Sample Questions

Computer Engineering

Subject Name: Distributed Computing **Semester: VIII**

Multiple Choice Questions

Choose the correct option for following questions. All the Questions carry equal marks	
1.	_____ and _____ are used to hide the access and location of the system.
Option A:	**Access transparency, location transparency.**
Option B:	migration transparency, replication transparency
Option C:	network transparency, location transparency
Option D:	failure transparency, network transparency
2.	The two popular remote object invocation models are
Option A:	RPC and RMI
Option B:	**CORBA and RMI**
Option C:	MOM and RPC
Option D:	MPI and MOM
3.	In distributed systems, a logical clock is associated with _____
Option A:	each instruction
Option B:	each register
Option C:	**Each process**
Option D:	none of the mentioned
4.	Process transfer policy in Load-balancing algorithms is _____
Option A:	Determines how to exchange load information among nodes
Option B:	Determines to which node the transferable process should be sent
Option C:	Determines the total number of times a process can migrate
Option D:	**Determines whether to execute a process locally or remotely**
5.	Client centric consistency model useful in applications where _____
Option A:	Data is static
Option B:	**One client always updates data store**
Option C:	Data updation is not required
Option D:	Data storage is not required
6.	In distributed file system, file name does not reveal the file's _____
Option A:	Local name
Option B:	Global name
Option C:	Cache location
Option D:	**Physical storage location**

7.	The Ricart & Agrawala distributed mutual exclusion algorithm is:
Option A:	More efficient and more fault tolerant than a centralized algorithm.
Option B:	More efficient but less fault tolerant than a centralized algorithm.
Option C:	Less efficient but more fault tolerant than a centralized algorithm.
Option D:	**Less efficient and less fault tolerant than a centralized algorithm.**
8.	The kernel is _____ of user threads.
Option A:	a part of
Option B:	the creator of
Option C:	**unaware of**
Option D:	aware of
9.	What is stub?
Option A:	**transmits the message to the server where the server side stub receives the message and invokes procedure on the server side**
Option B:	Perform encryption and decryption
Option C:	Perform Routing operation
Option D:	Perform Retransmission of message
10.	In a distributed file system, _____ is mapping between logical and physical objects.
Option A:	Client interfacing
Option B:	**Naming**
Option C:	Migration
Option D:	Heterogeneity
11.	RPC is an example of ----------
Option A:	**synchronous communication**
Option B:	asynchronous communication
Option C:	persistent communication
Option D:	time independent operation
12.	What is a remote object reference?
Option A:	The variables referenced by the Method Invocation
Option B:	An identifier for the skeleton referred by a client
Option C:	An identifier for the proxy referenced by a client
Option D:	**An identifier for a remote object that is valid throughout a distributed system**
13.	In a distributed file system, _____ is mapping between logical and physical objects.
Option A:	Client interfacing
Option B:	**Naming**
Option C:	Migration
Option D:	Heterogeneity
	10 & 13 are same

14.	Concurrency transparency is
Option A:	Where users cannot tell where an object is physically located in the system
Option B:	Hide differences in data representation and how an object is accessed
Option C:	**Hide that an object may be shared by several independent users**
Option D:	Hide that an object is replicated
15.	Client centric consistency model useful in applications where_____
Option A:	Data is static
Option B:	**One client always updates data store**
Option C:	Data updates not required in the local store
Option D:	Data storage is not required **5 & 15 are same**
16.	The ring election algorithm works by
Option A:	Having all nodes in a ring of processors send a message to a coordinator who will elect the leader
Option B:	Sending a token around a set of nodes. Whoever has the token is the coordinator.
Option C:	Sending a message around all available nodes and choosing the first one on the resultant list
Option D:	**Building a list of all live nodes and choosing the largest numbered node in the list**
17.	What is a stateless file server?
Option A:	It keeps tracks of states of different objects
Option B:	**It maintains internally no state information at all**
Option C:	It maintains only client information in them
Option D:	It maintains only client access information in them
18.	In which file model, a new version of the file is created each time a change is made to the file contents and the old version is retained unchanged
Option A:	Unstructured files
Option B:	Structured files
Option C:	**Immutable files**
Option D:	Mutable files
19.	The Ricart Agrawala distributed mutual exclusion algorithm is:
Option A:	More efficient and more fault tolerant than a centralized algorithm.
Option B:	More efficient but less fault tolerant than a centralized algorithm.
Option C:	Less efficient but more fault tolerant than a centralized algorithm.
Option D:	**Less efficient and less fault tolerant than a centralized algorithm.**
20.	Which of the following is NOT a technique for achieving scalability
Option A:	**Centralization**
Option B:	Distribution
Option C:	Replication
Option D:	Caching

21.	A layer which lies between an operating system and the applications running on it is called as -
Option A:	Firmware
Option B:	Hardware
Option C:	Software
Option D:	**Middleware**
22.	Goals of Distributed system does not include-
Option A:	Resource sharing
Option B:	Access to remote resources
Option C:	**Sharing memory space**
Option D:	Concurrent process execution
23.	which of the following is not the commonly used semantics for ordered delivery of multicast messages-
Option A:	Absolute ordering
Option B:	**Persistent ordering**
Option C:	Consistent ordering
Option D:	Casual ordering
24.	The type of transparency that enables resources to be moved while in use without being noticed by users and application is-
Option A:	Location Transparency
Option B:	Migration Transparency
Option C:	**Relocation Transparency**
Option D:	Access Transparency
25.	A paradigm of multiple autonomous computers, having a private memory, communicating through a computer network, is known as-
Option A:	**Distributed computing**
Option B:	Cloud computing
Option C:	Centralized computing
Option D:	Parallel computing
26.	Following is not the common mode of communication in Distributed system-
Option A:	RPC
Option B:	RMI
Option C:	Message Passing
Option D:	**Shared memory**
27.	Following is not the physical clock synchronization algorithm-
Option A:	**Lamport's Scalar Clock synchronization**
Option B:	Christians clock synchronization
Option C:	Berkley clock synchronization
Option D:	Network time protocol

28.	Distributed Mutual Exclusion Algorithm does not use-
Option A:	**Coordinator process**
Option B:	Token
Option C:	Logical clock for event ordering
Option D:	Request and Reply message
29.	Vector Timestamp Ordering Algorithm is an example of-
Option A:	Centralized Mutual Exclusion
Option B:	Distributed Mutual Exclusion
Option C:	Physical Clock Synchronization
Option D:	**Logical Clock Synchronization**
30.	What is fault tolerance in distributed Computing?
Option A:	Ability of system to continue functioning in the event of a complete failure.
Option B:	**Ability of system to continue functioning in the event of a partial failure.**
Option C:	Ability of system to continue functioning when system is properly working.
Option D:	Ability of distributed system to work in all conditions.
31.	In Task Assignment Approach, we have to-
Option A:	**Minimize IPC cost**
Option B:	Maximize IPC cost
Option C:	Fix IPC cost
Option D:	Keep constant IPC cost
32.	Backward error recovery requires-
Option A:	Grouping
Option B:	Assurance
Option C:	**Check pointing**
Option D:	Validation
33.	Which of these consistency models does not use synchronization operations?
Option A:	**Sequential**
Option B:	Weak
Option C:	Release
Option D:	Entry
34.	Which is not possible in distributed file system?
Option A:	File replication
Option B:	**Migration**
Option C:	Client interface
Option D:	Remote access
35.	X.500 is a-
Option A:	**Directory services**
Option B:	Naming services
Option C:	Replication services
Option D:	Consistency services

36.	A DFS is executed as a part of-
Option A:	System specific program
Option B:	**Operating system**
Option C:	File system
Option D:	Application program
37.	Processes on the remote systems are identified by-
Option A:	Host ID
Option B:	Identifier
Option C:	**Host name and identifier**
Option D:	Process ID
38.	The function of load-balancing algorithm is-
Option A:	**It tries to balance the total system load by transparently transferring the workload from heavily loaded nodes to lightly loaded**
Option B:	It helps the process to know the time by simply making a call to the operating system.
Option C:	allows a process to access named entity
Option D:	It synchronizes the clocks
39.	A Multi-threaded Server has following threads-
Option A:	Dispatcher Thread
Option B:	Client Thread
Option C:	Worker Thread
Option D:	**Client and Server Thread**
40.	Maekawa's Mutual Exclusion Algorithm is based on-
Option A:	Coordinator selection
Option B:	Token
Option C:	**Voting**
Option D:	Tickets

DC QB Answers

Sample Questions

Computer Engineering

Subject Name: Distributed Computing **Semester: VIII**

Descriptive Questions

1. What are the different architecture models of Distributed System? Explain with suitable diagrams.

ARCHITECTURAL MODEL:

1. Architectural model defines the way in which the components of the system interact with each other and the way in which they are mapped onto an underlying network of computers.
2. It **simplifies and abstracts the functionality** of the individual components of a distributed system.
3. It describes responsibilities distributed between system components and how these components are placed.
4. **Examples: Client Server Model and Peer to Peer Model**.

I) Client Server Model:

- In the basic client-server model, processes in a distributed system are divided into two groups i.e. **server and client**.
- A server is a process implementing a specific service, for example, a file system service or a database service.
- A client is a process that requests a service from a server by sending it a request and subsequently waiting for the server's reply.
- The client-server model is usually based on a **simple request/reply protocol**.
- Figure 1.5 represents client server model in distributed system.
- A server can itself request services from other servers; thus, in this new relation, the server itself acts like a client.

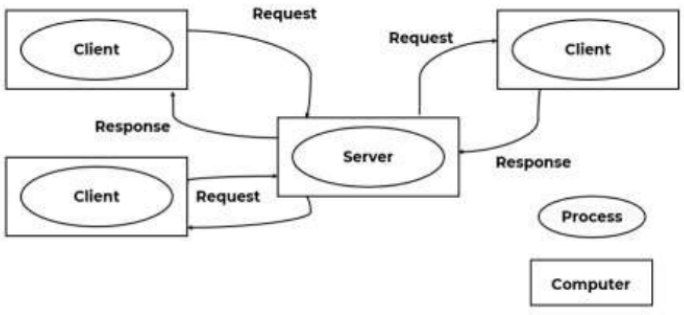


Figure 1.5: Client Server Model

II) Peer to Peer Model:

- The general idea behind peer to peer is where there is no central control in a distributed system.
- The basic idea is that, each node can either be a client or a server at a given time.
- If the node is requesting something, it can be known as a client, and if some node is providing something, it can be known as a server. In general, each node is referred to as a Peer.
- This is the most general and flexible model.
- In this network, any new node has to first join the network.
- After joining in, they can either request a service or provide a service.
- In this model, all the processes has the **same capabilities and responsibilities**.
- It tries to solve some problems which are associate with client server model.
- Figure 1.6 represents peer to peer model in distributed system.

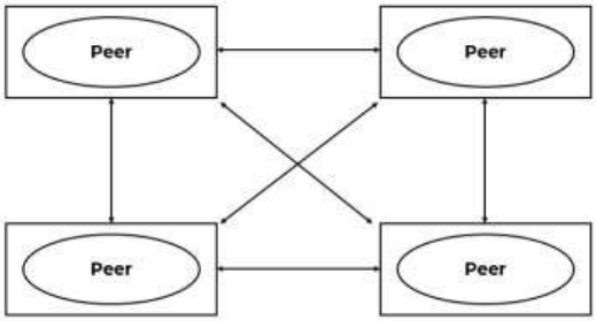


Figure 1.6: Peer to Peer Model

2. ‘’Write a short note on Raymond’s Tree based Mutual exclusion algorithm.

RAYMOND'S TREE BASED ALGORITHM:

1. Raymond’s Tree Based Algorithm is a **token based algorithm**.
2. In Raymond’s Tree Based Algorithm, a **directed tree is formed**.
3. Nodes are arranged as a tree.
4. Every node has a parent pointer.
5. Each node has a **FIFO queue** of requests.
6. There is one token with the root and the owner of the token can enter the critical section.
7. Figure 3.9 shows the example of Raymond’s Tree Based Algorithm.

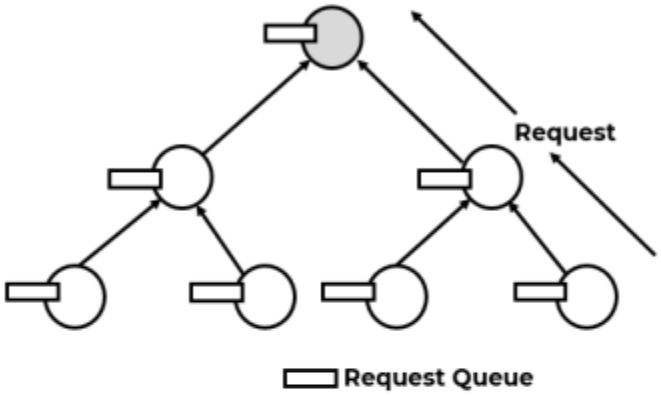


Figure 3.9: Raymond’s Tree Based Algorithm.

8. As the token moves across nodes, the parent pointers change.
9. They always point towards the holder of the token as shown in figure 3.10.
10. It is thus possible to reach the token by following parent pointers.

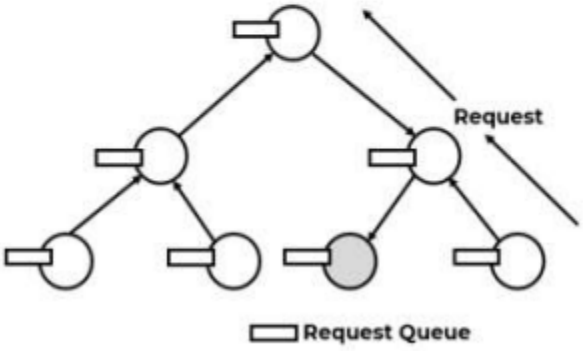


Figure 3.10: Raymond’s Tree Based Algorithm.

Requesting a Token:

1. The node adds “self” in its request queue.
2. Forwards the request to the parent.

3. The parents adds the request to its request queue.
4. If the parent does not hold the token and it has not sent any requests to get the token, it sends a request to its parent for the request.
5. This process continues till we reach the root (holder of the token).

Releasing a Token:

1. Ultimately a request will reach the token holder.
2. The token holder will wait till it is done with the critical section.
3. It will forward the token to the node at the head of its request queue.
    a. It removes the entry.
    b. It updates its parent pointer.
4. Any subsequent node will do the following:
    c. Dequeue the head of the queue.
    d. If "self" was at the head of its request queue, then it will enter the critical section.
    e. Otherwise, it forwards the token to the dequeued entry.
5. After forwarding the entry, a process needs to make a fresh request for the token, if it has outstanding entries in its request queue.

3. What is RPC? Explain model of RPC.
4. Define remote procedure call (RPC)? Describe the working of RPC in detail.
5. Describe the role of stubs in Remote Procedure Calls.

REMOTE PROCEDURE CALL:

1. Remote Procedure Call is as known as **Remote Function Call or a Remote Subroutine Call.**
2. A remote procedure call is an Interprocess communication technique that is used for client-server based applications.
3. RPC provides a general IPC protocol that can be **used for designing several distributed applications**.
4. Features of a RPC for distributed applications are:
   a. Simple call syntax.
   b. Familiar semantics.
   c. Ease of use.
   d. Generality.
   e. Efficiency.

RPC MODEL:

Figure 2.2 shows the RPC Model.

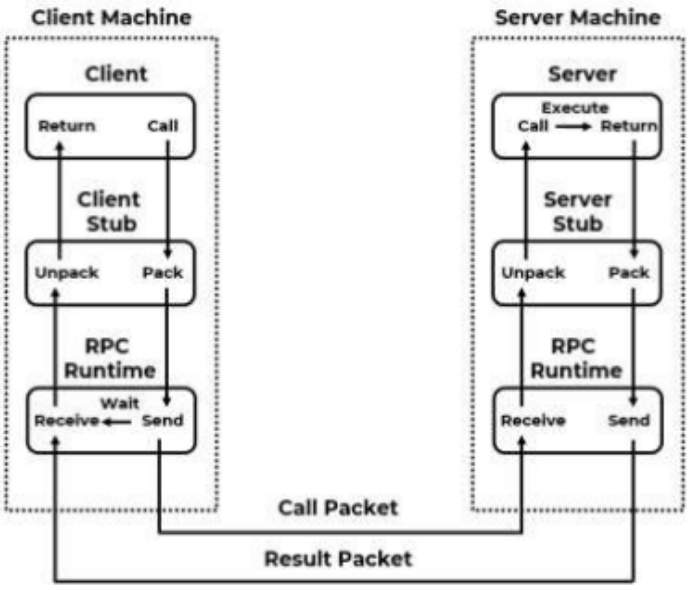


Figure 2.2: RPC Model.

IMPLEMENTATION OF RPC:

1. To achieve the goal of semantic transparency the implementation of an RPC mechanism is based on the concept of **STUBS**.
2. Stubs provide a **local procedure call abstraction.**
3. The implementation of an RPC mechanism usually involves the following five elements.
4. **Client:** The user process that initiates a remote procedure call.
5. **Client Stub:** It is responsible for:
    a. On receipt of a call request from the client, it packs a specification of the target procedure and the arguments into the message.
    b. On receipt of the result of procedure execution, it unpacks the result and passes it to the client.
6. **RPC Runtime:** It handles transmission of messages across the network between client and server machines including encryption, routing acknowledgement.
7. **Server Stub:** The job of the server stub is very similar to client stubs.
8. **Server:** The server executes the appropriate procedure and returns the result.

ADVANTAGES OF REMOTE PROCEDURE CALL

1. Remote procedure calls support process oriented and thread oriented models.
2. The internal message passing mechanism of RPC is hidden from the user.
3. The effort to re-write and re-develop the code is minimum in remote procedure calls.
4. Remote procedure calls can be used in distributed environment as well as the local environment.

DISADVANTAGES OF REMOTE PROCEDURE CALL

1. The remote procedure call is a concept that can be implemented in different ways. It is not a standard.
2. There is no flexibility in RPC for hardware architecture. It is only interaction based.
3. There is an increase in costs because of remote procedure call.

6. What are different data centric consistency model?

DATA CENTRIC CONSISTENCY MODEL:

1. Data-centric consistency models aim to provide system wide consistent view of the data store.
2. A data store may be physically distributed across multiple machines.
3. Each process that can access data from the store is assumed to have a local or nearby copy available of the entire store.
4. Figure 5.5 shows the example of data centric consistency model.

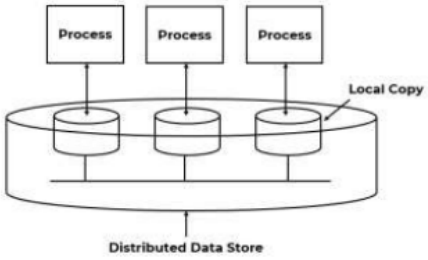
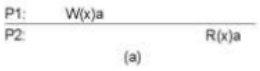


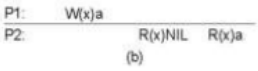
Figure 5.5: Example of Data Centric Consistency Model.

I) Strict Consistency Model:

1. Any read on a data item X returns a value corresponding to the result of the most recent write on X.
2. This is the strongest form of memory coherence which has the most stringent consistency requirement.



(a)

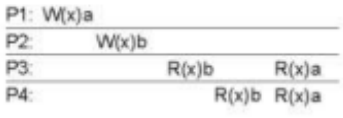


(b)

3. Behavior of two processes, operating on the same data item.
   a. A strictly consistent store.
   b. A store that is not strictly consistent.

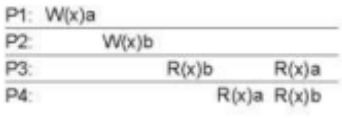
II) Sequential Consistency:

1. Sequential consistency is an important data-centric consistency model which is a slightly weaker consistency model than strict consistency.
2. A data store is said to be sequentially consistent if the result of any execution is the same as if the (read and write) operations by all processes on the data store were executed in some sequential order and the operations of each individual process should appear in this sequence in a specified order.



(a)

A sequentially consistent data store.



(b)

A data store that is not sequentially consistent.

III) Linearizability:

1. It that is weaker than strict consistency, but stronger than sequential consistency.
2. A data store is said to be linearizable when each operation is timestamped and the result of any execution is the same as if the (read and write) operations by all processes on the data store were executed in some sequential order.
3. The operations of each individual process appear in sequence order specified by its program.
4. If $ts_{OP1}(x) < ts_{OP2}(y)$, then operation OP1(x) should precede OP2(y) in this sequence.

IV) Causal Consistency:

1. It is a **weaker model than sequential consistency**.
2. In Casual Consistency all processes see only those memory reference operations in the correct order that are potentially causally related.
3. Memory reference operations which are not related may be seen by different processes in different order.
4. A memory reference operation is said to be casually related to another memory reference operation if the first operation is influenced by the second operation.
5. If a write (w2) operation is casually related to another write (w1) the acceptable order is (w1, w2).

V) FIFO Consistency:

1. It is **weaker than causal consistency**.
2. This model ensures that all write operations performed by a single process are seen by all other processes in the order in which they were performed like a single process in a pipeline.
3. This model is simple and easy to implement having good performance because processes are ready in the pipeline.
4. Implementation is done by sequencing write operations performed at each node independently of the operations performed on other nodes.

P1: W(x)a						
P2:	R(x)a	W(x)b	W(x)c			
P3:				R(x)b	R(x)a	R(x)c
P4:				R(x)a	R(x)b	R(x)c

VI) Weak Consistency:

1. The basic idea behind the weak consistency model is enforcing consistency on a group of memory reference operations rather than individual operations.
2. A Distributed Shared Memory system that supports the weak consistency model uses a special variable called a synchronization variable which is used to synchronize memory.
3. When a process accesses a synchronization variable, the entire memory is synchronized by making visible the changes made to the memory to all other processes.

VII) Release Consistency:

1. Release consistency model tells whether a process is entering or exiting from a critical section so that the system performs either of the operations when a synchronization variable is accessed by a process.
2. Two synchronization variables acquire and release are used instead of single synchronization variable. Acquire is used when process enters critical section and release is when it exits a critical section.
3. Release consistency can be viewed as synchronization mechanism based on barriers instead of critical sections.

VIII) Entry Consistency:

1. In entry consistency every shared data item is associated with a synchronization variable.
2. In order to access consistent data, each synchronization variable must be explicitly acquired.
3. Release consistency affects all shared data but entry consistency affects only those shared data associated with a synchronization variable.

7. Write a short note on code migration.
8. What is the need for Code Migration? Explain the code migration issues in detail.
9. Write a short note on the advantages of code migration.

CODE MIGRATION:

1. In process migration, an entire process has to be moved from one machine to another.
2. But this may be a **crucial task**, but it has good overall performance.
3. In some cases, a code needs to be migrated rather than the process.
4. Code Migration refers to transfer of program from one node to another.
5. For **example**: Consider a client server system, in which the server has to manage a big database.
6. The client application has to perform many database operations.
7. In such situation, it is better to move part of the client application to the server and the result is send across the network.
8. Thus code migration is a better option.
9. Code Migration is used to improve overall performance of the system by exploiting parallelism.
10. Example of code migration is shown below in figure 4.4.
11. Here the server is responsible to provide the client's implementation, when the client binds to the server.
12. The advantage of this approach is that, the client does not need to install all the required software. The software can be moved in as when necessary and discarded when it is not needed.

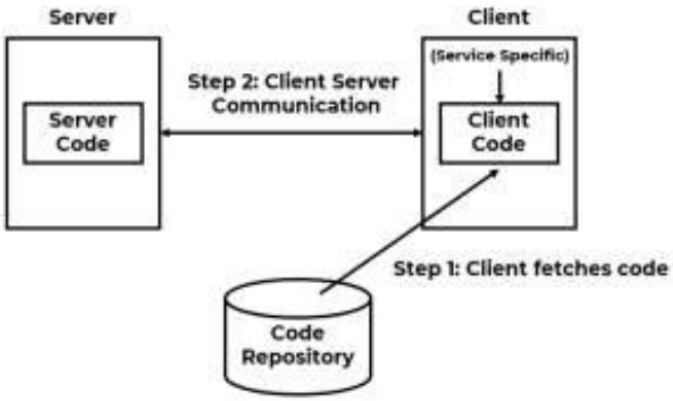


Figure 4.4: Example of Code Migration.

Reasons for Migrating Code

- Performance is improved by moving code from heavily loaded machines to lightly loaded machine, although it is costlier operation. If server manages huge database and client application perform operations on this database that involves large quantity of data. In this case, it is better to transfer part of client application to server machine. In other case, instead of sending requests for data validation to server it is better to carry out client side validation. Hence, to transfer part of server application to client machine.
- Code migration also helps to improve performance by exploiting parallelism where there is no need to consider details of parallel programming. Consider the example of searching for information in the Web. A search query which is small mobile program (mobile agent) moves from site to site. Each copy of this mobile agent can be transferred to different sites to achieve a linear speedup compared to using just a single program instance. Code migration also helps to configure distributed system dynamically.
- In other case, no need to keep all software at client. Whenever client binds with server, it can dynamically download the required software from web server. Once work is done, all these software can be discarded. No need to preinstall client side software.

Reasons of code Migration

1. Improve computing Performance by moving process from Heavily loaded machine to lightly loaded Machine

2. Improve Communication time by Shipping code to System where large dataset reside.

CODE MIGRATION ISSUES:

1. Communication in distributed systems is concerned with exchanging data between processes.
2. Code migration in the broadest sense which deals with moving programs between machines, with the intention to have those programs be executed at the target.
3. In code migration framework, a process consists of 3 segments i.e. **code segment, resource segment and execution segment**.
4. Code segment contains the actual code.
5. Resource segment contains references to resources needed by the process.
6. Execution segment stores the execution state of a process.
7. Consider the figure 4.5 of code migration.
8. Where,
   a. CS: Client-Server,
   b. REV: Remote evaluation.
   c. CoD: Code-on-demand.
   d. MA: Mobile agents.
9. Mobile agents moves from site to site.
10. Code-on-demand is used to migrate part of server to client.
11. Remote evaluation is used to perform database operations involving large amount of data.

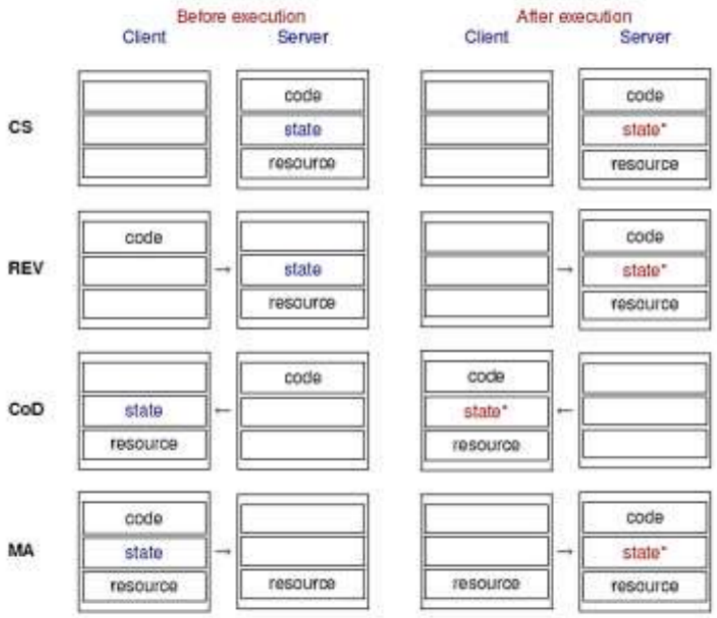


Figure 4.5: Code Migration.

Advantages:

Code migration is often used for load distribution, reducing network bandwidth, dynamic customization, and mobile agents. Code migration increases scalability, improves performance, and provides flexibility.

-

10. Explain Bully election algorithm with example.
11. Explain Bully election algorithm with an example and different scenarios. Use neat diagrams for the same.
12. Discuss the Bully algorithm with appropriate example. State its advantages and disadvantages.

1. Many algorithms used in distributed systems require a **coordinator**.
2. In general, all processes in the distributed system are equally suitable for the role.
3. Election algorithms are designed to **choose a coordinator**.
4. Any process can serve as coordinator.
5. Any process can "call an election".
6. There is no harm in having multiple concurrent elections.
7. Elections may be needed when the system is initialized, or if the coordinator crashes or retires.
8. Example of election algorithm is **Bully Algorithm**.

BULLY ALGORITHM:

1. Bully Algorithm was proposed by **Garcia-Molina**.
2. This algorithm is planned on assumptions that:
   a. Each process involved within a scenario has a process number that can be used for unique identification.
   b. Each process should know the process numbers of all the remaining processes.
   c. Scenario is synchronous in nature.
   d. A process with highest process number is elected as a coordinator.
3. Process 'p' calls an election when it notices that the coordinator is no longer responding.
4. Process 'p' sends an election message to all higher-numbered processes in the system.
5. If no process responds, then p becomes the coordinator.
6. If a higher-level process (q) responds, it sends 'p' a message that terminates p's role in the algorithm.
7. The process 'q' now calls an election (if it has not already done so).
8. Repeat until no higher-level process responds.
9. The last process to call an election "wins" the election.
10. The winner sends a message to other processes announcing itself as the new coordinator.

Example:

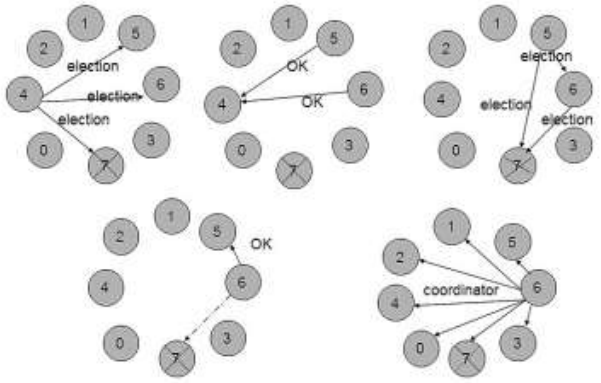


Figure 3.6: Example of Election Algorithm

1. Process (4) calls an election when it notices that the coordinator is no longer responding.
2. Process (4) sends an election message to all higher-numbered processes in the system.
3. If no process responds, then Process (4) becomes the coordinator.
4. If a higher-level process (5, 6) responds, it sends process (4) a message that terminates (4)'s role in the algorithm.
5. Now process (5) now calls an election (if it has not already done so).
6. Repeat until no higher-level process responds.
7. The last process to call an election "wins" the election.
8. The winner sends a message to other processes announcing itself as the new coordinator.
9. Example is shown in figure 3.6.

Advantages & Disadvantages:

The advantages of Bully algorithm are that this algorithm is a distributed method with simple implementation. This method requires at most five stages, and the probability of detecting a crashed process during the execution of the algorithm is lowered in contrast to other algorithms. Therefore other algorithms impose heavy traffic in the network in contrast to Bully algorithm. Another advantage of this algorithm is that only the processes with higher priority number respect to the priority number of process that detects the crash coordinator will be involved in election, not all process are involved. However the two major limitations of Bully algorithm are the number of stages to decide the new leader and the huge number of messages exchanged due to the broadcasting of election and OK messages.

- In Fig. 3.3.1, initially process 17 (higher-numbered) is coordinator. Process 14 notices that, coordinator 17 has just crashed as it has not given response to request of process 14. Process 14 now initiates election by sending **Election** message to process 15, 16 and 17 which are higher-numbered processes.

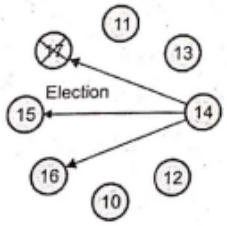


Fig. 3.3.1

- As shown in Fig. 3.3.2, process 15 and 16 responds with **OK** message. Process 17 is already crashed and hence does not send reply with **OK** message. Now, job of process 14 is over.

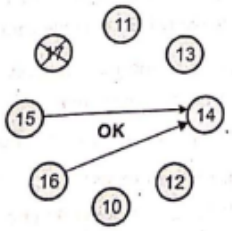


Fig. 3.3.2

- Now process 15 and 16 each hold an election by sending **Election** message to its higher-numbered processes. It is shown in following Fig. 3.3.3 (a). As process 17 is crashed, only process 16 replies to 15 with **OK** message. This is shown in Fig. 3.3.3 (b).

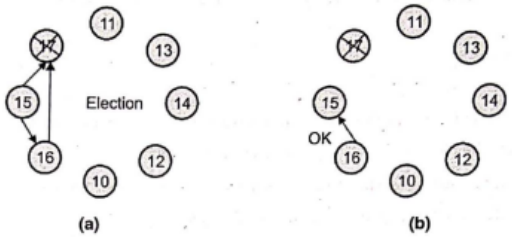


Fig. 3.3.3

- Finally, process 16 wins the election and send **coordinator** message to all the processes to inform that, it is now new coordinator as shown in Fig. 3.3.4. In this algorithm, if two processes detect simultaneously that the coordinator is crashed then both will initiate the election. Every higher-number process will receive two **Election** messages. Process will ignore the second **Election** message and election will carry on as usual.

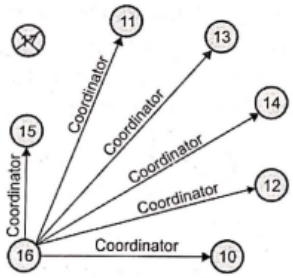


Fig. 3.3.4

13. Define fault tolerance. Describe different types of faults.

FAULT TOLERANCE:

1. A system fails when it does not match its promises.
2. An error in a system can led to a failure.
3. The cause of an error is called a **fault**.
4. Faults are generally **transient, intermittent or permanent**.
5. The ability of the system to continue functioning in the event of partial failure is known as fault tolerance.
6. Fault tolerance is the important issue in designing a Distributed file system.
7. There are various types of fault which harms the integrity of a system.
8. For example: If the processor loses the contents of its main memory in the event of a crash it leads to logically complete but physically incomplete operations, making the data inconsistent.
9. Decay of disk storage devices may result in the loss or corruption of data stored by a file system.
10. Decay is when the portion of device is irretrievable.

Types:

- **Omission Fault:**
  - When a process or channel fails to do something that it is expected to it is termed an **omission failure**.
  - There are two categories of omission fault i.e. process and communication omission fault.
  - **Process omission failures:**

- **Communication omission failures:**

Arbitrary Failures

- In this type of failure process or communication channel behaves arbitrarily. In this failure, the responding process may return wrong values or it may set wrong value in data item. Process may arbitrarily omits intentional processing step or carry out unintentional processing step.
- Arbitrary failure also occur with respect to communication channels. The examples of these failures are : message content may change, repeated delivery of the same messages or non-existent messages may be delivered. These failures occur rarely and can be recognized by communication software.

- **Timing Fault:**
  - Synchronous distributed system is one in which each of the following are defined:
    - Upper and lower bounds on the time to execute each step of a process;
    - A bound on the transmission time of each message over a channel;
    - A bound on the drift rate of the local clock of each process.
  - If one of these bounds is not satisfied in a synchronous system then a **timing failure** is said to have occurred.

-

14. Explain Hadoop distributed file system.

Hadoop File System was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly faulttolerant and designed using low-cost hardware.

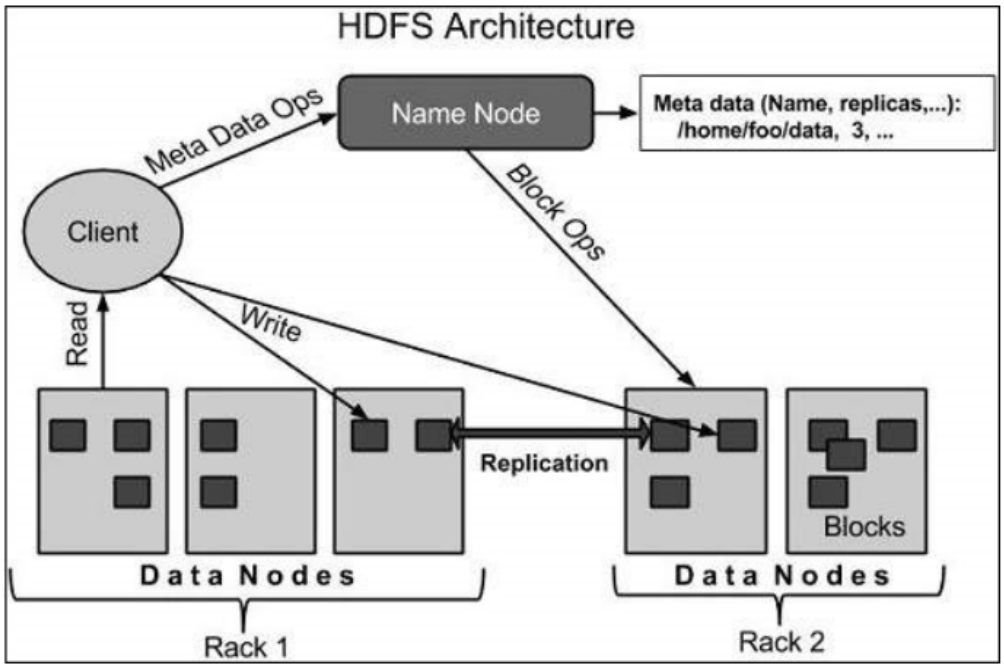
HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

Features of HDFS

- It is suitable for the distributed storage and processing.
- Hadoop provides a command interface to interact with HDFS.
- The built-in servers of namenode and datanode help users to easily check the status of cluster.
- Streaming access to file system data.
- HDFS provides file permissions and authentication.

HDFS Architecture

Given below is the architecture of a Hadoop File System.



HDFS follows the master-slave architecture and it has the following elements.

Namenode

The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware. The system having the namenode acts as the master server and it does the following tasks −

- Manages the file system namespace.
- Regulates client's access to files.
- It also executes file system operations such as renaming, closing, and opening files and directories.

Datanode

The datanode is a commodity hardware having the GNU/Linux operating system and datanode software. For every node (Commodity hardware/System) in a cluster, there will be a datanode. These nodes manage the data storage of their system.

- Datanodes perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.

Block

Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

15. Draw and explain the general architecture of a Message-Queuing System

2.5.4 Message-Oriented Persistent Communication

Message-queuing systems or Message-oriented Middleware (MOM) supports persistent asynchronous communication. In this type of communication, sender or receiver of the message need not be active during message transmission. Message is stored in intermediate storage.

Message-Queuing Model

- In message-queuing system, messages are forwarded through many communication servers. Each application has its own private queue to which other application sends the message. In this system, guarantee is given to sender that its sent message will be delivered to recipient queue. Message can be delivered at any time.
- In this system, receiver need not be executing when message arrives in its queue from sender. Also, sender need not be executing after its message is delivered to the receiver. In exchange of messages between sender and receiver, message is delivered to receiver with following execution modes.
  - Sender and receiver both are running.
  - Sender running but receiver passive.
  - Sender passive but receiver running.
  - Both sender and receiver are passive.

General Architecture of a Message-Queuing System

- In the message-queuing system, source queue is present either on the local machine of the sender or on the same LAN. Messages can be read from local queue. The message put in queue for transmission contains specification of destination queue. Message-queuing system provides queues to senders and receivers of the messages.
- Message-queuing system keeps mapping of queue names to network locations. A queue manager manages queues and interacts with applications which sends and receives the messages. Special queue managers work as routers or relays which forward the messages to other queue managers. Relays are more suitable as in several message-queuing systems, dynamically mapping of queue-to-location not available.
- Hence, each queue manager should have copy of queue-to-location mapping. For larger size message-queuing system, this approach leads to network management problem. To solve these problems, only routers need to be updated after adding or removal of the queues.
- Relays thus provide help in construction of scalable message-queuing systems. Fig. 2.5.7 shows relationship between queue-level-addressing and network-level-addressing.

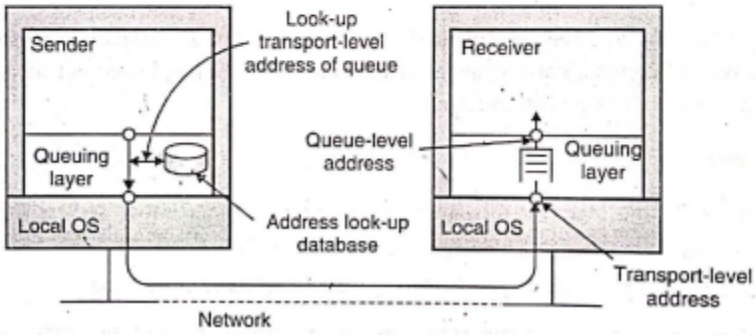


Fig. 2.5.7

16. What are the features of Andrew File System? Define File service architecture of AFS?

ANDREW FILE SYSTEM (AFS):

1. The Andrew File System (AFS) is a distributed file system.
2. It was developed by Carnegie Mellon University as part of the Andrew Project.
3. Originally named "Vice", AFS is named after Andrew Carnegie and Andrew Mellon.
4. Its primary use is in **distributed computing**.
5. It uses a **session semantics.**
6. AFS supports information sharing on large scale.

FEATURES:

Andrew File System is designed to:

1. Handle terabytes of data.
2. Handle thousands of users.
3. Working in WAN Environment.

AFS ARCHITECTURE:

1. Figure 6.2 shows the AFS Architecture.
2. AFS is divided into two types of nodes:
   a. **Vice Node:** It is dedicated file server.
   b. **Virtue Node:** It is a Client Machines.
3. In VFS, the entire file is copied to the Virtue Node (local machine) from the Vice Node (Server) when opened.
4. When the file is changed – it is copied to the server when closed.

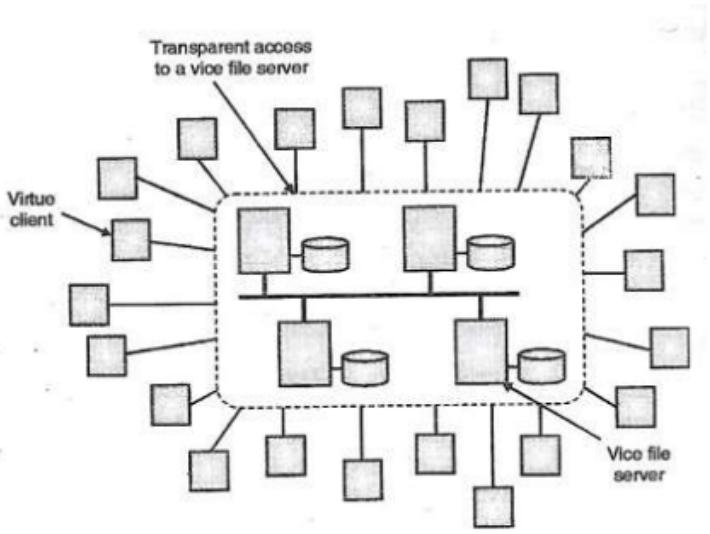


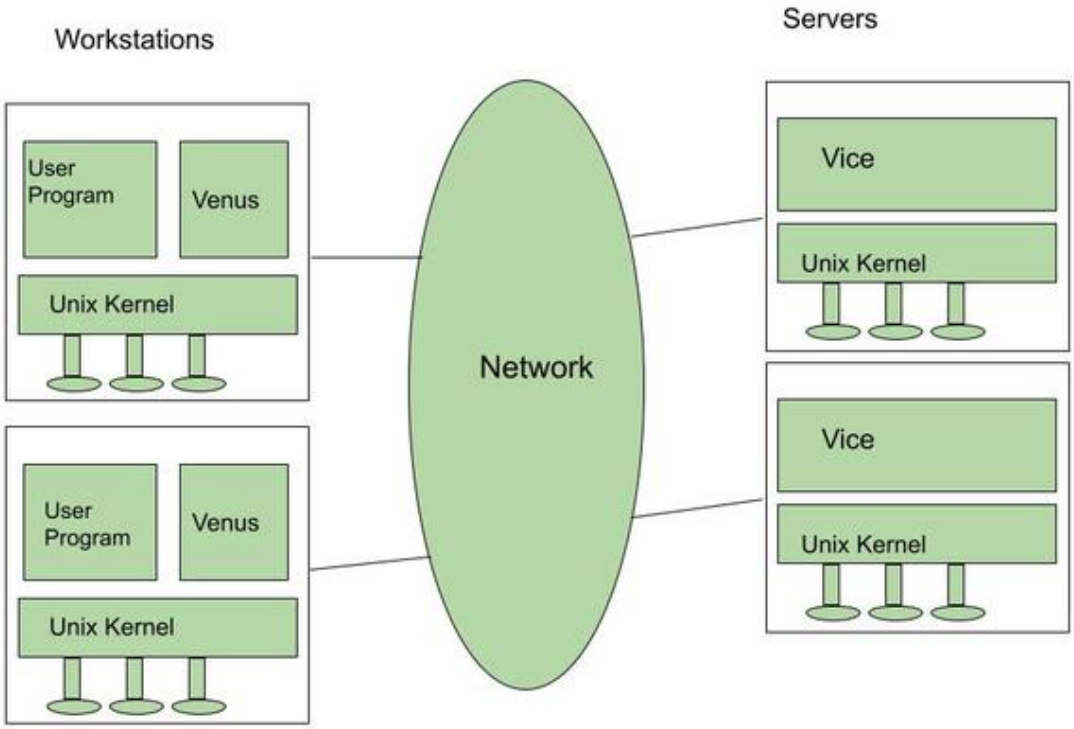
Figure 6.2: AFS Architecture.

LIMITATION OF AFS:

1. Non-availability of services when servers and network components fail.
2. Scaling problem.

IMPLEMENTATION

1. Afs implementation contains 2 softwares components vice and virtue.
2. Vice processes run on server machines and Virtue processes run on client machines both as User level UNIX processes.
3. At client side, local and shared files are available. Shared files are stored on the server and cached by clients in disk cache.
4. At client side, one of the file partitions is used as cache storing files, cached copies of files from shared space. The management of cache is carried out by virtue process.
5. It removes LRU(Least recently used) files so that new files accessed from the server get space. The size of the disk cache at the client is large.
6. Vice server implements the flat file service. Virtue processes at client side implements hierarchic directory structure that is needed by user programs.



-

17. Briefly describe the architecture and server operations of NFS.
18. Write short note on - Network File System (NFS)

NFS:

1. NFS stands for **Network File System**.
2. NFS is a **platform independent remote file system** technology created by Sun Microsystems (Sun) in 1984.
3. It is a **client/server application** that **provides shared file storage** for clients across a network.
4. It was designed to simplify the sharing of file systems resources in a network of non-homogeneous machines.
5. It is implemented using the RPC protocol and the files are available through the network via a **Virtual File System (VFS)**, an interface that runs on top of the TCP/IP layer.
6. Allows an application to access files on remote hosts in the same way it access local files.
7. NFS is generally implemented following the layered architecture shown in figure 6.8 below.

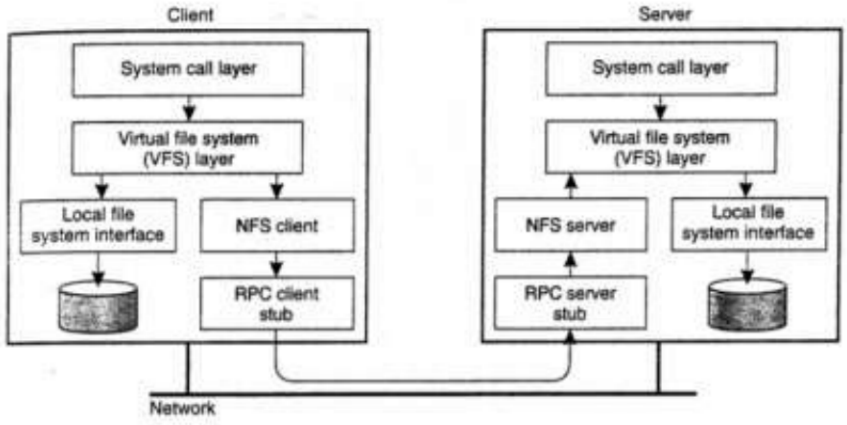


Figure 6.8: NFS Architecture

NFS SERVERS: COMPUTERS THAT SHARE FILES

1. During the late 1980s and 1990s, a common configuration was to configure a powerful workstation with lots of local disks and often without a graphical display to be a NFS Server.
2. "Thin," diskless workstations would then mount the remote file systems provided by the NFS Servers and transparently use them as if they were local files.

NFS Simplifies management:

1. Instead of duplicating common directories such as /usr/local on every system, NFS provides a single copy of the directory that is shared by all systems on the network.
2. Simplify backup procedures - Instead of setting up backup for the local contents of each workstation (of /home for exmaple), with NFS a sysadm needs to backup only the server's disks.

-

NFS Clients: Computers that access shared files

1. NFS uses a mixture of kernel support and user-space daemons on the client side.
2. Multiple clients can mount the same remote file system so that users can share files.
3. Mounting can be done at boot time. (i.e. /home could be a shared directory mounted by each client when user logs in).
4. An NFS client:

   (a) Mounts a **remote file system** onto the client's **local file system** name space and

   (b) Provides an interface so that access to the files in the remote file system is done as if they were local files.

GOALS OF NFS DESIGN:

1. **Compatibility:** NFS should provide the same semantics as a local unix file system. Programs should not need or be able to tell whether a file is remote or local.
2. **Easy deployable:** Implementation should be easily incorporated into existing systems remote files should be made available for local programs without these having to be modified or relinked.
3. **Machine and OS independence:** NFS Clients should run in non-unix platforms Simple protocols that could be easily implemented in other platforms.
4. **Efficiently:** NFS should be good enough to satisfy users, but did not have to be as fast as local FS. Clients and Servers should be able to easily recover from machine crashes and network problems.

NFS PROTOCOL

1. Uses Remote Procedure Call (RPC) mechanisms
2. RPCs are synchronous (client application blocks while waits for the server response)
3. NFS uses a stateless protocol (server do not keep track of past requests) - This simplify crash recovery. All that is needed to resubmit the last request.
4. In this way, the client cannot differentiate between a server that crashed and recovered and one that is just slow.

19. Explain the different issues and steps involved in a good Load Balancing algorithm
20. Explain different load estimation policies used by load balancing approach. `

LOAD BALANCING APPROACH:

1. Load Balancing is used in **Process Management**.
2. Scheduling Algorithm that uses Load Balancing Approach are called as **Load Balancing Algorithms**.
3. The main goal of load balancing algorithms is to **balance the workload** on all the nodes of the system.
4. Load balancing aims to:
   a. Optimize resource use.
   b. Maximize throughput.
   c. Minimize response time.
   d. Avoid overload of any single resource.

DESIGNING ISSUES:

Designing a load balancing algorithm is a critical task. It includes

I) Load Estimation Policy:

1. The first issue in a load balancing algorithm is to decide which method to use to estimate the workload of a particular node.

2. Estimation of the workload of a particular node is a difficult problem for which no completely satisfactory solution exists.
3. A nodes workload can be estimated based on some measurable parameters below:
   a. Total number of processes on the node.
   b. Resource demands of these processes.
   c. Instruction mixes of these processes.
   d. Architecture and speed of the node's processor.

II) Process Transfer Policy:

1. The idea of using this policy is to transfer processes from heavily loaded nodes to lightly loaded nodes.
2. Most of the algorithms use the threshold policy to decide whether the node is lightly loaded or heavily loaded.
3. Threshold value is a limiting value of the workload of node which can be determined by:
   a. **Static Policy:** Each node has a predefined threshold value.
   b. **Dynamic Policy:** The threshold value for a node is dynamically decided.
4. Below threshold value, node accepts processes to execute.
5. Above threshold value node tries to transfer processes to a lightly loaded node.
6. Single threshold policy may lead to unstable algorithm.
7. To reduce instability double threshold policy has been proposed which is also known as high low policy.
8. Figure 4.1 (a) shows single threshold policy and (b) shows double threshold policy.

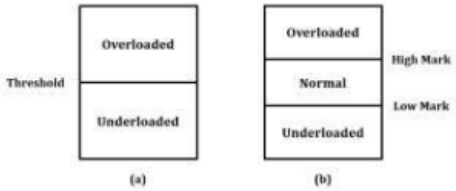


Figure 4.1: Threshold Policy.

-

III) Location Policy:

1. When a transfer policy decides that a process is to be transferred from one node to any other lightly loaded node, the challenge is to find the destination node.
2. Location policy determines this destination node.
3. It includes following:
   a. **Threshold:** This policy selects a random node and checks whether the node is able to receive the process then it transfers the process. If the node rejects the transfer then another node is selected randomly. This continues until the probe limit is reached.
   b. **Shortest method:** In this 'm' distinct nodes are chosen at random and each is polled to determine its load with minimum value.
   c. **Bidding method:** In this method, nodes contain **managers** to send processes and **contractors** to receive processes.
   d. **Pairing:** It is used to reduce the variance of load only between nodes of the system.

IV) State Information Exchange Policy:

1. It is used for frequent exchange of state information within the nodes.
2. It includes:
   a. **Periodic Broadcast:** Each node broadcasts its state information after the elapse of T units of time.
   b. **Broadcast when state changes:** Each node broadcasts its state information only when a process arrives or departures.
   c. **On demand exchanges:** Each node broadcasts its state information request message when its state switches from normal to either under load or over load.
   d. **Exchanges by polling:** The partner node is searched by polling the other nodes on by one until poll limit is reached.

V) Priority Assignment Policy:

1. The priority of the execution of local and remote processes at a particular node should be planned.
2. It includes:
   a. **Selfish priority assignment rule:** In this local process are given higher priority than remote process.
   b. **Altruistic priority assignment rule:** Remote processes are given higher priority than local process.
   c. **Intermediate priority assignment rule:** Priority in this rule is decided depending upon the number of local and remote processes on a node.

VI) Migration Limiting Policy:

1. This policy determines the total number of times a process can migrate from one node to another.
2. It includes:
   a. **Uncontrolled Policy:** The remote process is treated as local process. Hence, it can be migrated any number of times.
   b. **Controlled Policy:** Migration count is used for remote processes.

-

21. Explain the Centralized algorithms for Mutual Exclusion in Distributed Systems.

CENTRALIZED APPROACH:

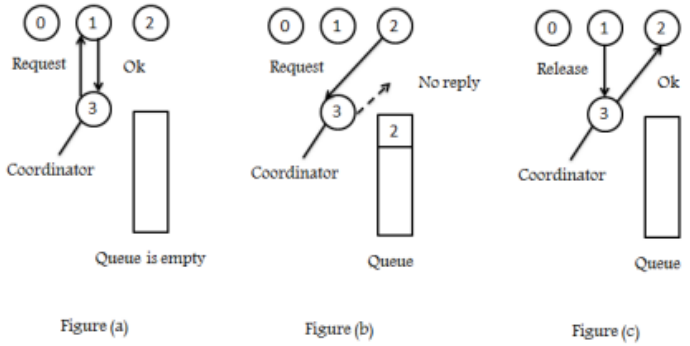


Figure 3.8: Centralized Approach for Mutual Exclusion

1. In centralized algorithm one process is elected as the **coordinator**.
2. Coordinator may be the machine with the highest network address.
3. Whenever a process wants to enter a critical region, it sends a request message to the coordinator stating which critical region it wants to enter and asking for permission.
4. If no other process is currently in that critical region, the coordinator sends back a reply granting permission, as shown in figure 3.8 (a).
5. When the reply arrives, the requesting process enters the critical region.
6. Suppose another process 2 shown in figure 3.8 (b), asks for permission to enter the same critical region.
7. Now the coordinator knows that a different process is already in the critical region, so it cannot grant permission.
8. The coordinator just refrains from replying, thus blocking process 2, which is waiting for a reply or it could send a reply saying 'permission denied'.
9. When process 1 exits the critical region, it sends a message to the coordinator releasing its exclusive access as shown in figure 3.8 (c).
10. The coordinator takes the first item off the queue of deferred requests and sends that process a grant message.
11. If the process was still blocked it unblocks and enters the critical region.
12. If an explicit message has already been sent denying permission, the process will have to poll for incoming traffic or block later.
13. When it sees the grant, it can enter the critical region.

Advantages:

1. No starvation problem.
2. Easy to implement & use for general resources allocation.

Disadvantages:

1. If coordinator crashes, the entire system will fail.
2. In large system a single coordinator can become a performance bottleneck.

-

22. Describe File caching schemes in brief.

FILE CACHING:

1. In order to improve the file I/O performance in centralized time sharing systems, file caching has been implemented and executed in numerous distributed file systems.
2. Main goal of file caching scheme is to retain recently accessed data in main memory.
3. So that repeated access to the same information can be efficiently handled.
4. In implementing file-caching scheme, one has to make several key decisions like granularity of cached data.
5. Three design issues are involved in file caching which are:

I) Cache location:

1. Cache location refers to the place where the cached data is stored.
2. There exists three possible cache locations in servers DFS i.e. server's main memory, client's disk and client's main memory as shown in figure 6.1.

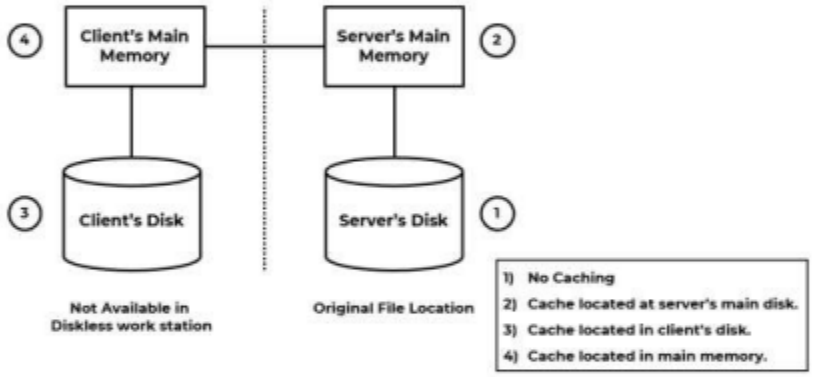


Figure 6.1: Cache Location.

Server's main memory:

1. In this the file must be first transferred to the server's main memory and then across the network
2. Hence the disks access cost is reduced.

Client's disks: The cache located in the client's disk eliminates network access cost but requires disk access cost.

Client's main memory: The cache located in a client's main memory eliminated both network access cost and disk access cost.

II) Modification propagation:

1. When the cache is located on clients' node; a file's data may simultaneously be cached on multiple nodes.

-

2. It is possible for caches to become inconsistent when the file data is changed by one of the clients and the corresponding data cached at other nodes are not changed or discarded.
3. A variety of approaches is used to handle these issues:
   a. When to propagate modifications made to a cached data to corresponding file server?
   b. How to verify the validity of cached data?
4. It consists of the following techniques:

Write through scheme:

When cache entry is modified new value is immediately sent to the server for updating the master copy of the file.

Delayed-write scheme:

1. In this scheme when a cache entry is modified the new value is written only to the cache.
2. The client just makes a note f the cache entry that has been updated.
3. All the cache entries are collected together and sent to the server later on.
4. Its approaches are as follows:
   a. **Write on ejection from cache:**
      - Modified data in the cache is sent to the server when the cache replacement policy has decided to eject it from the client's cache.
   b. **Periodic write:**
      - Here cache is scanned at regular intervals.
      - And cached data is been modified right from the last scan is sent to the server.
   c. **Write on close:**
      - Modification made to a cached data by a client is sent to the server when the corresponding file is closed by the client.

III) Cache Validation Schemes:

1. A file data resides in the cache of multiple nodes.
2. It becomes necessary to check if the data cached at the client node is consistent with the master node.
3. If it is not then the cached data must be invalidated and the updated version must be fetched from the server.
4. Validation is done in two ways:

Client initiated approach:

1. In this method the client contacts the server to check if locally cached data is consistent with the master copy.
2. File sharing semantics depends on the frequency of the validity check and can be use below approaches:
   a. **Check before every access:** The main purpose of caching is defeated in this process because the server has to be contacted on every access.
   b. **Periodic check:** For this method a check is initialized after fixed interval of time.
   c. **Check on file open:** A client cache entry is validated only when the client open a corresponding file for use.

Server initiated approach:

1. In this method a client informs the file server when it opens a file indicating whether the file is being opened for reading, writing or both.
2. The file server keeps a record of which client has which file opened and in what mode.
3. Thus server keeps a track of the file usage modes being used by different clients and reacts in case of inconsistency.
4. Inconsistency occurs when more clients try to open a file in conflicting modes.
5. **Example:** If a file is open for reading, other clients may be able to open it to read a file without any problem, but not for writing.
6. Similarly, a new client must not be allowed to open a file which is already open for writing.
7. When client closes a file it sends intimation to the server along with any modification made to the file.
8. On receiving intimation the server updates its record of which client has which file open in what mode.

23. What is an open distributed system and what benefits does openness provide?

- Openness is the important goal of the distributed system. An **open distributed** system provides services as per standard rules that tell the syntax and semantics of those services. As, in computer networks, standard rules state the message format, its contents and meaning of sent and received messages. All these rules are present in protocols.
- Similar to above, in distributed systems, services are usually specified through interfaces, which are expressed in an Interface Definition Language (IDL). The definitions of the interfaces written in an IDL almost capture only the syntax of these services.
- They state exactly the names of the available functions together with parameters type; return values, exceptions likely to be raised etc. The semantics of interfaces means specification of what these interfaces can perform. Actually, these specifications are specified in an informal way through natural language.
- Processes make use of interfaces to communicate with each other. There can be different implementation of these interfaces leading to different distributed system that function in the same manner.

- Appropriate specifications are complete and neutral. Complete specifications specify the whole thing that is essential to make an implementation. But, many interface definitions does not obey completeness.
- So that it is required for a developer to put in implementation-specific details. If specifications do not impose what an implementation should look like : they should be neutral. Completeness and neutrality are significant for interoperability and portability.
- An open distributed system should allow configuring the system out of different components from different developers. Also, it should be trouble-free to put in new components or replace existing ones without disturbing those components that stay in place. It means, an open distributed system should also be extensible. Open systems interfaces are published.
- Monolithic approach should be avoided for building the system. There should be separation between policy and mechanism. For example, apart from storing the documents by browser, users should be able to make a decision which documents are stored and for how much duration. User should be able to set it dynamically.
- User should be able to implement his own policy as a component that can be plugged into the browser. Certainly, implemented component must have an interface that the browser can recognize so that it can call procedures of that interface.
- Open distributed system provides uniform communication mechanism and it is also based on published interfaces in order to access the common resources. It also considers heterogeneous environment in terms of hardware and software.

24. Explain Cristian's algorithm for physical clock synchronization

I) Cristian's Algorithm:

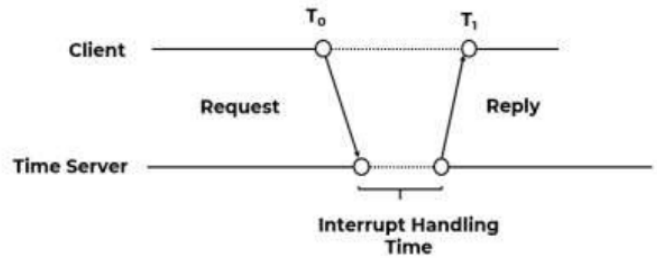


Figure 3.1: Cristian's Algorithm

1. Cristian's Algorithm is a clock synchronization algorithm is used to synchronize time with a time server by client processes.
2. This algorithm works well with low-latency networks where Round Trip Time is short as compared to accuracy.
3. Here Round Trip Time refers to the time duration between start of a Request and end of corresponding Response.
4. In this method, each node periodically sends a message to server.
5. When the time server receives the message it responds with a message T, where T is the current time of server node.
6. Assume the clock time of client be T_0 when it sends the message and T_1 when it receives the message from server.
7. T_0 and T_1 are measured using same clock so best estimate of time for propagation is $(T_1-T_0)/2$.
8. When the reply is received at clients node, its clock is readjusted to $T + (T_1-T_0)/2$.
9. Figure 3.1 represents cristian's algorithm.

Advantage: It assumes that no additional information is available.

Disadvantage: It restricts the number of measurements for estimating the value.

25. Explain Stream oriented communication with example.

STREAM ORIENTED COMMUNICATION:

1. RPC and Message Oriented Communication are based on the **exchange of discrete messages**.
2. Timing might affect performance, but not correctness.
3. In Stream Oriented Communication the message content must be delivered at a certain rate, as well as correctly.
4. **Example:** Music or video.
5. Stream Oriented Communication supports **continuous media communication.**

TRANSMISSION MODES IN STREAM ORIENTED COMMUNICATION:

1. **Asynchronous Mode:** No restrictions with respect to when data is to be delivered
2. **Synchronous Mode:** Define a maximum end-to-end delay for individual data packets.
3. **Isochronous Mode:** Define a maximum end-to-end delay and maximum delay variance.

STREAM:

1. A data stream is a **connection-oriented communication facility**.
2. It supports **isochronous data transmission**.
3. Some common stream characteristics:
   a. Streams are **unidirectional**.
   b. There is generally a single **source.**
4. Stream types:
   a. **Simple**: It consists of a single flow of data (e.g., audio or video)
   b. **Complex**: It consists of multiple data flows (e.g., stereo audio or combination audio/video)

EXAMPLE:

Token Bucket Algorithm:

1. Figure 2.9 shows the implementation of Token Bucket.
2. The token bucket can be easily implemented with a counter.
3. The token is initialized to zero.
4. Each time a token is added, the counter is incremented by 1.
5. Each time a unit of data is dispatched, the counter is decremented by 1.
6. If the counter contains zero, the host cannot send any data.

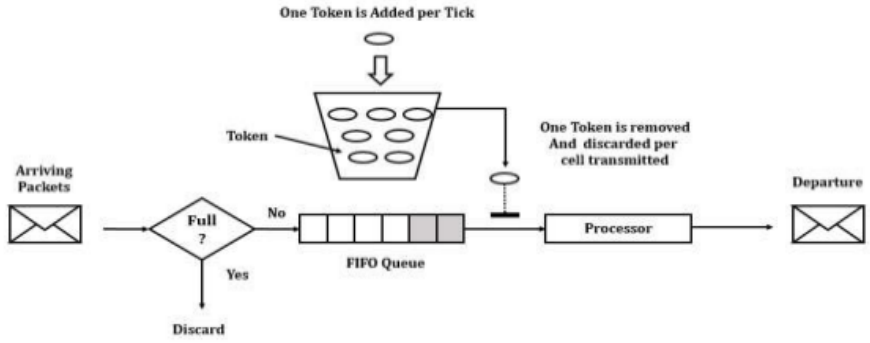


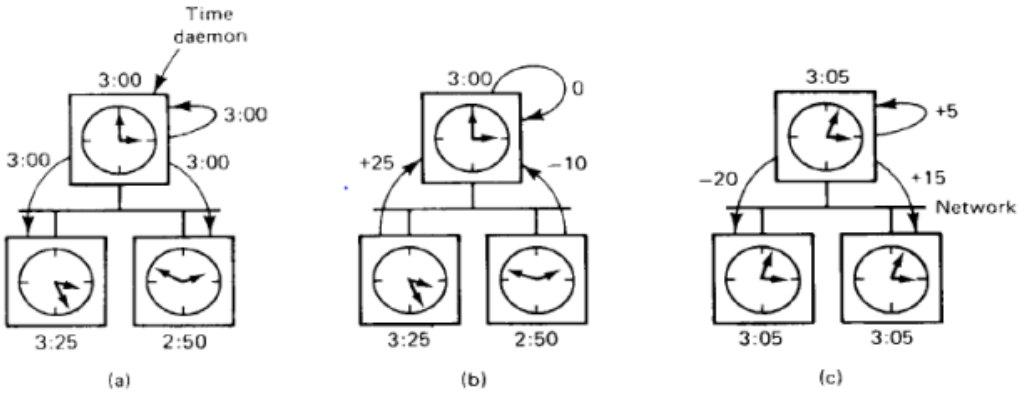
Figure 2.9: Implementation of Token Bucket.

-

26. Explain Berkeley physical clock algorithm

II) Berkley Algorithm:

1. The Berkeley algorithm is a method of clock synchronization in distributed computing which assumes no machine has an accurate time source.
2. It was developed by Gusella and Zatti at the University of California, Berkeley in 1989
3. This is an **active time server approach**.
4. In this approach, the time server periodically sends a message to all the computers in the group of computers.
5. When this message is received each computer send backs its own clock value to the time server.
6. The time server has a prior knowledge of the approximate time required for propagation of a message which is used to readjust the clock values.
7. It then takes average of clock values of all the computers.
8. The calculated average is the current time to which all clocks should be readjusted.
9. The time server readjusts its own clock to this value and instead of sending the current time to other computers it sends the amount of time each computer needs for readjustment.
10. This can be positive or negative value.
11. Figure 3.2 represents berkley algorithm.



(a) The time daemon asks all the other machines for their clock values. (b) The machines answer. (c) The time daemon tells everyone how to adjust their clock.

Figure 3.2: Berkley Algorithm

-

27. Differentiate between NOS, DOS and Middleware in the design of a distributed systems.

COMPARISON BETWEEN NOS & DOS:

Table 1.1: Comparison between NOS & DOS

Parameters	NOS	DOS
Definition:	It is a computer operating system that is designed primarily to support workstation, personal computer which is connected on a LAN.	It is an operating system which manages a number of computers and hardware devices which make up a distributed system.
Goal:	Goal is to offer local service to remote client.	Goal is to hide and manage hardware resources.
OS type:	It is tightly coupled operating system.	It is loosely coupled operating system.
Used for:	Used for Heterogeneous multi computers.	Used for multi-processor and homogeneous multi computers.

Architecture:	Follows '2' tier client server architecture.	Follows 'n' tier client server architecture.
Types:	Multicomputer and multiprocessor operating system.	Peer to peer and client server architecture.
Communication mode:	It uses file for communication.	It uses messages for communication.
Transparency:	Degree of transparency is low.	Degree of transparency is high.
Reliability:	Low.	High.
Example:	Novell NetWare.	Microsoft distributed component object model.
Diagram:	Refer Figure 1.15	Refer Figure 1.14

-

28. Differentiate between Data Centric and Client centric Consistency models with examples.

DIFFERENCE BETWEEN DATA CENTRIC AND CLIENT CENTRIC CONSISTENCY:

Table 5.1: Difference between Data Centric and Client Centric Consistency.

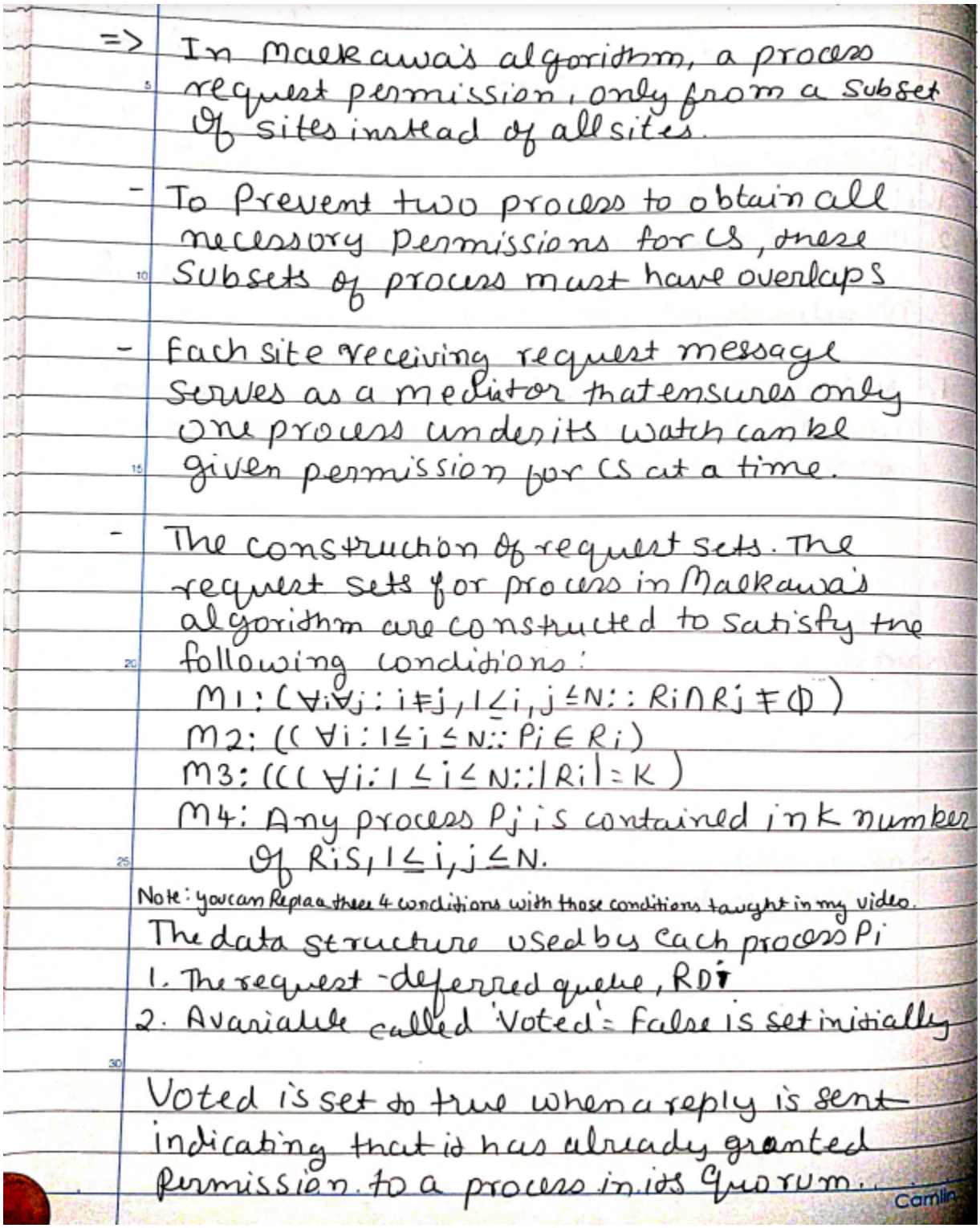
Data Centric Consistency	Client Centric Consistency
It is used for all client in a system.	It is used for individual client.
It does not have lack of simultaneous updates.	It has lack of simultaneous updates.
It is data specific.	It is client specific.
Globally accessible.	Not globally accessible.
It aims to provide system wide consistent view on a database	It aims to provide client specific consistent view on a database.

Examples:

- **Data Centric Consistency**
  - Strict Consistency Model
  - Sequential Consistency
  - Linearizability
  - Casual Consistency
  - FIFO Consistency
  - Weak Consistency
  - Release Consistency
  - Entry Consistency

- **Client Centric Consistency**
  - Eventual consistency
  - Monotonic reads
  - Monotonic writes
  - Read your writes
  - Writes follow reads

29. What are the steps involved in the execution of Maekawa's Algorithm for Mutual Exclusion



=> In Maekawa's algorithm, a process request permission, only from a subset of sites instead of all sites.

- To Prevent two process to obtain all necessary permissions for CS, these subsets of process must have overlaps

- Each site receiving request message serves as a mediator that ensures only one process under its watch can be given permission for CS at a time.

- The construction of request sets. The request sets for process in Maekawa's algorithm are constructed to satisfy the following conditions:

M1: $(\forall i \forall j : i \neq j, 1 \leq i, j \leq N :: R_i \cap R_j \neq \phi)$

M2: $(\forall i : 1 \leq i \leq N :: P_i \in R_i)$

M3: $((\forall i : 1 \leq i \leq N :: |R_i| = K)$

M4: Any process P_j is contained in k number of R_is, $1 \leq i, j \leq N$.

Note: you can Replace these 4 conditions with those conditions taught in my video.

The data structure used by each process P_i

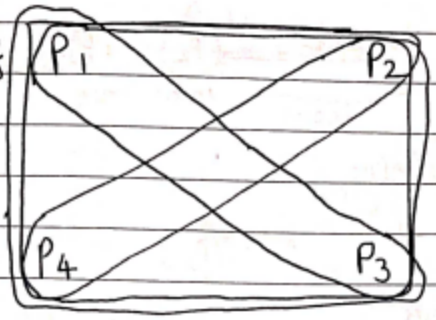
1. The request - deferred queue, RD
2. A variable called 'Voted' = False is set initially

Voted is set to true when a reply is sent indicating that is has already granted permission to a process in its Quorum.

Camlin

Example:

Step:1



$R_1 = \{P_1, P_2, P_4\}$

$R_2 = \{P_2, P_3, P_1\}$

$R_4 = \{P_4, P_3, P_1\}$

$R_3 = \{P_3, P_4, P_2\}$

Step:2

$R_1 = \{P_1, P_4, P_2\}$ (P1) Voted = F

(P2) Voted = F $R_2 = \{P_2, P_3, P_1\}$

$R_4 = \{P_4, P_3, P_1\}$ (P4) Voted = F

(P3) Voted = F $R_3 = \{P_3, P_2, P_4\}$

Step 3: P_1 wants to enter the Critical Section

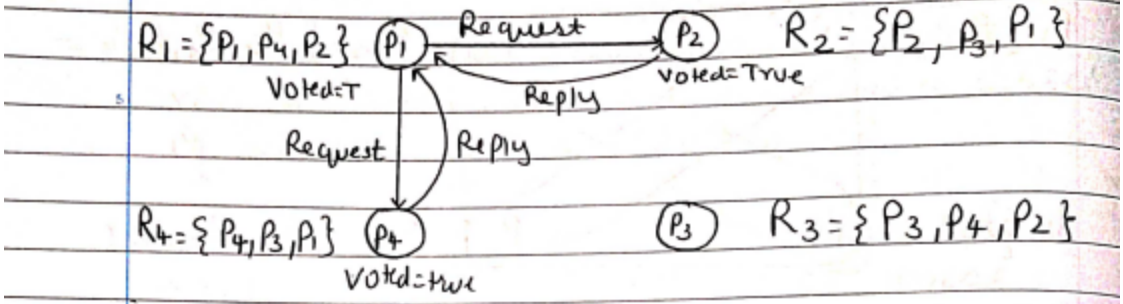
$R_1 = \{P_1, P_4, P_2\}$ (P1) Voted = F —Request→ (P2) Voted = F $R_2 = \{P_2, P_3, P_1\}$

(P1) —Request→ (P4)

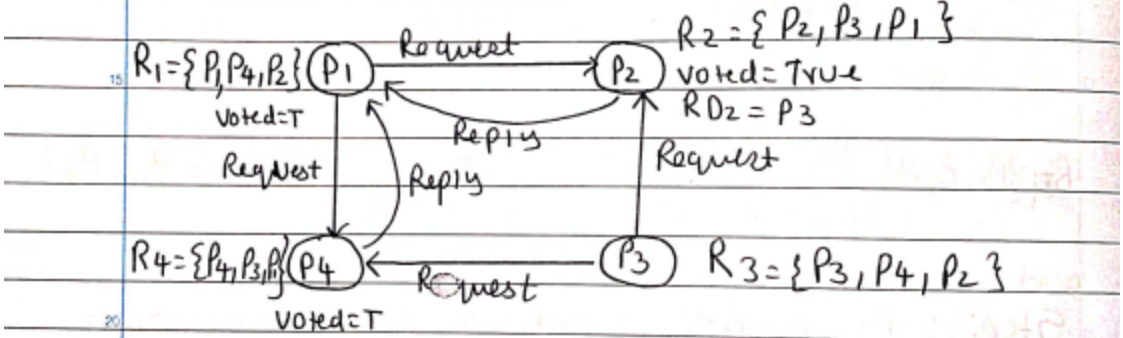
$R_4 = \{P_4, P_3, P_1\}$ (P4) Voted = F

(P3) Voted = F $R_3 = \{P_3, P_2, P_4\}$

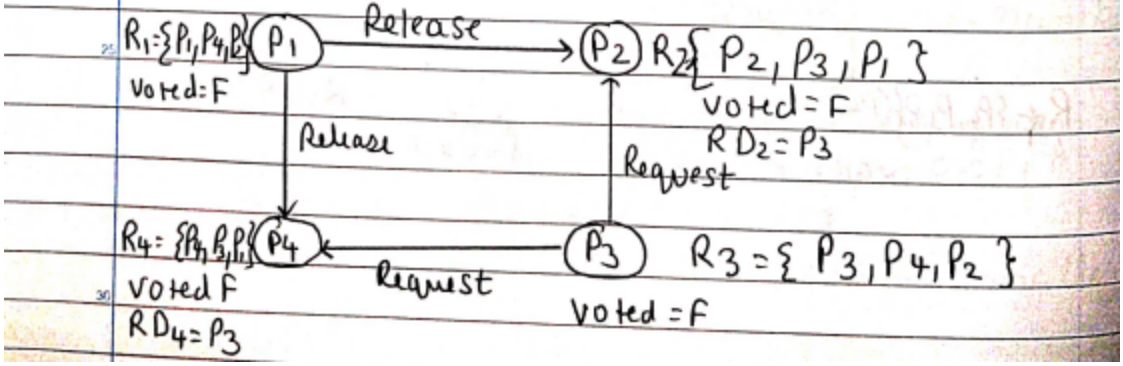
Step4:



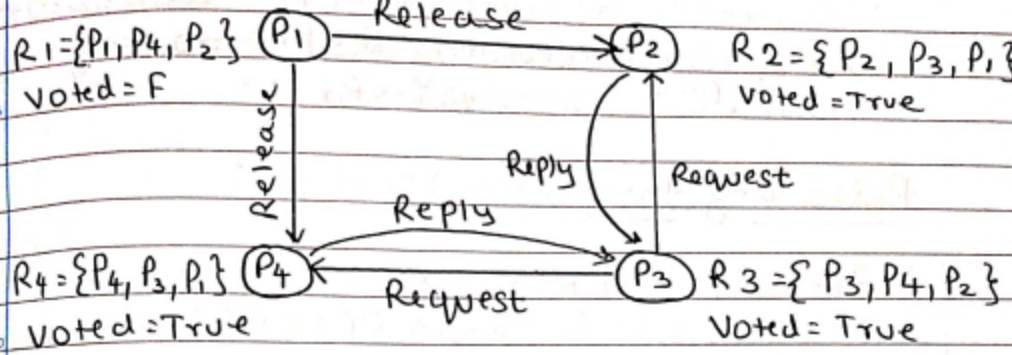
Step5: P3 wants to enter into the critical Section.



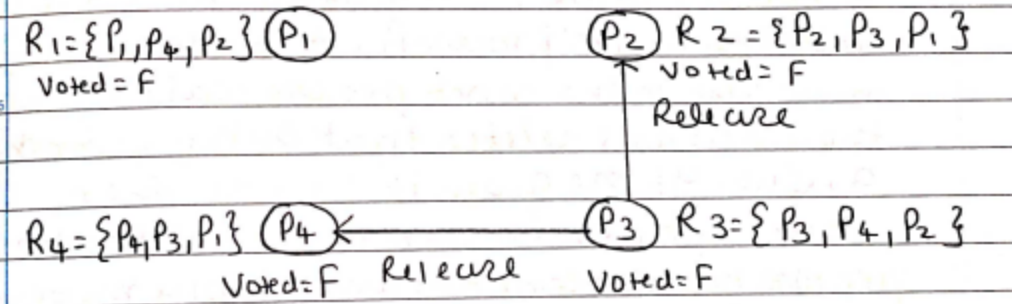
Step6: P1 wants to exit critical Section



Step: 7



Step: 8



Algorithm:

Requesting the critical section:

1. A process Pi request s access to the critical Section by sending ReQuesti messages to ale the process in its request set Ri.
2. When a process Pj receives the Requesti message, it sends a replyj message to Pi, provided it hasent sent a reply message to a process from the time it received the last Release message. Otherwise it queues up the REQUEST for later consideration.

Executing the critical section.

1. Process Pi accesses the critical Section only after receiving REPLY message from all the process in Ri.

Releasing the critical section:

1. After the execution of the critical section is over, site Pi sends RELEASE(i) message to all the process in Ri

2. When a process Pj receaves a RELEASEi message from process Pi, it sends a Reply message to the next process waiting in the queue and deletes that entry from the queue. If the queue is empty, then process updates its state to reflect that the process has not sent out any Reply message.

Advantages:

1. No Single point of failure
2. Mutual Exclusion Successfully achieved
3. Better bandwidth Utilisation

Disadvantage:

1. It might lead to deadlock
   Solution: Use timestamp.

30. Write short note on - Group Communication.

1) Group Communication

- In distributed system, it is necessary to have support for sending data to multiple receivers.
- This type of communication is called group communication.
- To support group communication, many network-level & transport-level solutions have been implemented.
- In group communication, messages sent to a group of processes will deliver to all members of the group.
- Group Communication also helps the processes from different hosts to work together and perform operations in synchronized manner, therefore increases overall perfomance of system.
- There are 3 types of group communication i.e one to many, many to one & many to many communication.

ONE – TO – MANY COMMUNICATIONS:

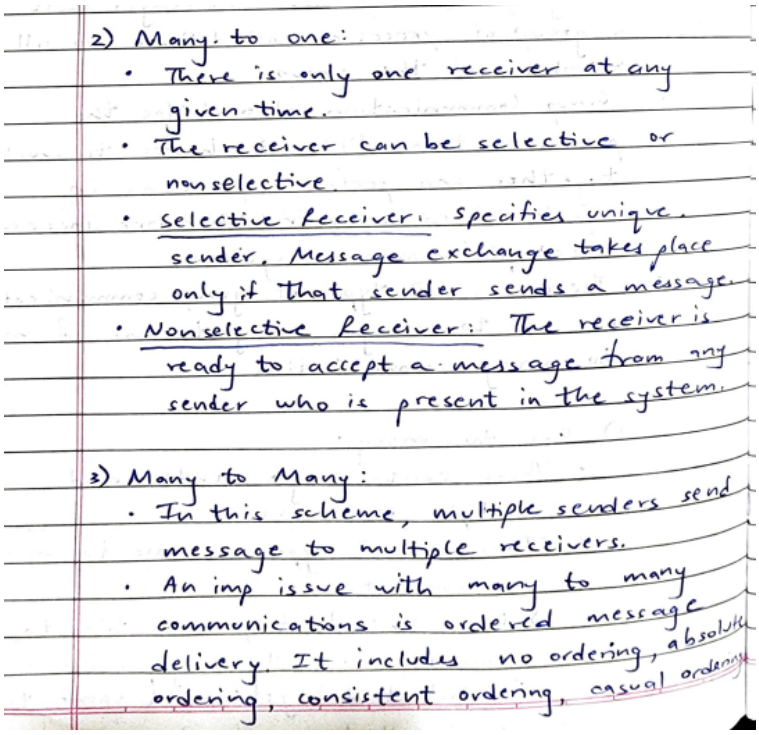
1. There exist single sender and multiple receivers.
2. It is also called as **multicast communication**.
3. There exists a special case of, multicast communication as broadcast communication in which message is sent to all processors connected to the network.
4. One to many communication includes:
   a. **Group Management:** There two types of groups open group and closed group.
   - A closed group is the one in which only the members of the group can send a message to the group as a whole, although it may send a message to an individual.
   - An Open Group is the one in which any processes in the system can send a message to the group as a whole.

   b. **Group Addressing:**
   - It is possible to create a special network address to which multiple machines can listen.
   - This network address is called **multicast address**.
   - Networks with broadcasting facility declare a certain address, such as zero, as Broadcast Address.

   c. **Buffered and Unbuffered Multicast:**
   - For Unbuffered Multicast the message is not buffered for the receiving process and it is lost if the receiving process is not in the state ready to receive it.
   - For Buffered Multicast, the message is buffered for receiving processes so each process of the multicast group will eventually receive the message.

   d. **Send to all and Bulletin Semantics:**
   - **Send-to-all semantics:** A copy of message is sent to each process for multicast group and the message is buffered until it is accepted by the process.
   - **Bulletin-board Semantics:** A Message to be multicast is addressed to a channel instead of being sent to every individual process of multicast group.

   e. **Flexible Reliability in Multicast Communication:**
   - **0 reliable:** No response is expected by the sender.
   - **1 reliable:** The sender expects at least one acknowledgement.
   - **M out of N reliable:** The sender decides as to how many acknowledgement it expects out of N.



2) Many to one:
- There is only one receiver at any given time.
- The receiver can be selective or nonselective.
- Selective Receiver: Specifies unique sender. Message exchange takes place only if that sender sends a message.
- Nonselective Receiver: The receiver is ready to accept a message from any sender who is present in the system.

3) Many to Many:
- In this scheme, multiple senders send message to multiple receivers.
- An imp issue with many to many communications is ordered message delivery. It includes no ordering, absolute ordering, consistent ordering, casual ordering

31. What is replication in distributed system? Explain the advantages of replication.

- At lower-level different lower-level names are used for different replicas. Replication control such as determination of the degree of replication and placement of replicas should be provided to higher levels. As one replica updates then from user's point of view, the changes should be reflected in other copies as well.

6.6.1 Replication and Caching

- In replication, replica is created at server, whereas cached copy is associated with clients.
- A replica is more persistent as compared to cached copy. Replica is widely known, secure, accurate, available and complete.
- Cached copy existence depends on locality in access patterns. Existence of replica depends on availability and performance.
- Cache copy is dependent on replica. It needs to be periodically verified with replica then it becomes useful.

6.6.2 Advantages of Replication

Following are the advantages of replication :

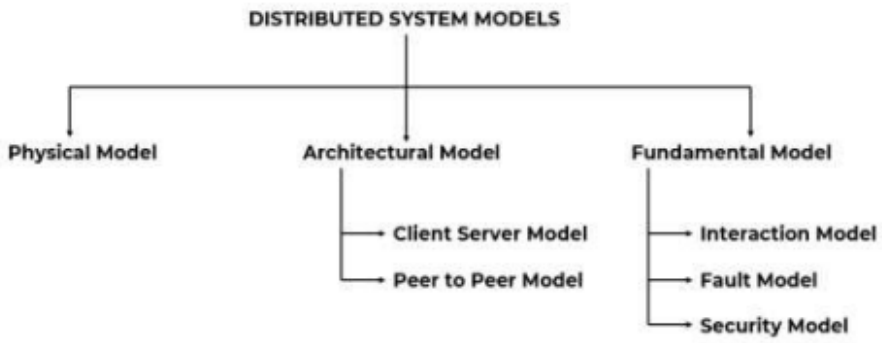
1. **Increased Availability :** Replication offers availability of the system. Although failure occurs, the system remains operational and available to users. The critical data can be replicated on several servers. If primary copy fails, still it can be accessed from other server.
2. **Increased Reliability :** As several copies of the files are available on different servers, recovery in case of failure is possible. Permanent loss of data due to catastrophic failure is also easy to recover from other replicated copy. Hence, replication offers reliability.
3. **Improved Response Time :** Replication allows data to be accessed locally or from the node whose access time is less than that of primary copy access time.
4. **Reduced Network Traffic :** If replica of file is available with file server that resides on client machine then client access request is serviced locally. Hence, it results in reduced network traffic.
5. **Improved System Throughput :** As different client's requests are serviced by different servers in parallel, it results in improved throughput.
6. **Better Scalability :** If file is replicated at multiple file servers then all the client's requests for that file would not arrive at one file server. In this way, load gets distributed and different client's requests are serviced by different servers. It improves scalability.
7. **Autonomous Operation :** All the files needed by clients for limited time period can be replicated on file server that resides on client machine. This ensures temporary autonomous operation of client node.

6.6.3 Replication Transparency

- User should not be aware about file replication. Although file is replicated, it should appear as a single logical file to its user.
- There should be same client interface for replicated and non-replicated file as well. Following are the two important issues related to replication transparency are naming of replicas and replication control.

32. What are the different model of distributed system? Explain.

DISTRIBUTED SYSTEM MODELS:



PHYSICAL MODEL:

1. A physical model is a representation of the underlying hardware elements of a distributed system.
2. It captures the hardware composition of a system in terms of computers and other devices and their interconnecting network.
3. Physical Model can be categories into three generations of distributed systems i.e. early distributed systems, Internet-scale distributed systems and Contemporary distributed systems.

ARCHITECTURAL MODEL:

1. Architectural model defines the way in which the components of the system interact with each other and the way in which they are mapped onto an underlying network of computers.
2. It **simplifies and abstracts the functionality** of the individual components of a distributed system.
3. It describes responsibilities distributed between system components and how these components are placed.
4. **Examples: Client Server Model and Peer to Peer Model**.

FUNDAMENTAL MODEL:

1. The purpose of fundamental model is to make explicit all relevant assumptions about the system we are modeling.
2. It includes interaction, fault and security model.

I) Interaction Model:

- Interaction model **deals with performance and are used for handling time** in distributed systems i.e. for process execution, message delivery, clock drifts etc.
- The two variants of the interaction model are **synchronous and asynchronous distributed systems**.

II) Fault Model:

- Failures can occur both in processes and communication channels.
- The reason can be both software and hardware faults.
- The failure model defines how the failure occurs in the system and what its effects are.
- Fault models are needed in order to build systems with predictable behavior in case of faults (systems which are fault tolerant).
- There are three categories of fault – **omission fault, arbitrary fault and timing fault**.

III) Security Model:

- Security model is based on establishing the trustworthiness and role of each component such as trusted users, trusted servers, trusted administrators and client.
- Security model is used for:
  - Protecting access to objects - Access rights and authentication of clients
  - Protecting processes and interactions.
  - Protecting communication channel.

33. How Monotonic Read consistency model is different from Read your Write consistency Model? Support your answer with suitable example.

II) Monotonic Reads:

1. A data store is said to offer monotonic read consistency if the following condition holds:
   "If process P reads the value of data item x, any successive read operation on x by that process will always return the same value or a more recent one."
2. Consider a Distributed e-mail database.
3. It has distributed and replicated user-mailboxes.
4. Emails can be inserted at any location.
5. However, updates are propagated in a lazy (i.e. on demand) fashion.
6. Suppose the end user reads his mail in Mumbai. Assume that only reading mail does not affect the mailbox.
7. Later when the end user moves to Pune and opens his mail box again, monotonic read consistency assurances that the messages that were in the mailbox in Mumbai will also be in the mailbox when it is opened in Pune.
8. Figure 5.1 shows the read operations performed by a single process 'P' at two different local copies of the same data store.

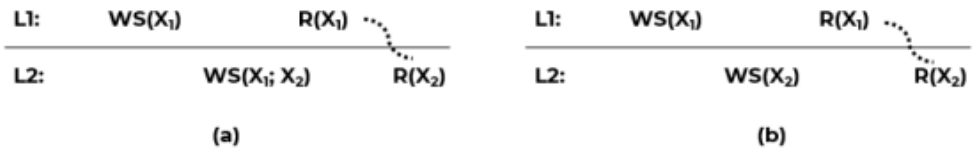


Figure 5.1

9. Figure 5.1 (a) Demonstrate a monotonic-read consistent data store.
10. Figure 5.2 (b) Demonstrate a data store that does not provide monotonic reads.

IV) Read Your Writes:

1. A data store is said to offer read your write consistency if the following condition holds:
   "The effect of a write operation by a process P on a data item x at a location L will always be seen by a successive read operation by the same process."
2. **Example:** Updating a Web page and guaranteeing that the Web browser shows the newest version instead of its cached copy.
3. Figure 5.3 (a) Demonstrate a data store that delivers read your writes consistency.
4. Figure 5.3 (b) Demonstrate a data store that does not delivers read your writes consistency.

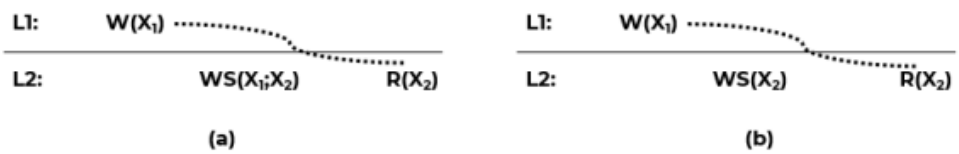


Figure 5.3

-