S741PE R22

ENGINEERING COLLEGE: HYDERABAD
UGC AUTONOMOUS

B.Tech II - semister End Examinations (Regular) - Dec - 2025
Distributed systems
(CSE)

1. a) What are the characteristics of Distributed systems ?

Ans. characteristics of Distributed systems

* Resource sharing
* concurrency
* Scalability
* fault tolerance
* Transparency

b) List the problems of distributed systems.

Ans problems of Distributed systems

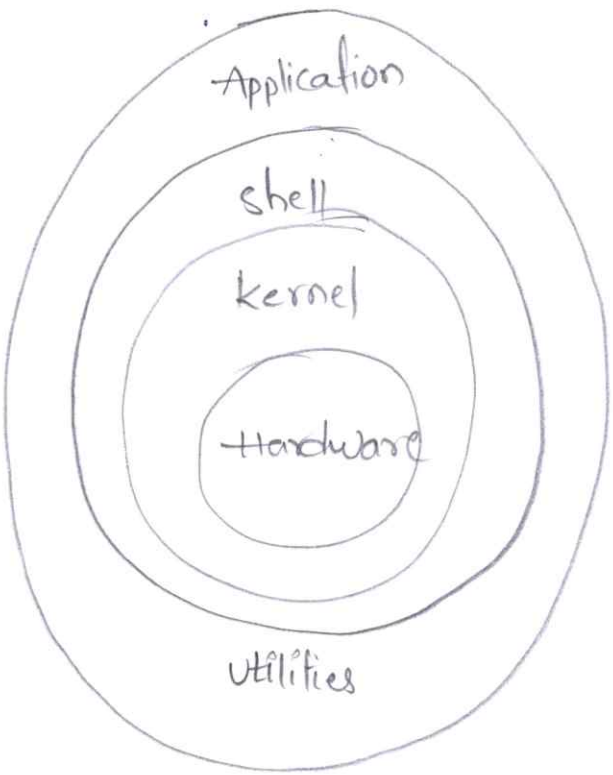
* Security issues
* Network failures and delays
* synchronization problems
* Deadlocks and resource conflicts
* Scalability

c) What is flat file Service interface?

The flat file service is concerned with implementing operations on the contents of files.

---

d) Draw the operating system Architecture?



---

e) Write about notes on Distributed Debugging?

Distributed Debugging is the process of finding and fixing errors in systems with multiple interconnected components.

---

f) When Does a Transaction Abort?

A transaction abort stopping on ongoing operation before its finalized (commited)

→ A transaction aborts when:

* A system failure occurs
* Deadlock happens
* Data conflicts is detected.

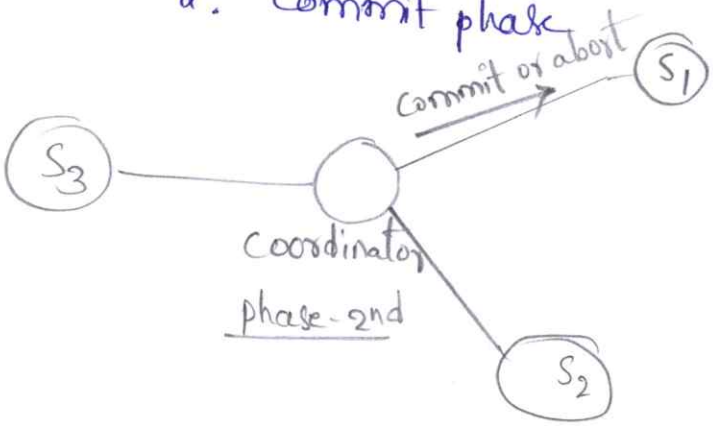
---

g) Define two-phase commit protocol for nested transactions?

Ans:- It is a protocol is a set of rules used to ensure atomicity in distributed nested transactions.

→ It has two phases:

1. voting phase
2. commit phase



---

h) State the nested transactions?

Ans:- nested transactions are transactions inside another transaction (parent-child structure).

* child transactions execute independently.
* parent commit only if all children commit.

PART-B 50 Marks

2. What is Distributed system? Discuss about the challenges for constructing distributed systems?

Ans:- A Distributed system is a group of independent computers that work together and appear to the user as a single unified system. These computers communicate with each other through network and share resources such as files, Data bases / printers, and processing power.

A distributed system is a collection of autonomous computers through a network the cooperate to achieve a common goal and appear as a single system to users.

Example :-

* Internet
* cloud computing (Google Drive, AWS)
* online banking systems
* Distributed databases
* peer-to-peer networks

Challenges in constructing Distributed systems.

1. Heterogeneity :-

Different computers may use :

* Different hardware
* Different operating systems

Different programming languages

Challenges : The system must work smoothly across all these differences.

2. Network Reliability and failures

* Network can fail at any time
* messages may be lost, delayed, or duplicated

Challenge : the system must continue working even if some machines or network links fails.

3. Security :

Adequate protection of shared resources can be provided by using the encryption. the sensitive information can be made confidential when transmitted over the network.
The problem can be still caused by denial of service attacks.

4. Scalability :-

If the cost of adding a user is constant in terms of resources that need to be added, then the distributed system is said to be scable. The performance bottlenecks should be avoided by the algorithms that are used to access the shared data and this data must be structured hierarchically to obtain best access times.

(5) Failure Handling :-

Any computer or network may fail independently, but can effect the functioning of others. Hence each of the component must be aware of possible ways to recover from the failure if the component on which it depends fails.

6. Concurrency :-

Many users and processes run at the same same time. Hence, each of it must be designed to be unaffected with each other in such environment.

7. Transparency :-

The aim of transparency is to hide some aspects of the distributed systems. This means that the systems should be made available as a single instead of integration of multiple components.

The system should hide :

* location of resources
* Failures
* Data movement
* Replication

3 a (a) Discuss how distributed systems are scalable than the centralized systems ? 7

Ans:- **Distributed systems :-**

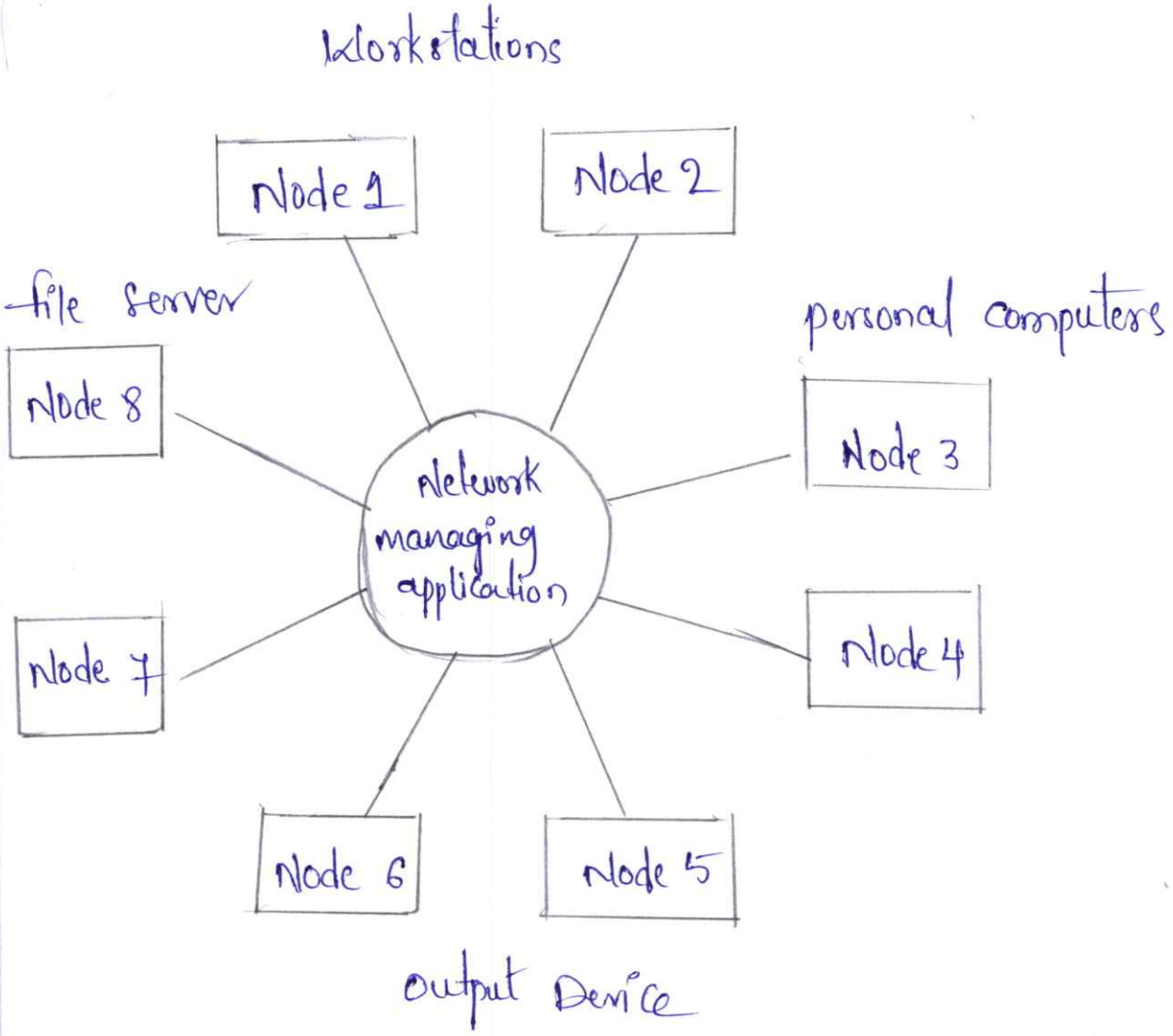
A Distributed system is a collection of independent computers connected through a network that work together and appear to the users as one single system. Each computers in the system has its own processor and memory, and all systems communicate by message passing.

Key features of Distributed systems:

* multiple computers work together
* Resources are shared
* Workloads is divided among many machines
* No single point of failure
* Supports parallel processing
* Easily expandable

Examples :-

* Cloud computing
* Google, facebook servers
* online banking systems
* Distributed databases.



Centralized systems :- A centralized system is a system where all processing, data storage, and control are done by a single central computer. All users depends on this one main server for every operation.

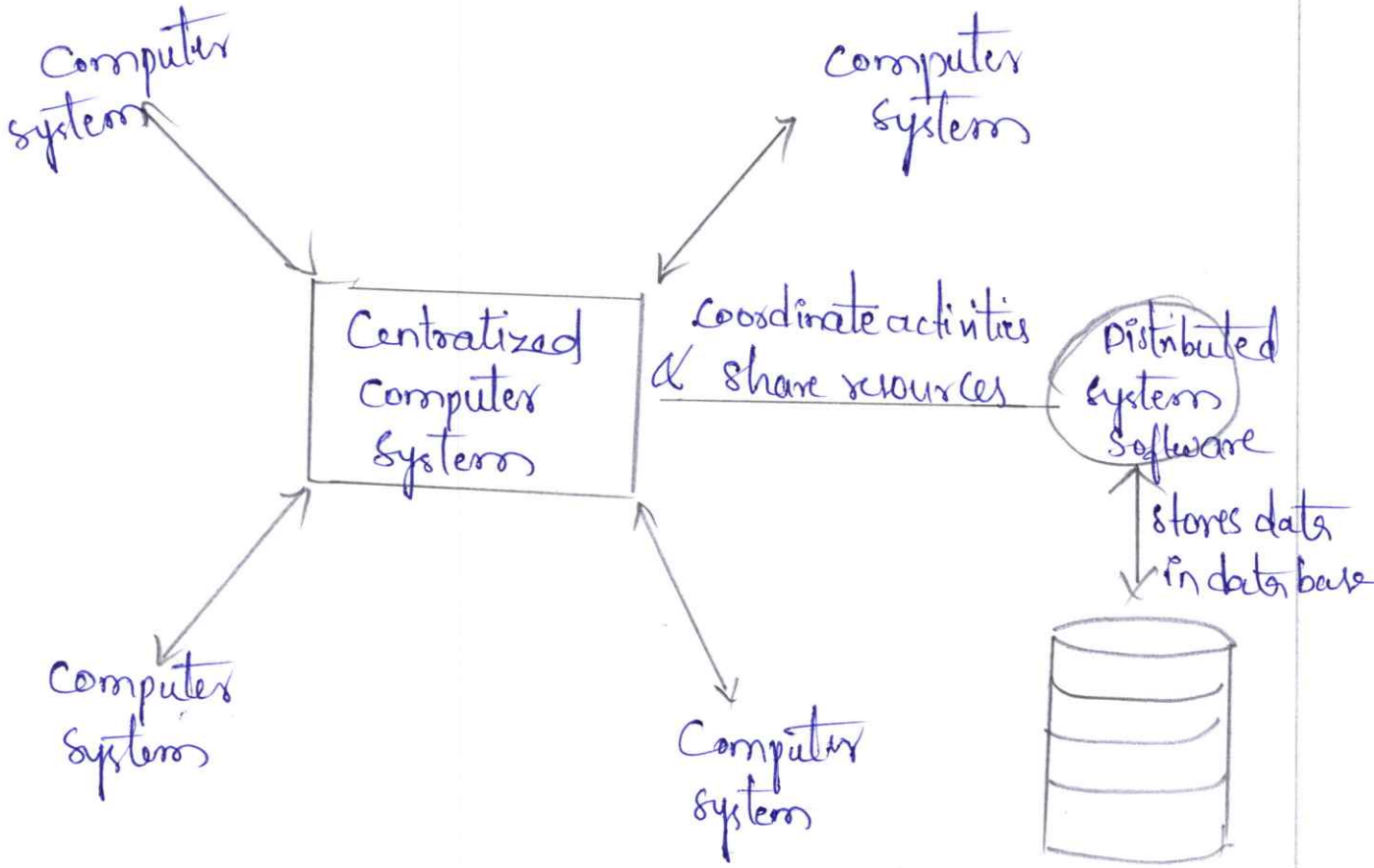
Key features of centralized systems :

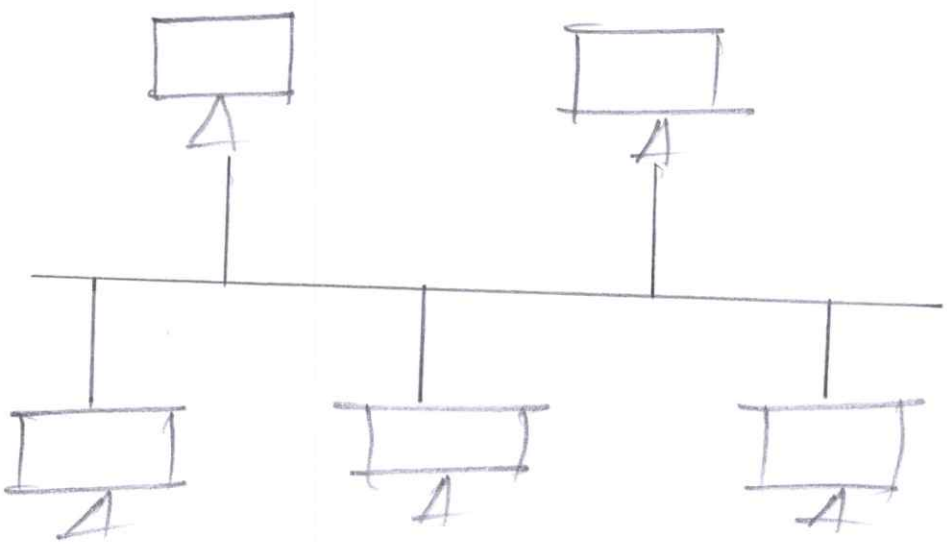
* only one main computer
* All users connect to the same server
* processing and data storage are centralized
* single point of failure
* limited performance

* Difficult expand

Examples :

* old mainframe systems
* single - server office setup
* Traditional database Server systems





10

3) (b) Explain brief the communication between distributed objects ?

Ans:- Distributed objects are objects that exist on different computers in a network but interact with each other as if they are local. The communication between them happens through remote method calls message passing.

1. client - server interaction :

One object acts as a client and sends a request, while the other acts as a server and returns the result.

2. Remote Method Invocation (RMI)/Remote procedure Call (Rpc)

A method of a remote object is called just like a local method.

Example :- A client Calling Data getData() on a remote server object.

3. Marshalling and unmarshalling :-

Marshalling : Converting data into a format suitable for network transmission.

Unmarshalling : converting received data back to original form.

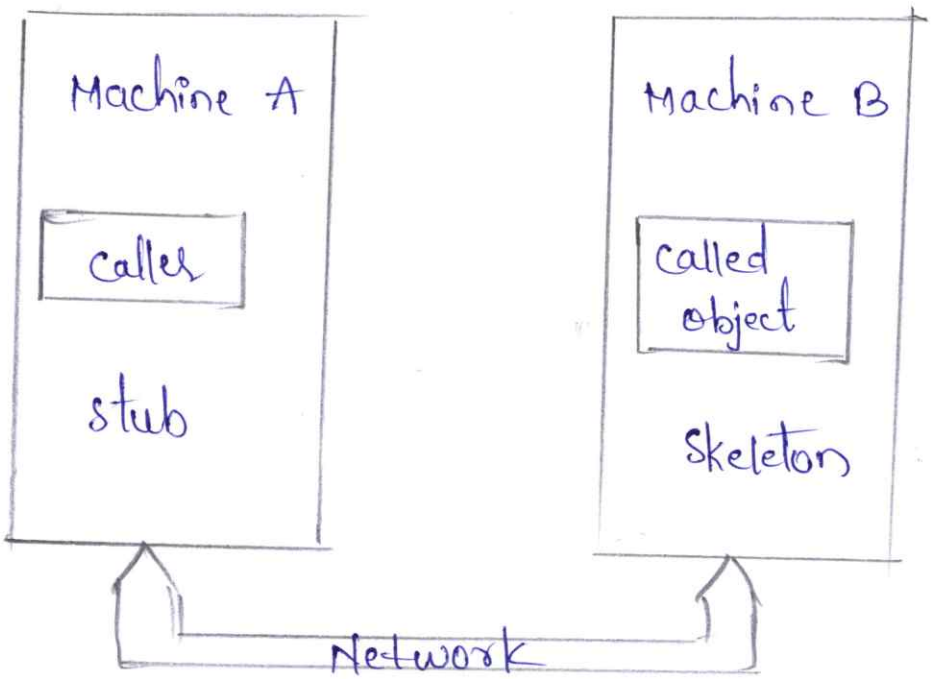
4. Middleware support :

middleware like Java RMI, CORBA, web services manages :-

* object lookup
* Communication
* Data conversion
* Security

5. Message passing :-

objects exchange data using request and reply messages over the network.



12

4 a) Explain the components and operation of a distributed file system.

A Distributed file system (DFS) allows users to store and access files over a network as if they were stored on their local computer. Files may be stored on different machines, but the system provides transparent access.

→ file system are responsible for the organization, storage retrival, naming, sharing and protection of file. They are provide a programming interface that characterises the file abstraction, freeing programmers from concern with details of storage allocation and layout.

file attribute structure :

file length
Creation time stamp
Read time stamp
Write time stamp
Attribute time stamp
Reference count
owner
file type
Access control list

Components :

1. NameNode
2. DataNode
3. Secondary namenode
4. Resource manager
5. Data manager

Functions :

* stores actual data blocks on local disks
* Handles read and write requests from clients
* performs block replication as instructed by nameNode
* sends heartbeat signals to nameNode
* Sends block reports periodically
* Detects and recovers from data failures.

4 b) Discuss the role of invocation and communication in file Servicel. 14

Ans: file Services provide operations like, It provide transparent, network-wide access to files stored across multiple Servers.

* Create
* open
* Read
* write
* close
* Delete

-These operations are performed on remote file Servers.

Key Roles of Invocation :-

1. Remote Access to files :-

A client can invoke file operations on a remote server.

2. location Transparency :

The user does not need to know where the file is stored.

3. Use of Rpc/RMI

-file operations like read() and write() are executed using:

Role of communication in file Services:

Communication is the actual exchange of messages between client and file server over the network.

Key role of communication :-

1. Request-reply Mechanism
   * client sends a request
   * Server sends a reply
2. Data Transfer

Actual file data is transferred through communication.

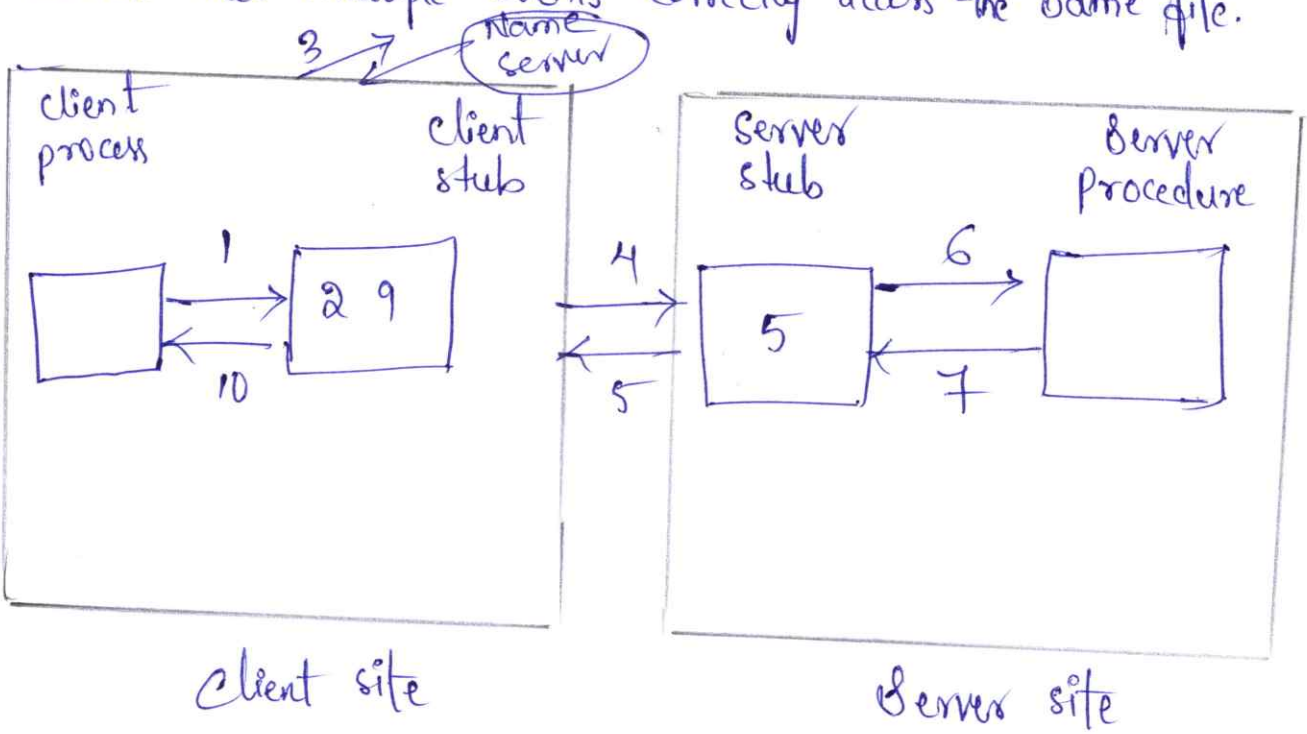
3. Error and Status Reporting

Server sends:

* Success messages
* Error messages

4. Synchronization

Ensure that multiple clients correctly access the same file.



5 (a) Discuss process and thread management in distributed environments. 16

In a distributed environment, programs do not run on a single machine. Instead, processes and threads execute on multiple computers connected over a network. Managing these processes and threads efficiently is essential for performance, reliability, and synchronization.

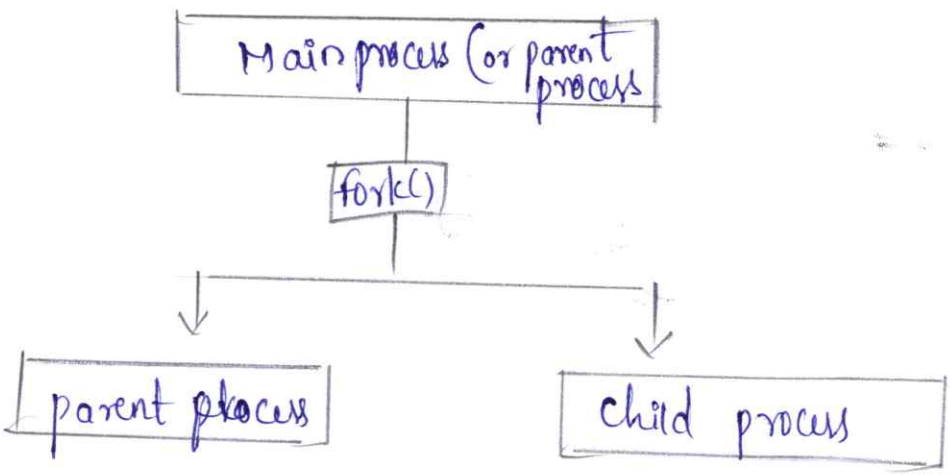
process :

A process is a program in execution. In distributed systems, processes may run on different machines and still work together.

1. process creation

* processes may be created:
  * locally on the same machine
  * Remotely on another machine

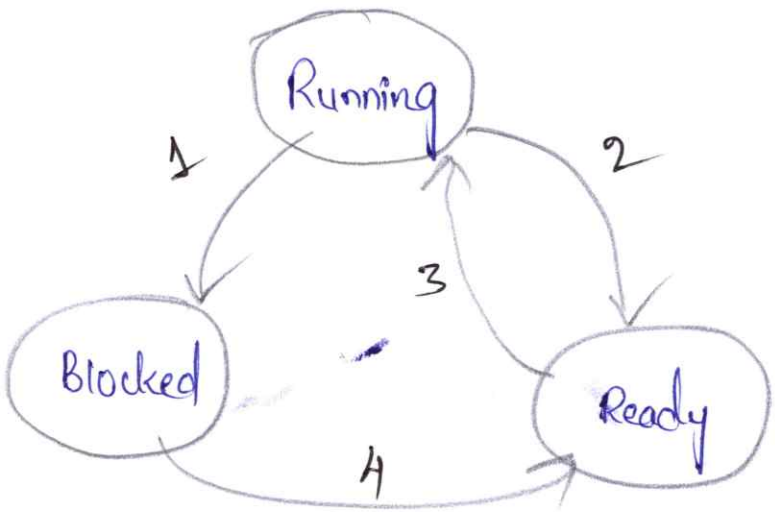
This is done using remote process creation mechanisms.



process means any program is in execution. process control block controls the operation of any proces. process control block contains information about processes for Example process priority, proces id, process state, Cpu, register etc

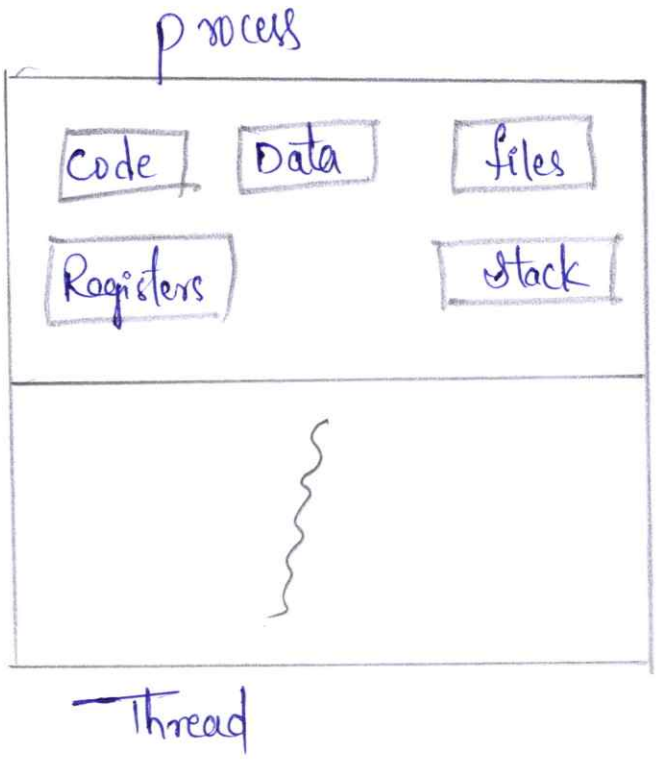
→ A process can create other processes which are known as child processes.

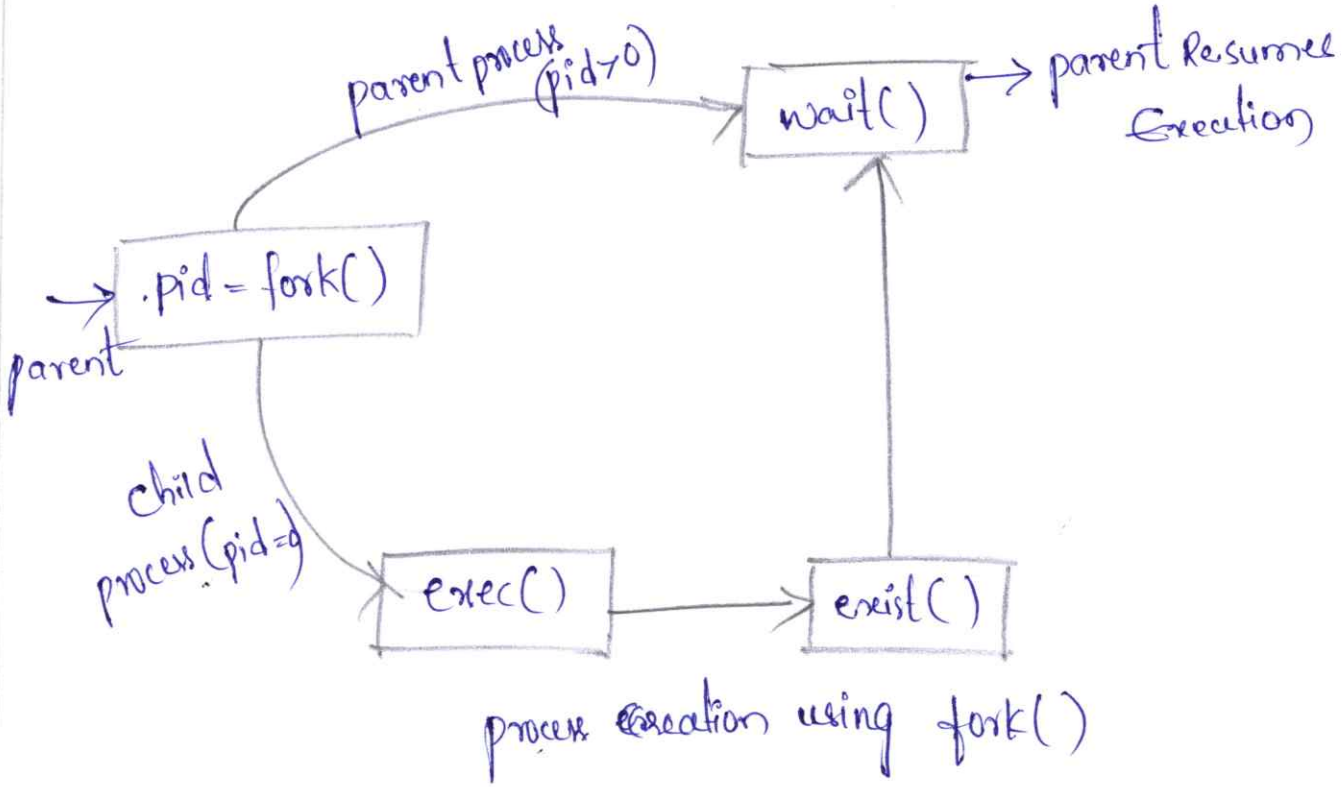
→ proces takes more time to terminate and it is isolated.



1. process block for Input
2. Schedular picks another process
3. Schedular picks their process
4. Input becomes available

Thread have 3 states : running, ready and blocked
Thread take less time to terminate as compared to process and like process threads do not isolate.

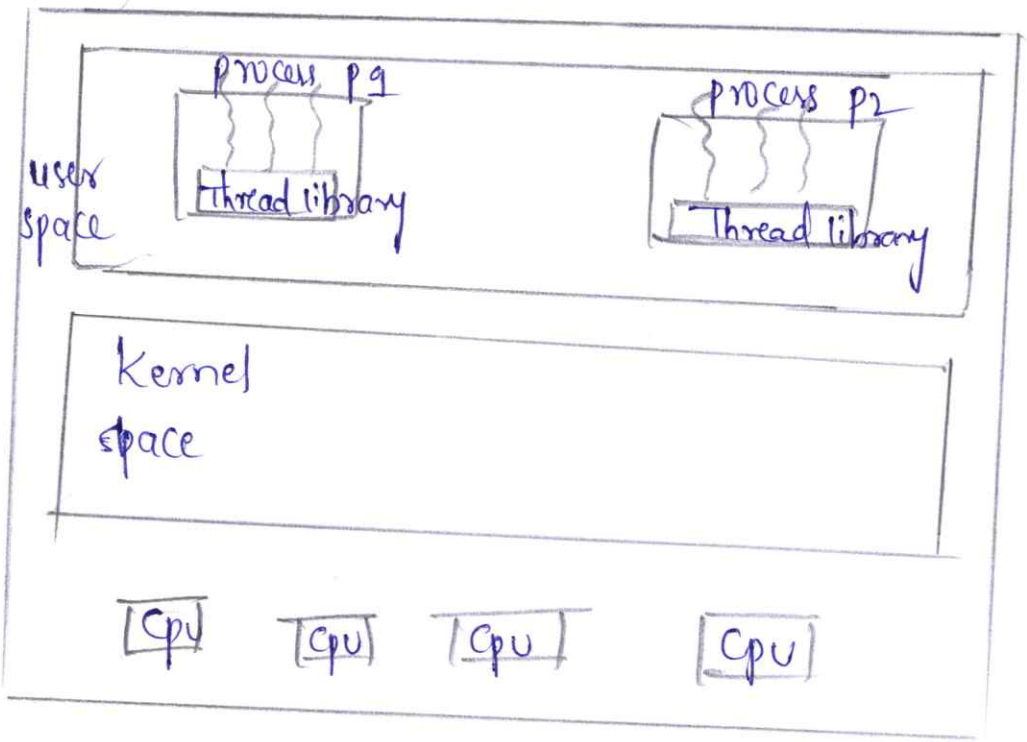




process creation using fork()

Types of threads :-

1. user level thread
2. Kernel level thread



5) b) Explain the layer of operating systems support in distributed systems.

Ans To support distributed systems, the operating system is organized into layers. Each layer provides specific services to enable communication, transparency, and resource sharing across multiple machines.

→ Communication
→ Transparency
→ resource sharing
→ fault tolerance

Layers of operating system support :—

1. Hardware layer

- This is the lowest layer
  - → Includes
    - * cpu
    - * Memory
    - * Network interface card
    - * Disk storage
  - * It provides basic computing and communication capability.

2. Kernel layer

This is the core of the operating system

It provides :-

* process management
* Memory management
* file management
* Device drivers
* Network communication support

The Kernel enables:

* Inter-process communication (Ipc)
* Scheduling of processes and threads

3. Middleware Layer :-

This is the most important layer in distributed systems. It hides the complexity of the network.

Middleware provides :-

* Rpc / RMI
* Naming services
* Time synchronization
* Security services
* Distributed file Services
* Transaction management

Examples :-

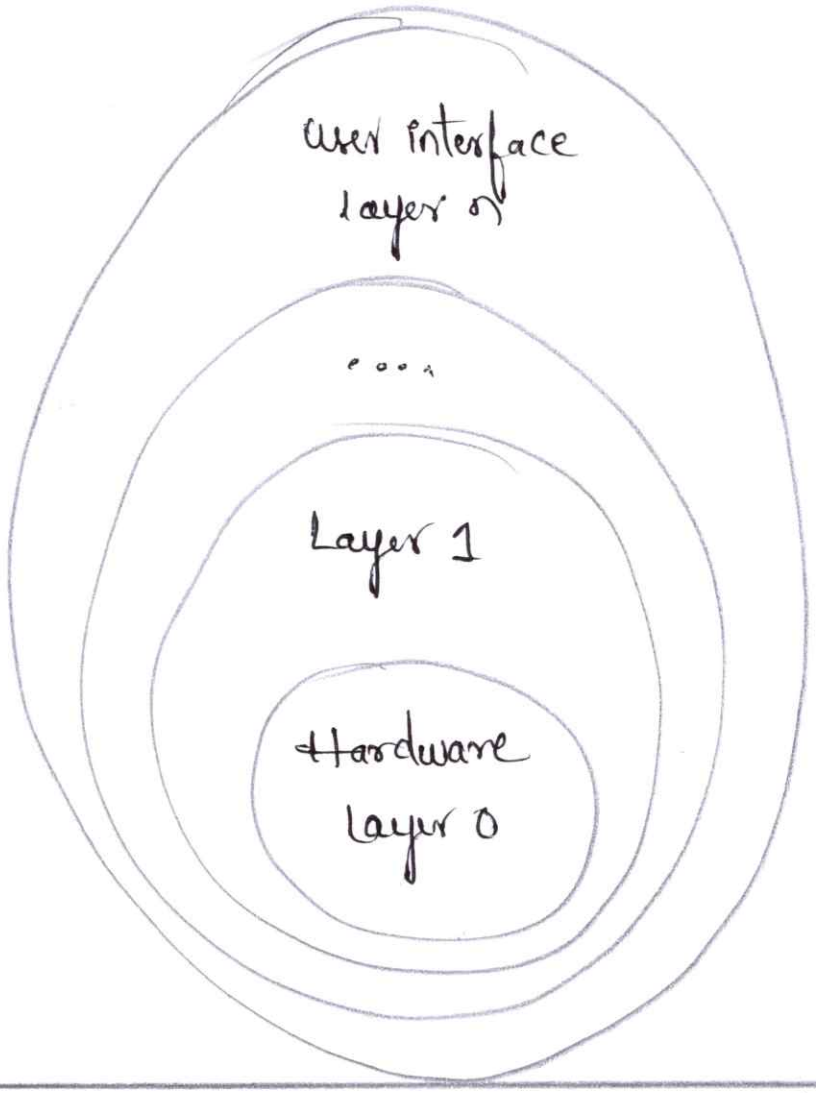
* CORBA
* Java RMI
* Web services

4. Application layer :-

This is the topmost layer:

* User programs and distributed applications run here,

Examples:-
* web applications
* Online banking systems
* Cloud services



---

6 a) with neat sketch explain Routing overlay in detail?

Ans- In distributed systems, especially in peer-to-peer (p2p) networks, node are connected over a physical network like the Internet. On the top of this physical network, a logical network called an overlay network is created.

Routing in this logical network is called Routing overlays.

Why Routing overlays are Nested:

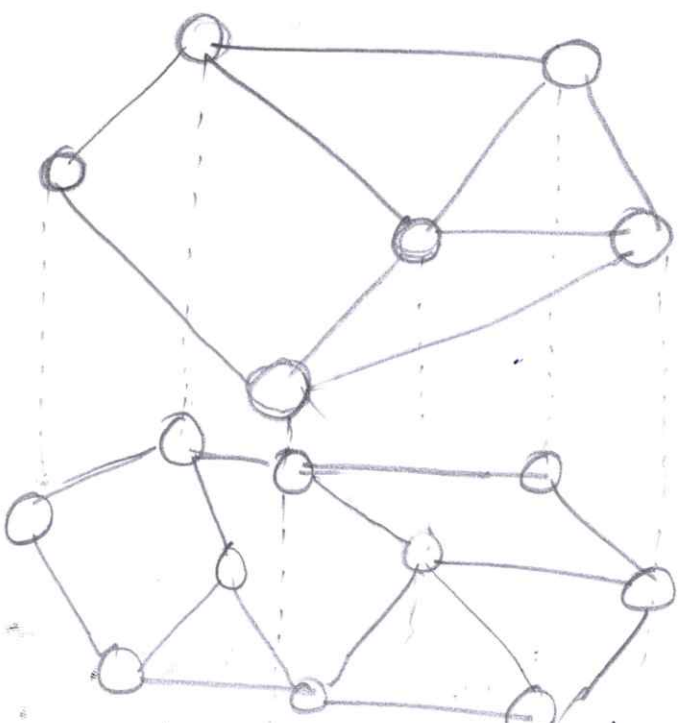
* physical network does not give direct control over routing
* Nodes frequently join and leave
* Need for scalable and fault-tolerant routing
* Support content lookup and resource sharing

Basic Idea of overlay Routing

Instead of using Ip routing only:

* Each node keeps a routing table
* Messages a forward from one overlay node to another
* finally, the message reaches the destination node

Neat sketch of an overlay network:-



Overlay network layered on top of a physical n/w

* physical links are actual network cables
* overlay links are logical connections between selected nodes.

Types of overlays :-

1. unstructured overlays
2. structured overlays

1. Unstructured overlays :-

Examples : Gnutella, Freenet

* nodes are connected randomly
* No fixed structure
* Searching is done using :
  * Flooding
  * Random walk

Advantages :

* Simple to implement
* Good for highly dynamic networks.

Disadvantages :-

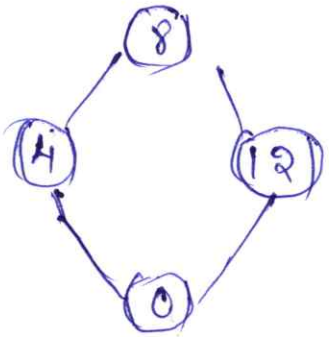
* high network traffic
* Search is inefficient

2. Structured overlays :-

Examples : chord, CAN, pastry, Tapestry

* Nodes are arranged in a fixed logical structure
* Based on Distributed Hash Table (DHTs)
* Each node is responsible for a certain key range

Example : Chord Ring



* Each node stores a set of keys.
* lookup takes O(log N) time

Advantages :-

* very efficient searching
* scalable
* low message overhead

Disadvantages :-

* Complex to design
* Node joining and leaving needs updating

Functions of Routing overlays :-

* Node discovery
* Efficient message routing

* load balancing
* fault tolerance
* Resource location

Applications of Routing overlays :-

* peer-to-peer file sharing
* content distribution networks
* Blockchain networks
* Distributed storage systems.

---

6 (b) Explain clock synchronization and logical time in distributed systems ?

Ans:- In distributed systems, each computer has its own clock. There is no global clock. This creates problems in:

* Event ordering
* Transaction management
* Debugging
* Consistency

So we use :

1. clock synchronization
2. Logical Time

1. Clock synchronization :-

clock synchronization is the process of Making all computers show nearly the same physical time.

It needed has to,

* to correctly timestamp transactions
* To order events
* for distributed databases
* for Security and logging

problems in clock synchronization :-

* clock drift (clocks run at different speeds)
* Network delay
* Message loss

Methods of clock synchronization :-

(a) Cristian's algorithm (b) Berkeley algorithm
(c) Network Time protocol (NTP)

(a) Cristian's algorithm :-

* One nodes acts as time server
* Other needs request time and adjust their clocks
* Based on round-trip delay

(b) Berkeley algorithm

* No global time server
* one node becomes master
* It collects time from all nodes and sends average time

(c) Network Time protocol (NTP)

* used on the Internet
* Synchronizes time with millisecond accuracy

Limitations of physical clock synchronization:-

* Cannot guarantee perfect synchronization
* Affected by:
  * Traffic
  * Delay
  * Failures

2) Logical Time :-

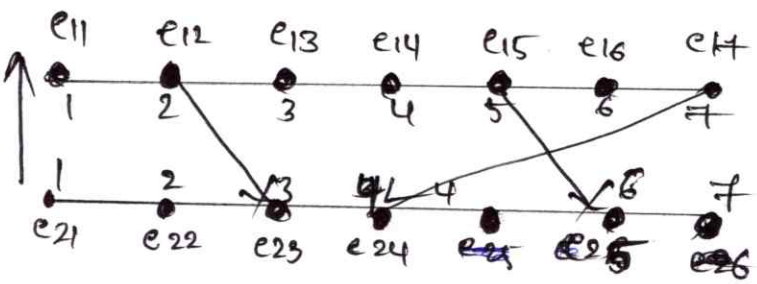
Logical time is a software-based method to order events, without using actual physical time.

It answers:
Which events happened before which?

Lamport logical clock :-

Rules :-

1. Each process maintains a counter L
2. Before executing an Event
3. L = L+1
4. When a message is sent → send L
5. On receiving message
6. L = max (L, received value) + 1



Vector clocks :-

* Extension
* uses a vector of counters
* Gives exact causal ordering
* more accurate but needs more memory.

7a) Explain the architecture and functioning of peer-to-peer systems with suitable examples.

A peer-to-peer (p2p) system is a type of distributed system in which all nodes (peers) are Equal. Each peer can act as both.

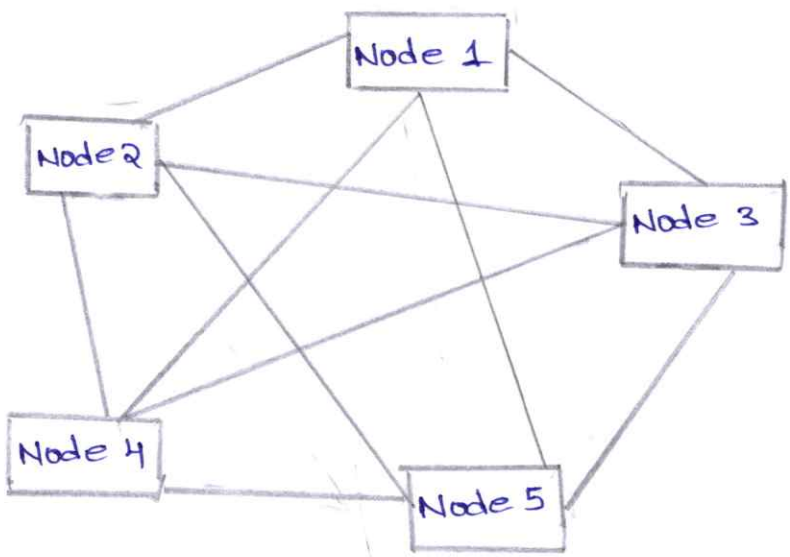
- client (requests services)
- server (provides services)

There is no central controlling server.

Examples:

- BitTorrent
- Napster (early version)
- Gnutella
- Blockchain networks

Architecture of peer-to-peer systems



P2P Architecture

Functioning (Working) of peer-to-peer systems

The Working of a p2p system follows these main steps:

1. Peer Joining
   - A new peer joins the network
   - It connects to one or more existing peers

2. Resource Advertisement
   - Each peer advertises:
     - Files
     - storage
     - services

3. Resource Discovery (searching)
   - A peer searches for required data using:
     - Flooding (unstructured p2p).
     - DHT lookup (structured p2p).

4. Data Transfer
   - once the peer is found, direct peer-to-peer data transfer takes place.

5. Peer Leaving
   - Peers can leave the system at any time
   - The system dynamically adjusts.

Example 1:- online Banking system
- uses replication and backup servers.
- Even if one server fails, transactions continue safely

Example 2:- cloud storage (Google Drive, one Drive)
- Data is stored on multiple servers
- If one server crashes, data is still available.

Example 3:- Airline Reservation system
- uses backup servers and replication.
- continuous service is critical.

7b) Briefly explain the bully algorithm in election.

An election algorithm is used in a distributed system to select one process as the coordinator (leader) when:

- The current coordinator fails
- A new coordinator is needed

The Bully Algorithm is an election algorithm in which the process with the highest process ID is always selected as the coordinator.

Working of Bully Algorithm

1. When a process detects that the leader has failed, it starts an election.
2. It sends an ELECTION message to all processes with higher IDs.
3. If no process replies, it becomes the new coordinator.
4. If any higher-ID process replies with an OK message, then
   - The higher-ID process takes over the election.
5. Finally, the process with the highest ID becomes the coordinator and sends a COORDINATOR message to all others.

Examples.

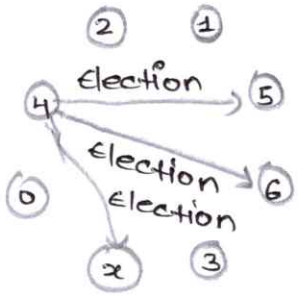
If process IDs are:

P1, P2, P3, P4, P5, P6

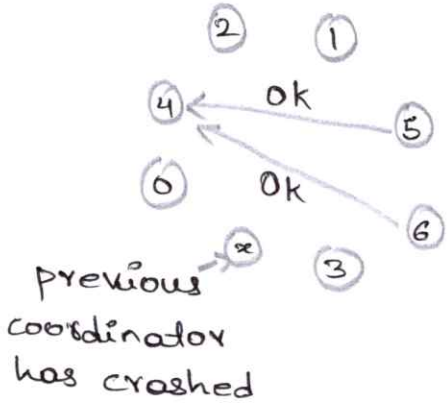
If P3 starts the election:

- It sends messages to P4 & P5
- If only P5 responds

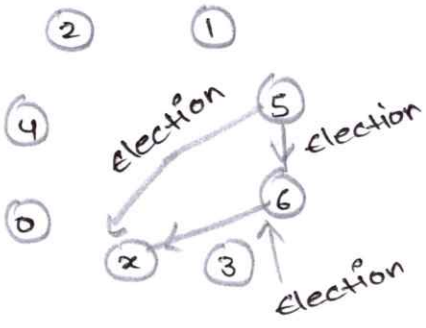
- P_6 becomes the new coordinator



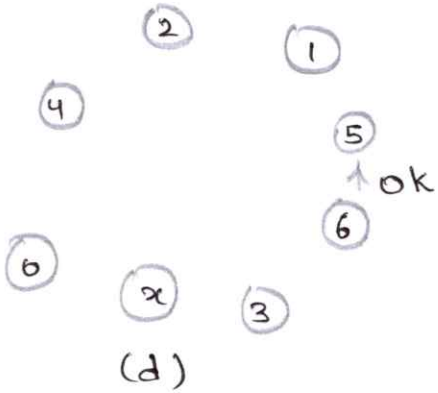
(a)



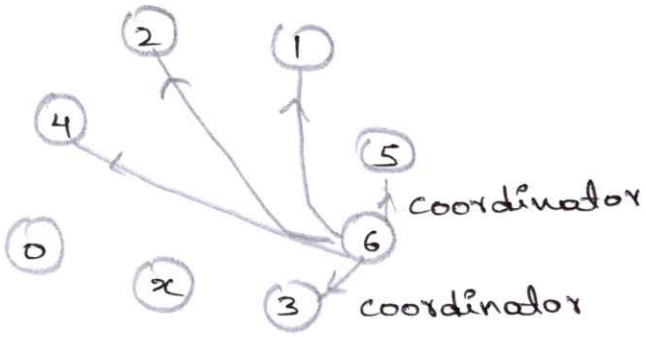
(b)



(c)



(d)



(e)

The Bully Algorithm ensures that the strongest (highest IDs) process becomes the coordinator. It is simple and reliable but causes high ~~measur~~ message traffic.

8a] What are the locking rules for nested transitions?

A nested transaction is a transaction that contains sub-transactions (child transactions).

- The main transaction is called the parent transaction
- sub transactions execute independently
- Final commit happens only when the parent Commits.

Locking in Nested Transactions

In Nested Transactions, locking is more complex than in flat Transactions because:

- Multiple child transactions run under one parent.
- Locks must be managed in a way that ensure data consistently and isolation.

Locking Rules for Nested Transactions

1. Lock Inheritance Rule

- A child transaction inherits all locks held by its parent.
- This means a child can acess all data items already locked by the parent

2. Lock Retention on Commit

- When a child transaction Commits
  - Its locks are not released immediately
  - They are transfered to the parent commits (or) abborts
- Actual lock release happens only when the parents commits.

3. Lock Release (or) Abbort

- If a child transaction Aborts
  - Only the childs changes are Undone.
  - But parent continues execution
  - child locks are released

4. Two-Phases Locking Still Applies

Nested transactions must follow the Two-Phases locking (2PL) protocol:

- Growing Phase: Locks are acquired.
- Shrinking Phase: Locks are released
- Once a transaction releases a lock, it cannot acquire new locks.

5. No conflicts Between Siblings

- Two child transactions of the same parent:
  - Can share locks safely
  - since they belong to the same parent transaction

8b] Explain about two phase commit protocol

The Two-Phase Commit Protocol (2PC) is a distributed transaction protocol that ensures atomicity

It guarantees that a transaction is either:

- Commited at all sites, or
- Aborted at all sites.

There is no partial Commit

Components of 2PC

1. Coordinator - controls the transaction
1. Participants - sites involved in the transactions.

Phases of Two-Phase Commit Protocol

Phase 1: Voting Phase (Prepare Phase)

1. The coordinator sends a PREPARE message to all participants.
2. Each participant
   - checks if it can commit safely
   - writes changes to log
   - sends:
     - VOTE-YES: if it is ready
     - VOTE - NO: if it not commit

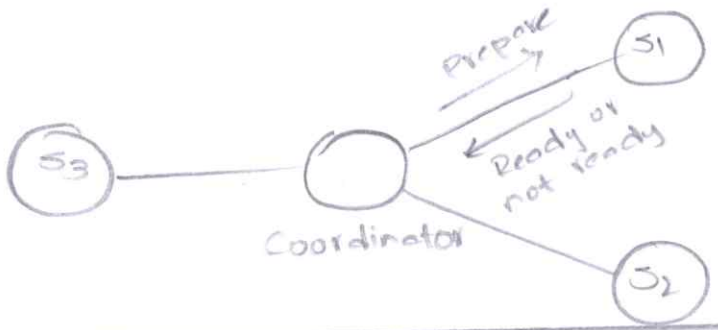
Phase 2: Commit Phase (Decision Phase)

Case 1: if all voters are Yes

- Coordinator sends GLOBAL-COMMIT to all participants
- All participants:
  - Commit the transaction
  - Release locks

Case 2: If Any Vote is NO

- Coordinator sends GLOBAL-ABORT to all participants
- All participants:
  - Abort the transaction
  - Undo all changes



9] Discuss in breif about the two problems associated with aborting transactions and also discuss the way to overcome them.

A transaction abort stopping an ongoing operation before it's finalized (commited). When a transaction aborts in a disturbed system, it creates several issues

Two important problems are security and scalability.

1. Security Problem

When a transaction aborts:

- Practically executed operations may expose sensitive data
- Logs are temporary data may be acessed by attackers
- Roll back operations cand be tampered with.

Result:

This can lead to:

- Partially executed operations may expose sensitive data
- Logs and temporary data may be accessed by attackers
- Unauthorized acess
- Loss of trust in system

2. Scalability Problem

In large distributed systems:

- Frequent aborts cause network overhead
- Repeated rollbacks increase processing load.
- Many sites must coordinate again.

Result:

- Reduce system thoughtput.
- Increased response time.
- Poor performance in large-scale systems

How to Overcome these Problems?

1. Use strong Authentication and Authorization

Ensures only authorized users and systems can acess transaction data, improving security during aborts.

2. Encrypt Transaction Data and Logs

Protects sensitive data during:

- Transmission
- Logging
- Rollback.

This prevents data leakage.

3. Efficient Concurrency Control Mechanisms

Using:

- Two phase Locking
- Time stamp ordering

Reduces conflicts and minimizes aborts, helping scalability

4. Optimizing Commit Protocols

Using improved protocols like:

- Pressured Abort
- Three Phase Commit

Reduces blocking and improves performance in large systems.

5. Load Balancing and Replication

Distributing workload across servers:

- Prevents overload
- Reduces transaction failures
- Improves system scalability and reliability.

10 a) Explain the system model for replication and the role of group communication.

Replication means storing multiple copies of the same data or services at different nodes in a distributed system.

The main goals are:

- High avaliability
- Fault tolerance
- Better perfomance.

Example: A database stored on multiple servers.

System Model for Replication

The system model for Replication explains how replicas are organized and how they behave.

Main components of the Replication System Model.

1. Clients
   - Users or applications that send requests.
   - They can contact any replica.
2. Replicas
   - Multiple copies of the same data or service
   - Stored on different machines
   - All replicas must remain consistent.
3. Communication Network.
   - Connects clients and replicas
   - Used for:
     - Sending requests
     - Updating replicas
     - Synchronizing data

4. Failure Model

Defines possible failures:

* Node failure
* Network failure
* Message loss
* Delay

The system must continue working even if some replicas fail.

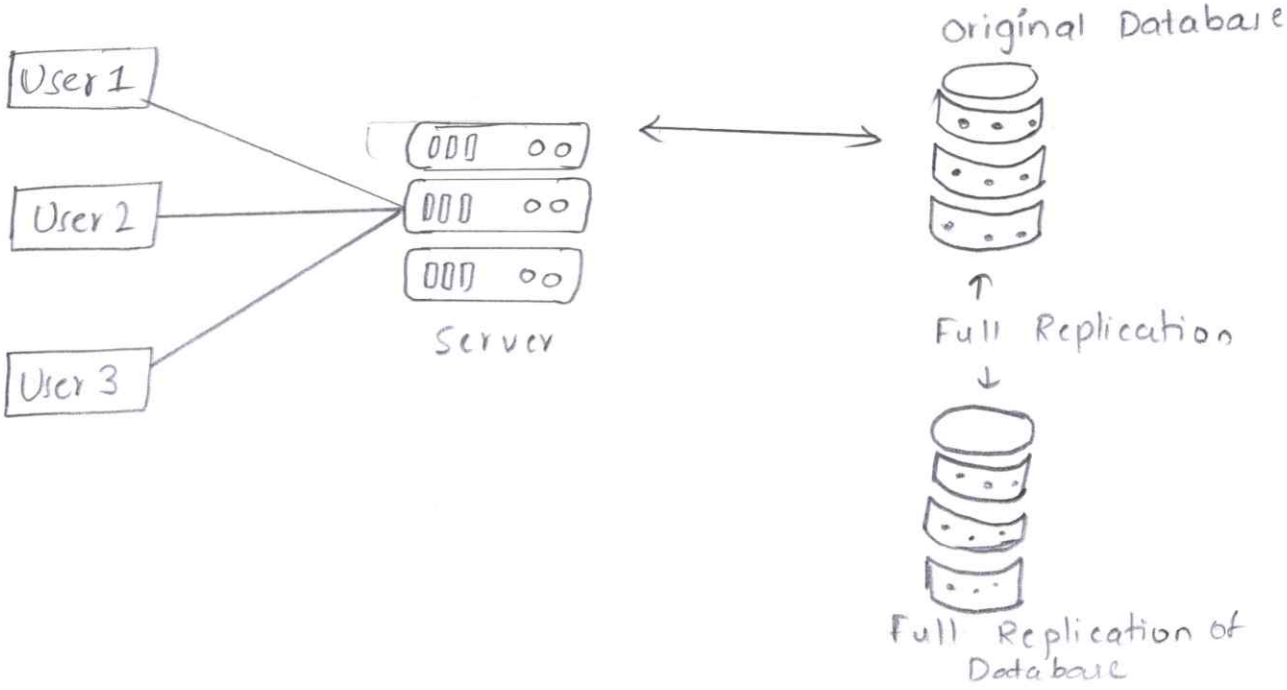
5. Consistency Model

Defines how updates are seen by users, such as:

* Strong consistency
* Weak consistency
* Eventual consistency

Working of Replication System.

1. Client sends requests to a replica
2. That replica updates its local copy
3. Update is propagated to other replicas
4. All replicas reach a consistent state.



Role of Group Communication in Replication.

Group communication is a communication method in which:

* A message is sent to a group of processes at one.
* All members of the Group recieve the message.

Why Group Communication Is Needed in Replication.

In Replication, updata must be sent to all replicas. This is done using group communication.

Roles of Group Communication in Replication.

1. Update Propagation

When one replica updates data, the update is sent to all other replicas using multicast.

2. Consistency Maintainance

Ensures that all replicas apply updates in the same order, maintaining consistency.

3. Fault Detection

Group communication helps in:

* Detecting failed replicas
* Updating group membership

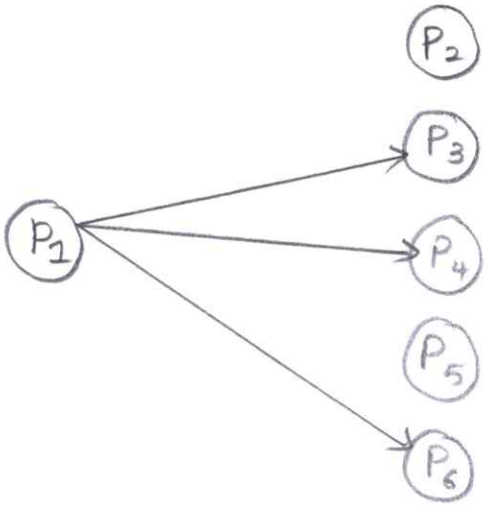
4. Synchronization

Keeps all replicas synchronized during reads and writes.

5. Efficient Communication

One message is delievered to many replicas, reducing:

* Network load
* Communication cost.



10 b) Discuss fault-tolerant services with suitable examples.

A fault-tolerant service is a service that continues to operate correctly even when one or more components fail.

Goal:

* High reliability
* Continuous service
* No system crash.

Why fault tolerance is needed.

* Hardware can fail
* Software can crash
* Network can go down
* Power failure can occur.

Fault tolerance ensures the system doesn't stop working.

Techniques Used to Achieve Fault-Tolerant Services.

1. Replication

Multiple copies of the same service are maintained. If one replica fails, another one continues.

2. Primary - Backup Model

* One server is the primary
* One or more are backups

If the primary fails, a backup takes over.

3. Checkpointing and Roll back

* System state is saved periodically.
* On failure, system restarts from the last checkpoint.

4. Message Logging

All messages are logged. After crash, the process:

* Replays logged messages
* Recovers its state.

5. Heartbeat Mechanism

Nodes regularly send heartbeat messages to show they are alive.

If heartbeat stops → failure is detected.

Example:

1. Telecommunication Networks.

Mobile and Internet service providers use replication and backup routing.

If one communication tower or route fails, calls and data are automatically rerouted through another path, so service continues without interruption.

2. Stock Market Trading System.

Stock exchanges use replicated servers and real-time backups.

No data loss & trading interuption.

3. Hospital Information systems.

Patient records & monitoring systems uses fault-tolerant databases and backup servers.

If one system fails, doctors can still:

- Access history
- View reports
- Monitor vital signs

This is critical for saving lives.

---

11 a) Explain Sequential consistency in ~~Dat~~ Distributed Shared Memory.

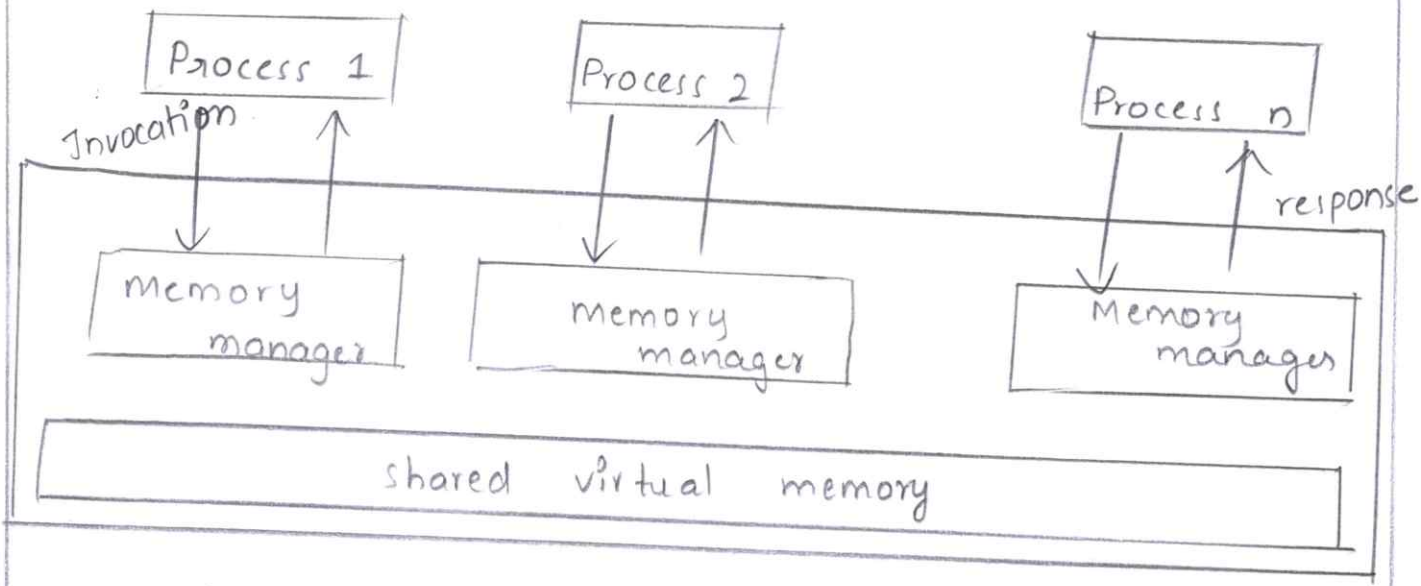
DSM is a memory system that allows process on different computers to share a common logical memory space, even though physically the memory is distributed across multiple machines.

In simple words:

- All processes see the same order of memory updates.
- Each process instructions appear in program order

Key properties

1. Single Global Order of Operations
2. Program Order is maintained
3. Same view for all process.



11 b) What are the requirements for the design of distribute File system?

A DFS allows users to store and access files over a network as if they were stored on their local computer. Files may be stored on different machines, but the system provides transparent access.

Main requirements of DFS.

DFS follows a Master-Slave architecture. It has three main components:

1. NameNode :
   - Stores metadata
   - Manages the entire file system namespace
   - Controls read and write access
2. DataNode (Slave Node)
   - Stores actual data blocks
   - Performs read and write operations

3. Secondary NameNode.

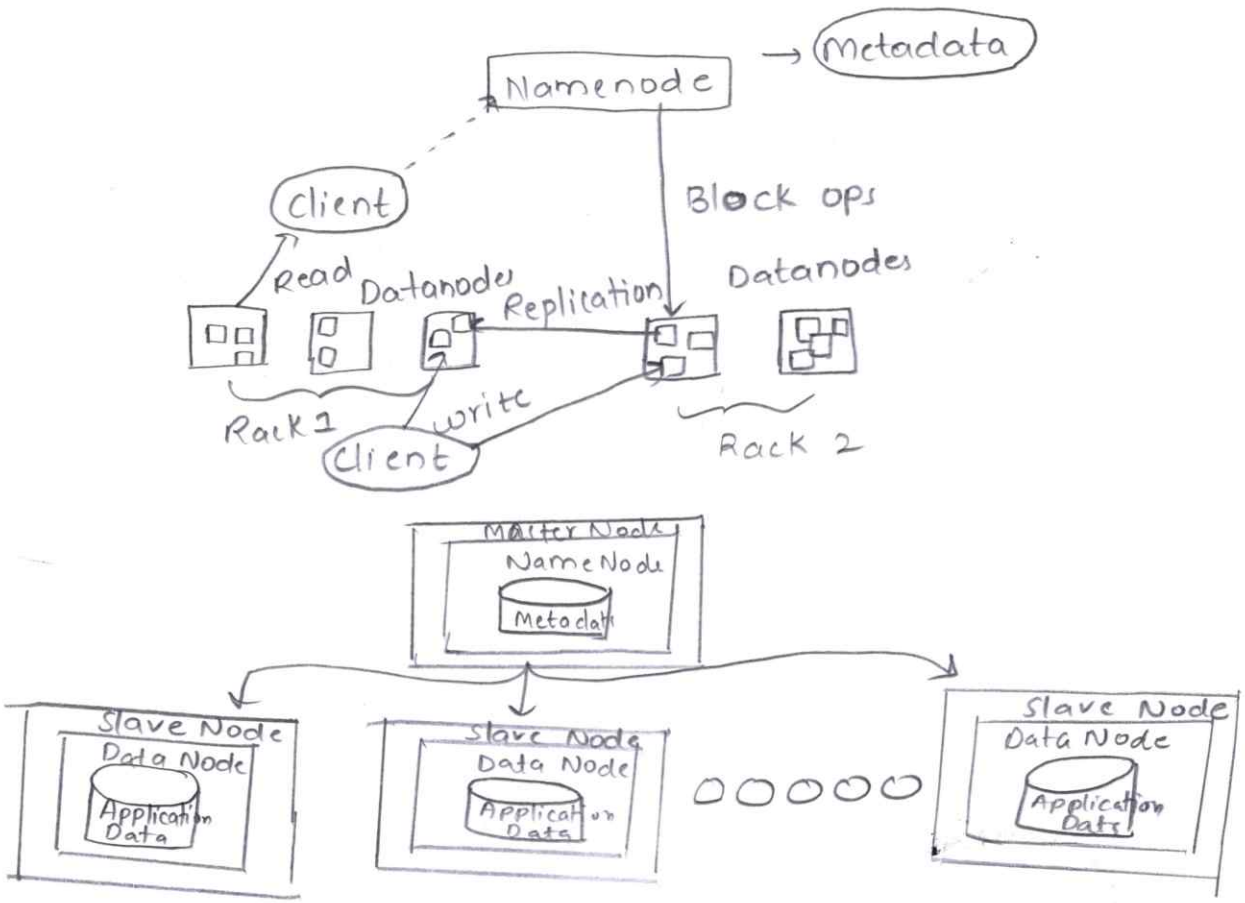
* Creates checkpoints
* Merges: fsimage, edit logs
* Reduces load on NameNode
* Helps in fast recovery.

4. Resource Manager

* Manages CPU, memory etc.
* Allocates resources to different applications
* Schedules jobs based on resource
* Ensures fair resource

5. Data Manager

* Stores actual data blocks on local disks
* Handles read and write requests from clients
* Sends heartbeat signals to NameNode
* Sends block reports periodically
* Detects and recovers from data failures.



SCHEME OF EVALUATION

Subject Code: **R22CS741PE**

Subject Name: **DISTRIBUTED SYSTEM**

Branch-**CSE**

Year/Sem**:-IV/I**

IV-B.TECH-I-SEMESTER END EXAMINATIONS (Regular (R22)-December-2025

Q.NO	DEFINITION	DESCRIPTION	EXPLANATION/ PROGRAMS	DIAGRAM /OUTPUT	MARKS
1.a)	1	-	-	-	1
1.b)	1	-	-	-	1
1.c)	1	-	-	-	1
1.d)	1	-	-	-	1
1.e)	1	-	-	-	1
1.f)	1	-	-	-	1
1.g)	1	-	-	-	1
1.h)	1	-	-	-	1
1.i)	1	-	-	-	1
1.j)	1	-	-	-	1
2	2	-	8	-	10
3a.		2	3	-	5
3b.	-	2	3	-	5
4a.	-	2	3	-	5
4b.	-	2	3	-	5
5a.	-	2	3	-	5
5b.	-	-	3	2	5
6a.	-	-	3	2	5
6b.	-	-	3	2	5
7a.	-	-	3	2	5
7b.	-	2	3	-	5
8a.	-	2	3	-	5
8b.	-	2	3	-	5
9.	-	3	5	2	10
10.a)	-	2	3	-	5
10.b)	-	2	3	-	5
11.a)	-	-	3	2	5
11.b)	-	2	3	-	5