

## Introduction: =)

Networks of computer's are everywhere. The internet is one, as are the many networks of which it is composed. Mobile phone networks, Corporate networks, factory networks, Campus networks, home networks, in-car networks, all of these separately and in combination, share the essential characteristics that make them relevant subjects for study under the heading distributed system.

### 1.2 Features of Distributed Systems

Although distributed systems are to be found everywhere, their design is quite simple and there is still a lot of scope to develop more ambitious services and applications.

#### 1.1.1 Concurrency

Both services and applications provide resources that can be shared by clients in a distributed system. There is therefore a possibility that several clients will attempt to access a shared resource at the same time.

For example, a data structure that records bids for an auction may be accessed very frequently when it gets close



to its deadline.

### 1.1.2 No Global clock

The portability of many of these devices, together with their ability to connect conveniently to networks in different places, makes mobile computing possible. Mobile computing called nomadic computing is the performance of computing tasks while the user is on the move, or visiting places other than their usual environment.

### 1.1.3 Heterogeneity

The internet enables users to access services and run applications over a heterogeneous collection of networks and computers. It is applied to:-

- networks
- computer hardware
- operating system
- programming languages
- implementations by different developers.

### 1.1.4 Openness

The openness of computer system is the characteristic that determines whether the system can be extended and re-implemented in various ways.



### 1.1.5 Security

Security of information resources has three components:-

- Confidentiality
- Integrity
- Availability

The internet allows a program in one computer to communicate with a program in another computer irrespective of its location, security issues are associated with allowing free access to all of the resources in the internet.

### 1.1.6 failure handling

failures in a distributed system are partial - that is, some components fail while others continues to function.

- Detecting failures
- Masking failure
- Terminating failures
- Recovery failure
- Redundancy

### 1.2 Nodes of Distributed System

In communication networks, a node is either a connection point or a distributed point or a communication end point.

The definition of a node depends on the network and protocol layers referred to a physical layer.



### 1.2.1 Server

In computing a server is a system that responds to requests across a computer network to provide, or help to provide a network service.

### 1.2.2 Clients

- In computing client software that accesses a remote service on another computer.
- Client, a recipient of goods or services in return for monetary.
- Client, in the system of patronage in ancient Rome, an individual protected and sponsored by a patron.

### 1.2.3 Peer

- A member of the peerage, a system of honours or nobility in various countries.
- A network entity with which one performs peering operations.
- Peer-to-peer, a kind of computer network in which participants act as both client and server.

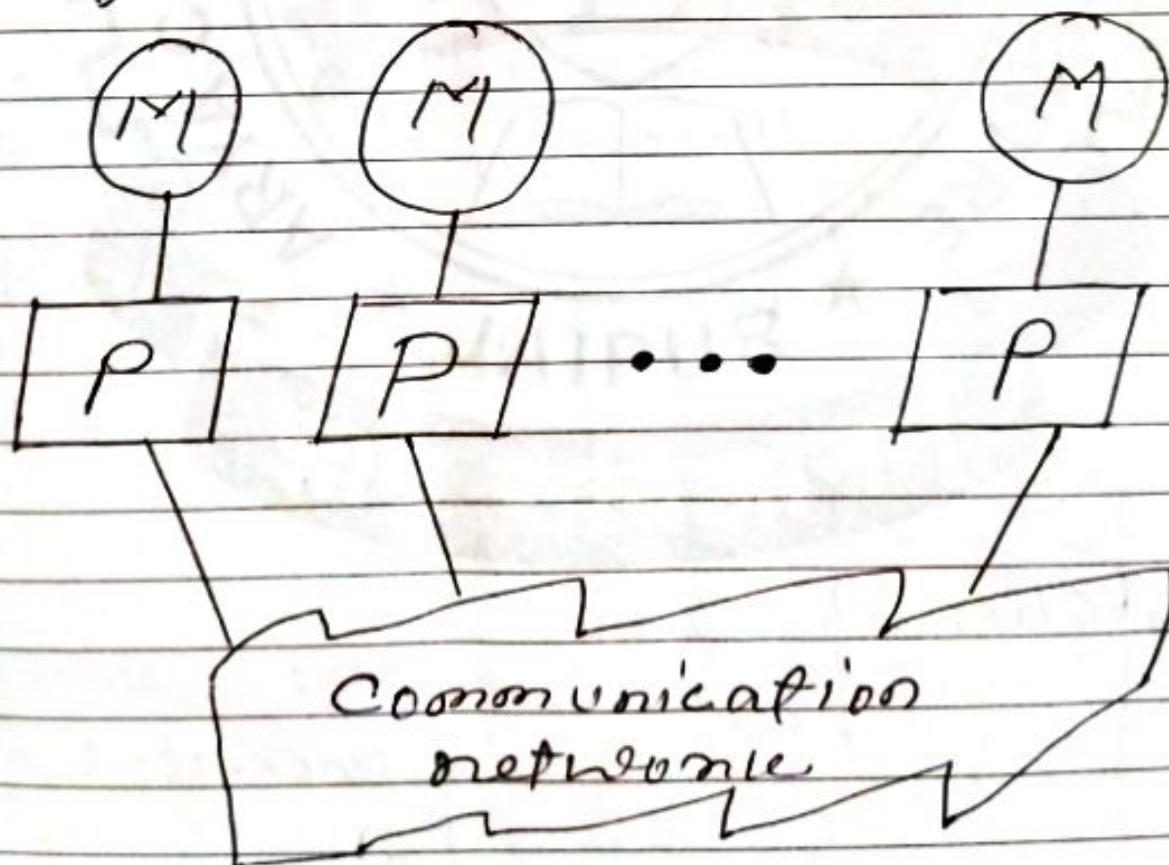


### 1.3 Distributed computing paradigms

The architecture of a system is its structure in terms of separately specified components.

#### 1.3.1 Message-passing paradigm

- Many instances of sequential paradigm considered together.
- Programmer imagines several processes each with own memory, and writes a program to run on each processor.
- Processes communicate by sending messages to each-other.



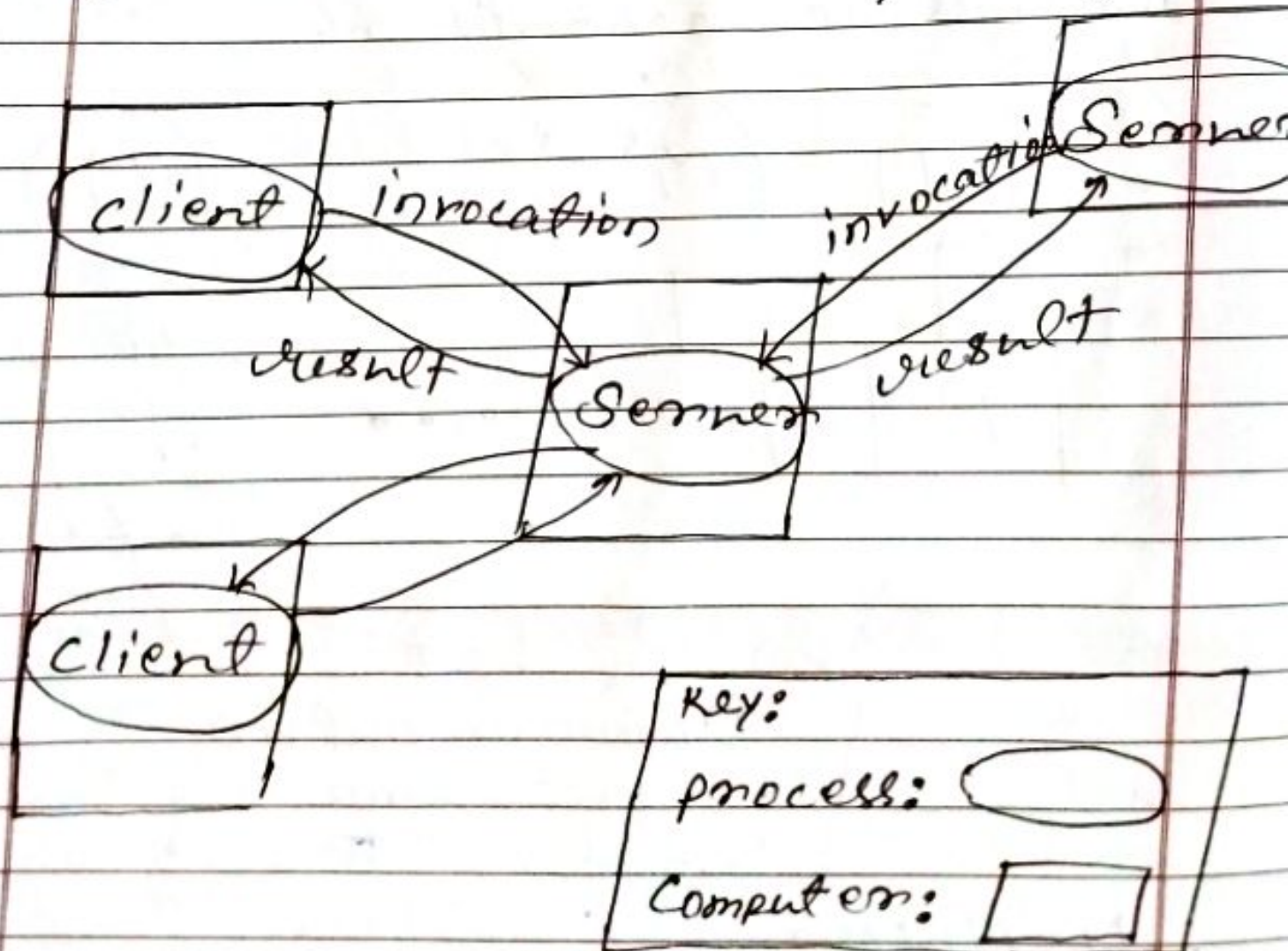
M - Message  
P - Processor



- Message Transfer occurs when data moves from variables in one sub-program to variables in another sub-program.
- Sending and receiving processes cooperate to provide required information for message transfer.

### 1.3.2 Client Server Paradigm

This is the architecture that is most often cited when distributed systems are discussed. It is historically the most important and remains the most widely employed.

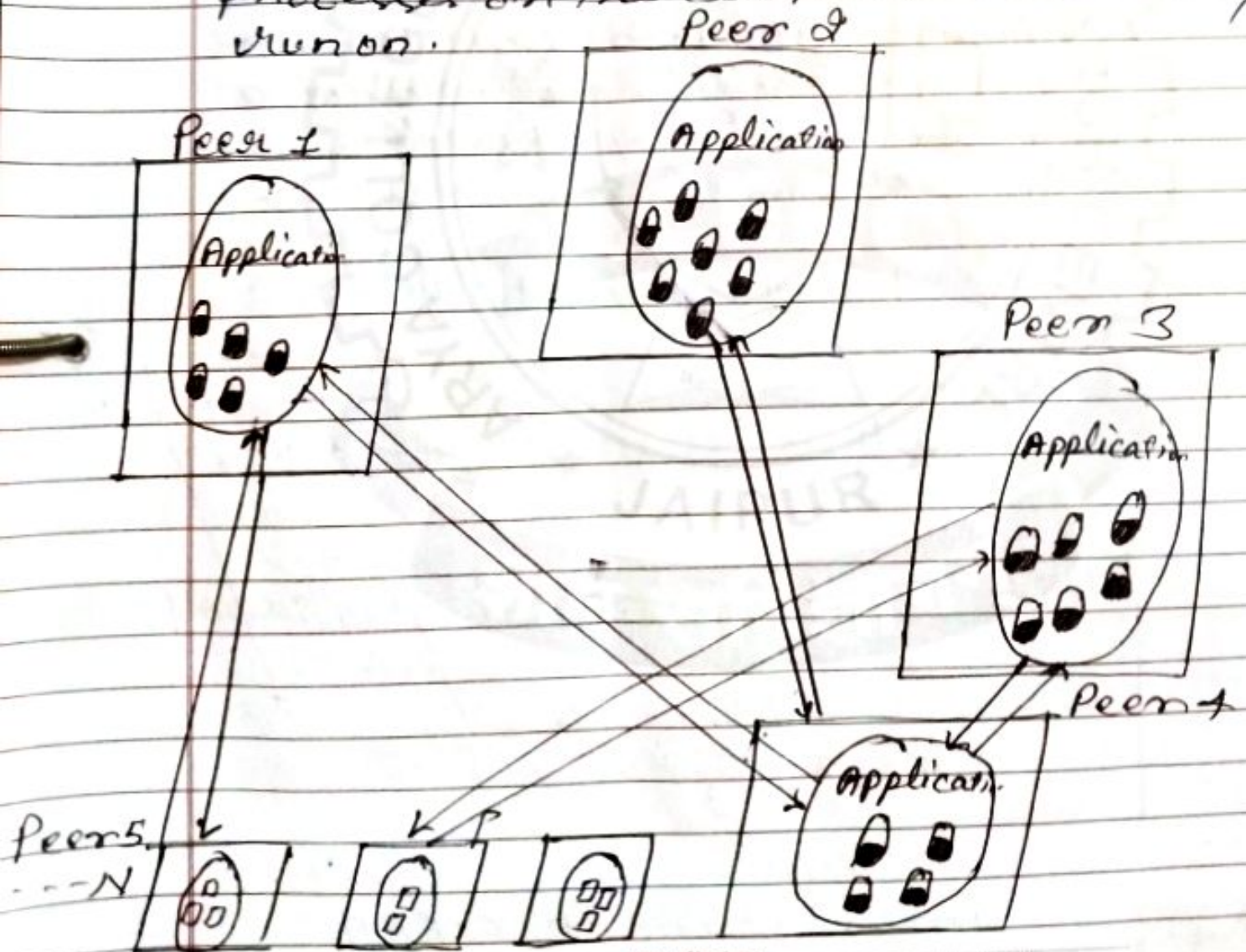




- A web server is often a client of a local file server that manages the files in which the web pages are stored.

### 1.3.3 Peer to Peer System architecture

In this architecture all of the processes involved in a task or activity play similar roles, interacting co-operatively as peers without any distinction between client and server processes on the computers that they run on.

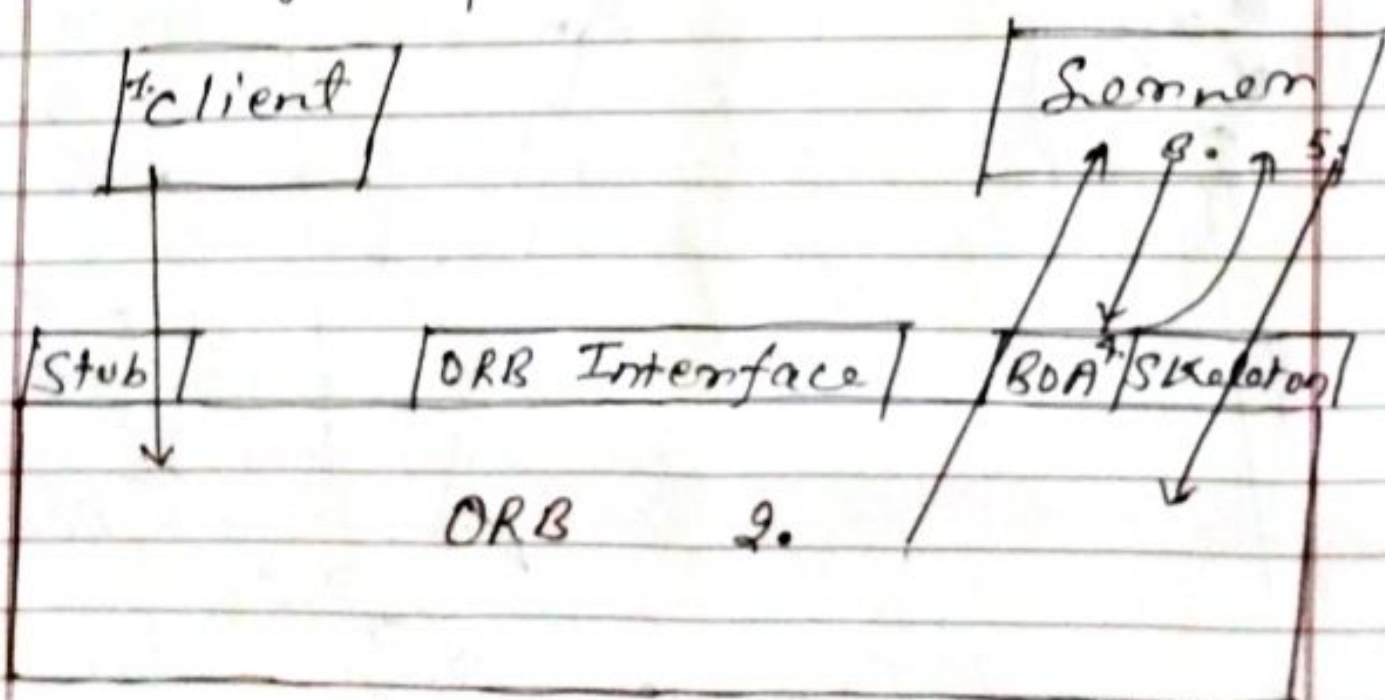


- The hardware capacity and as functionality of today's desktop computers exceeds that of yesterday's servers and majority are equipped with always-on broadband network connection.

### 1.3.4 Distributed Object Paradigm

This paradigm exploits the benefits of object-oriented technology and use the internet and its communications infrastructure as a vehicle for the delivery of a wide range of sophisticated value-added distributed services.

CORBA:- Common Object Request Broker Architecture



Client Server object interactions in CORBA



- The client object called a server method through the stub.
- The ORB (Object Request Broker) handles the request to the BOA (Basic Object Adapter) which activates the implementation.
- The implementation invokes the BOA to say it is active and available.
- BOA passes the request to a method in an object implementation via its skeleton.
- The implementation returns the results back to the client object through ORB.

### 3.1.1 Remote Method Invocation

Remote Method Invocation is a way that a programmer, using programming language and development environment, can write object-oriented programming in which objects on different computers can interact in a distributed network.

RPC procedure call

RMI (Remote Method Invocation)

TCP	UDP	Applications, Services	
Header size 20 byte	8 byte	RMI and RPC	HTTP - 80
Connection oriented	Connection less	request-reply protocol	FTP - 20
slow	fast	marshalling and external data representation	SMTP - 25
HTTP	DNS		HTTPS - 443
FTP	DHCP		DNS - 53
SMTP	RIP		DHCP - 67
HTTPS		UDP and TCP	RIP - 68
slow	fast		

Name of Lecturer : \_\_\_\_\_

Dynamic host configuration



### 1-3.4.2 Object Request Broker

In common object request broker architecture, an object request broker is the programming that acts as a "broker" between a client request for a service from a distributed object or component and the completion of that request.

ORB may be thought of as a strategic middleware that is more sophisticated conceptually and in its capabilities than earlier middleware including Remote Procedure call (RPC) Services.

### 3.4.3 Object Space

- Many objects can be encapsulated in server processes, and references to them are passed to other processes so that their method can be accessed by remote invocation.
- This is the approach adopted by CORBA and by Java with its remote method invocation mechanism.



### 1.3.5 Mobile Agents

- A mobile agent is a running program that travels from one computer to another in a network carrying out a task on someone's behalf, such as collecting information, eventually returning with the results.
- A mobile agent may make many invocations to local resources at each site it visits - for example, accessing individual database entries.

### 3.6 Network Services

- A network service is an application running at the network application layer, that provide data storage, manipulation, presentation and communication.
- In computer networking programme the application layer is an abstraction layer reserved for communications protocols and methods designed for process-to-process comm" across an internet protocol computer network.



### 1.3.7 Collaborative applications

- There are different kinds of the applications available in which it is used in:
- CORBA
  - Java RMI
  - Web Services
  - Microsoft's Distributed Component Object Model (DCOM)
  - The ISO/ITU-T's Reference Model for Open Distributed Processing (RM-ODP).

### 1.4 Model of Distributed System

#### 1.4.1 Work station Model

- The distributed computing based on the work station model consists of several work stations inter-connected by a communication network. A company's office and department has the work stations scattered throughout the campus.
- It has been often found that in such an environment, at any one time a significant proportion of the work stations are idle.



### 1.3.7 Collaborative applications

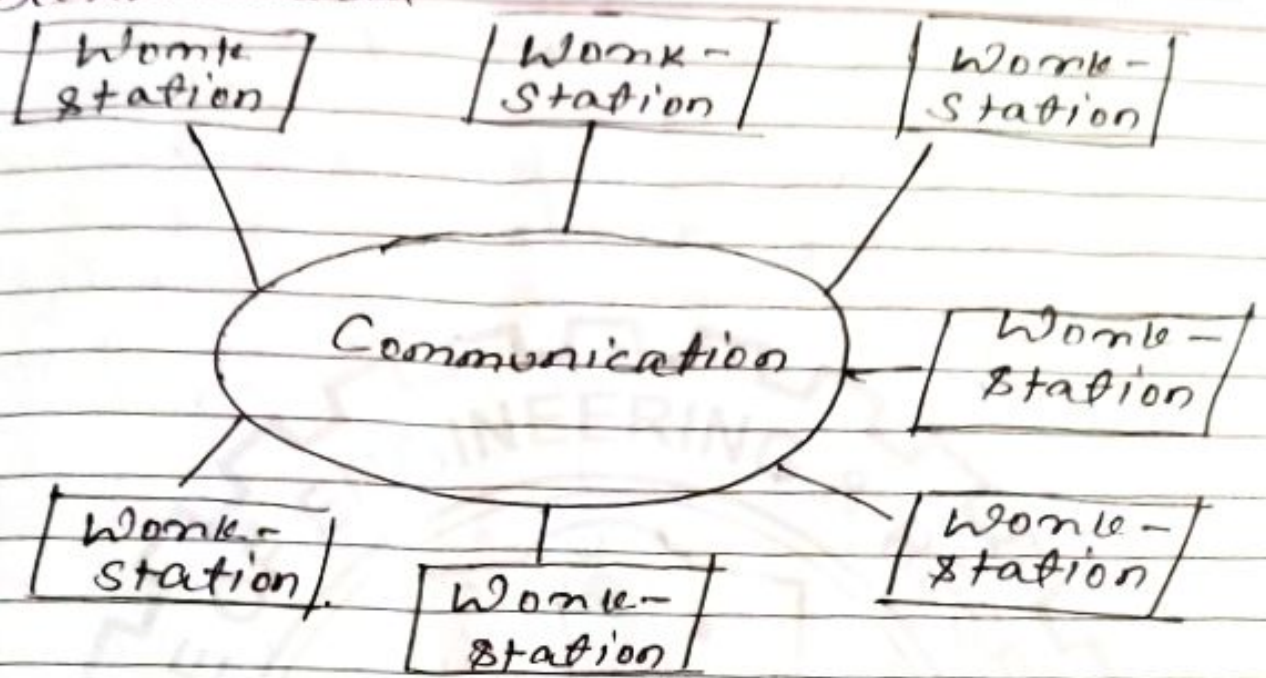
- There are different kinds of the applications available in which it is used in:
- CORBA
  - Java RMI
  - Web Services
  - Microsoft's Distributed Component Object Model (DCOM)
  - The ISO/ITU-T's Reference Model for Open Distributed Processing (RM-ODP).

### 1.4 Model of Distributed System

#### 1.4.1 Work station Model

- The distributed computing based on the work station model consists of several work stations inter-connected by a communication network. A company's office and department has the work station scattered throughout the campus.
- It has been often found that in such an environment, at any one time a significant proportion of the work stations are idle.

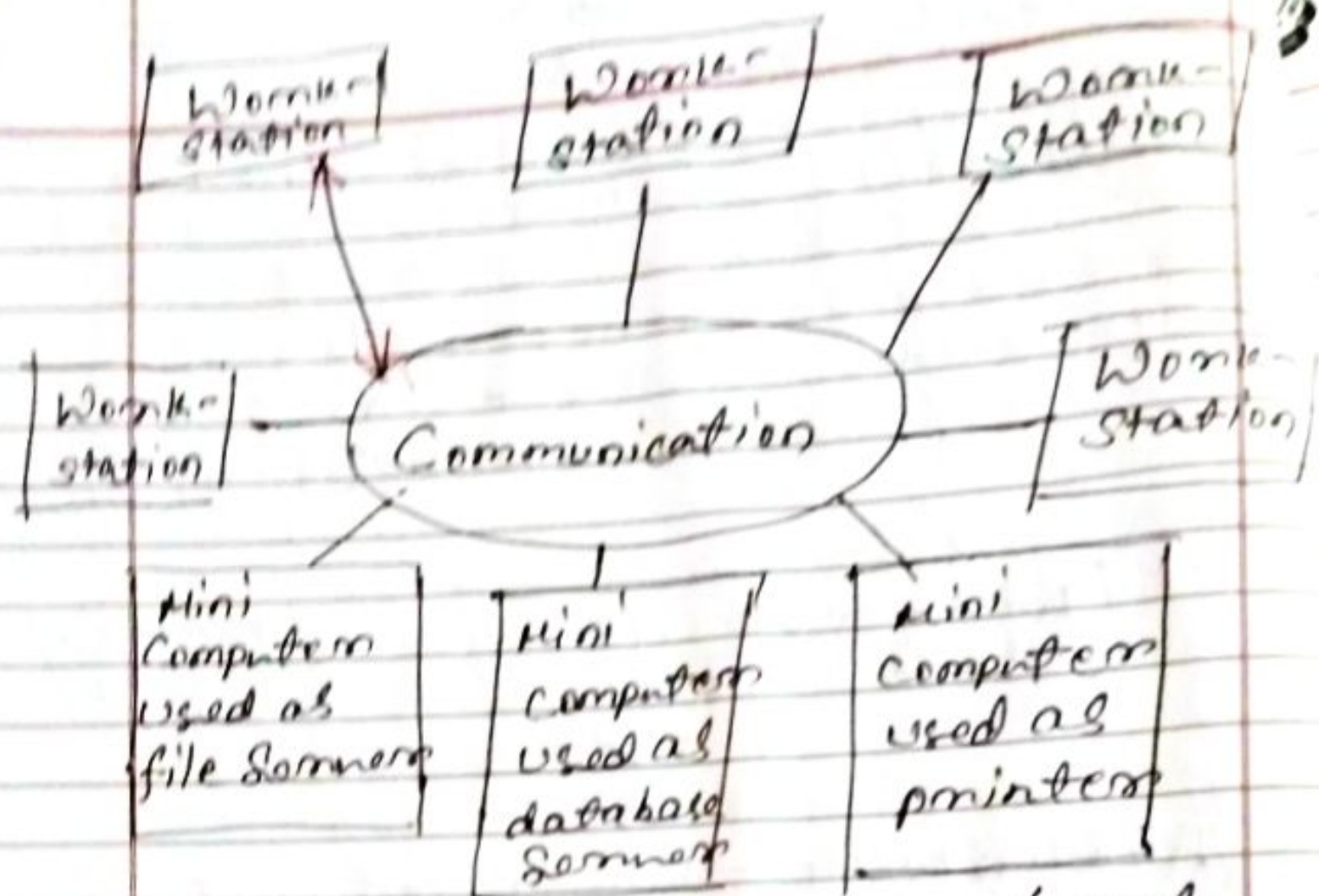




#### 4.2 Work Station - Semnet Model

- The workstation model is a network of personal workstations, each with its own disk and a local file system.
- Environments than disful workstations making the workstation-semnet model more popular than the workstation model for building distributed computing system.
- A distributed computing system based on the workstation semnet model consists of a few mini-computers and several workstations.



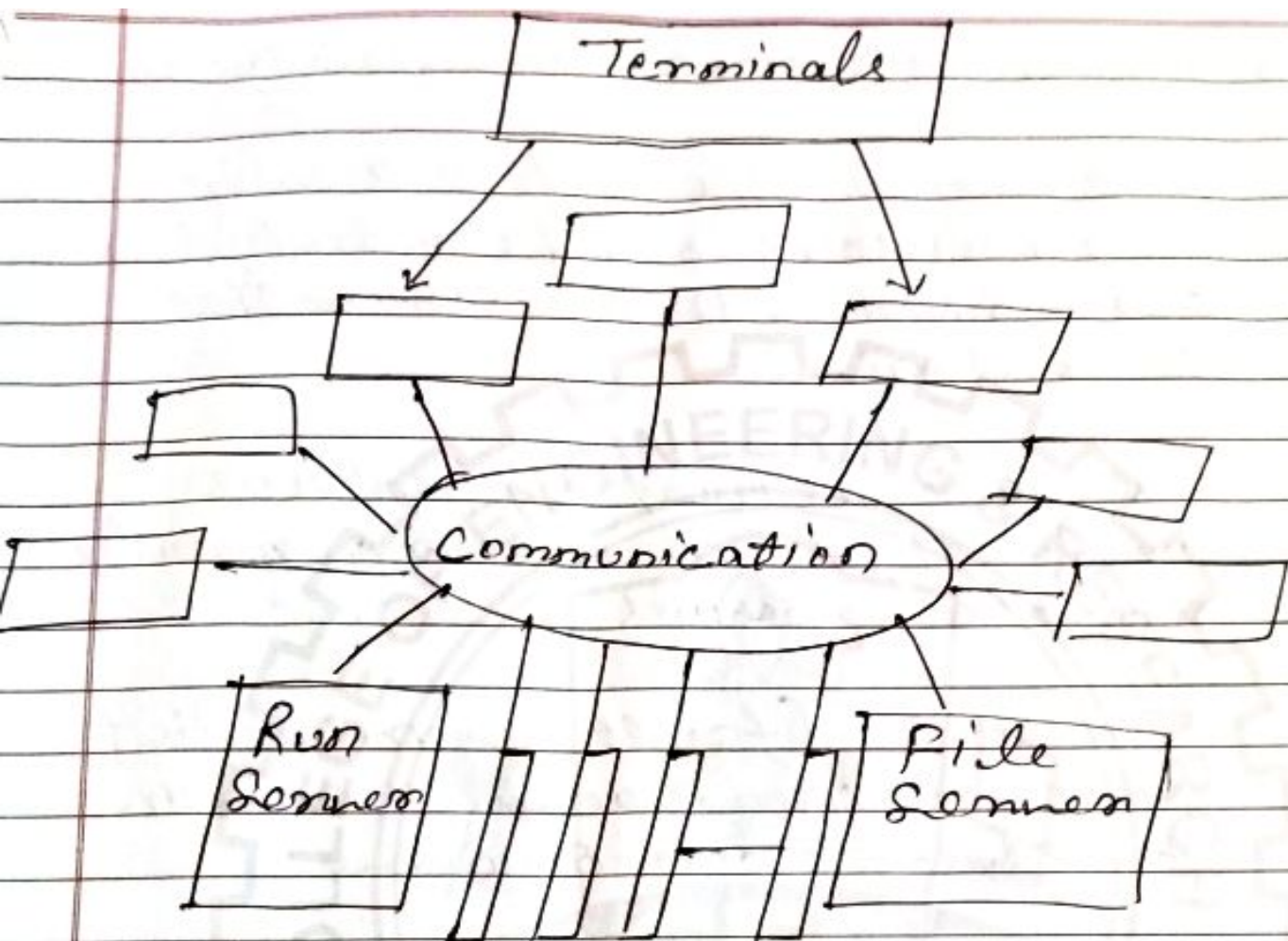


Distributed computing system based on workstation server model

### 1.4.3 Processor - Pool Model

- The processor pool model is based on the observation that most of the time a user does not need any computing power but once in a while he or she may need a very large amount of computing power.
- In processor pool model the processors are pooled together to be shared by the user's as needed.





A Distributed Computing system based on the process pool model

#### 4. Comparison of the Distributed computing models

A conventional distributed environment assumes a pool of computational nodes. A node is a collection of resources (CPU, memory, storage, I/O devices etc) + treated as a single unit. The differences and comparison is of the

- virtual machine level
- virtual pool level
- physical level



## 1.4.5 Advantages of Distributed System

- a) Performance: Very often a collection of processors can provide higher performance than a centralized computer.
- b) Distribution: Many applications involve, by their nature, spatially separated machines.
- c) Reliability: (fault tolerance): If some of the machines crash, the system can survive.
- d) Incremental growth: As requirements or processing power grow, new machines can be added incrementally.
- e) Sharing of data/resources: Shared data is essential to many applications. Other resources can also be shared.
- f) Communication: facilitates human-to-human communication.



#### 4.6 Disadvantages of Distributed Systems:-

- a) Difficulties of developing distributed software: How should operating systems, programming language and application look like.
- b) Networking Problems: Several problems are created by the network infrastructure which have to be dealt with: loss of messages, overloading.
- c) Security Problem: Sharing generates the problem of data security.

#### 1.5 Type of Operating System

##### 1.5.1 Network Operating System

User's are aware of multiplicity of machines. Access to resources of various machines is done by:

- Remote logging into the appropriate remote machine.
- Transferring data from remote machines to local machines, via the FTP mechanism.



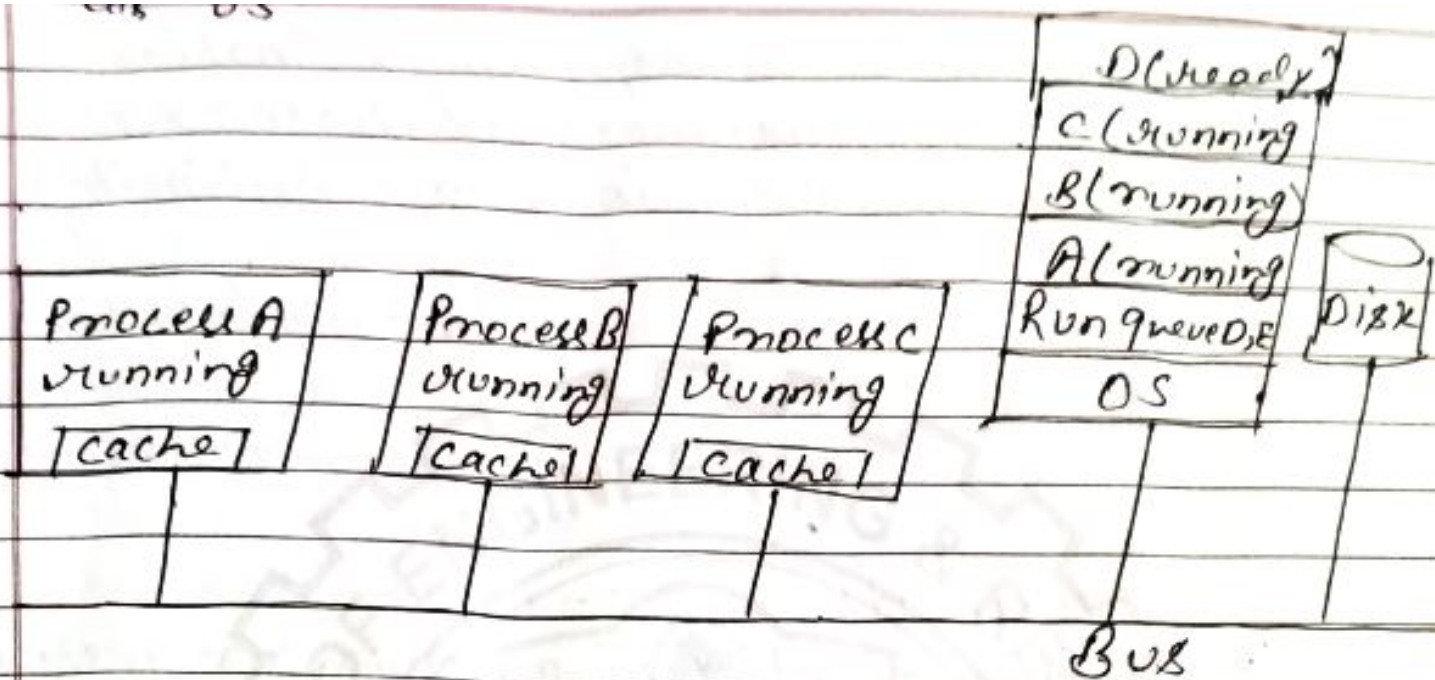
1.5.2 Distributed Operating System  
User's not aware of multiplicity of machines. Access to remote resources similar to access to local resources.

- Data Migration: - Transfer data by transferring entire file, or transferring only those portions of the file necessary for the immediate task.
- Computation Migration: - Transfer the computation, rather than the data, across the system.
- Process Migration: - Execute an entire process, or parts of it, at different sites.

### 1.5.3 Multiprocessor Time Sharing System

The combination is tightly coupled software with hardware, while various special purpose machine exist in this category, the most common example are multi-processors that are operated as UNIX time sharing.





Multi-processor with a single run queue

## 1.6 Issues of Designing Distributed Operating System

### 1.6.1 Transparency

Transparency is defined as the concealment from the user and the application programmer of the separation of components in a distributed system.

- **Location Transparency:** - Enable resources to be accessed without knowledge of their physical network.
- **Concurrency Transparency:** - Enables several processes to operate concurrently using shared resources without interference between them.



- Replication Transparency: Enables multiple instances of resources to be used to increase reliability.
- Failure Transparency: Enables the concealment of faults, allowing user and application program to complete.

### 1.6.2 flexibility

- The flexibility of a computer system is the characteristic that determines whether the system can be extended and re-implemented in various ways.
- flexibility can be achieved unless the specification and documentation of the key software interfaces of the components of a system are made available to the developers.

### 1.6.3 Reliability:

- Data types such as integers may be represented in different ways on different sorts of hardware.
- The OS of all computers on the internet need to include an implementation of the internet protocols.



#### 1.6.4 Performance

- The challenges arising from the distribution of resources extend well beyond the need for the management of concurrent updates. This is arising from the limited processing and communication capacities of computers.
- In addition, transfer of data between processes and the associated synchronizing of control is relatively slow, even when the processes reside in the same computer.

#### 1.6.5 Scalability

- Distributed systems operate effectively and efficiently at many different scales, ranging from a small intranet to the global internet.

#### 1.6.6 Security

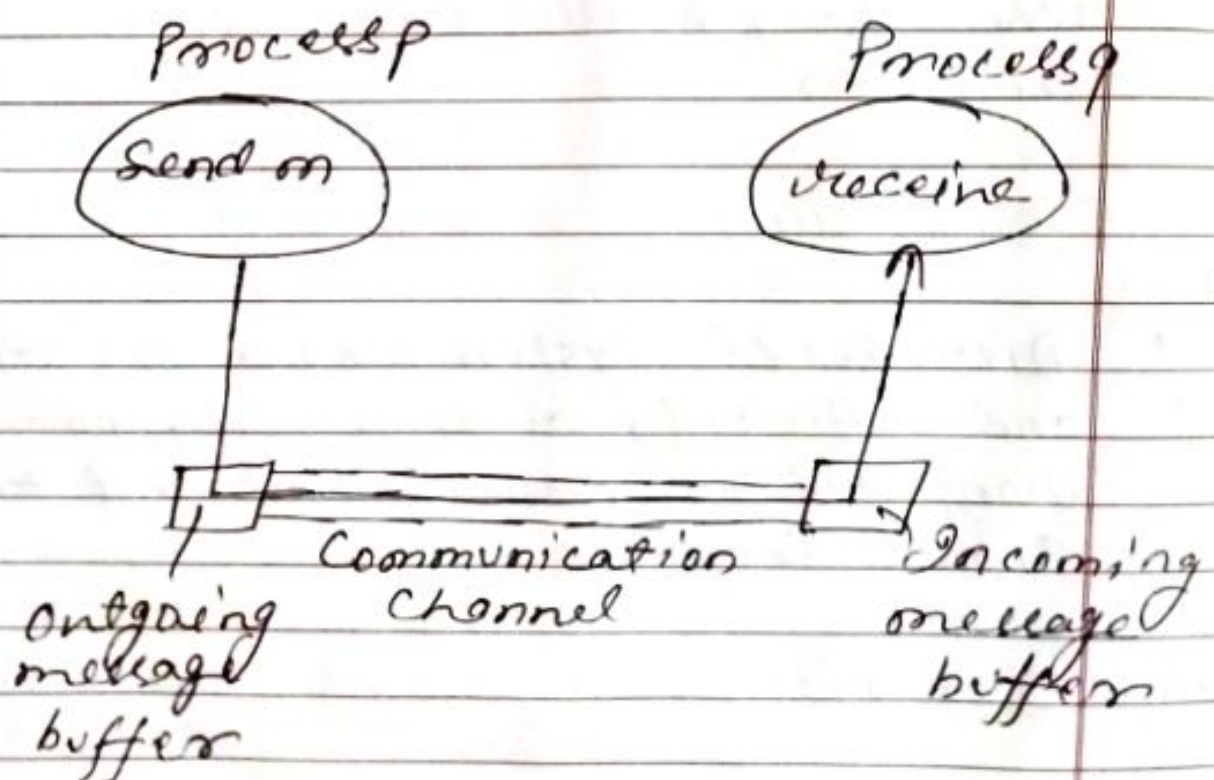
- Security for information resources has three components: (i) confidentiality (ii) integrity (iii) availability and these are the major aspects of the security.



- Although a firewall can be used to form a barrier around an internet, restricting the traffic that can enter and leave.

## 1.6.7 Fault Tolerance

- In a distributed system both processes and communication channels may fail - that is, they may depart from what is considered to be correct or desirable behaviour.



- The two phase commit protocol for transactions. It is designed to complete well defined failure.



1.7

## System Concepts and architecture

The division of responsibilities between system components applications, server and other processes and the placement of the components on computers in the network is perhaps the most evident aspect of distributed system design.

### 1.7.1 Goals

The different kinds of the goals are available for designing the system architecture:

#### 1.7.1.1 Efficiency

- System management components are software processes that defines the nodes policies.
- These components are the part of the OS outside the kernel. These components provide higher level communication, process and resource management.

#### 1.7.1.2 Flexibility

- Flexibility in a distributed operating system is enhanced through the



modular and characteristics of the distributed operating systems and by providing a richer set of higher-level services.

- The completeness and quality of the kernel/microkernel simplifies implementation of such services, and potentially enables service providers greater choice of providers for such services.

### 1.7.1.3 Consequences

#### a) Concurrency:

- Autonomous: Computers carry out tasks independently.
- Cooperative: Computers coordinate actions.

#### b) No global clock

- Hard to synchronize their clocks precisely
- Coordinate by exchanging messages

#### c) Independent failures

- Part of network or node faults does not stop the system.



#### 1.7.1.4 Robustness | property of being strong, and healthy in constitution.

↓ is the ability of a comp. system to

- The synchrony assumed is that of eventual bounded delay message delivery. The best way to give some initial intuition to the difficulties to the problem.
- Distinct Synchronized clapping is typically also achieved in-spite of a certain fraction of the audience that are reluctant to cooperate and even intentionally clap in their own pace.

#### 1.7.2 Transparency

The distributed systems should be perceived as a single entity by the users on the application programs rather than as a collection of autonomous systems, which are co-operating.

##### 1.7.2.1 Access Transparency

Clients should be unaware of the distribution of the files. The files could be present on a totally different set of servers which are physically distant apart and



single set of operations should be provided to access these resources as well as the local files.

### 1.7.2.2 File Location Transparency

- Clients should be able to see a uniform file name space, files or groups of files may be relocated without changing their path names.
- A location transparent name contains no information about the named object's physical location.

### 1.7.2.3 Migration Transparency

- This transparency allows the user to be unaware of the movement of information or processes within a system without affecting the operations of the user's and the applications that are running.
- This mechanism allows for the load balancing of any particular client, which might be overloaded. The system that implement this transparency are NFS and Web pages.



#### 1.7.2.4 Replication Transparency

- This kind of transparency should be mainly incorporated for the distributed file systems, which replicate the data at two or more sites for more reliability.
- The client should not be aware that a duplicated copy of the data exists.

#### 1.7.2.5 Parallelism Transparency

- The system is responsible for exploiting any ability to parallelize task execution without user knowledge or interaction.
- Arguably the most difficult aspect of transparency and described by authors for distributed system designers.

#### 1.7.2.6 Failure Transparency

- Enables the concealment of faults, allowing user and application programs to complete their tasks despite the failure of hardware or software component.
- Fault tolerance is provided by the mechanisms that relate to access transparency.



### 1.7.2.7 Performance Transparency

- The system is responsible for the detection and remediation of local or global performance shortfalls.
- Note that the system policies may prefer some user classes over others. No user knowledge is involved.

### 1.7.2.8 Size Transparency

- The system is responsible for managing its geographic reach, number of nodes, level of nodes capability without any required user knowledge or interaction.

### 1.7.2.9 Revision Transparency

- The system is responsible for upgrades and revisions and change to system infrastructure without user knowledge or action.



## 7.2.10 Scaling Transparency

- A system should be able to grow without affecting application algorithms. Graceful growth and evolution is an important requirement for most enterprises.
- A system should also be capable of scaling down to small environments where required, and be space and time efficient as required.

## 7.3 Services

- Basic services provided by OS follows
- Program development
  - Access to I/O devices
  - Controlled access to files
  - System access
  - program execution

### 7.3.1 Primitive Services

- Primitive services are those that must be implemented on kernel of each node in the system.

In distributed system when communications are done through message passing set of primitives for send and receive



must be defined and implemented.

### 1.7.3.2 Services by System Schemas

- The services which can be implemented anywhere in the system and perform the functions which is basic to the operation of a distributed system.
- The service provider of the system is called system schema. To achieve transparency in DS schema map physical objects with logical system names.

### 1.7.3.3 Value added Services

- Services that are not needed in the implementation of a distributed system but useful in supporting distributed application.
- Used to increase the computational performance and enhancement of fault tolerance or by need for cooperative activities.



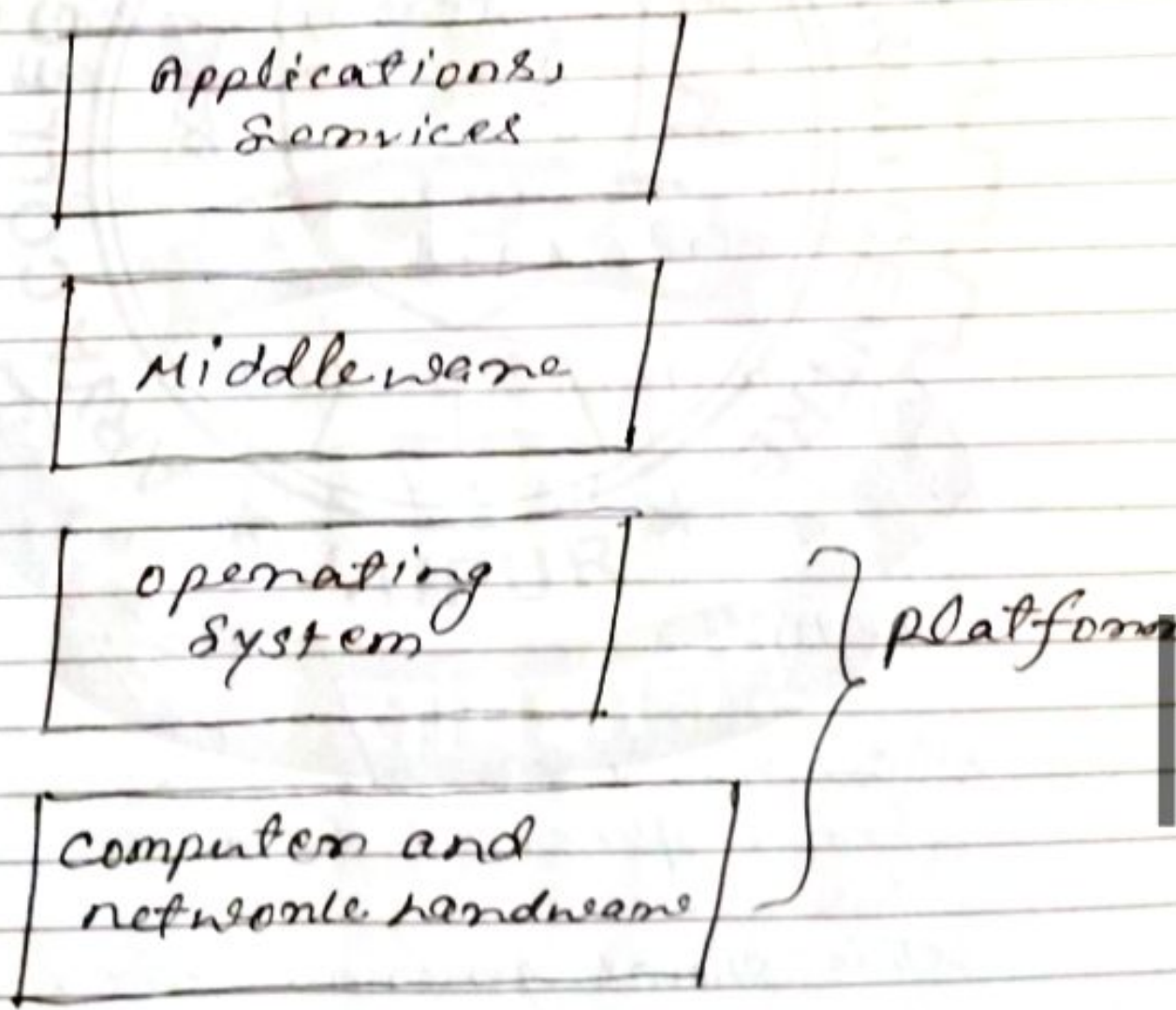
1.8

## Architecture Models

1.8.1

### Distributed System Architectures

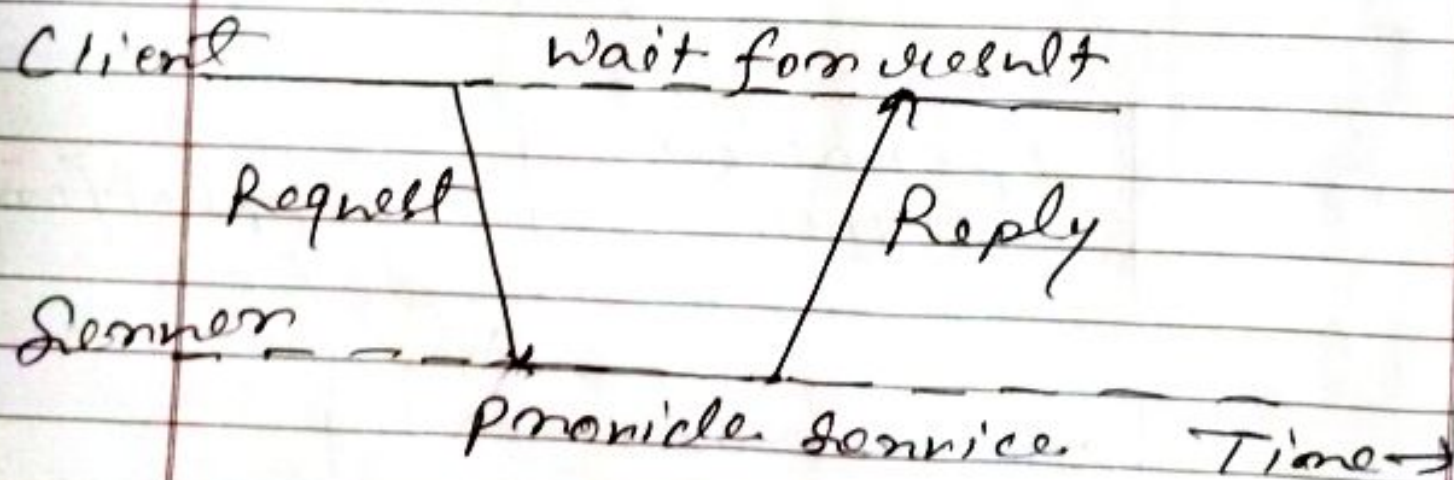
The architecture of a system is its structure in terms of separately specified components. The overall goal is to ensure that the structure will meet present and likely future demands on it.





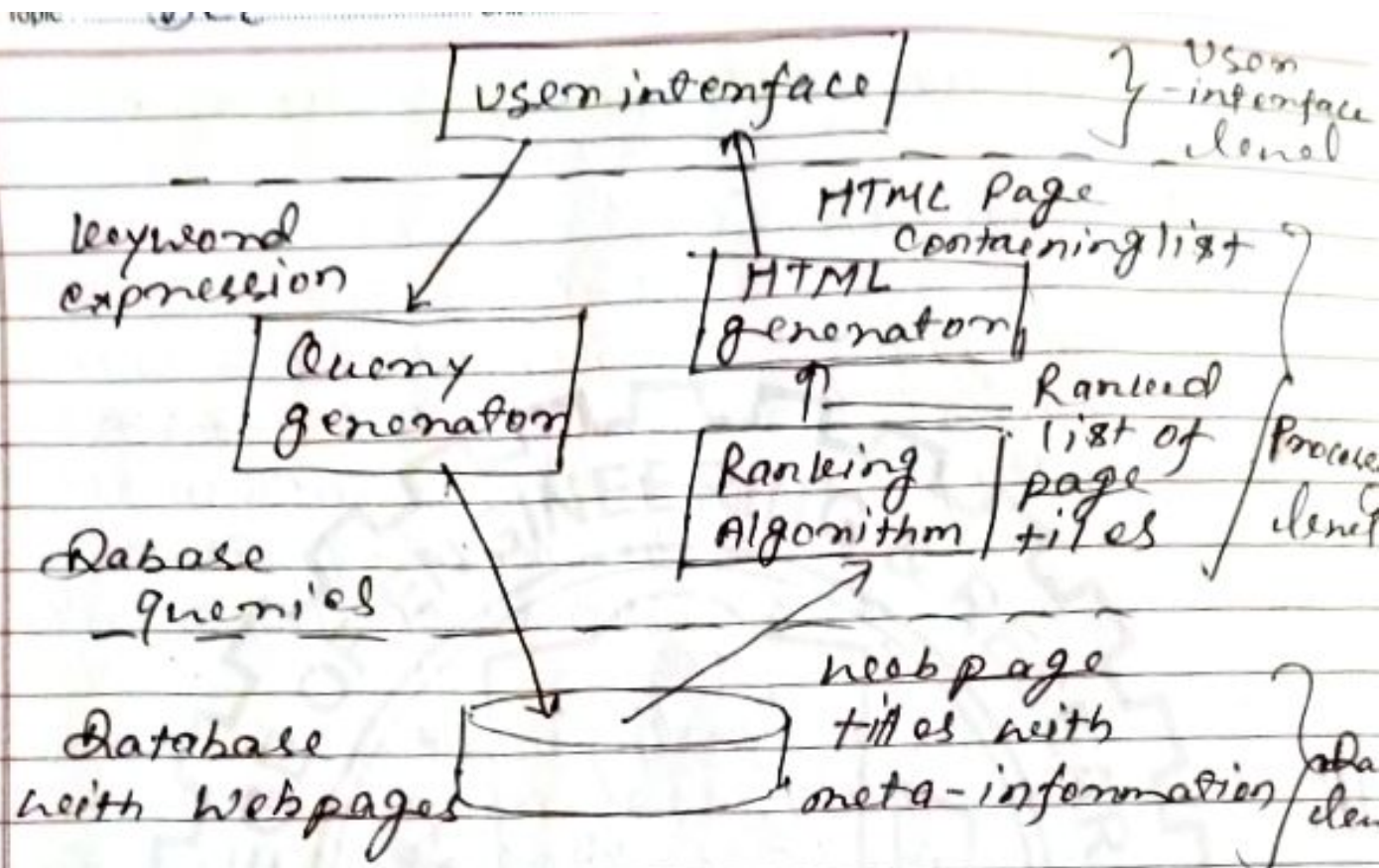
- ⇒ The lowest level hardware and software layers are often referred to as a platform for distributed systems and applications.
- ⇒ Middleware is defined as a layer of software whose purpose is to make heterogeneity and to provide a convenient programming model to application programmers.
- ⇒ Many distributed applications rely entirely on the services provided by the available middleware to support their needs for comm. and data sharing.

### 1.8.2 Communication Network Architecture



This shows general interaction between a client and a server.





Simplified organization of an internet search engine into three different layers

- ⇒ Vertical organization of communication and control paths
- ⇒ Logical separation of functions into client (requesting client) and server (responder)
- ⇒ Classification of a system as centralized or decentralized depends to communication and control organization primarily



1.9

## Distributed computing Environment

1.9.1

What is DCE?

- In business enterprises, distributed computing generally has meant putting various steps in business processes at the most efficient places in a network of computers.
- The distributed computing environment is a widely-used industry standard that supports kind of distributed computing.

1.9.2

How was DCE created?

- The distributed computing environment is a software system developed in the early 1990s by a consortium that include Apollo Computer, IBM and others. The DCE supplies a framework and toolkit for developing client/server applications. The framework includes a remote procedure call (RPC) mechanism known as DCE.
- DCE was a big step in direction to standardisation of architectures



which were manufacturer dependent before. Similar to OSI model, DCE was come about to a large degree as part of the Unix wars of the 1980s.

### 1.9.3 DCE Components

DCE comprises several technology components:

#### 1.9.3.1 DCE Threads

In a traditional computer program, there is only one thread of control. Execution of the program proceeds sequentially, and, at any given time, there is only one point in the program that is currently executing.

#### 1.9.3.2 DCE Remote Procedure call

RPC gives programmer's the ability to express an interaction between the client and server of a distributed application as if it were a procedure call.

In RPC, as in a traditional local procedure call, control is passed from one code segment to another, and then returns to the original segment.



### 1.9.3.3 DCE Directory Service

- When a directory service is available, it is no longer necessary to maintain local copies of information such as database of user, hosts and server locations.

### 1.9.3.4 DCE Distributed Time Service

- DTS provides a way to periodically synchronize the clocks on the different hosts in a distributed system.
- DTS also provides a way of keeping that synchronized notion of time reasonably close to the current time.

### 1.9.3.5 DCE Security Service It comprises of several parts

- Authentication service • privilege service • registry service • ACL failure • Login facility
- Password strength service
- Audit service

These are the different parts of security.



### 9.3.6 DCE Distributed file Service

- A distributed file system is an example of an application that can take advantage of all of these aspects of a distributed system.
- DCE DFS is a distributed client/server application built on the underlying DCE services. It takes full advantage of the cluster-level DCE components.

### 9.4 DCE cells

The largest unit of management in DCE is a cell. The highest privileges within a cell are assigned to a role called cell administrator.

Major cells of DCE within an area are:

- a) The security server that is responsible for authentication.
- b) The cell directory server (CDS) that is the repository of resources and ACL.
- c) The distributed time server that provides an accurate clock for proper functioning of the entire cell.



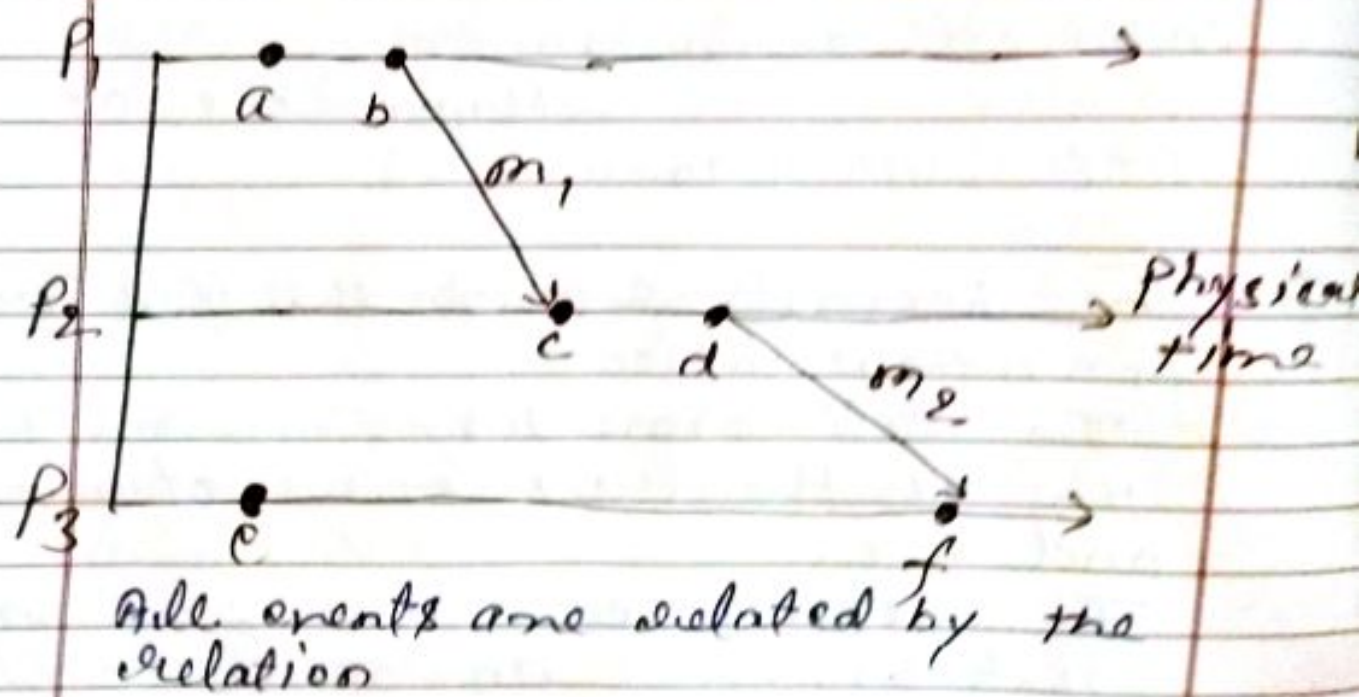
1.10

# Theoretical issues in distributed systems

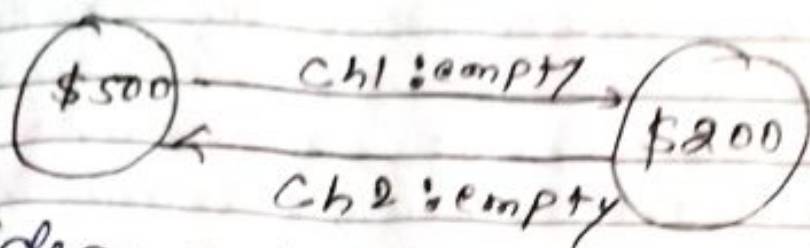
1.10.1

## Notations of Time and State

- If two events occurred at the same process  $P_i$  ( $i = 1, 2, \dots, n$ ) then they occurred in the order in which  $P_i$  observes them.
- Whenever a message is sent between processes, the event of sending the message occurred before the event of receiving the message.







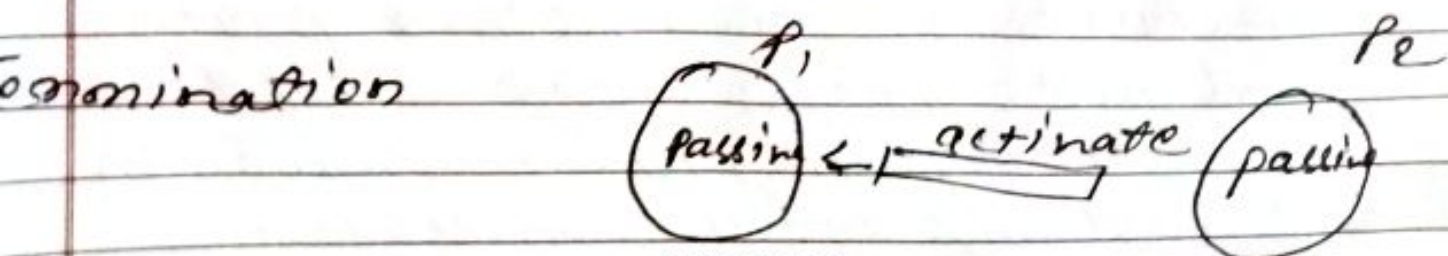
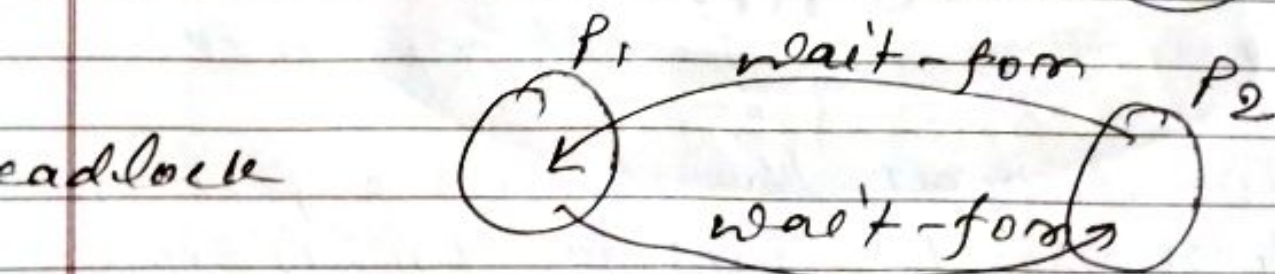
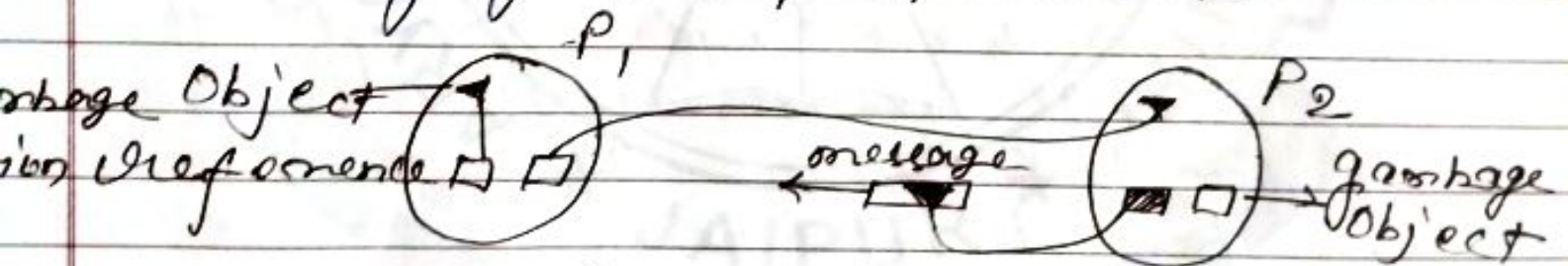
consider a bank system with two accounts A and B at two different site; need transfer \$50 between A and B.

## 2. States and events in a distributed system

### 2.1 Global and Local state

Local state of each process, state about messages each process has sent or received.

Detecting global properties:-





- A distributed deadlock occurs when each of a collection of processes wait for another process to send it a message, and where there is a cycle in the graph of this "waits-for" relationship.
- The problem here is to detect that a distributed algorithm has terminated. Detecting termination is a problem that sounds deceptively easy to solve.

### 1.10.2.2 Events

We can use a system of clocks satisfying the clock condition to place a total ordering on the set of all system events.

We simply order the events by the times clocks with Jules IR1 and IR2, and use them to define a total ordering  $\Rightarrow$  of all events.

1. To request the resource, process  $P_i$  sends the message  $T_{on}: P_i \text{ request resource to every other process, and puts the message on its request queue, where } T_{on} \text{ is the timestamp of the message.}$



- 2) When process  $P_j$  receives the message  $T_{m_i}$ ,  $P_i$  re-gets resource, it places it at its request queue and sends acknowledgment message to  $P_i$ .
- 3) To release resource, process  $P_i$  removes any  $T_{m_i}$   $P_i$  requests resource message from its request queue and sends  $P_i$  releases resource message to every other process.
- 4) When process  $P_j$  receives a  $P_i$  releases resource message, it removes any  $T_{m_i}$   $P_i$  requests resource message from its request queue.
- 5) Process  $P_i$  is granted the resource when the following two conditions are satisfied
  - i) There is a  $T_{m_i}$   $P_i$  requests resource message in its queue by the relation  $\Rightarrow$
  - ii)  $P_i$  has received a message from every other process time-stamped later than  $T_{m_i}$ .



1.10.3 Time, clock and Event Precedence  
The concept of one event happening before another in a distributed system is examined, and is shown to define a partial ordering of the events.

- A distributed algorithm is given for synchronizing a system of logical clocks which can be used to totally order the events.

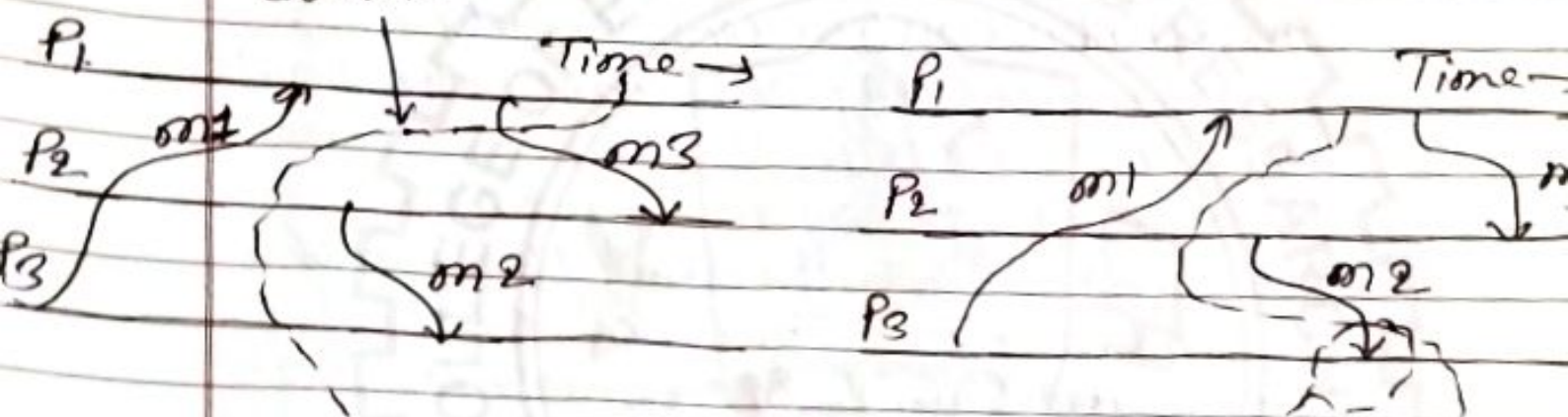
### 1.10.3.1 Global and local clock

- Logical clock is invented by Lamport a single mechanism by which the happened-before ordering can be captured numerically, called a logical clock.
- LC1:  $L_i$  is incremented before each event is issued at process  $P_i$ :  $L_i := L_i + 1$
- LC2: When a process  $P_i$  sends a message  $m$ , it piggybacks on  $m$  the value  $t = L_i$ .



(b) On receiving  $(m, t)$ , a process  $p_j$  computes  $L_j := \max(L_j, t)$  and, then applies LCT before timestamping the event receive( $m$ ).

A distributed snapshot should reflect a consistent state:  
Consistent cut



(a)

Sender of  $m_2$  cannot be identified with this cut (b)

### 10.3.2 Event Precedence

(1) We define a causal relationship between synchronous message based on the Lamport happened before relation on events. We present an online algorithm to timestamp message using a vector of size less than  $n$ . We prove that the vector timestamps accurately capture the order relationship b/w messages.



b) Using timestamp assigned to messages, we align timestamp to all events in the computation. Our timestamps for events use only few additional integers than timestamps for messages. We also show that, similar to Fidge/Mattern's timestamps, our timestamp can be used to test for precedence between any two events in  $O(1)$  time.

c) We have to prove that the problem of computing an optimal edge decomposition is NP-complete in general. We present a heuristic algorithm decomposition is at most twice the size of an optimal edge decomposition.

d) We have to show that the vector size  $\lceil N/2 \rceil$  is sufficient to capture relationship between synchronous message using an offline algorithm.

e) We show that, for every  $N \geq 2$ , there exists a synchronously ordered computation on  $N$  processes  $2\lceil N/6 \rceil$  entries in the vector.



#### 4. Recording the state of distributed system

The snapshot algorithm selects a cut from the history of execution. The cut, and therefore the state recorded by this algorithm, is consistent.

To see this let  $e_i$  and  $e_j$  be events occurring at  $P_i$  and  $P_j$  respectively such that  $e_i \rightarrow e_j$ . We assert that if  $e_j$  is in the cut then  $e_i$  is in the cut.

We shall find a permutation of  $Sys$ ,  $Sys' = e'_0, e'_1, e'_2, \dots$  such that all three states  $S_{init}$ ,  $S_{snap}$  and  $S_{final}$  occur in  $Sys'$ .

$S_{snap}$  is reachable from  $S_{init}$  in  $Sys'$  and  $S_{final}$  reachable from  $S_{snap}$  from  $S_{snap}$  to  $Sys'$ .

We derive  $Sys'$  from  $Sys$  by first categorizing all events in  $Sys$  as pre-snap events and post-snap events.



- We shall show how we may order all pre-snap events before post-snap events to obtain  $Sys'$ . Suppose that  $e_j$  is a post-snap event at one process, and  $e_{j+1}$  is a pre-snap event at a different process.

- A marker message would have to have preceded the message, making the reception of the message a post-snap event, but by assumption  $e_{j+1}$  is a pre-snap event.

