⊛ Distributed Systems ⊛

⊛ Chapter : ① : ⊛

⊛ Topics :

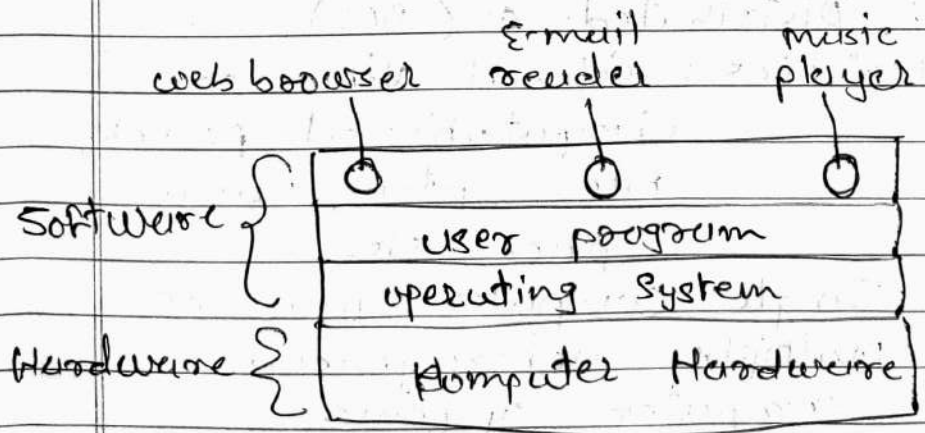
(1) Definition of a Distributed System
(2) Goals of a Distributed system
(3) Types of Distributed system
(4) Basics of Operating System & Networking

0.① What is operating system?

- An operating system (OS) is a system software that manages computer hardware and software resources and provides common services for computer programs.

Ex. Linux, windows, mac os, etc.

- it is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.



∴ OS Architecture ∴

⊛ Evolution of Modern OS:

[1] First Generation OS:

- System: centralized OS
- Characteristics: process management
  memory management
  I/O management
  file management
- Goals: Resource Management

[2] Second Generation OS:

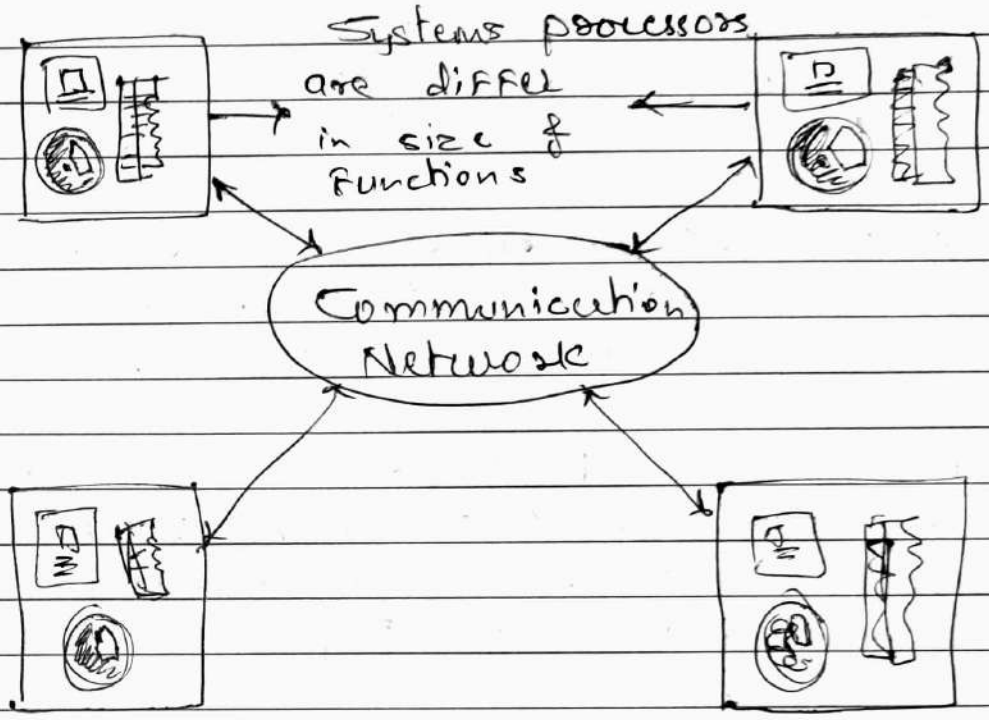
- System: Network OS (NOS)
- Characteristics: Remote access
  Information exchange
  Network browsing
- Goals: Interoperability - sharing of resources between the systems.

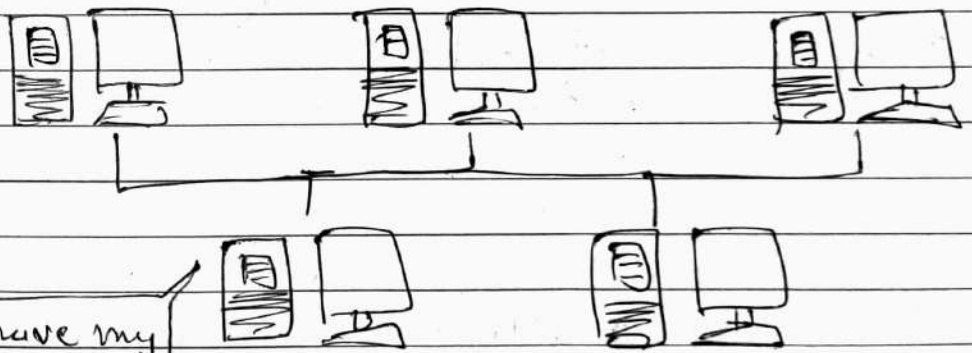
[3] Third Generation OS:

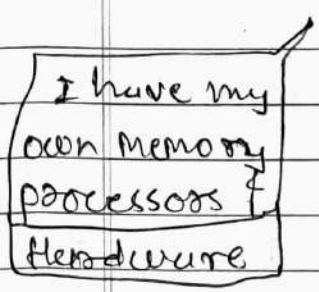
- System: Distributed OS (DOS)
- characteristics: Global view of computational power, file system, name space, etc.
- Goals: Single computer view of multiple heterogeneous computer systems.

Q.② What is Distributed Operating System?

- "A Distributed System is collection of independent computers which are connected through network."







⑧ characteristics : seperate physically Networked Independent Communicating

- Definition by, Coulouris, Blair, Dollimore, Kindberg

↓

"A distributed system is defined as one in which components at networked computers communicate and coordinate thier actions only by passing messages."

↓

"A Distributed system is collection of independent computers which are Connected through network."

- This system looks to its users like an ordinary Centralized operating System but runs on multiple, independent central processing units.

⊛ Examples of Distributed systems:

- Telephone networks & Cellular Networks
- Computer network such internet
- ATM Machines
- Mobile computing
- web Search Engines etc.
- Massively multiplayer online games

* * *

Q.(3) Difference Between Network & Distributed

No.	Network OS	Distributed OS
(1)	A network os is made up of software & associated protocols that allow a set of computer network to be	A distributed os is an ordinary centralized os but runs on multiple independent CPUs.
(2)	Environment users are aware of multiplicity of machines.	Environment users are not aware of multiplicity of machines.
(3)	Control over file placement is done manually by the user.	It can be done automatically by the system itself.
(4)	No implicit sharing of ~~goods~~ clouds.	Sharing of code between nodes.
(5)	Highly Scalable.	Less Scalable.
(6)	performance is badly affected if certain part of the hardware starts malfunctioning.	It is more reliable or fault tolerant. i.e, distributed os performs even if certain part of the hardware starts malfunctioning
(7)	Easy to Implement.	Difficult to Implement.
(8)	Low Transperency.	High Transperency.

Q.④ Goals of Distributed System.

(1) The relative simplicity of the software:

- Each processor has a dedicated function.

(2) Incremented growth:

- If we need 10 percent more computing power, we just add 10 percent more processors.

(3) Reliability and availibility:

- A few parts of the system can be down without disturbing people using the other parts.

(4) Openness:

- An open distributed system is a system that offers services according to standerd rules that describe the syntax and semantics of those services.

(5) Making Resources Accessible:

- Make it easy for the users to access remote resources.
- To share them in a controlled and efficient way.

Page No.
Date

Q. Advantages of DS over Centralized systems & Disadvantages

[1] Economics:

- A collection of microprocessors offer a better price/performance than mainframes. It is an cost effective way to increase computing power.

[2] Speed:

- A distributed system may have more total computing power than a mainframe.

[3] Inherent distribution:

- Some applications are inherently distributed. Eg. a supermarket chain, Banking, etc.

[4] Reliability:

- If one machine crashes, the system as a whole can still survive. Higher availibility and improved reliability.

Eg. Control of nuclear reactors or aircraft.

[5] Data Sharing:

- Allow many users to access to a common database.

[6] Resource sharing:

- Expensive peripherals such as color laser printers, photo-type setters and massive arrived storage devices are also among the few things that should be shareable.

[7] Communication:

- Enhance human-to-human communication, e.g. Email, chat.

[8] Flexibility:

- Spread the workload over the available machines.

(2) Disadvantages of DS over Centralizes System:

[1] Software: would be complex.

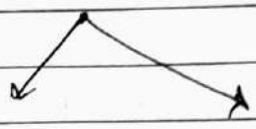
[2] Network problem:

- Network saturation.
- Malfunctioning of network.

[5] Security:

- possibility of security violation since the private data are visible to others over the network.

Q. (6) Classification of Distributed System:

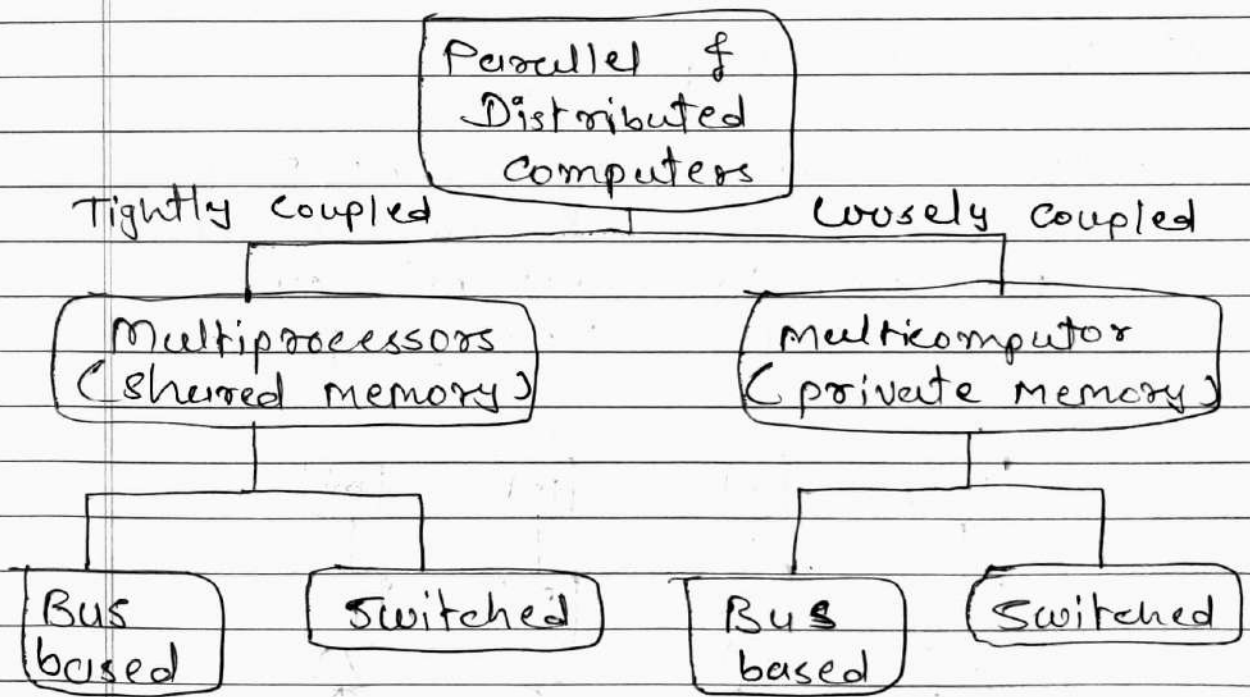


Based on Hardware

Based on number of instructions and Datastream

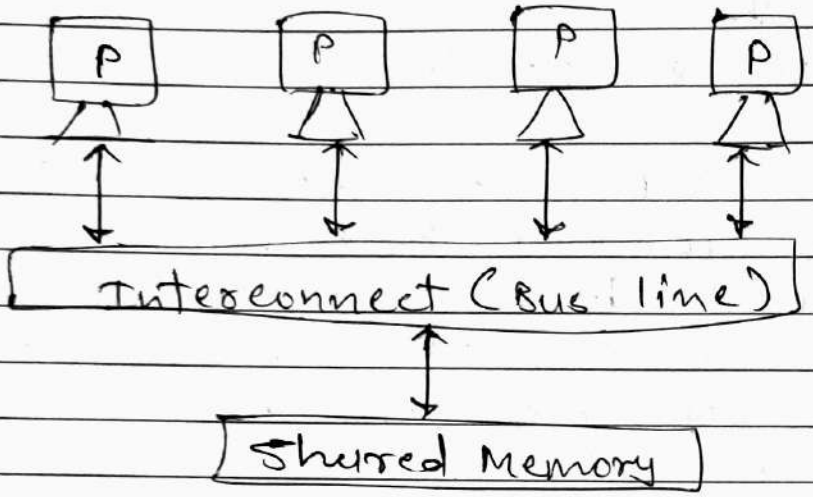
[i] Based on Hardware:

- Even though all distributed system consists of multiple CPUs, there are several different ways the hardware can be organized, specially in terms of how they are interconnected and communicate.



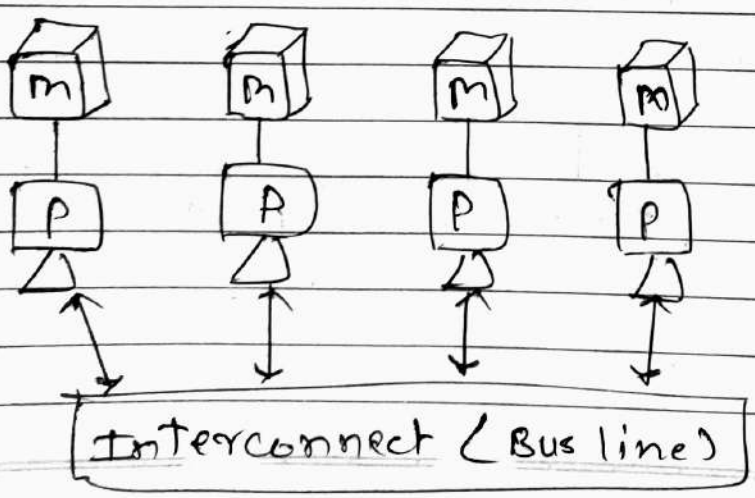
[1] Tightly - Coupled OS:

- Shared Memory Machine: There is processors shared physical address space. Communication can be done through shared memory.



[2] Loosely - Coupled OS:

- Private Memory Machine: Each processor has its own local memory. Communication can be done through message passing.



Page No.
Date

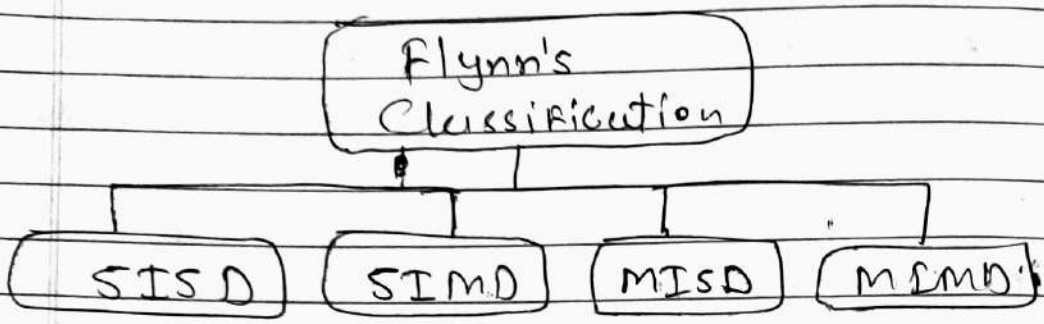
⑧ Difference between Tigutly & coosely coupled.

	Coosely Coupled	Tigutly Coupled.
(1)	Each processor has its own local memory.	The n processors shares physical address space.
(2)	Communication can be done through message passing.	Communication can be done through shared memory.
(3)	Manages heterogeneous multicomputer OS.	Manages multiprocessors & heterogeneous multicomputers.
(4)	Similar to "local access feel" as a non-distributed, standalone OS.	provide local services to remote clients via remote logging.
(5)	Data migration or computation migration modes.	Data transfer from remote OS to local OS via FTP.
(6)	Distributed operating System (DOS).	Network Operating system (NOS).

⑨ (2) Based on Instruction & Datastream.

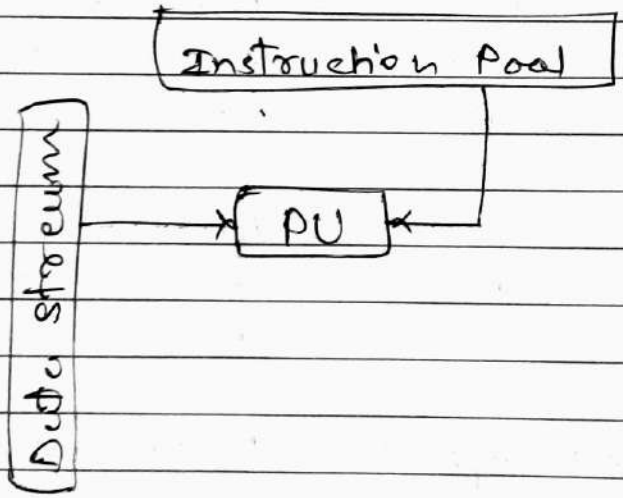
- According to Flynn's classification can be done based on the number of instruction

streams & number of data streams.



[1] Single instruction stream single datastream (SISD):

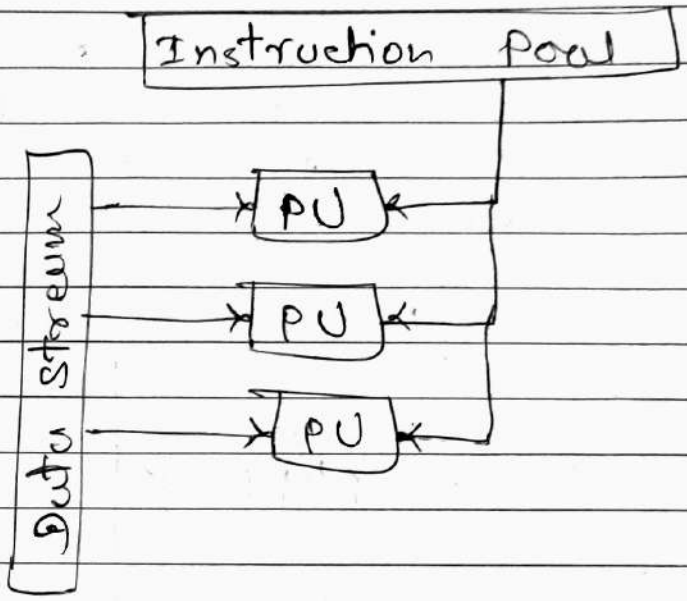
- one program counter & one path to data memory.
- A computer is capable of executing one instruction at a time operating on one piece of data.
- An ordinary (sequential) computer.



[2] Single instruction stream, multiple data streams (SIMD):

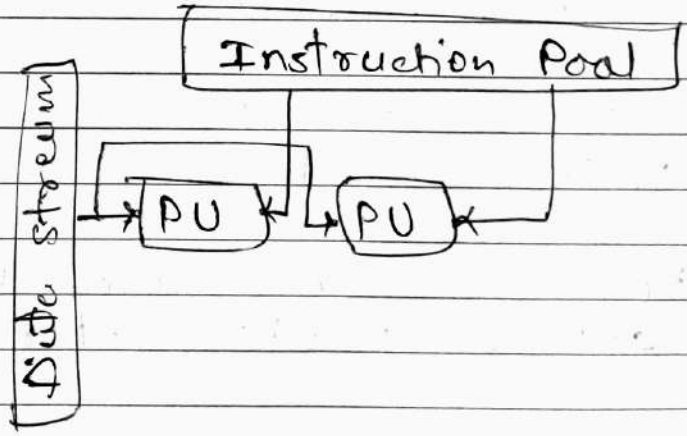
- one program counter & multiple paths to data memory.

- A computer is capable of executing one instruction at a time, but operating on different ~~of~~ pieces of data.



[3] Multiple instruction streams, single data stream (MISD):

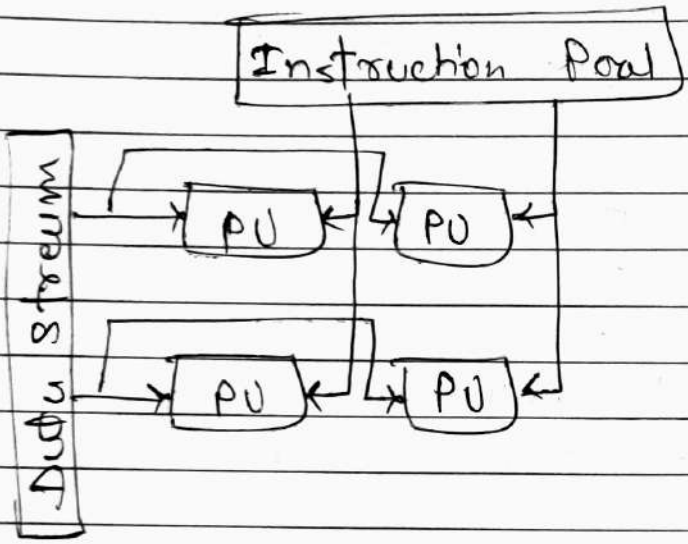
- No more computers fit this model.
- Uncommon architecture which is generally used for fault tolerance.



[4] Multiple instruction streams, multiple data stream (MIMD):

- A group of independent computers, each with its own program counter, program, and data.

- A computer that can run multiple processes or threads that are cooperating towards a common objective.



All distributed systems are MIMD, we divide all MIMD computers into two groups:

- Have shared memory, usually called multiprocessors.

- Do not have shared memory, called multicomputer.

Q.⑦ Distributed Computing System Models:

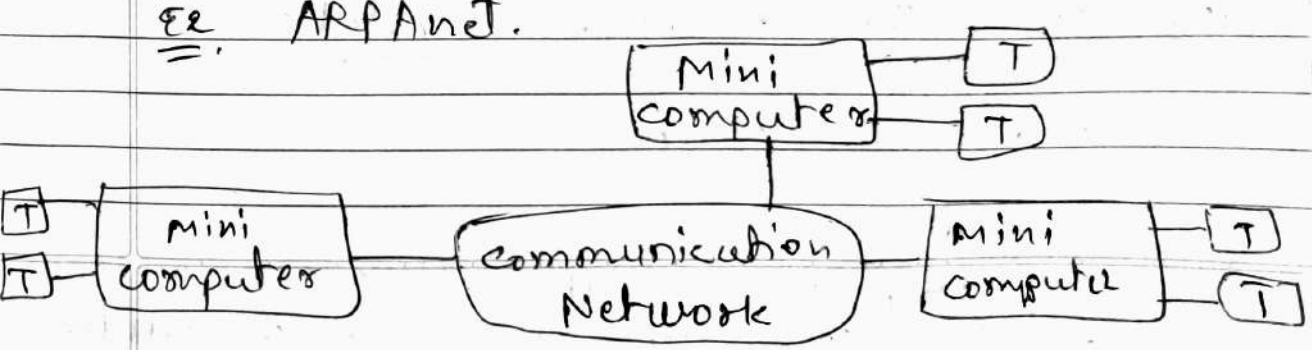
- Distributed computing system models can be broadly classified into five categories.

[1] Minicomputer model
[2] Workstation model
[3] Workstation-server model.
[4] Processor-pool model
[5] Hybrid model.

[1] Minicomputer Model:

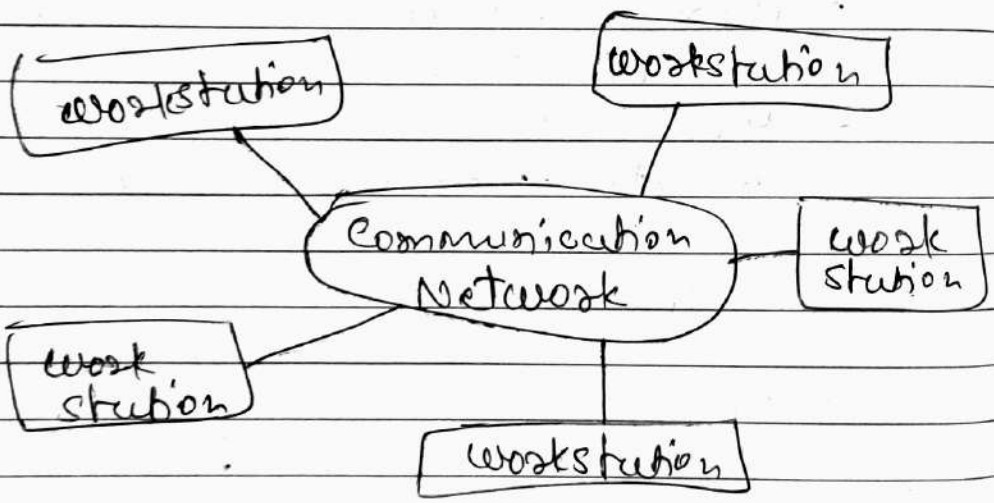
- Extension of Time Sharing system:
  - user must log on his/her home minicomputer.
  - Thereafter, he/she can log on a remote machine by telnet.
- Resource Sharing
  - Database
  - High-Performance devices

Eg. ARPAnet.



[2] Workstation Model:

- Process migration
  - users first log on his/her personal workstation.
  - If there are idle remote workstations, a heavy job may migrate to one of them.
- problems:
  - What if a user log on the remote machine.
  - How to find an idle workstation
  - How to migrate a job.



[3] Workstation - Server Model:

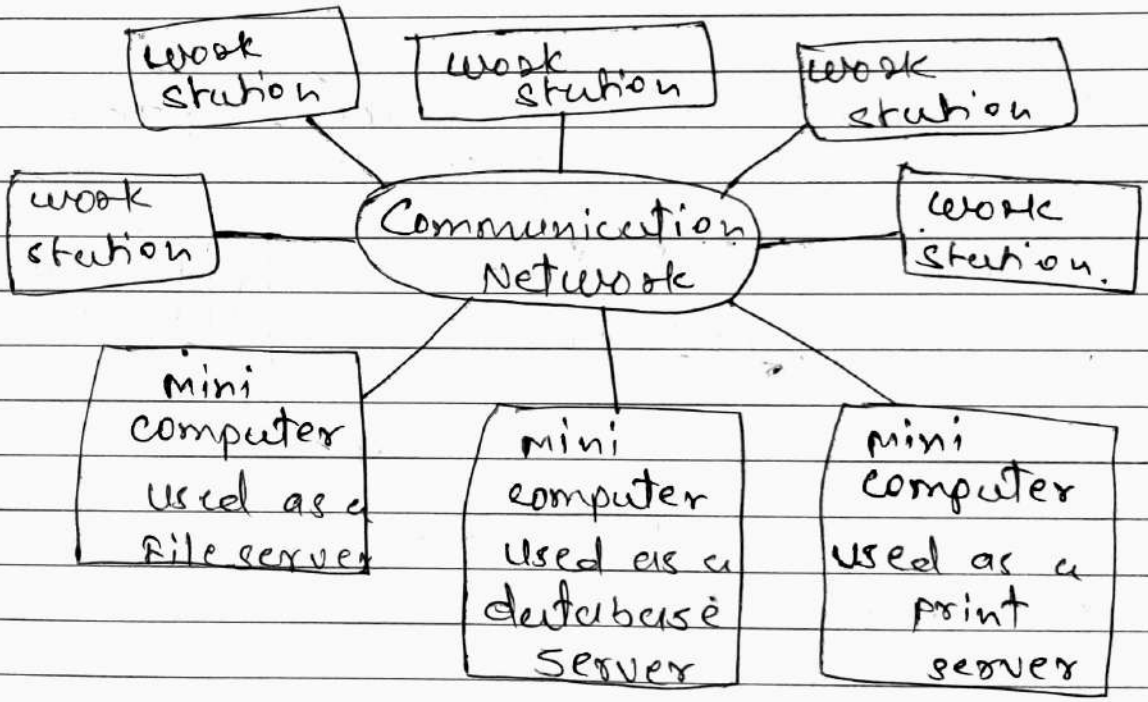
- clients workstations
  - Diskless
  - Graphic application processed in local
  - All file, print, http & even cycle computation

- Server minicomputers

  - Each minicomputer is dedicated to one or more different types of services.

- client-server model of communication

  - RPC (Remote Procedure Call)
  - RMI (Remote Method Invocation)
    - A client process calls a server process function.
    - No process migration invoked.



[4] Processor-Pool Model :

- clients :

  - they log in one of terminals
  - All services are dispatched to servers.

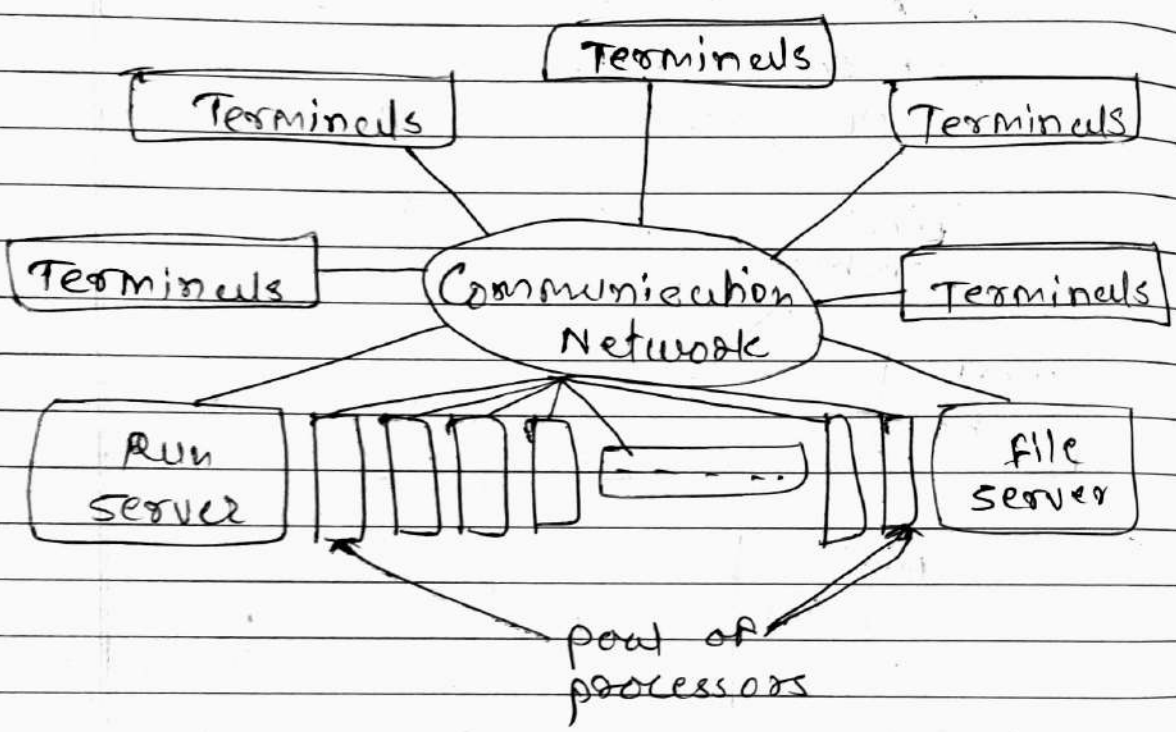
requests are sent to servers.

- Servers :

- Necessary number of processors are allocated to each user from the pool.

- Better utilization of resources.

Ex. Web search Engines.



[5] Hybrid Model :

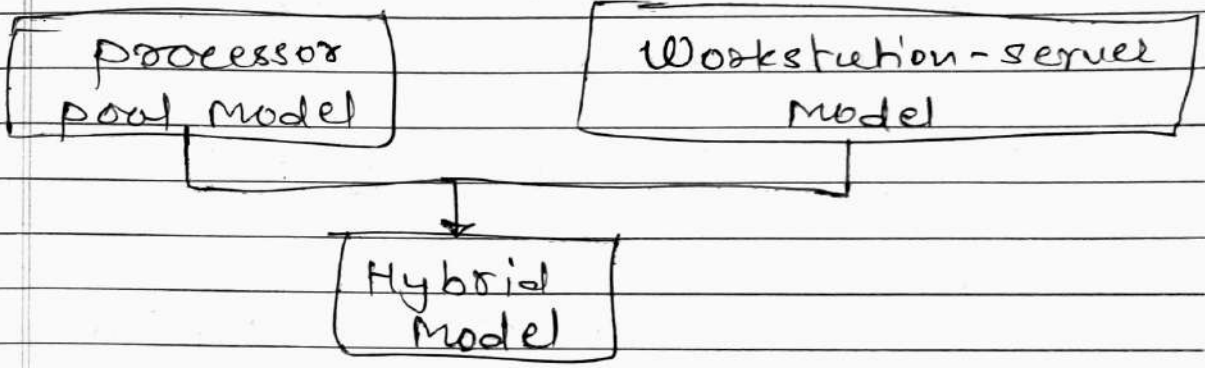
- Advantages of the workstation-server and processor-pool models are combined to build a hybrid model.

- it is built on the workstation-server model with a pool of processors.

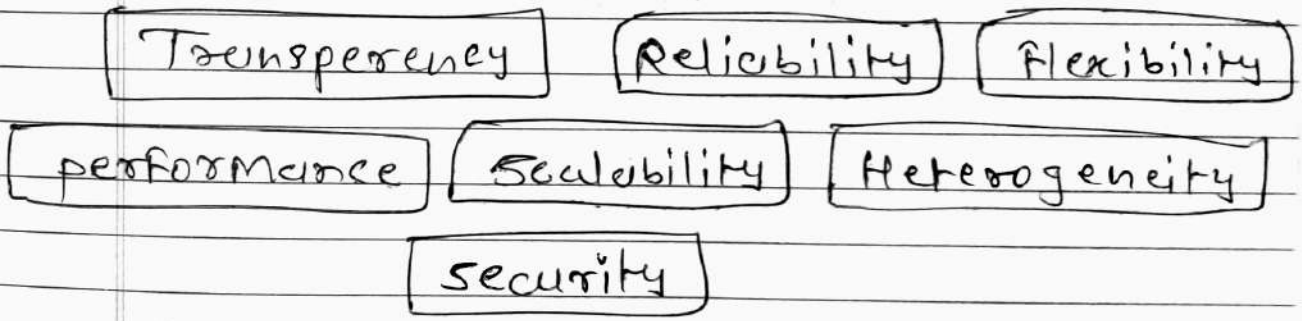
- processors in the pool can be allocated dynamically for large

computations, that cannot be handled by the workstations, & require several computers running concurrently for efficient execution.

- This model is more expensive to implement than the workstation-server model or the processor-pool model.



Q.⑧ Issues in Designing DS.



[1] Transperency :-

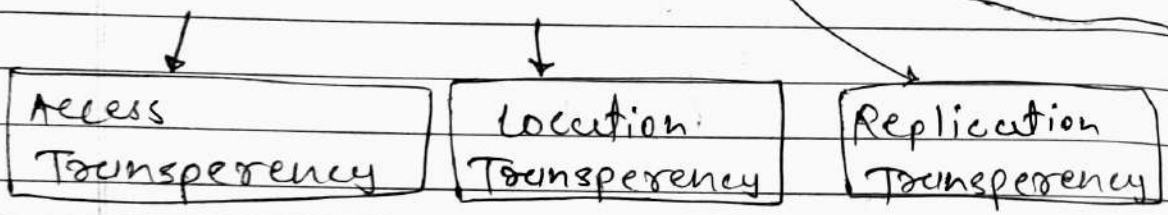
- main goal of Distributed system is to make the existance of multiple computers invisible (transperent) and

provide single system image to user.

- A transparency is some aspect of the distributed system that is hidden from the user (programmer, system developer, application).

- while users hit search in google.com, they never notice that their query goes through a complex process before google shows them a result.

② Types of Transparency:

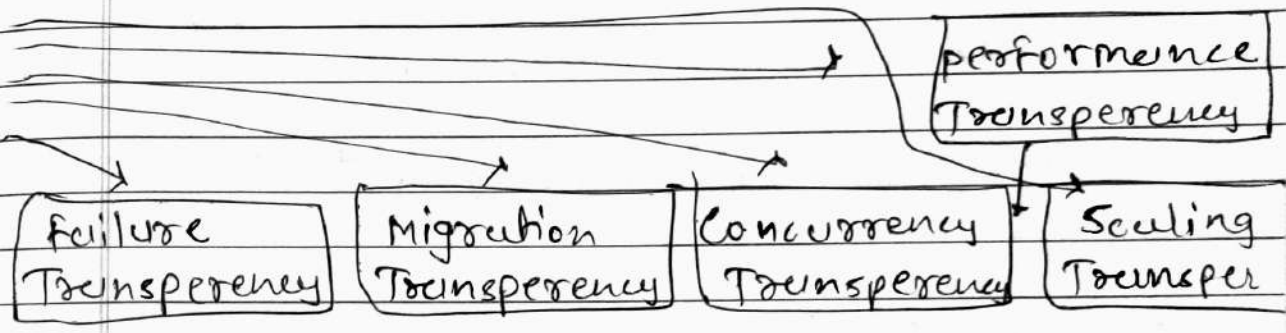


(1) Access Transparency:

- Local and remote objects should be accessed in a uniform way.
- User should not find any difference in accessing local or remote objects.
- Hide differences in data representation & resource access.
- Ex = Navigation in the web

[2] Location Transperency:

- objects are reffered by logical names which hide the physical location of the objects.
- Resource should be independent of the physical connectivity or topology of the system or the current location of the resources.



- Hide location of resources.

Ex perges in the web

[3] Replication Transperency:

- The provision of create replicas (additional copies) of files and other resources on different node of the distributed system.
- Hide the possibility that multiple copies of the resources exist.

- Replica of the files & data are transperent to the user.

[4] Failure Transperency:

- It deals with the masking from the users partial failures in the system, such as a common link failure, a machine failure, or a storage device crash.

- Hide failure & recovery of the resources.

Ex Database Management System.

[5] Migration Transperency:

- Resource object is to be moved from one place to another automatically by the system.

- Hide possibility that a system may change location of resource.

- Load balancing is one among many reason for migration of objects.

[6] Concurrency Transpency:

- Each user has the feeling that he or she is the sole user of the system & other user do not exists in the system.

- Hide the posibility that the resource may be shared concurrently.

Eg. Automatic teller machine network, DBMS.

Performance Transperency:

- It allow the system to be automatically reconfigured to improve performance, as load vary dynamically in the system.

- As far as practicable, a situation in which one processor of the system is overloaded with jobs while another process or is idle should not be allowed to occur.

Scaling Transperency:

- It allows the system to expand in scale without disrupting the activities of the users.

- Eg: world-wide-web.

* * *

Reliability :-

Distributed systems are expected to be more reliable than centralized systems due to the existence of multiple instances of resources.

- System failure are of two types:

(1) Fail-stop:

- The system stop functioning after detecting the failure.

~~(2) Byzant~~

(2) Byzantine failure:

- The system continues to function but gives wrong results.

- The fault-handling mechanism must be designed properly to avoid faults, to tolerate faults and to detect & recover from faults.

(*) Fault Avoidance:

- Fault avoidance deals with designing the components of the system in such a way that the occurrence of fault is minimized.

(*) Fault Tolerance:

- Redundancy Technique: To avoid single point of failure.

- Distributed control: To avoid simultaneous functioning of the servers.

⑥ Fault detection & Recovery:

- Atomatic translation
- stateless server.
- Acknowledgement and timeout-based re-transmissions of messages.

---

[3] Flexibility :-

- The design of Distributed operating system should be Flexible due to Following reasons:

(1) Ease of Modification:

- It should be easy to incorporate changes in the system in a user transperent manner or with minimum interruption caused to the users.

(2) Ease of Enhancement:

- New functionality should be added from time to time to make it more powerful and easy to use.

- A group of users should be able to add or change the services as per the Comfortability of thier use.

---

[4] Performance :-

- A performance should be better than or at least equal to that of running the same application on a single-processor system.

- Some design principles considered useful for better performance are as below:

(1) Batch if possible:

- Batching often helps in improving performance.

(2) Cache whenever possible:

- caching of data at clients side frequently improves over all system performance.

(3) Minimize copying of data:

- Data copying overhead involves a subsequential CPU cost of many operations.

(4) Minimize network traffic:

- It can be improved by reducing internode communication costs.

Page No.
Date

[5] Scalability :-

- Distributed system must be scaleable as the number of user increases.

"A system is said to be scaleable if it can handle the addition of users and resources without suffering a noticeable loss of performance or increase in administrative complexity."

- Scalability has 3 dimensions:

[1] size : Number of users and resources to be processed. problem associated is overloading.

[2] Geography : Distance between users and resources. problem associated is Communication reliability.

[3] Administration : As the size of distributed systems increase, many of the system needs to be controlled. problem associated is administrative mess.

- Guiding principles for designing scalable distributed systems :

- Avoid centralized entities.
- Avoid centralized algorithms.
- perform most operations on client workstations

[6] Heterogeneity :-

- This term means the diversity of the distributed system in terms of hardware, software, platform, etc.
- Modern distributed system will likely span different :
  - Hardware devices : computers, tablets, mobile phones etc.
  - operating System : Ms windows, Linux, Mac, Unix, etc.
  - Network : Local Network, wireless network, etc.
  - Programming languages : C, C++, Java, python, php, etc.
  - Different roles of software developers, designers, system managers.

---

[7] Security :-

- System must be protected against destruction and unauthorized access.
- Enforcement of security in a ds has the following additional requirements as compared to centralized systems:

- Sender of the message should know that message was received by the intended receiver.

- Receiver of the message should know that the message was sent by genuine sender.

- Both sender and receiver should be guaranteed that the content of message were not changed while it is in transfer.

---

④ Brief (Issues in Designing DS) :-

[1] Transperency : provide a single system image to its users.

[2] Reliability : Degree of fault tolerence should be low.

[3] Flexibility : Ease of modification and Enhancement.

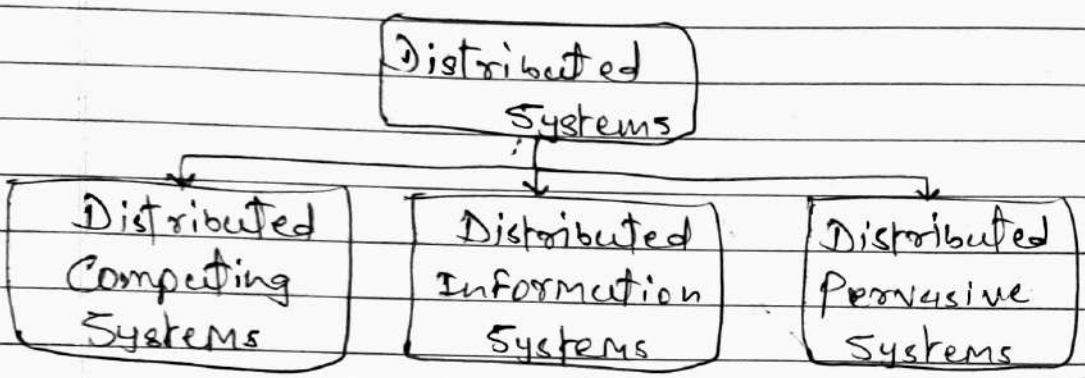
[4] Performance : Performance should be better than centralized system.

[5] Scalability : Capability of a system to adopt increased service load.

[6] Heterogeneity : It consits of dissimilar hardware or software systems.

[7] Security: Must be protected against destruction and unauthorized access.

Q.(a) Types of Distribution System.



[1] Distributed Computing Systems:

- An important class of distributed systems is the one used for higher performance computing tasks.
- Here, computers in a network communicate via message passing.
- Two Types of Distributed computing Systems:

(1) Clustering Computing Systems
(2) Grid computing systems

(1) Clustering Computing systems:

- A collection of similar workstations or PCs connected by a high-speed local-area network (LAN).

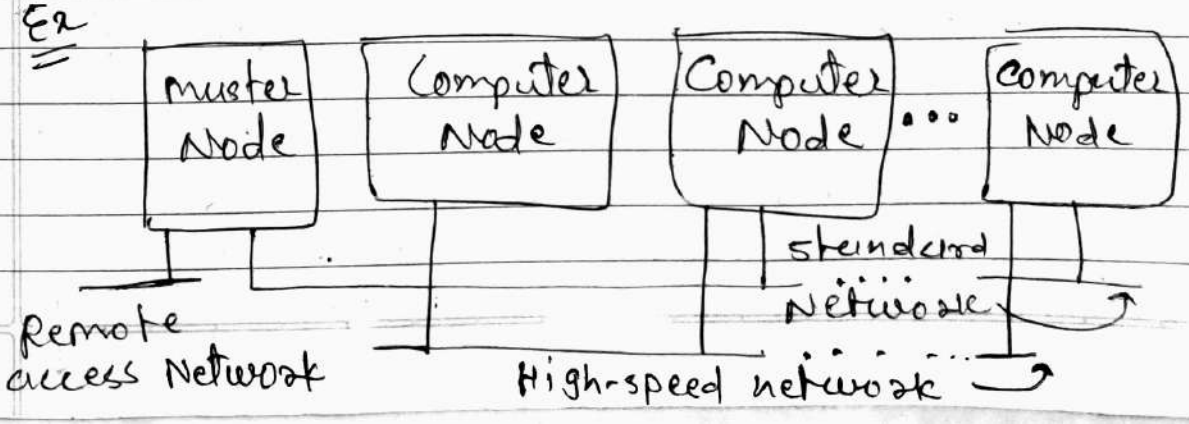
- It is homogenous given that each node runs the same OS.

- The ever increasing price / performance ration of computers makes it cheaper to build a supercomputer by putting together many simple computers, rather than buying high-performance one.

- Also, robustness is higher, maintenance and incremental addition of computing power is easier.

- Usage
  - parallel programming
  - Typically, a single computationally-intensive program is run in parallel on multiple machines.

Ex



(2) Grid Computing Systems:

- A characteristic feature of cluster computing is its homogeneity.

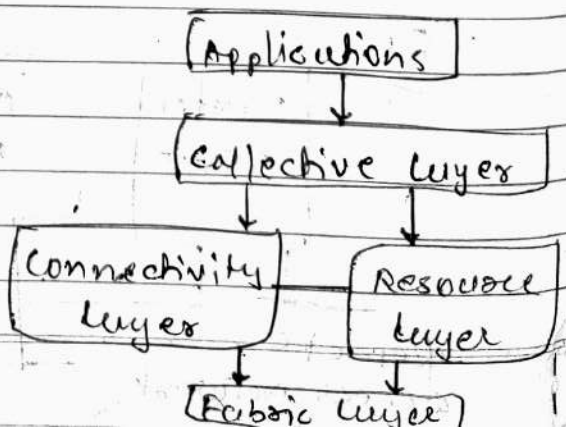
- In most cases, the computers in a cluster are largely the same, they all have the same operating system, and are all connected through the same Network.

- In contrast, grid computing systems have a high degree of heterogeneity: No assumptions are made concerning hardware, operating systems, networks, administrative domains, security policies, etc.

- A key issue in a grid computing system is that resources from different organizations are brought together to allow the collaboration of a group of people or institutions.

(*) Architecture of a Grid Computing System:

- Fabric layer
- Connectivity layer
- Resource layer
- Collective layer
- Application layer



Page No.
Date

⑧ Grid Middleware layer:

- provide access to the management of resources that are potentially dispersed across multiple sites.

- shift towards a service-oriented architecture in which sites offer access to the various layers through a collection of web services.

- Led to the definition of an alternative architecture known as the openGrid Services Architecture (OGSA).

- Consists of various layers & many components, making it rather complex.

---

[2] Distributed Information System:-

- Evolved in organizations that were ~~confirmed~~ confronted with a wealth of networked applications, but for which interp interoperability turned out to be problematic.

- Many of the existing middleware solutions are the result of working with an infrastructure in which it was easier to integrate applications into an enterprise-wide information system.

- Several levels at which integration took place:

  - Several non-interoperating servers shared by a number of clients:

    - distributed queries, distributed transactions.

    Ex: Transaction Processing System

  - Several sophisticated applications:

    - not only databases, but also processing components - requiring to directly communicate with each other.

    Ex: Enterprise Application Integration (EAI).

① Transaction Processing Systems:

- A transaction is a collection of operations on the state of an object (database, object composition, etc.) that satisfies the following properties: (ACID).

[Atomicity]:

- All operations either succeed, or all of them fail. When the transaction fails, the state of the object will remain unaffected by the transaction.

[Consistency]:

- A transaction establishes a valid state transaction.

[Isolation]:

- concurrent transactions do not interfere with each other. It appears to each transaction T that other transactions occur either before T, or after T, but never both.

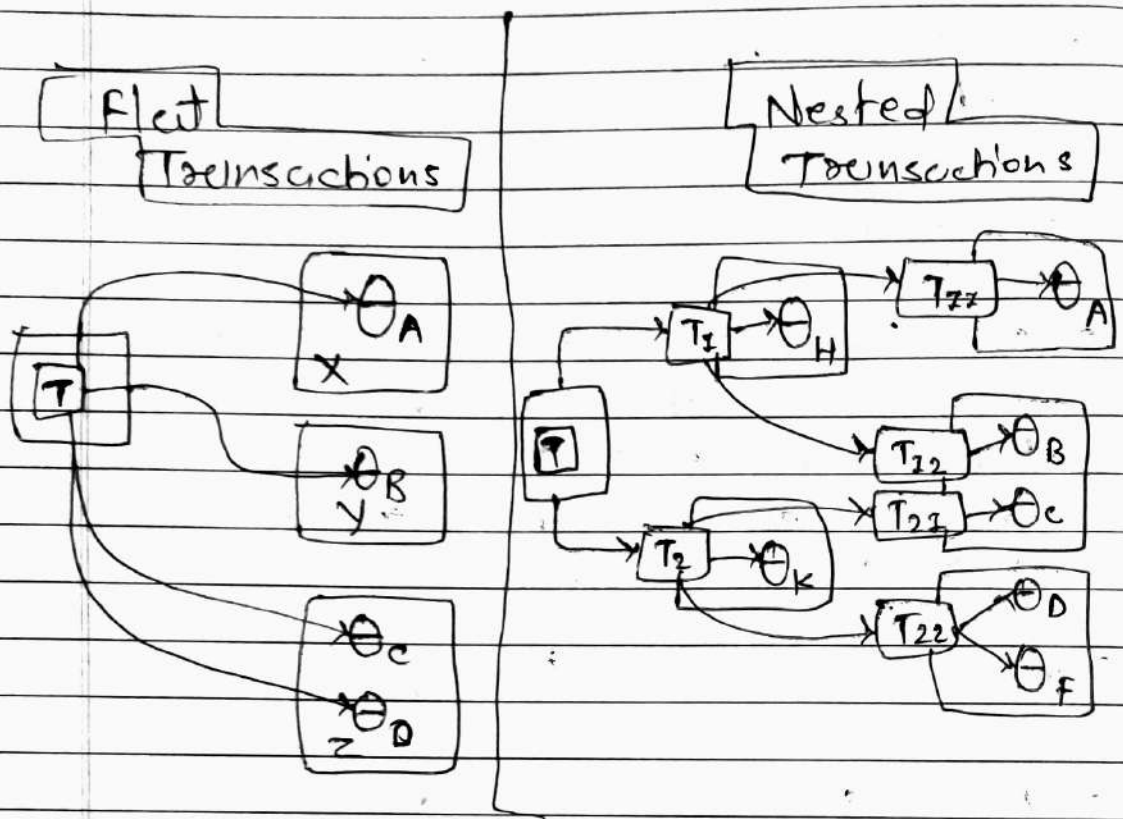
[Durability]:

- After the execution of a Transaction, its effects are made permanent.

Examples of primitives for Transactions.

primitive	Description
BEGIN_TRANSACTION	mark the start of a Tran.
END_TRANSACTION	terminate the transaction & try to commit
ABORT_TRANSACTION	kill the transaction & restore the old values
READ	read data from a File, a table, or otherwise
WRITE	write data to File, a table or otherwise

⊛ Distributed Transactions (Flat nested)



- A flat client transaction completes each of its requests before going on to the next one.
- Therefore, each transaction accesses a server's objects sequentially.

- In a nested transaction, the top-level transaction can open subtransactions, and each subtransaction can open further subtransactions down to any depth of nesting.

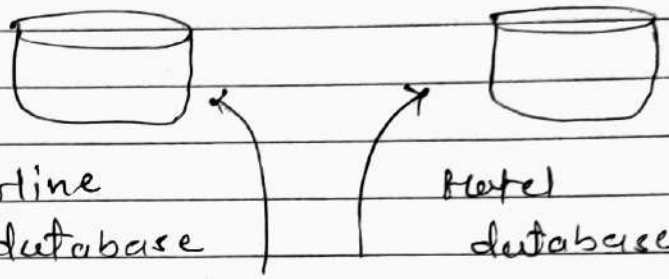
⊛ Nested Transactions :

- A Nested transaction is made of a number of subtransactions.

Page No.
Date

Nested transaction

Subtransaction Subtransaction



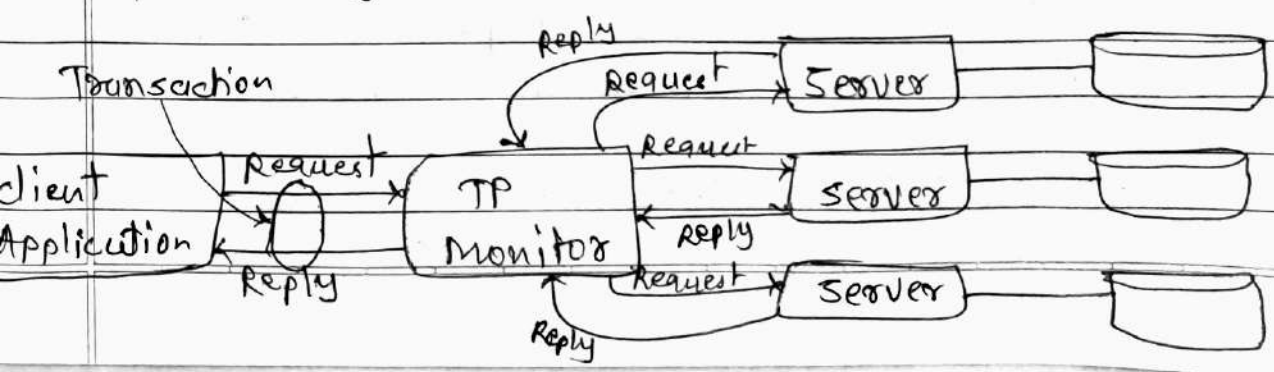
Two different databases (independent)

- The top-level transaction may fork off children that run in parallel with one another, on different machines, to gain performance or simply programming.

⑧ TP (Transaction Processing) Monitor:

- In the early days of enterprise middleware systems, the component that handled distributed (or nested) transactions formed the core for integrating applications at the server or database level.

- This component was called a Transaction processing monitor.



- Its main task was to allow an application to access multiple server/databases by offering it a transactional programming mode.

⊛ Popular TP monitor products:

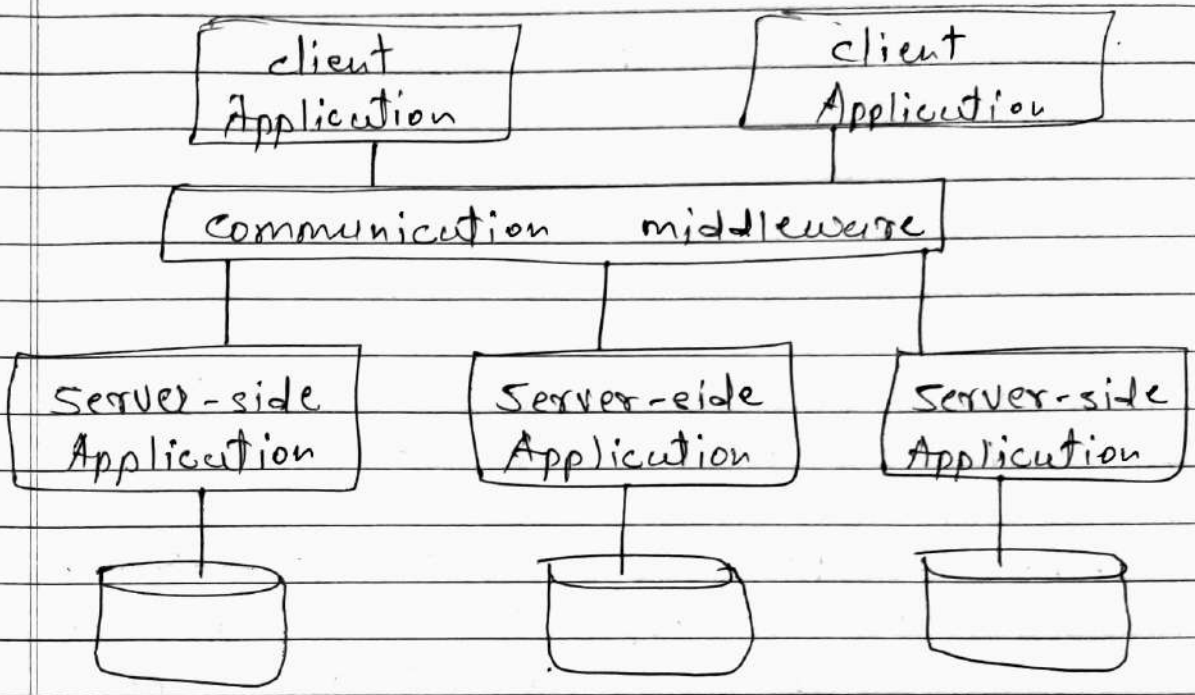
- TUXEDO from BAE systems
- Microsoft Transaction Server (MTS)
- IBM CICS (Customer Information control system).

⊛ Enterprice Application Integration:

- The more Applications become de-coupled from the databases they were built upon, the more evident it became the facilities were needed to integrate applications independent from thier databases.

- Application components should be able to communicate directly with each other and not merely by mean of the request/reply behaviour that was supported by transaction processing systems.

- Result: middleware as a communication facilator in enterprise application integration.



② Communicate Middleware

- Several types of communication middleware exist.

- Remote procedure calls (RPC):

  - an application component can send a request to another application component by doing a local procedure call, which results in the request being packaged as a message and sent to the callee.

- Remote method invocations (RMI):

  - An RMI is the same as an RPC, except that it operates on objects instead of applications.

- problems with RPC & RMI:

- The caller and callee both need to be up and running at the time of communication.

- They need to know exactly how to refer to each other.

- Results: message-oriented middleware (MOM) applications send msg to logical contact points, often described by means of a subject.

[3] Distributed Pervasive Systems:

- Above distributed systems characterized by their stability: nodes are fixed and have more or less permanent & high-quality connection to a network.

- Mobile & embedded computing devices: instability is the default behaviour. The devices in these, what we refer to as distributed pervasive systems.

- They are often characterized by being small, battery-powered, mobile,

and having only a wireless connection, although not all these characteristics apply to all devices.

⊛ Health Care Systems:

- Personal systems built around a Body Area Network.
- possibly minimizing impact on the person like, preventing free hand motion.

⊛ Features and Requirements of Distributed Pervasive Systems:

⊛ Feature:

- General lack of human administrative control.
- Devices can be configured by their owners
- They need to automatically discover their environment and fit in as best as possible.

⊛ Requirements for Pervasive systems applications:

- Embrace contextual changes
- Encourage ad hoc composition.
- Recognize sharing as the default.

Q. (11) Computer Network :-

- A computer network or data network is a telecommunications network which allows computers to exchange data.

- In computer networks, networked computing devices exchange data with each other using a data link.

- The connections between nodes are established using either cable media or wireless media.

- The best-known computer network is the Internet.

(*) Computer Network in DOS :-

- A distributed system is basically a Computer Network whose nodes have their own local memory and may also have other hardware & software resources.

- A distributed system relies on the Underlying computer Network for the communication of data & control the information between nodes.

- The performance & relibility of a Distributed system depends on the Underlying

(*) Classification of Computer Network:

- Network types depends on how large they are and how much of an area they cover geographically.

- Networks are classified based on

  - size
  - Capabilities
  - Geographical distance

[1] Local Area Network (LAN):

- A group of devices (computers, servers, ~~etc~~, switches and printers) that are located in the same building.

[2] Metropolitan Area Network (MAN):

- Larger than LAN.
- Spans over several buildings in a city or town.

[3] Wide Area Network (WAN):

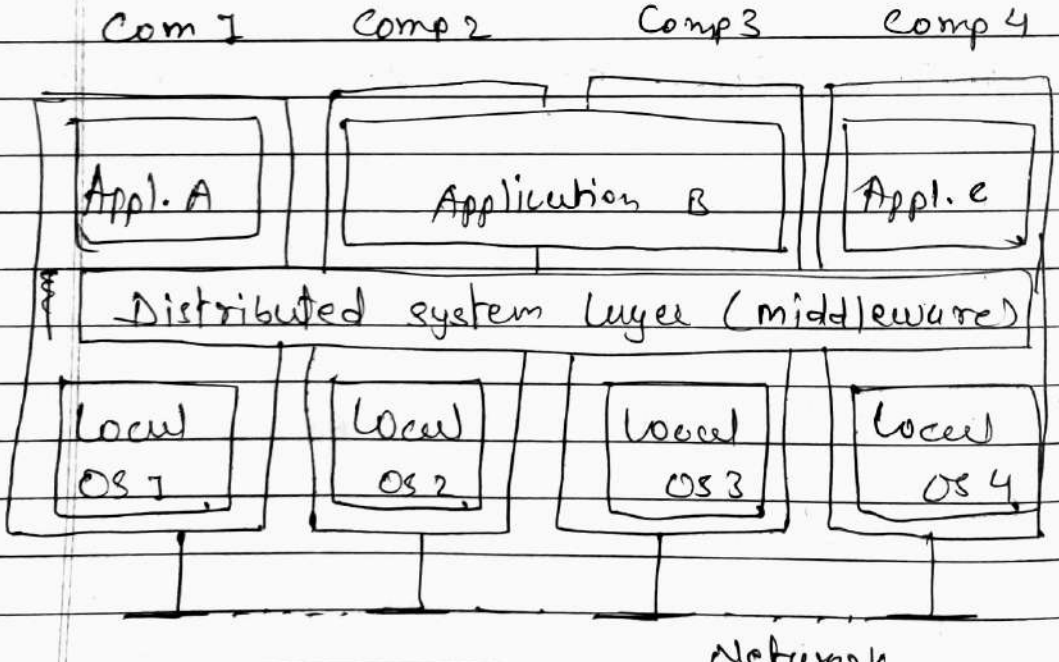
- Largest ~~Area~~ type of Network.
- Spans over large geographical area.
- the internet is example of WAN.

Q.(11) Wireless Network:

- A wireless LAN uses wireless transmission medium.
- wireless Network used to have
  - high prices
  - low data-rates
  - licensing requirements
- popularity of wireless LAN has grown rapidly.

* * *

Q.(12) Distributed Operating System Architecture:



- A distributed system organized as middleware.

- The middleware layer runs on all machines, and offers a uniform interface to the system.

- Middleware is software which lies between an operating system & the applications running on it.

⑥ Middleware (MW):

- Software that messages & supports the different components of a distributed system. In essence, it sits in the middle of the system.

- Essentially functioning as hidden translation layer, middleware enables communication & data management for distributed applications.

- It enables multiple systems to communicate with each other across different platforms.

- Examples:
  - Transaction processing monitors
  - Data converters
  - Communication controllers

Page No.
Date

Role of Middleware (MW):

- In some early systems:
  - Middleware tried to provide the illusion that a collection of seperate machines were a single computer.
- Today:
  - clustering software allows indep-endent computers to work together closely.
  - Middleware also supports seamless access to remote services, doesn't try to look like a general purpose OS.
- Other Middleware Examples:
  - CORBA (common Object Request Broker Architecture)
  - DCOM (Distributed Component object Model)
  - Sun's ONE RPC (Remote procedure call)
  - RMI (Remote Method Invocation)
  - SOAP (simple object Access Protocol)

Distributed System :-

Page No.
Date

Chapter : 7

Q.1 What is Distributed file system? Requirements of DFS. Features of DFS. Advantages & Disadvantages of DFS.

Requirements :-

(1) Transparency
(2) Concurrent file Updates
(3) File Replication
(4) Hardware and OS Heteroginity
(5) Fault Tolerance
(6) Consistency
(7) Security
(8) Efficiency

Distributed file system :-

- A distributed file system (DFS) is a file system with data stored on a server.
- The data is accessed and processed as if it was stored on the local client machine.
- The DFS makes it convenient to share information and files among users on a network in a controlled and authorized way.
- Servers allow the client users to share files and store data just like they are

Storing the information locally.

- Servers have full control over the data and give access control to the clients.

- DFS's primary goal is to enable users of physically distributed systems to share resources and information through the Common File System (CFS).

- it is a File system that runs as a part of operating systems.

- DFS has two components in its services.

  (1) Local Transperency
  (2) Redundancy

① Local Transperency :-

- it is achieved via the name space component.

② Redundancy :-

- it is achieved via the a file replication component.

- In the case of failure or heavy load, these components work together to increase data availability by allowing data from multiple places to be logically combined under a single folder known as

the "DFS root".

⑧ Features :-

(1) Transperency (Hiding):

↳ (1) structure Transperency
(2) Access Transperency
(3) Naming Transperency
(4) Replication Transperency

(2) User Mobility
(3) performance
(4) Simplicity and ease of use
(5) High availability
(6) Scalability
(7) High Reliability
(8) Data integrity
(9) Security
(10) Heteroginity

⑧ Working of Distributed File System :-

- there are two methods of DFS in which they might be implemented

(1) — Standalone DFS namespace
(2) Domain-based DFS namespace

⑧ Standalone DFS name space :-

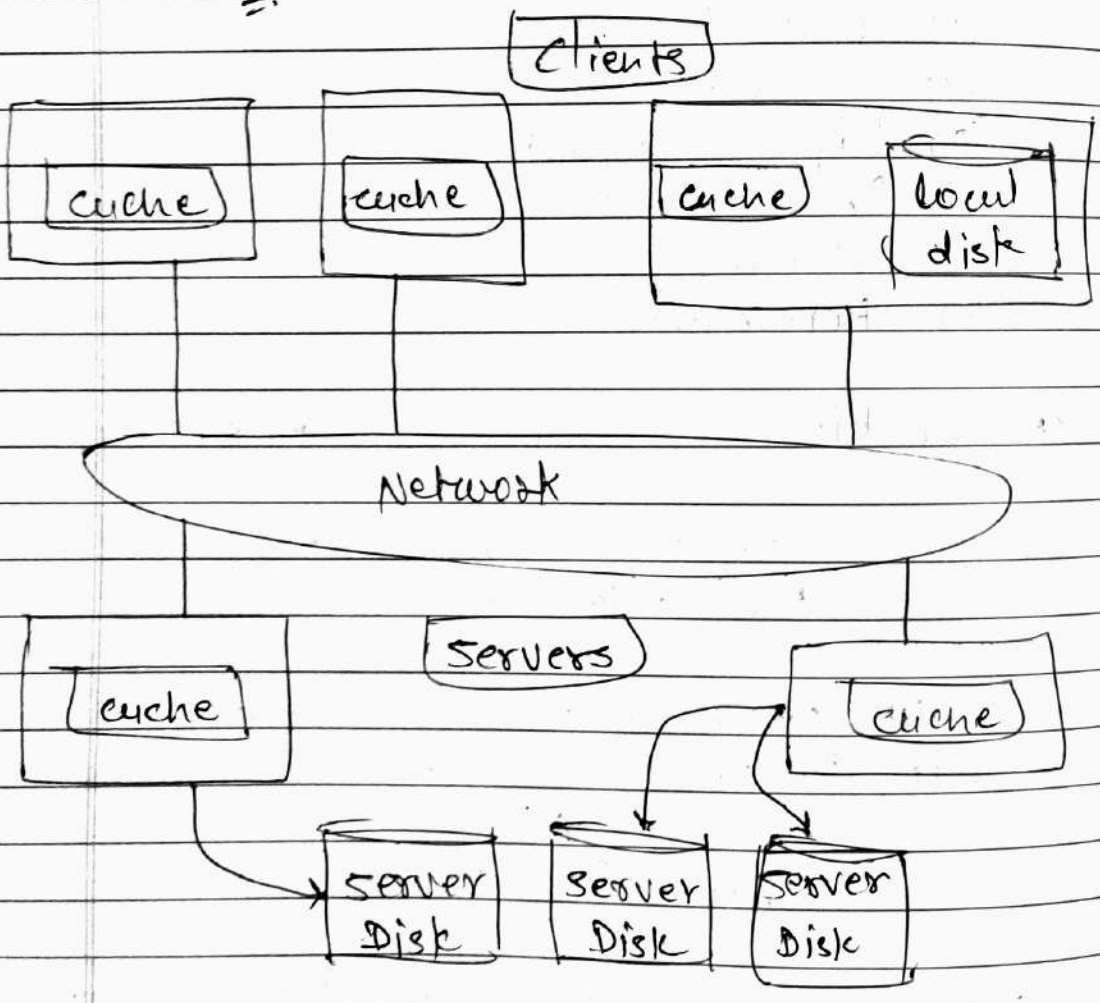
- it does not uses Active Directory and only permits DFS roots that exists on the Local system

- A standalone DFS may only be acquired on the systems that credited it.
- it offers no fault liberation and may not be linked to the other DFS.

⊛ Domain-based DFS name space :-

- it stores the DFS configuration in Active Directory and creating namespace for root at domainname > dfsroot> or FQDN > dfsroot>.

Ex. \\<DOMAIN.NAME>\<dfsroot>\<path>.



⊛ Distributed file system ⊛

Q.② What is Distributed web based System?

- The world wide web (www) can be viewed as a huge distributed system consisting of millions of clients and servers for accessing linked documents.

- Servers maintain collection of documents, while clients provide users an easy-to-use interface for presenting and accessing those documents.

① Architecture :-

- Many web-based systems are organized as simple client-server architecture.

- the simplest way to refer to a document is by means of a reference called a Uniform Resource Locator (URL).

- it specifies where a document is located, often by embedding the Domain name server (DNS) of its associated server.

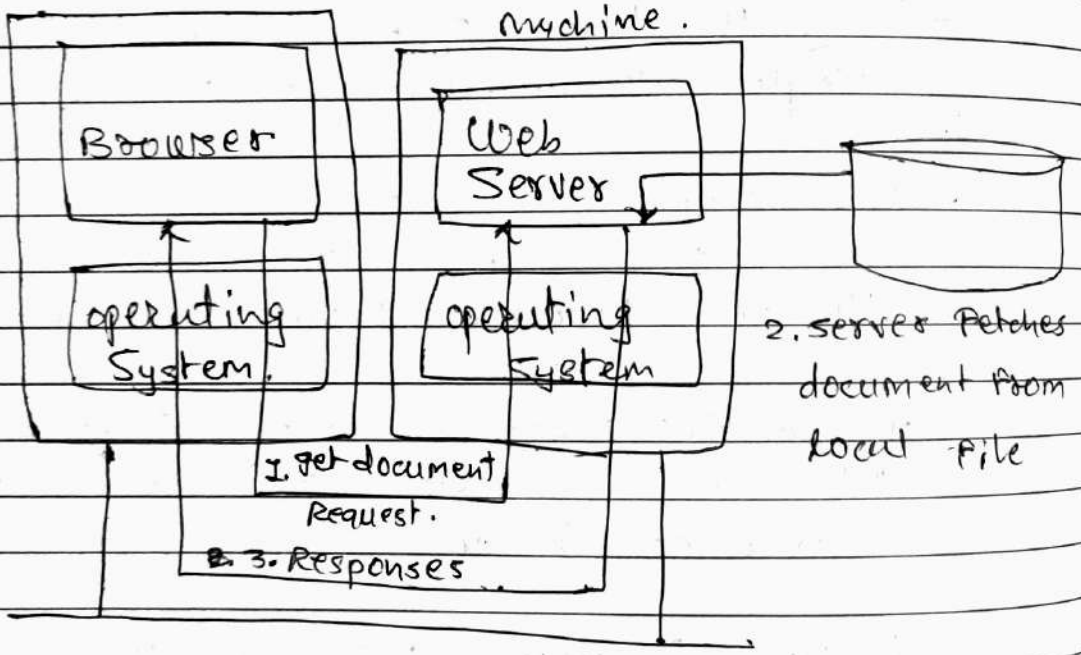
- A URL specifies the application-level protocol for transferring the document across the network.

- A client interacts with web servers through a special application known as a browser.

- A browser is responsible for properly displaying a document.

- A browser accepts input from a user by letting the user select a reference to another document, which it then subsequently fetches & displays.

- The communication between a browser and web server is standardized: they both adhere to the Hyper text Transfer protocol (HTTP).



① Traditional Architecture ①

② Web Documents :-

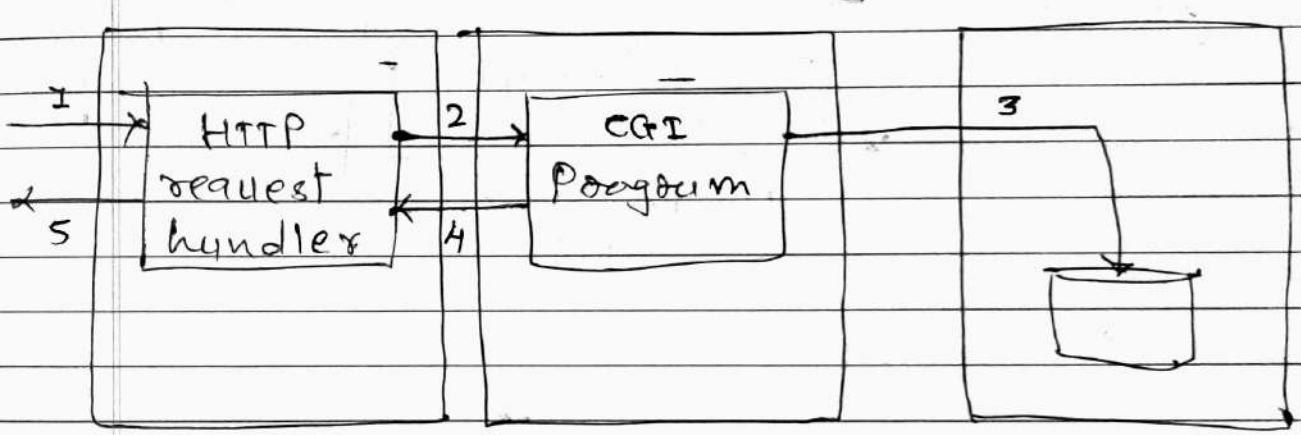
(1) main part act as a template
(2) second part consist of collection of pages / documents.

- main part of document written in HTML, XML.
- Each embedded document has an associated MIME (Multipurpose Internet Mail Exchange) type.

④ Multi-tire Architecture :-

- Common Gateway Interface (CGI) defines a standard way by which web server can execute a program taking user data as input.

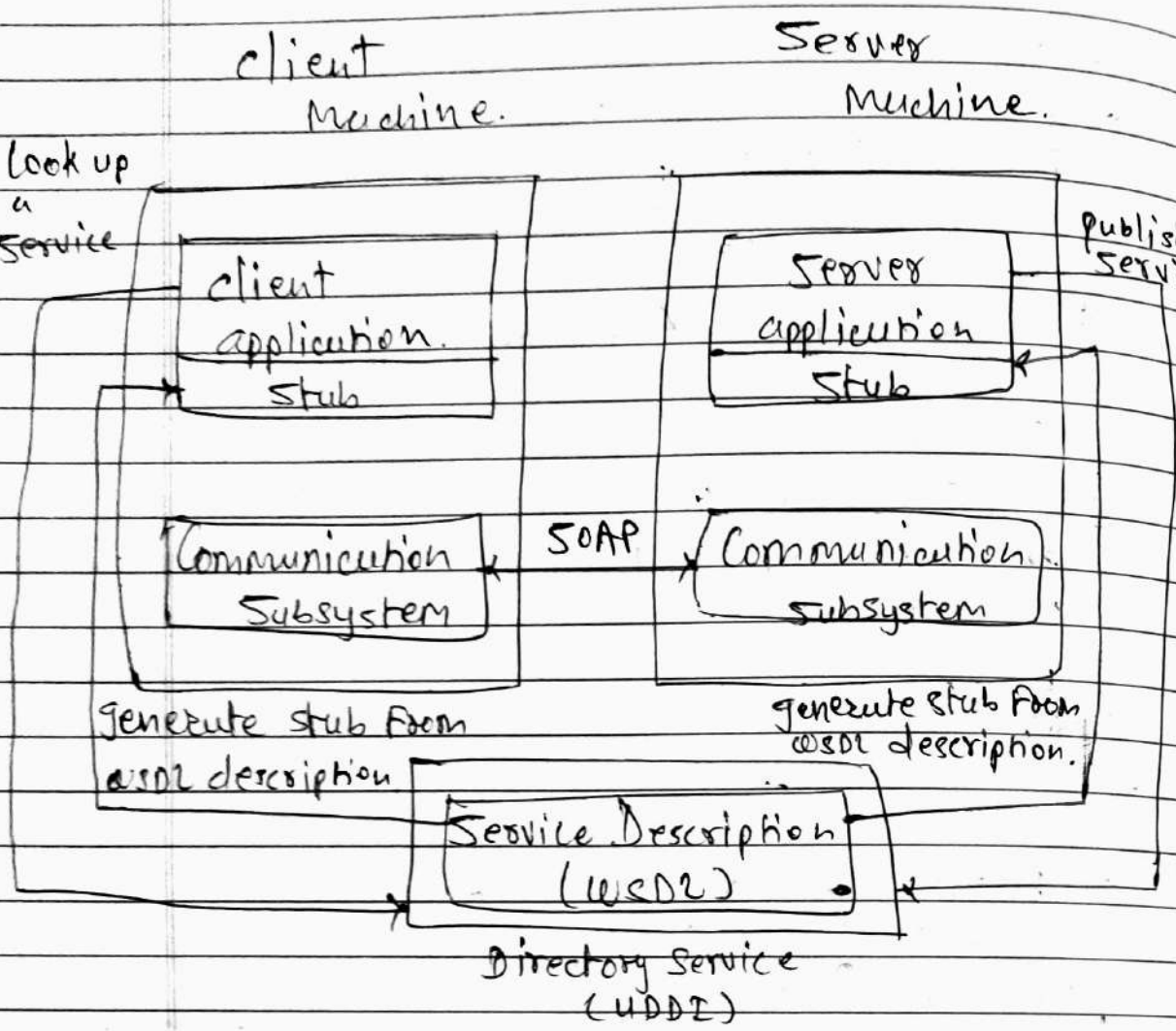
- User data come from an HTML forms.



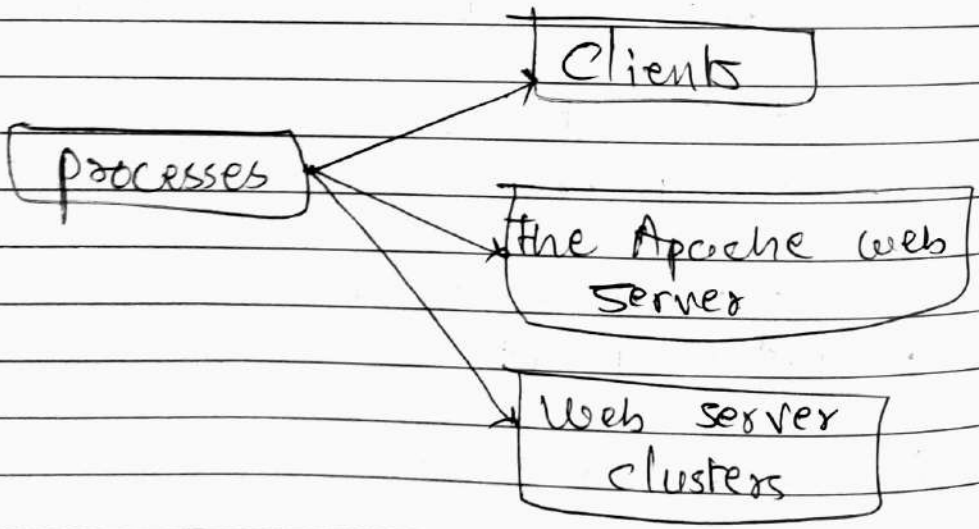
1. Get Request
2. Start Process to Fetch document
3. Database Interaction
4. HTML Document creator.
5. Return Result.

⑤ Web-Services :-

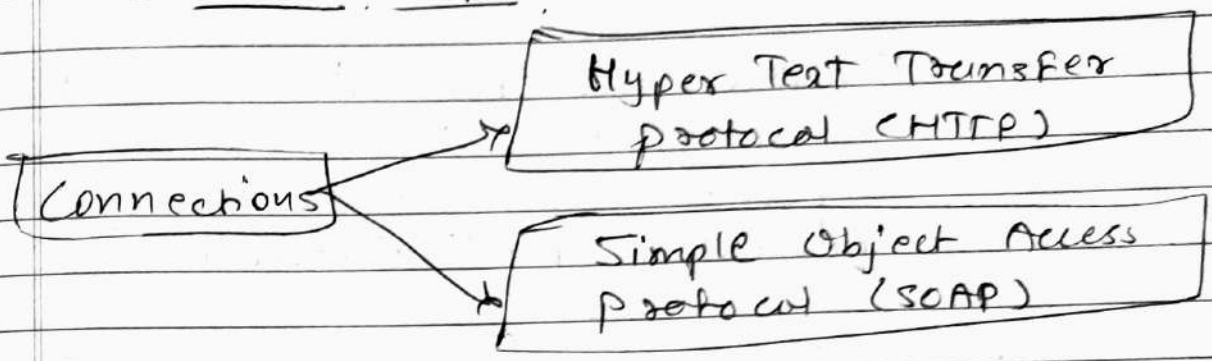
- SOAP, WSDL


client Machine.
Server Machine.
Look up a service
client application stub
Server application stub
publish service
Communication Subsystem
SOAP
Communication subsystem
Generate stub from WSDL description
Generate stub from WSDL description.
Service Description (WSDL)
Directory Service (UDDI)

④ Processes :-


Clients
processes
The Apache web server
Web server clusters

④ Connection Types :-



⑤ Domain Name System :-

- DNS stands for Domain Name system.
- DNS is a directory service that provides a mapping between the name of a host on the network and its numerical address.
- DNS is required for the functioning of the internet.
- Each node in tree has a domain name, and a full domain name is a sequence of symbols specified by dots.
- DNS is a service that translates the domain name into IP address.
  - This allows the users of networks to utilize user-friendly names when looking for other hosts instead of remembering IP addresses.

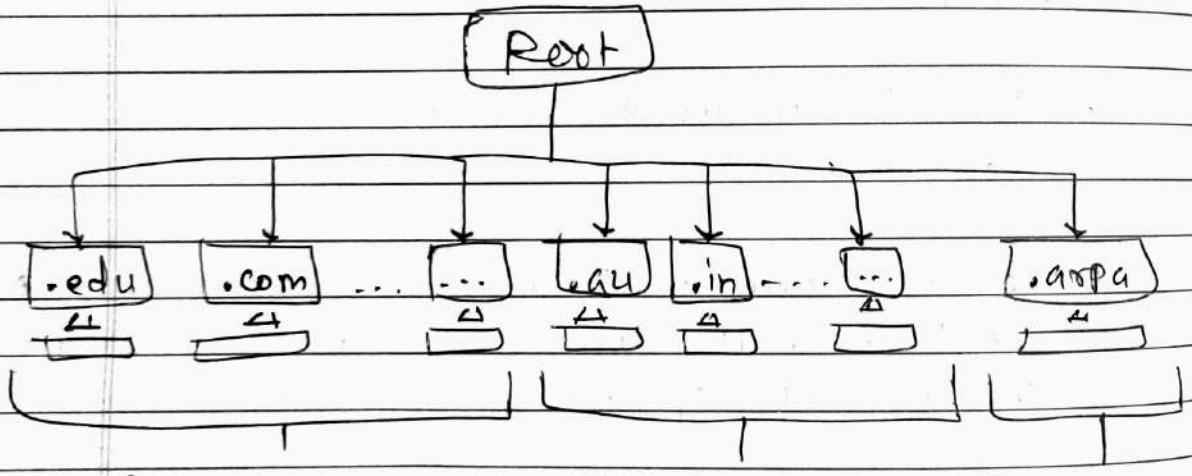
- For Example,

Suppose the FTP site EduSoft had an ip address 132.147.165.50, most people would reach this by specifying ftp.EduSoft.com.

- Therefore, the domain name is more reliable than IP address.

⊛ The domain name space is divided into three different sections:

(1) generic domains
(2) country domains
(3) inverse domains



Generic Domain
↓
defines the registered host according to their generic behaviour.
Ex. .com, .edu, .org, etc.

Country Domain
↓
Format of country name as in two characters.
Ex. .in, .au, .us, etc.

Inverse Domain
↓
used to mapping an address to a name.
Ex. .arpa.

Q. (4) Distributed Object-based System.

- In distributed object-based systems, the notation of an object plays a key role in establishing distribution transperency.

- In principle, everything is treated as an object and clients are offered services and resources in the form of objects that they can invoke.

- Few examples of object-based systems are CORBA., Java-based systems, & Globe.

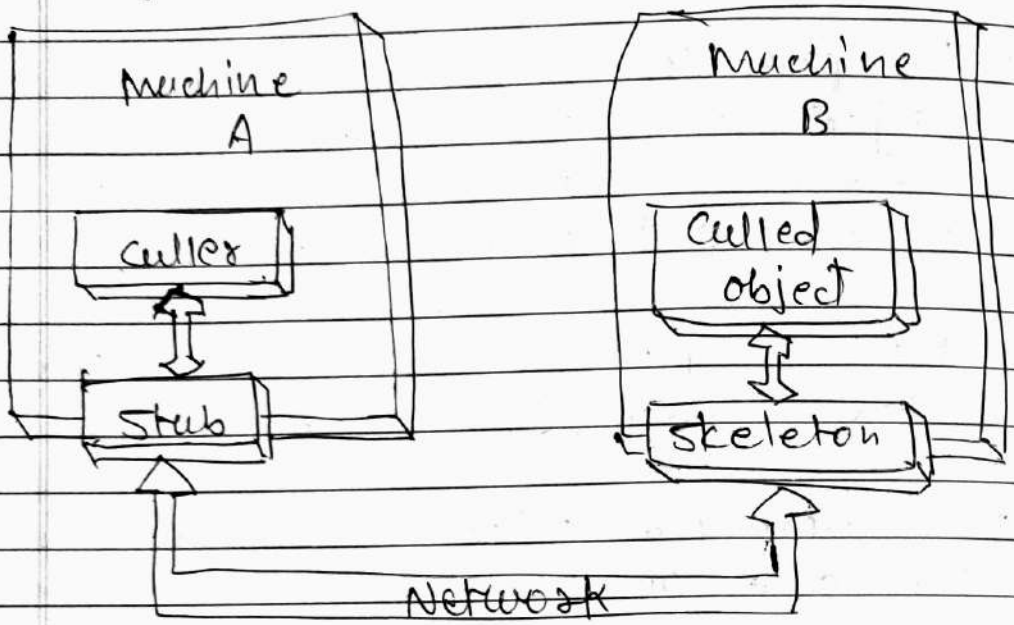
- A distributed object is one whose methods can be invoked by a remote process, a process running on a computer connected via a network to the computer on which the object exists.

- Distributed object is an object that can be accessed remotely. This means that a distributed objected can be used like a regular object, but from anywhere on the network.

- The location of ~~object-based~~ distributed object is not ~~crucial~~ critical to the user of the object.

key feature of an object: it encapsulates data, the state, and the operations on those data, the methods.

- methods are available through an interface. The seperation between interfaces and the objects implementing interfaces are crucial. For distributed systems.



- A stub is defined on client side. when a client binds to a distributed object, an implementation of the objects interface, called a proxy, is then loaded into the client's address space.

- then the stub passes caller data over the network to the server skeleton (machine B).

- the skeleton then passes received data to the called object. skeleton waits for a response and returns the result to the client stub (machine A).

- Client Stub responsible for:-

1. Initiate remote calls
2. Marshal arguments to be sent
3. Inform the remote reference layer to invoke the call.
4. Unmarshaling the return value
5. Inform remote reference the call is complete.

- Server skeleton responsible for:-

1. Unmarshaling incoming arguments from client.
2. Calling the actual remote object implementation.
3. Marshaling the return value for transport back to client.

Q.(5) What is CORBA? Explain briefly.

- Common Object Request Broker Architecture (CORBA) could be a specification of a regular design for middleware.

- it is a client-server software development model.

- Using a CORBA implementation, a shopper will transparently invoke a way on a server object, which may air a similar machine or across a network.

- The middleware takes the decision, associated is to blame for finding an object which will implement the request, passing it the parameters, invoking its methodology and returning the results of the invocation.

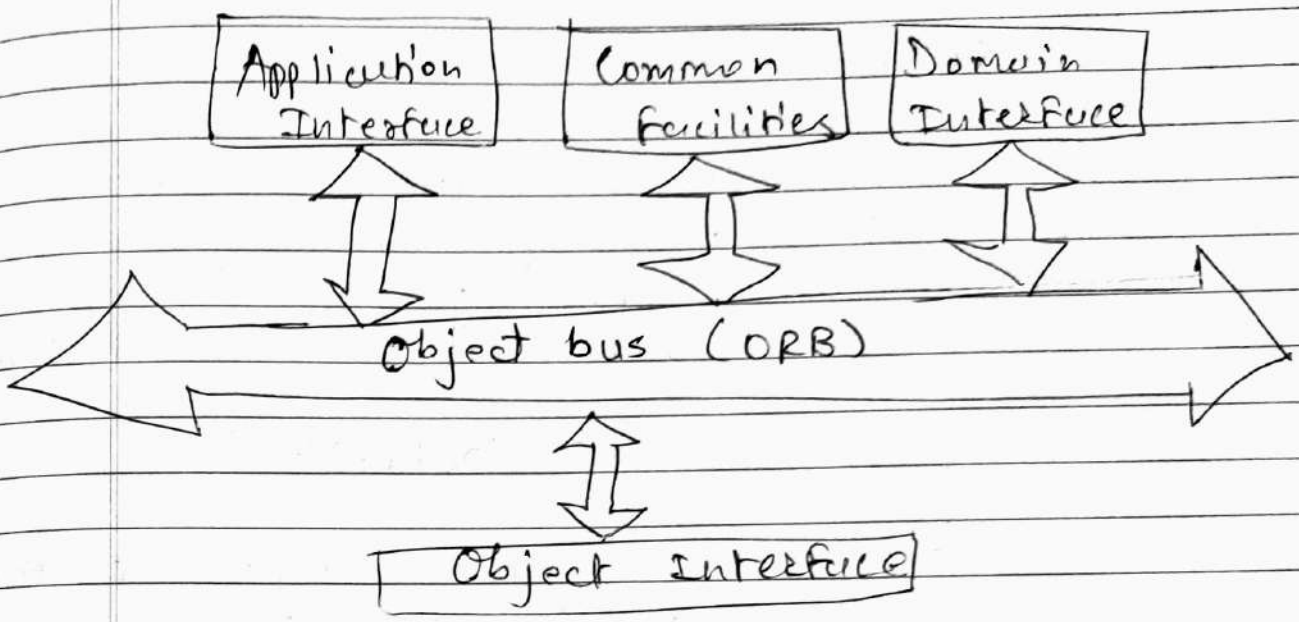
- The shopper doesn't need to remember of whenever the item is found, its programming language, its software package or the other aspects the don't seem to be a part of the associated objects interface.

⑧ CORBA Reference Model :-

- The CORBA reference model known as Object Management design (OMA) is shown in figure.

- The OMA is itself a specification (actually, a group of connected Specifications) that defines a broad vary of services for building distributed client-server applications.

- Several Services one may expect to search out in very middleware product like CORBA (like naming, dealings, & Asynchronous event management services) are literally fixed as services within the OMA.



⑤ Object Management Architecture (OMA) ⑤

- Different parts communicate victimization ORB.
- ORB is additionally referred as the item bus.
- An associate example of the application interface is a distributed document facility.
- In a very domain interface, it will have domain dependent services, for instance, producing domain.
- Object interface has some domain freelance services :

② CORBA :-

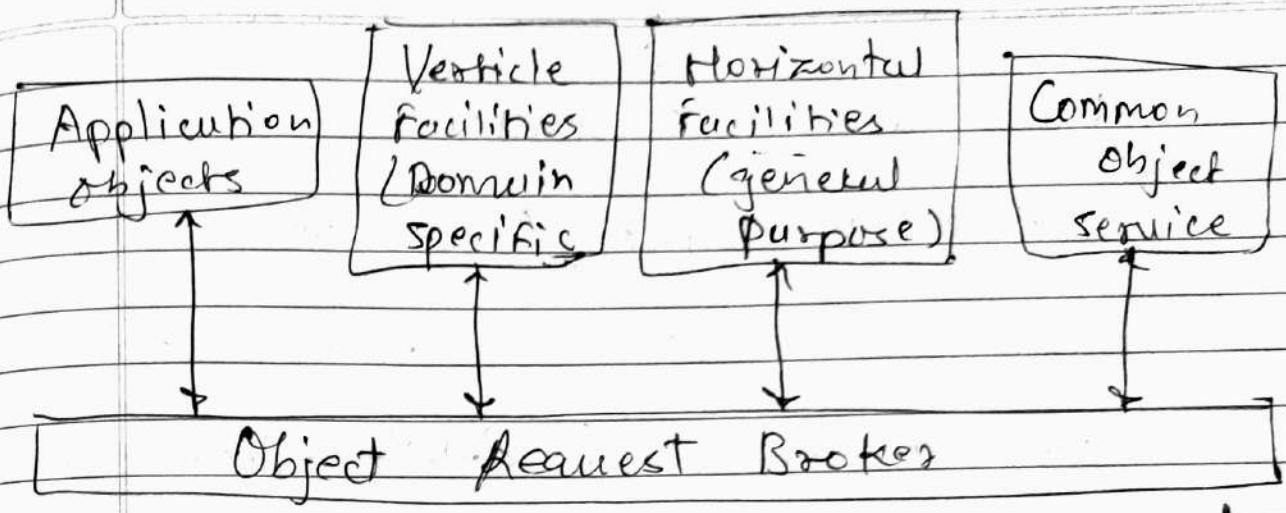
- CORBA (Common Object Request Broker Architecture) is a specification of an architecture supporting this (rather than an implementation such as RMI).

- ORB (Object Request Broker) :-
  - The runtime system responsible for communication between objects and their clients.

- CORBA facilities :-
  - Collection of classes and objects that provide general purpose capabilities that are useful in many applications.
  - Horizontal facilities: user interfaces, information management, system management, task management.
  - Vertical facilities: e.g. electronic commerce, banking, manufacturing, healthcare, telecommunication.

- ORB :-

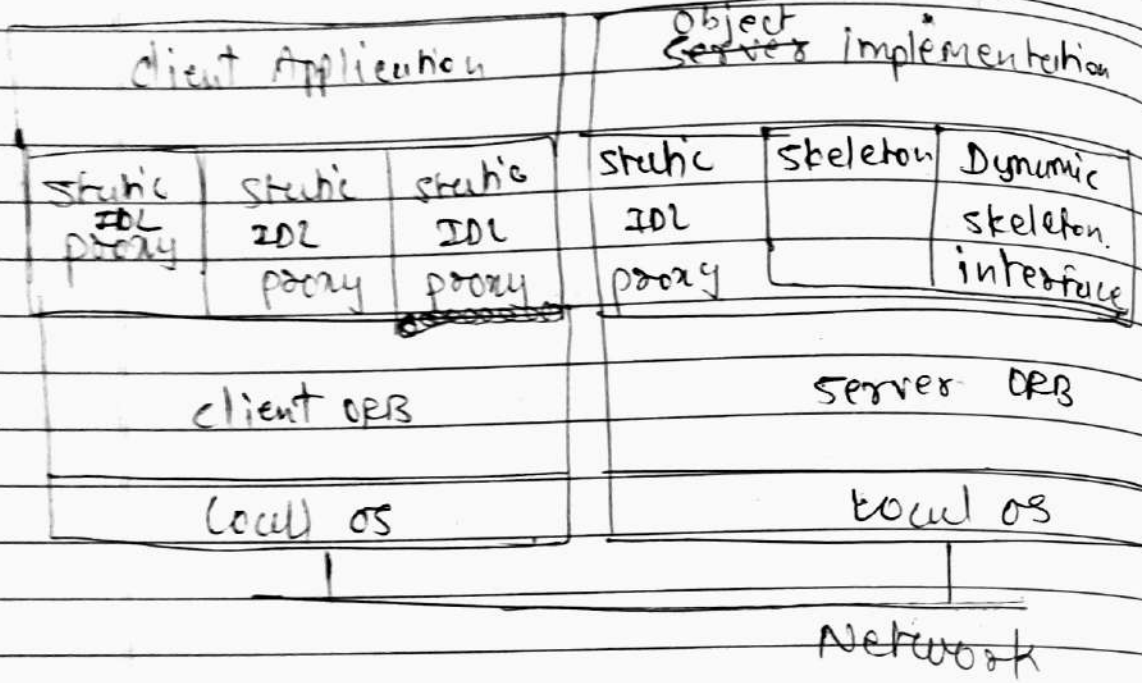
The runtime system responsible for communication between objects and their clients.



- An Interface Definition language (IDL) and its mapping onto the implementation language (c, c++, Java, .. etc.).
  - Specifies the syntax of objects and services
  - Cannot be used to describe semantics
  - Object can be implemented by a language without classes.
- An Interface Repository (IR) representing the interfaces of all objects available in the distributed system.
- A fully dynamic calling mechanism allowing:
  - run-time discovery of objects
  - discovery of available interfaces from an IR.
  - Sending of messages requests.
- Object Adapters: an abstraction mechanism for removing implementation details from

The message substrate.

Client & CORBA Organisation :-



ORB offers :-

- Basic communication
- Object references
- Initial finding of services

Interfaces :-

- Static (described in IDL)
- Run-time creation (Dynamic invocation interface

④ Chapter: ⑤

Q.① Explain why Replication is necessary. What is replication and consistency.

- Replication is necessary for:

OPS interface

- Improving performance: A client can access nearby replicated copies and save latency
- Increasing the availability of services: Replication can mask failures such as server crashes and network disconnection.
- Enhancing the scalability of systems: Repli Requests to data can be distributed across many servers which contain replicated copies of the data.

PASS
performance
Availability
Scalability
Security

- Securing against malicious attacks: Even if some replicas are malicious, security of data can be guaranteed by relying on replicated copies at non-compromised servers.

④ Data Replication :-

- Common Technique in distributed systems.

- Reliability - if one replica is unavailable or crashes, use another

  - protect against corrupted data

- performance - ~~Scale~~ scale with size of the distributed system (replicated web servers).

  - Scale in geographically distributed systems (web proxies)

- Key Issues:-

  - need to maintain consistency of data

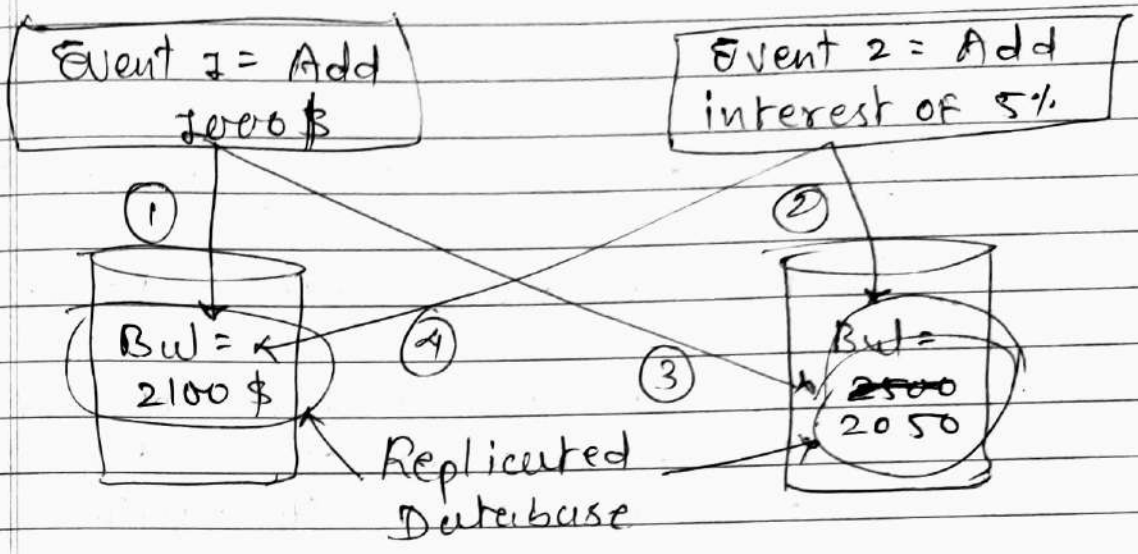
    1. if one copy is modified, others become inconsistent.

    2. When and how modifications need to be carried out, determines the price of replication??

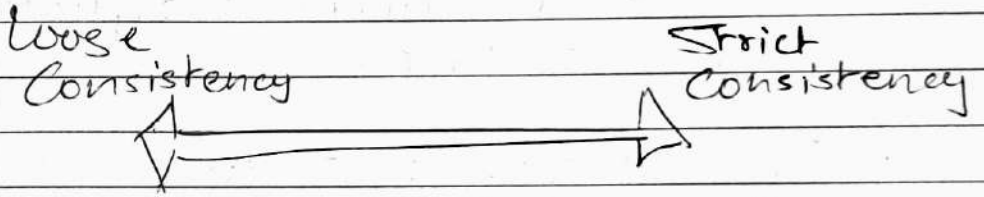
⊛ Consistency:-

- Server-side ~~consisten~~ replication comes with a cost, which is the necessity for maintaining consistency (or more precisely consistent ordering of updates)

Ex. A Bank Database



- maintaining consistency should balance between the strictness of consistency versus efficiency (or performance).



Easier to implement, and is efficient.

generally, hard to implement, and is inefficient.

⑤ Consistency Model :-

- A consistency model is contract between:
  - the process that wants to use the data
  - and the data-store
- A consistency model states the level (or degree) of consistency provided by the data-store to the process while

reading and writing data.

⑦ Types of consistency models :-

(1) Data-Centric Consistency Models:

- These models define how updates are propagated across the replicas to keep them consistent.

(2) Client-centric consistency models:

- These models assume that clients connect to different replicas at different times.

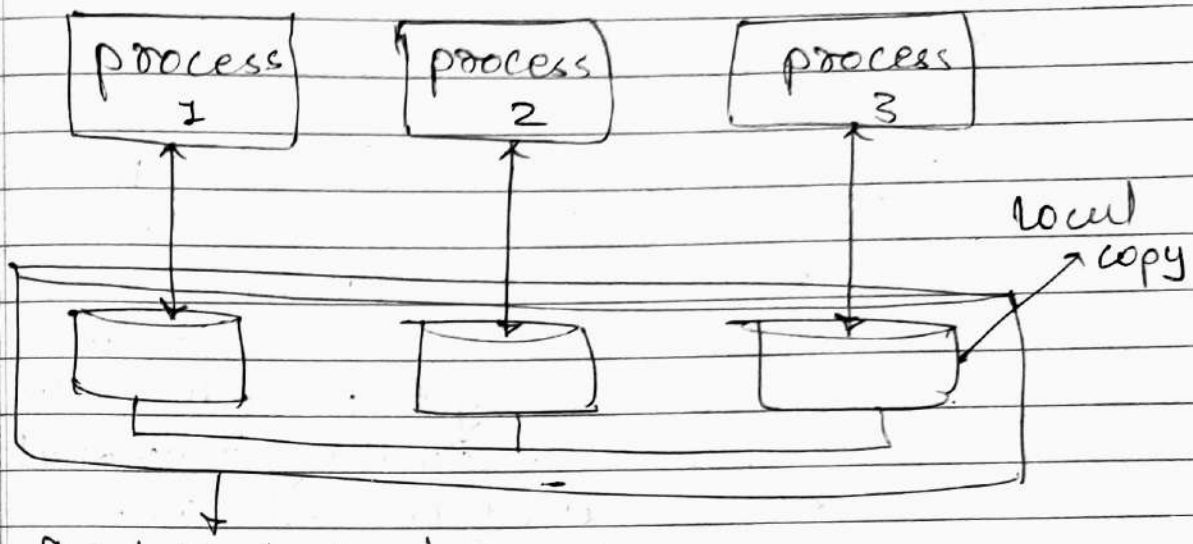
- They ensure that whenever a client connects to a replica, the replica is brought up to date with the replica that the client accessed memory.

⑧ Data-Centric Consistency Models :-

- A data-store can be read from or written to by any process in a distributed system.

- A local copy of the data-store (replica) can support "fast reads".

- However, a write to a local replica needs to be propagated to all remote replicas.



- The general organization of a logical data store, physically distributed and replicated across.

- Data-centric models are applicable when many processes are concurrently updating the data-store.

- Data-Centric Consistency model is too strict when,

  - one client process updates the data
  - other processes read the data and are ok with reasonably stable data.

⊛ Data-Centric Consistency models :-

(1) Strong Consistency Models : Operations on shared data are synchronized

(2) Weak Consistency Models : Synchronization occurs only when shared data is locked and unlocked.

① Strong Consistency models :-

- ~~Strict~~ Strong Consistency (related to time)
- Sequential Consistency (what we are used to)
- Causal Consistency (maintains only causal relations)
- FIFO Consistency (maintains only individual ordering) (or pipelined Random Access Memory Consi...)

② Weak Consistency models :-

- General Weak Consistency
- Release Consistency
- Entry Consistency

① Strong Consistency :-

- In a strongly consistent system, all nodes in the system agree on the order in which operations occurred.
- Reads will always return the most recent version of the data, and writes will be visible to all nodes immediately after they occur.
- These model provides the highest level of consistency.
- There are some performance and availability issues.

process	Write operation	Read operation
P1	Write(x)a	
P2		Read(x)a

⊛ FIFO consistency or pipelined Random Access Memory :-

- PRAM is one of the weak consistency models. Here, all processes view the operations of a single process in the same order that they were issued by that process, while operations issued by different processes can be viewed in a different order from different processes.

P1	P2	P3	P4
			⊗
Write(x)a			
	Read(X)a		
	Write(X)b		
			⊗
		Read(x)a	Read(x)b
		Read(x)b	Read(x)a

⊛ Sequential Consistency :-

- This model was proposed by Lamport.
- In this sequential consistency model, the result of any execution is the same as if the read and write operations by all processes were executed

in some sequential order and the operations of each individual process in this sequence in the order specified by its program.

- The sequential consistency model is weak as compared to the strict consistency.

process	write operation	Read operation
P1	write(x)a	
P2		Read(x)a
P3	write(x)b	

⊛ Causal Consistency model :-

- The causal consistency model was introduced by Hutto and Ahamad in 1990.

- It makes sure that all processes see causally-related shared accesses in the same order.

- The causal consistency model is weaker as compared to strict or sequential consistency model because the difference between causal consistency and sequential consistency is that causal consistency does not require a total order.

- All write operations that are potentially causally related are seen by all processes in the same (correct) order.

- Write operations that are not potentially causally related may be seen by different processes in different orders.

- if a write operation (w_2) is causally related to another write operation (w_1), the acceptable order is (w_1, w_2) because the value written by w_2 might have been influenced in some way by the value written by w_1.

- Therefore, (w_2, w_1) is not an acceptable order.

- Conversely, if two processes spontaneously and simultaneously write two different data items, there are not causally related.

- Operations that are not causally related are said to be concurrent.

P1	P2	P3	P4
Write(x) a	Read(x) a	Read(x) a	Read(x) a
	Write(x) b		
Write(x) c		Read(x) c	Read(x) b
		Read(x) b	Read(x) c

④ Weak Consistency Model :-

- A weakly consistent system provides no guarantees about the ordering of operations or the state of the data at any given time.

- Clients may see different versions of the data depending on which node they connect to.

- This model provides the highest availability and scalability but at the cost of consistency.

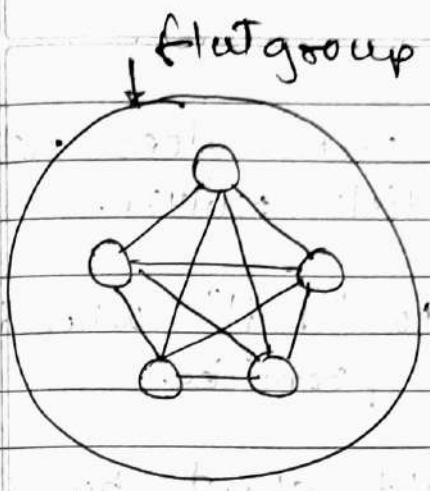
P1	P2	P3
Write(x) a		
Write(x) b		
	Read(x) a	Read(x) b
	Read(x) b	Read(x) a

Q.② ④ Process Resilience :-

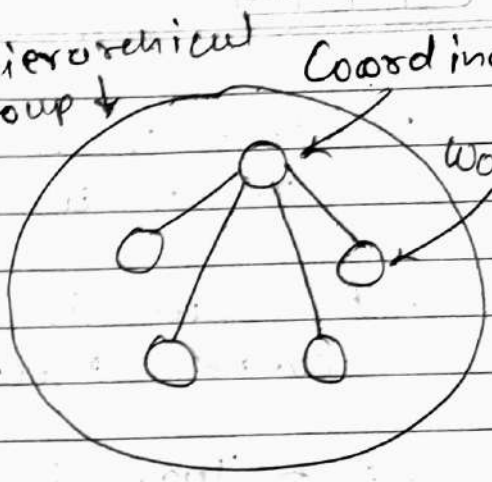
⑧ Process groups:

- protect yourself against faulty processes by replicating and distributing computations in a group.

(1). flat groups
(2). hierarchical groups



(a)



(b)

(a) Flat groups :-

- good for fault tolerance as information exchange immediately occurs with all group members.

- May impose more overhead as control is completely distributed (hard to implement).

(b) Hierarchical groups :-

- all communication through a single coordinator → not really fault tolerant and scalable, but relatively easy to implement.

Q.(8) Define failure ? list down various reasons for the occurrance of failure.

- A failure is a device from a specified behaviour.

Eg: Pressing brake pedal does not stop car
- brake failure. ~~(could be~~

Page No.
Date

Eg Read of a disk sector does not return content → disk failure.

- A system is said to "fail" when it cannot meet its promises.

- A failure is brought about by existence of "errors" in the system.

- The cause of an error is a "fault".

- Many failures are due to incorrect specified behaviour.

  - This typically happens when the designer misses addressing a scenario that makes the system perform incorrectly

  - it is especially true in complex systems with many subtle interactions.

④ Failure can happen due to variety of reasons:

(1) Hardware faults
(2) Software bugs
(3) Operator errors
(4) Network errors

Q.(4) What is distributed commit and recovery in distributed system ?

(*) Descributes commit:-

- Some applications perform operations on multiple databases.

Ex. Transfer funds between two bank accounts or debiting one account and crediting another.

- We would like a guarantee that either all the databases get updated or none does.

- Distributed Commit problem:

  - Operations is Commited when all participants can perform it. Once a commit decision is reached, this requirement holds even if some participants fail and later recover.

- Commit protocols are used to ensure atomicity across sites.

- Transaction which executes at multiple sites must either be commited at all the sites or aborted at all the sites. But it is not acceptable to have a transaction commited at one site and aborted at another.

- The two phase commit (2PC) protocal is widely used.

- The Three-phase commit (3PC) protocal is more complicated and more expensive, but avoids some drawbacks of two-phase commit protocal.

- Transaction behave as one operation:-

(1) Atomicity (all done, or no done).
(2) Consistency (no violation)
(3) Isolation (results are hidden)
(4) Durability (commited are permenant)

⊛ Recovery :-

- Recovery refers to restoring a system to its normal operational state.

- Once a failure has occured, it is essential that the process where the failure happened can recover to a correct state.

- Fundumental to fault tolerance is the recovery from an error.

- Resources are allocated to executing in a computer.

Eg. A processor has memory allocated to

it and a process may have locked shared resources, such as file and memory.

④ Following are some solution on process recovery :-

1. Reclaim resources allocated to process
2. Undo modification made to database and
3. Restart the process
4. Or: restart process from point of failure and resume execution.

Q.⑤ Two phase Commit protocol :-

- two phase commit is the standard protocol for making commit and abort atomic.

- it is designed to allow any participant to choose to abort a transaction. if one part of the transaction is aborted, then the whole transaction must also be aborted.

① protocol first phase :-

- every participants votes for the transactions to be commited or aborted.

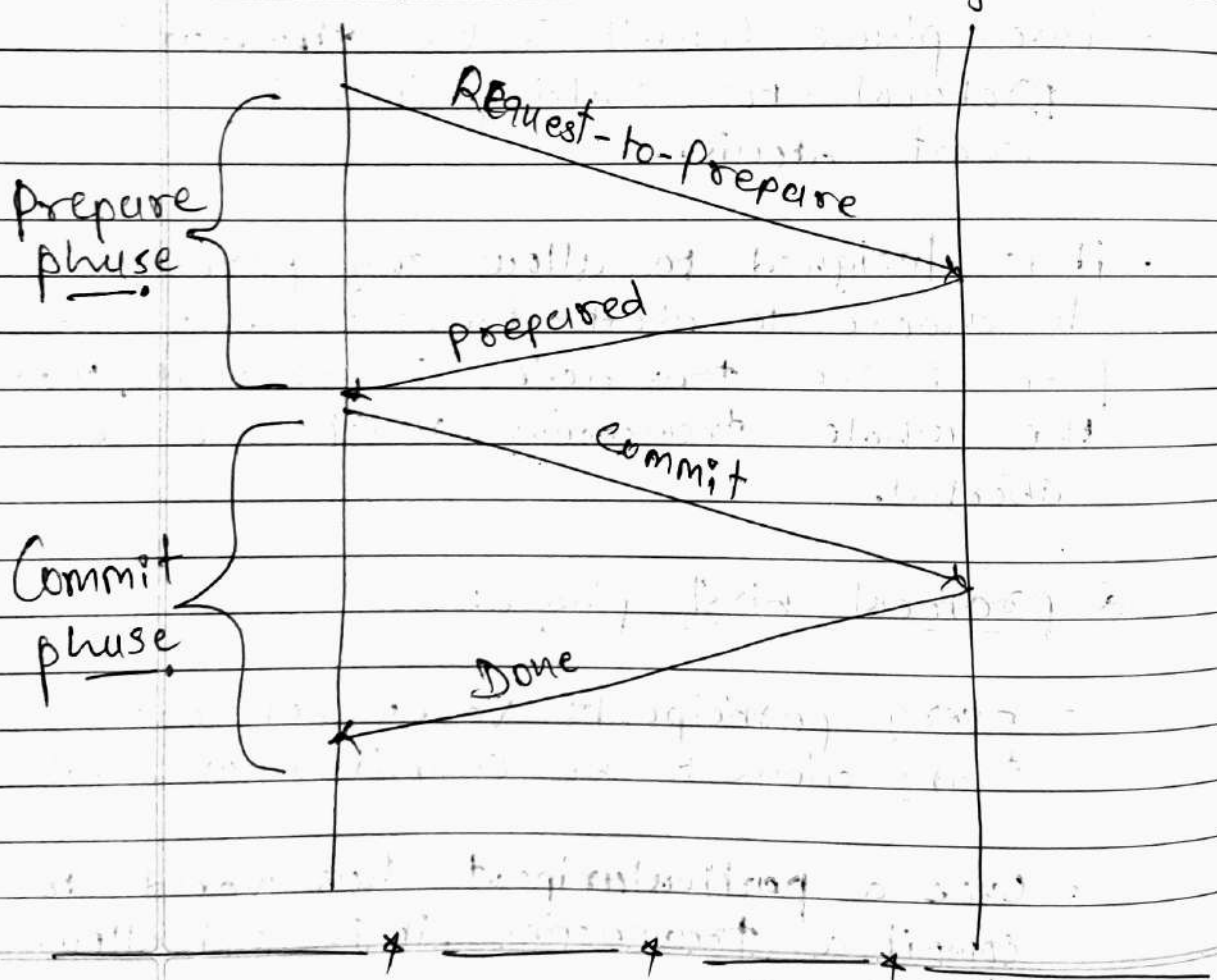
- Once a participant has voted to commit a transaction, it is not allowed to abort it.

④ protocol second phase :-

- Every participant in the transaction carries out the joint decision.

- If any one participants votes to abort then the decision must be to abort the transaction.

- if all the participants vote ~~for~~ to commit, then the decision is to commit the transaction.



④ chapter: ④

Q.① what is Logical clock? Explain how logical clocks are implemented in ds.

- A logical clock is a mechanism for capturing chronological and causal relationship in distributed system.

- Distributed systems may have no physically synchronous global clock, so a logical clocks allows global ordering on events from different processes in such systems.

- Suppose, we have more than 10 PCs in a distributed system and every PC is doing it's own work but then how make them work together.

- There comes the solution to this. i.e, LOGICAL CLOCK.

④ method: ①

- To Orders events across process, try to Sync clocks in one approach.

- This means that if one PC has a time 2:00 pm then every PC should have the same time which is quite not possible.

- Not every clock sync at one time. Then we can't follow this method.

④ Method: ②

- Another approach is to assign Timestamps to events.

- Taking the example into consideration, this means if we assign first place as 1, second place as 2, third place as 3 and so on.

- Then we always know that the first place will always come first and then so on. Similarly, if we give each PC their individual number then it will be organized in a way that 1st PC will complete its process first and then second and so on.

Q.② Types of clock Synchronisation :-

- (1) physical clock Synchronisation
  (2) Logical clock Synchronisation
  (3) Mutual Exclusion Synchronisation

④ physical clock Synchronization :-

- In a ~~rete~~ realm of distributed systems each ~~or~~ node operates with its clock, which can lead to time differences

- However the goal of physical clock synchronisation is to overcome this challange.

- This involves equipping each node with a clock that is adjusted to match Universal Coordinated Time (UTC) a recognised standard.

- By synchronizing thier clocks in this way diverse systems, across the distributed landscape can maintain harmony

⑧ Logical clock Synchronisation :-

- think of clocks as storytellers that prioritze the order of events than thier exact timing.

- these clocks enables the establishment of connections between events like weaving threads of cause and effect.

⑨ Mutual Exclusion Synchronisation :-

- Imagine multiple clock processes competing for access to the resource simultaneously.

- To address this issue mutual exclusion synchronisation comes into play as an expert technique that reduces chaos and promotes resource ~~memory~~ harmony.

Q.3 Mutual Exclusion & Types of mutual Exclusion algorithms.

- The design of distributed mutual Exclusion algorithms is complex because these algorithms have to deal with unpredictable message delays and incomplete knowledge of the system state on eliminating the mutual Exclusion problem in distributed system approach based on Message passing is used.

- Below are the three approaches based on messaging passing to implement mutual Exclusion in distributed system.

(1) Token based approach
(2) Non-Token based approaches
(3) Quorum-based approach.

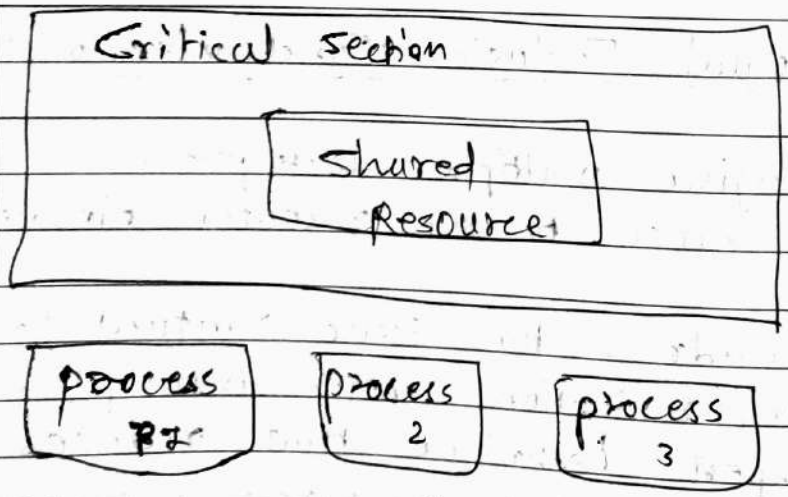


Fig: mutual Exclusion

④ Token Based Algorithm :-

- A token is shared among all the sites.
- if a site posseses the unique token to, it is allowed to enter its criticial section.
- This approach uses sequence number to order request for the criticul section.
- Each request for criticul section contains a sequence Number. this sequence number is used to distinuguish betwen old and current requests.
- This approach ensures mutuid exclusion as the token is uniaue.

⑤ Non-Token based approach :-

- A site communicates with each other sites in order to determine which site is should execute critiricil section next.
- This reauires ~~knowledge~~ exchange of two or more succesive round of messages among sites.
- This approach uses timestamps instead of sequence Number to order request for the criticul section.

- Whenever a site make request for critical section, it gets a timestamp. Timestamp is also used to resolve any conflict between critical section requests.

- All algorithm which follows non-token based approach maintains a logical clock. Logical clocks get updated according to Lamport's scheme.

③ Quorum-based approach :-

- Instead of requesting permission to execute the critical section from all other sites, each site requests only a subset of sites which is called a "Quorum".

- Any two subsets of sites or Quorum contains a common site.

- This common site is responsible ~~for~~ to ensure mutual exclusion.

④ Vector clock Timestamp with example.

- Vector clock is also called a logical clock, which is used to assign timestamps for events in a distributed system.

- Vector clock also gives a partial ordering of the events.

- In this, we use a vector of integer values to represent the timestamp.

- If we have N processes in the group, then each process will have a vector with N elements.

Vector-clock Algorithm :-

- Initially, all the clocks are set to zero.

- Every time, an Internal event occurs in a process, the value of the processes' logical clock in the vector is incremented by 1.

Two Rules :-

(1): (Logical Update rule):-

- Every time a process sends a message, the value of the processes' logical clock in the vector is incremented by 1 and then send a copy of its own vector.

(2): (message Rule):-

- Everytime, a process receives message, the value of the processes' logical clock in the vector is incremented by 1.

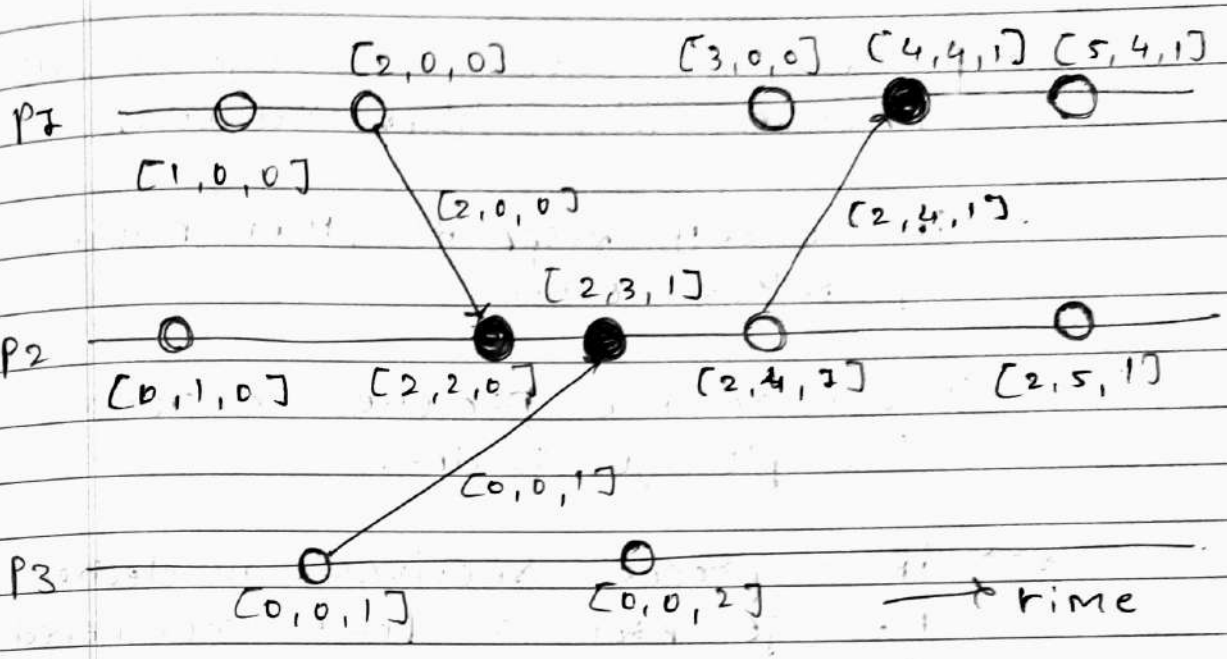
- Moreover, each element is updated by taking the maximum of the value in its own vector clock and the value in the vector in the received message.

- As a result, when a process receives a message, it merges its own time vector and the time stamp sent with the message - it selects the higher of the values for each element.

- This ensures that the sender has information that is at least updated as the receiver.

Ex: Consider a process (P) with a vector size N. For each process: the above set of rules mentioned are to be executed by the vector clock.

- The example depicts the vector clocks mechanism in which the vector clocks are updated after execution of internal events, the arrows indicate how the values of vector are sent between the processes (P_1, P_2, P_3).



○ = event executing Rule 1
● = event executing Rule 2.

Q.⑤ Election Algorithms

(1) Bully Algorithm
(2) Ring Algorithm

⊛ Bully Algorithm :-

- Bully algorithm specifies the process with highest identifier will be the coordinator of the group.

⊛ Working :-

- When a process P detects that the coordinator is not responding to requests, it initiates an election:

- P sends an election message to all processes with higher numbers.

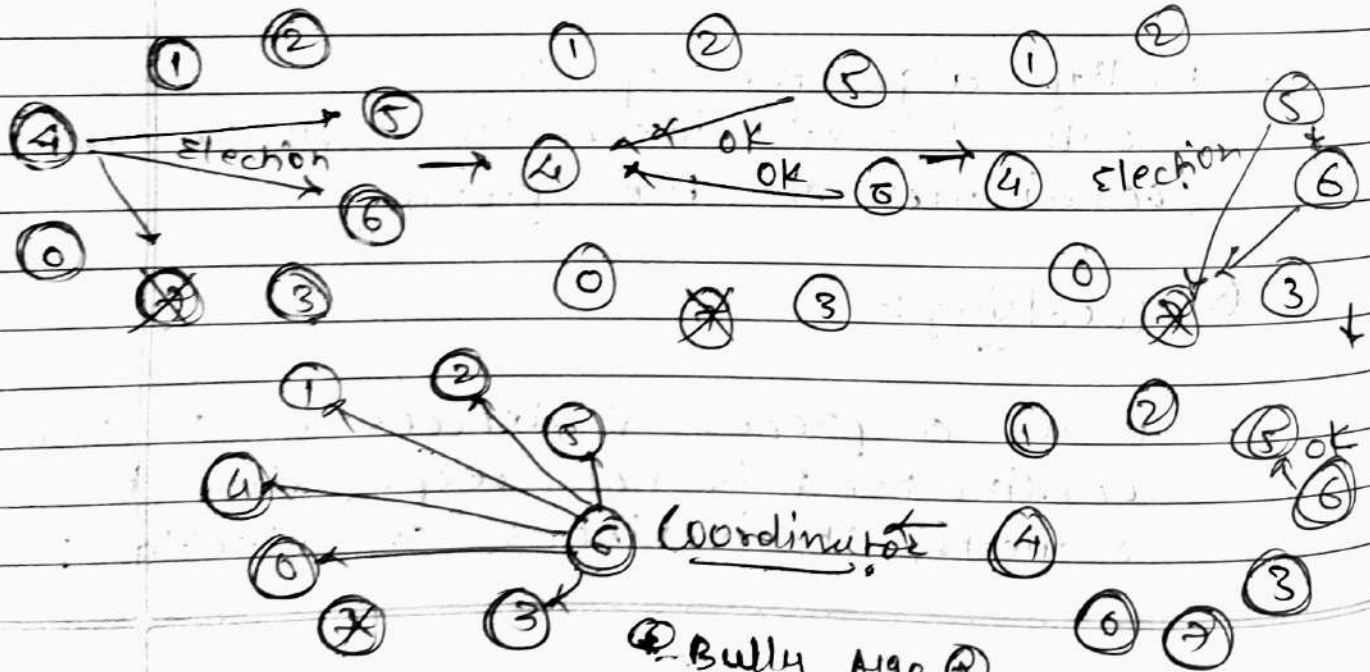
- if nobody responds, then P wins & takes over.

- if one of the processes answers, the P's job is done.

* → if a process receives an election message from a lower numbered process at any time, it:

  - sends an OK message back.
  - holds an election (unless its already holding one).

→ A process announces its victory by sending all processes a message telling them that it is the new Co-ordinator.



Bully Algo

④ Ring Algorithm :-

- The ring algorithm assumes that the processes are arranged in a logical ring and each process is knowing the order of the ring of processes.

- if any process detects failure, it constructs an election message with its process ID and send it to its neighbour.

- if the neighbor is down, the process skips over it and sends the message to the next process in the ring.

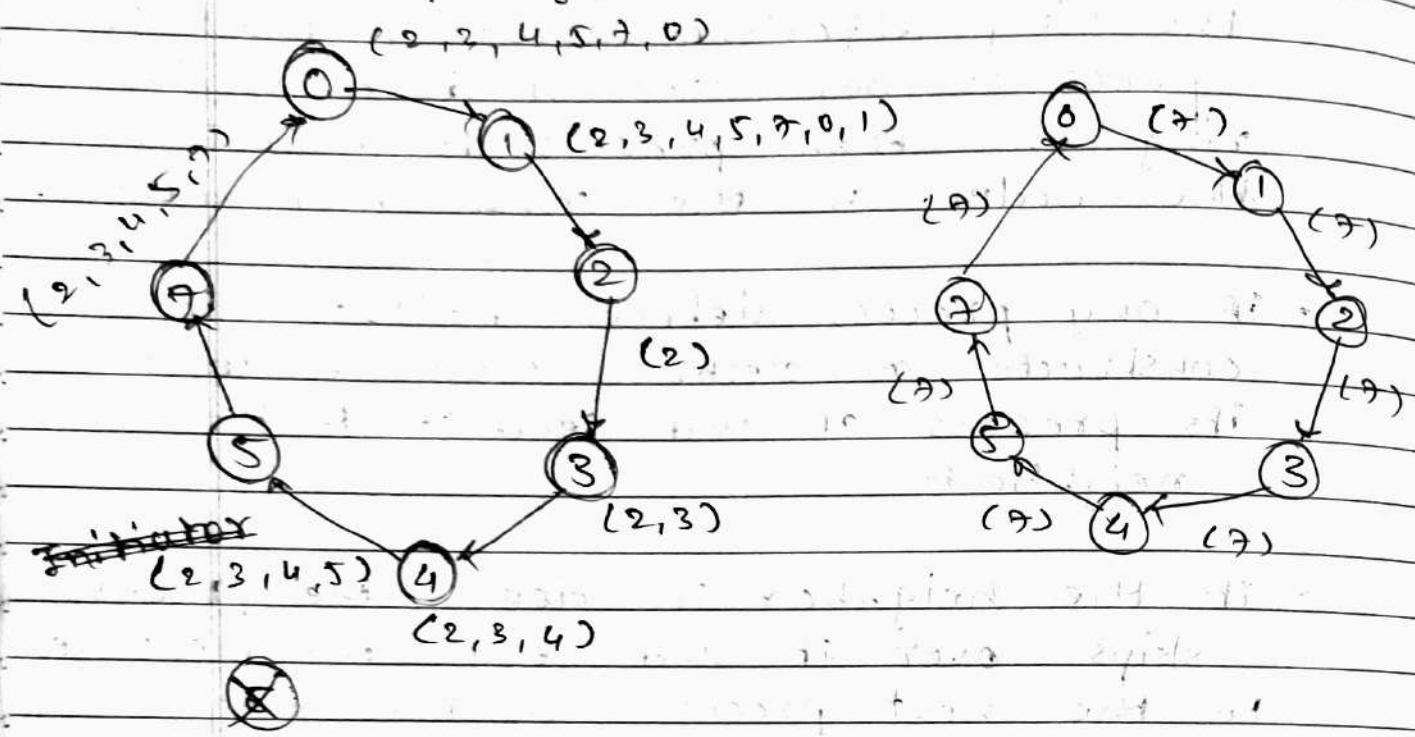
- This process is repeated until a running process is lowered.

- At each step, the process adds its own process ID to the list in the message and sends the message to its living neighbour.

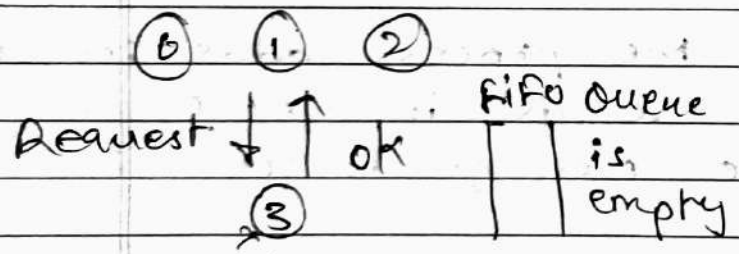
- Eventually, the election message come back to the process that started it.

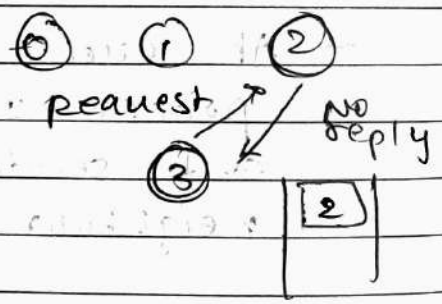
- The process then picks either the highest or lowest process ID in the list and sends out a message to the group informing them of new coordinator.

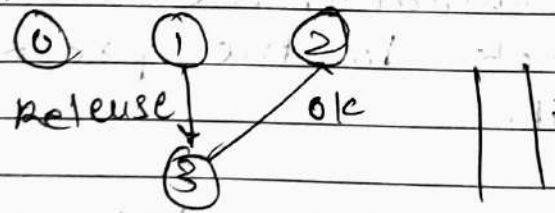
- this algorithm is generally used in ring topology.



6. Centralized Algorithm.







Detailed in ppt pg.no → 49 → 52.

Q.② Berkeley Algorithm :-

- A single server can fail, blocking time keeping.
- The Berkeley algorithm is a distributed algorithm for timekeeping.
- Assumes all machines have equally-accurate local clocks.
- Obtain average from participating computers and synchronizes clocks to that average.
- time serve periodically send a a message ("time = ?") to all computers in the group.
- Each computer in the group sends its clock value to the server.
- Server has prior knowledge of propagation time from node to server.
- Time server readjusts the clock values of the reply message using propergation time and then takes fault tolerent average.
- The time server adjust its own time and sends the adjustment (positive or negative)

to each node.

④ Approach :-

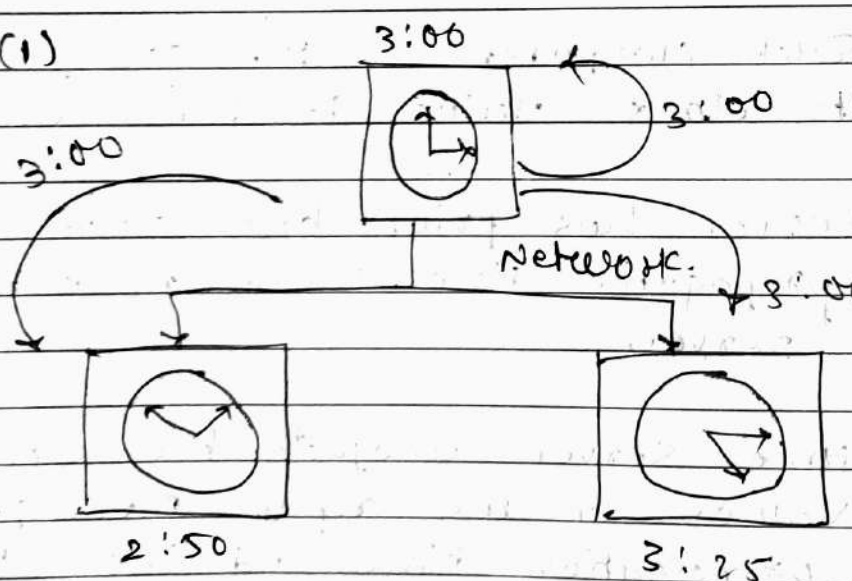
- A time server periodically (approx. once in 4 minutes) sends its time to all the computers and polls them for the time difference.

- The computers compute the time difference and then reply.

- The server computes an average time difference for each computer.

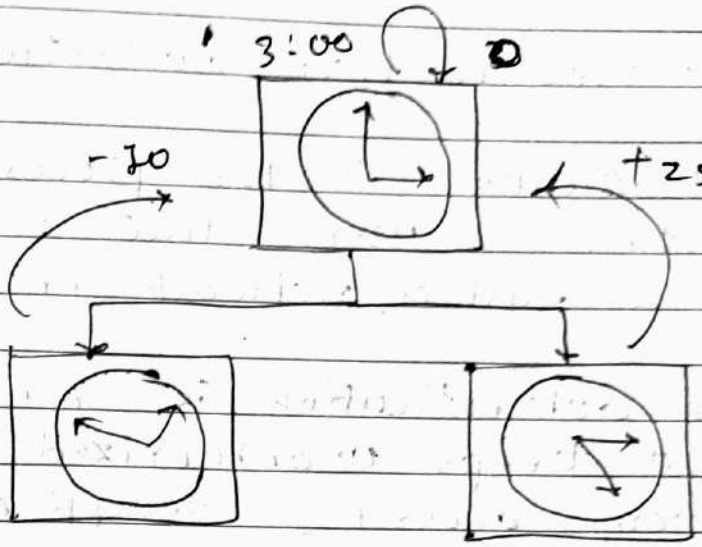
- The server commands all the computers to update their time (by gradual time synchronisation).

(1)



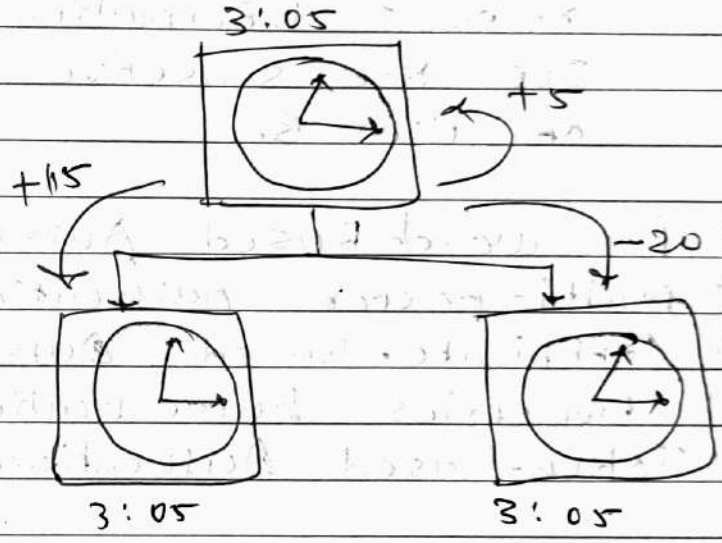
(a)

(2)



(b)

(3)



(c)

Q.(x) Approaches to User Authentication :-

- Authentication is the process of identifying users that request access to a system, network or device.

- User authentication is a method that keeps unauthorized users from accessing sensetive information.

- Eg: User A only has access to relevant information and can't see the sensetive information of user B.

(1) Password Based Authentication
(2) Multi-factor Authentication
(3) Certificate-based Authentication
(4) Biometrics Authentication
(5) Token-based Authentication.

Q.(x) ~~Multi-casting & characteristics of Multi-casting.~~

Q.(x) Layered Architecture :-

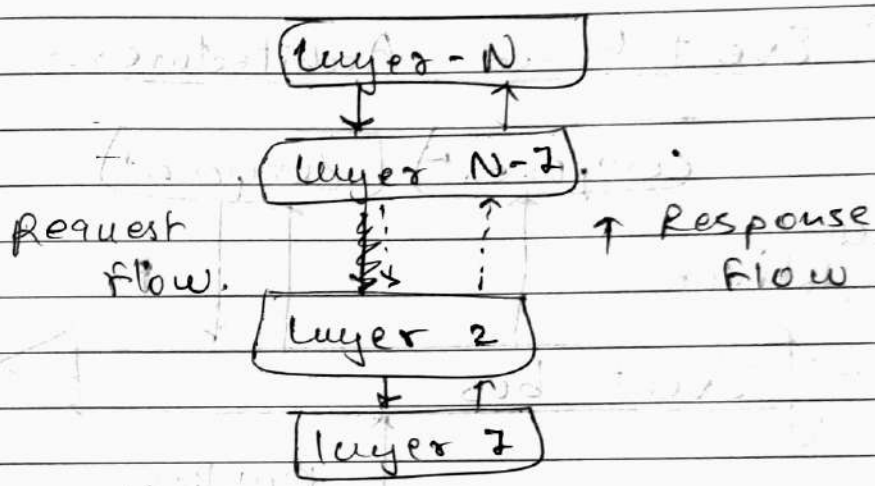
Q.(x) List out the types of system Architecture in DS.

- ~~There~~ Important styles of Architecture for Distributed systems:

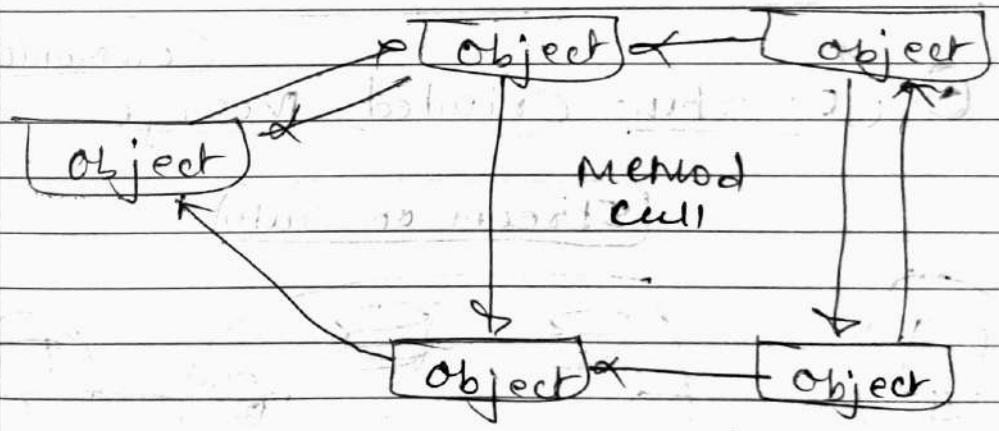
(1) Layered architectures
(2) Object-based architectures
(3) Data-centered architectures
(4) Event-based architectures.

(1) Layered Architectures :-

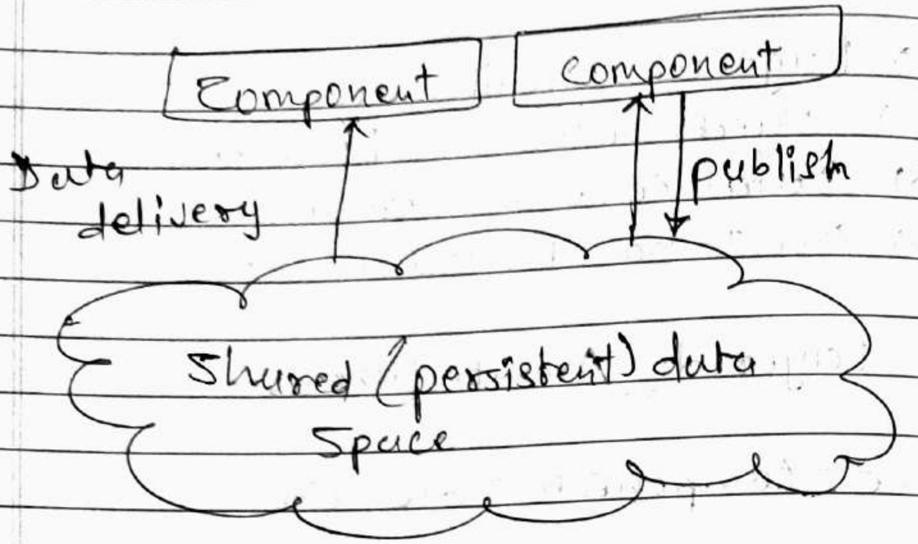
- the components are orga



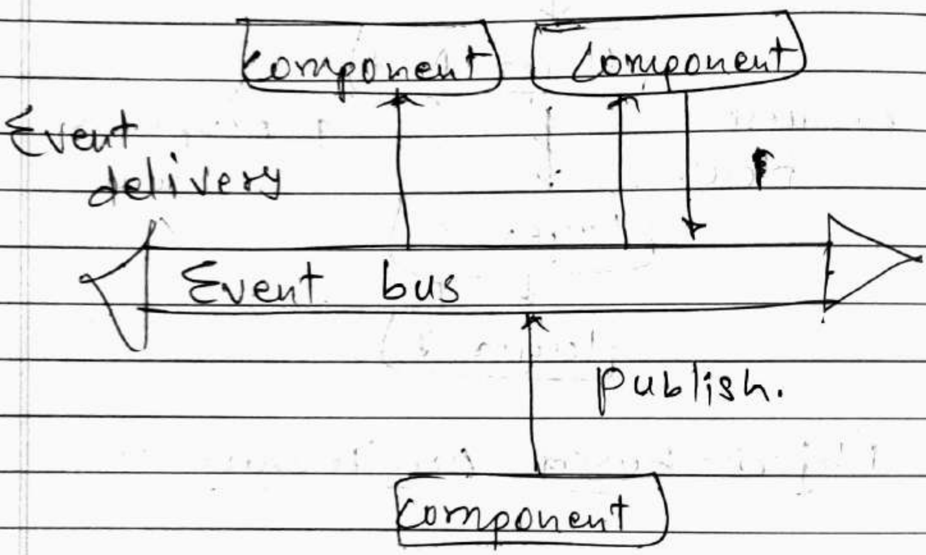
(2) Object-based Architectures :-



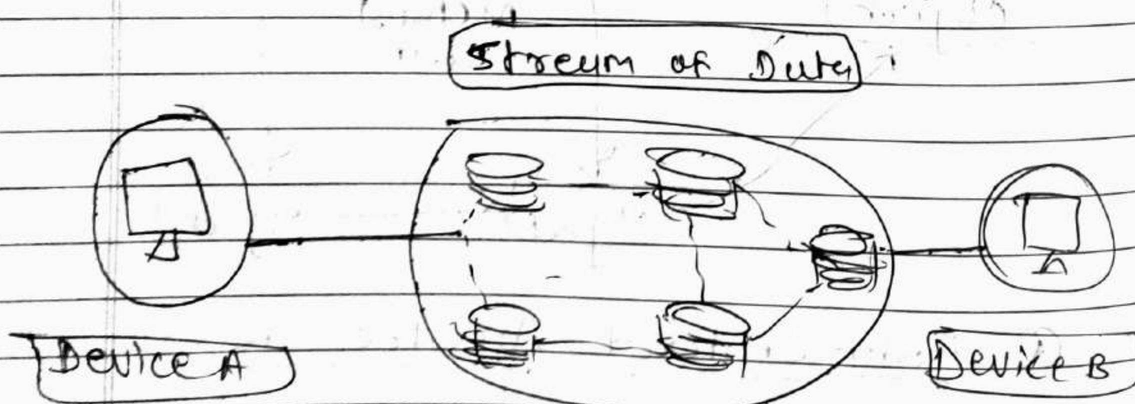
(3) Data-Centered Architectures :-



Event-based Architectures :-

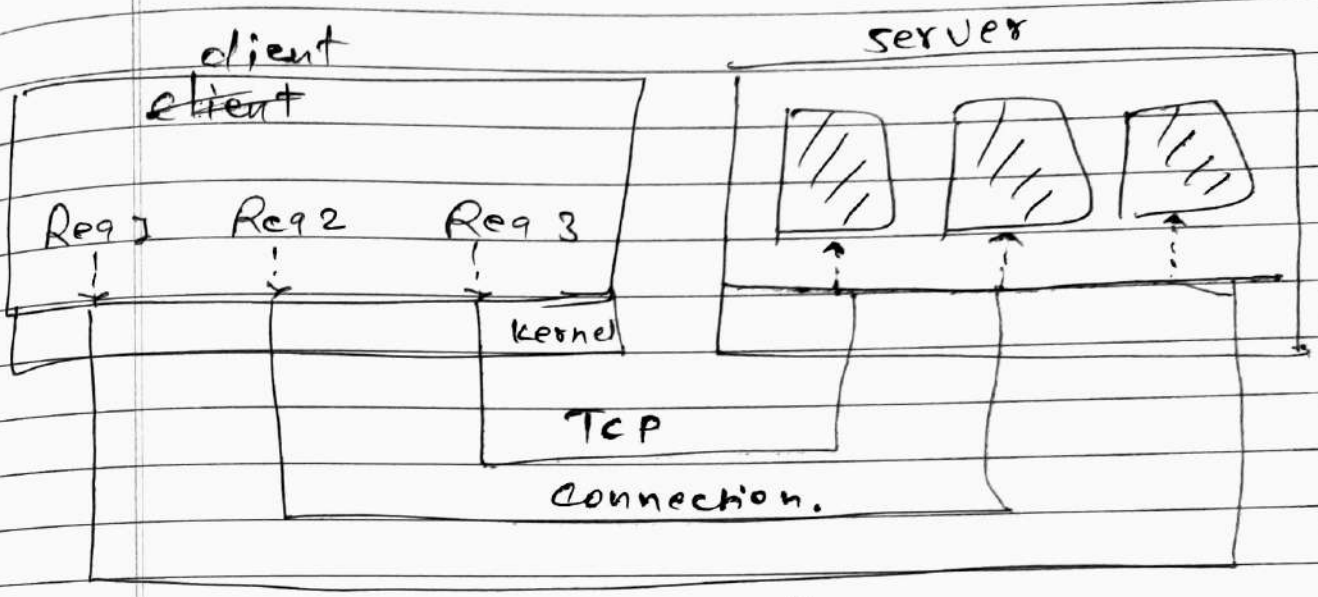


Connection-Oriented Message Communication



COMC

① Non-Persistent HTTP (1.0) :-



② Persistent HTTP (1.1) :-