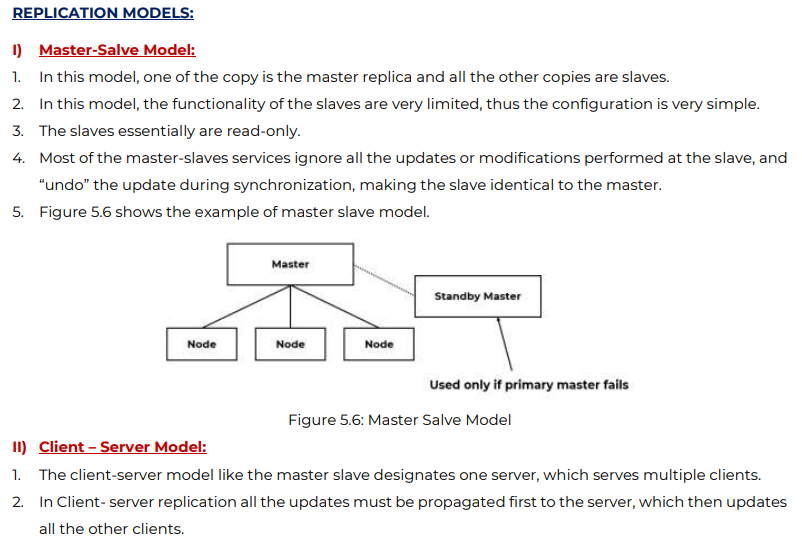
6. Transient Faults : It occurs when a particular network or a certain service becomes unavailable for a temporary period when required.

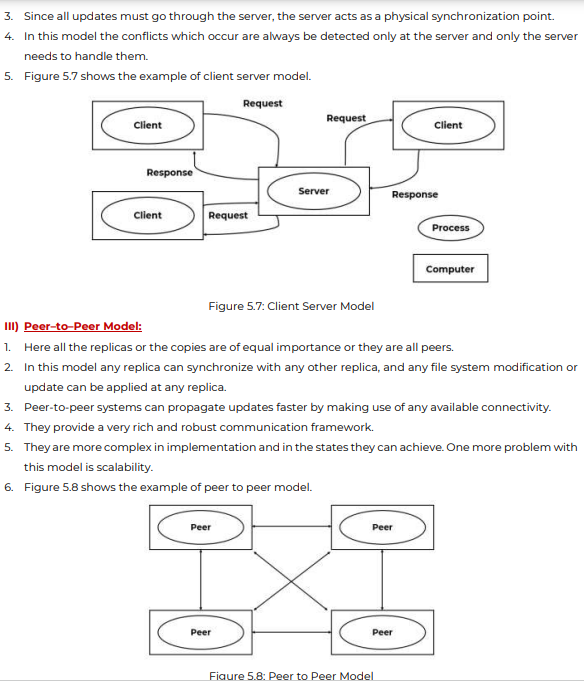
7. Permanent Faults : It is a type of fault that continues to exist until the faulty component is repaired.

What is replication in distributed system? Advantages of Replication.

Replication in computing involves sharing information so as to ensure consistency between redundant resources, such as software or hardware components, to improve reliability, fault-tolerance, or accessibility.

Give Example





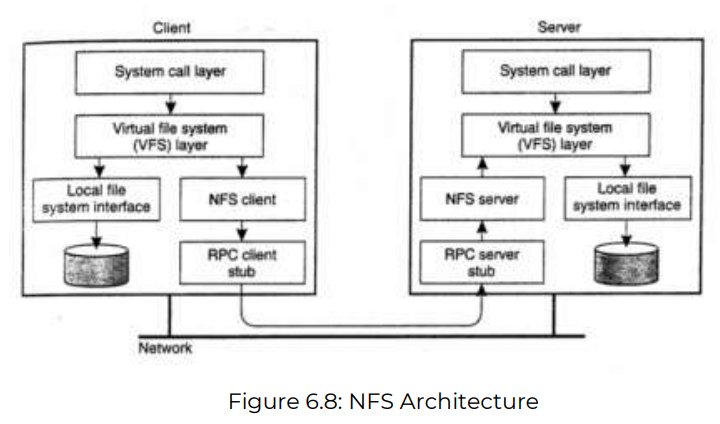
Advantages :

* Availability
* Reliability
* Fault Tolerant

Mod 6 :

Network File System :

* It is a platform independent remote file system tech created by Sun Mircosystems.
* It is a client/server application that provides shared file storage for clients across a network.
* It was designed to simplify sharing of file system resources in a network of machines that didn’t share same specification
* It is implemented by using the RPC protocol and is available through a network via a Virtual File System (VFS), an interface that runs on top of TCP/IP layer.
* It allows the access of remote files the same way it easily allows access to local files.



* It provides a single copy of directory that is shared by all the systems on the network.
* An NFS Client

a. Mounts a remote file system onto the clients local file system name space

b. Provides an interface so that access to remote files is same as that of local files.

Goals :

1. Compatibility : same semantics as local file system. User should not understand whether file is local or remote.

2. Easy Deployable : It should be easily incorporated into existing system remote files without having anything modified

3. Machine and OS Independence : NFS should run on non-unix platforms with simple protocols that can allow easy implementation in other platforms

4. Efficiently : Service should be efficient and recovery should happen from machine crashes and network problems.