LECTURE NOTES

ON

DISTRIBUTED SYSTEMS
ACADEMIC YEAR 2021-22

IV B.Tech.–II SEMESTER (R16)

G.K.HAVILAH, Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

V S M COLLEGE OF ENGINEERING
RAMCHANDRAPURAM
E.G DISTRICT
533255

IV Year – II Semester

L	T	P	C
4	0	0	3

DISTRIBUTED SYSTEMS

OBJECTIVES:

- Provides an introduction to the fundamentals of distributed computer systems, assuming the availability of facilities for data transmission, IPC mechanisms in distributed systems, Remote procedure calls.
- Expose students to current technology used to build architectures to enhance distributedComputing infrastructures with various computing principles

UNIT-I:

Characterization of Distributed Systems: Introduction, Examples of Distributed Systems, Resource Sharing and the Web, Challenges.

System Models: Introduction, Architectural Models- Software Layers, System Architecture, Variations, Interface and Objects, Design Requirements for Distributed Architectures, Fundamental Models- Interaction Model, Failure Model, Security Model.

UNIT-II:

Interprocess Communication: Introduction, The API for the Internet Protocols- The Characteristics of Interprocess communication, Sockets, UDP Datagram Communication, TCP Stream Communication; External Data Representation and Marshalling; Client Server Communication; Group Communication- IP Multicast- an implementation of group communication, Reliability and Ordering of Multicast.

UNIT-III:

Distributed Objects and Remote Invocation: Introduction, Communication between Distributed Objects- Object Model, Distributed Object Modal, Design Issues for RMI, Implementation of RMI, Distributed Garbage Collection; Remote Procedure Call, Events and Notifications, Case Study: JAVA RMI

UNIT-IV:

Operating System Support: Introduction, The Operating System Layer, Protection, Processes and Threads –Address Space, Creation of a New Process, Threads.

UNIT-V:

Distributed File Systems: Introduction, File Service Architecture; Peer-to-Peer Systems: Introduction, Napster and its Legacy, Peer-to-Peer Middleware, Routing Overlays.

Coordination and Agreement: Introduction, Distributed Mutual Exclusion, Elections, Multicast Communication.

UNIT-VI:

Transactions & Replications: Introduction, System Model and Group Communication, Concurrency Control in Distributed Transactions, Distributed Dead Locks, Transaction Recovery; Replication-Introduction, Passive (Primary) Replication, Active Replication.

OUTCOMES:

- Develop a familiarity with distributed file systems.
- Describe important characteristics of distributed systems and the salient architecturalfeatures of such systems.
- Describe the features and applications of important standard protocols which are used indistributed systems.
- Gaining practical experience of inter-process communication in a distributed environment

TEXT BOOKS:

1. Ajay D Kshemkalyani, MukeshSighal, "Distributed Computing, Principles, Algorithmsand Systems", Cambridge
2. George Coulouris, Jean Dollimore, Tim Kindberg, "Distributed Systems-Concepts andDesign", Fourth Edition, Pearson Publication

REFERENCE BOOKS

1. Distributed-Systems-Principles-Paradigms-Tanenbaum PHI

CHARACTERIZATION OF DISTRIBUTED SYSTEMS

* Introduction of Distributed System:-

Distributed System:- A distributed System is one in which the hardware and Software Components that are located at network computer are able to Communicate and also coordinate their actions by passing messages.

Significant Consequences:-

(i) **Concurrency:-** The connection of network of computers is the Concurrent execution. more than one user can work on their System at a time by sharing resources such as webpage or files. By adding more resources to the network, its capa of handling shared resources can be increa -sed.

(ii) **No Global clock:-** The programs that Coord -ate their action by menage exchanging When they need to cooperate. And this Coordination may depend upon the idea of time at which the actions of progra occur. But the accuracy with which the clocks of the computers in a network can be Synchronized may have limits. i.e there is no global nation of the correct time. This is a Consequence about t

that sending messages through the network is the communication.

(iii) Independent failures :- Any computer system may fail, and hence it is the system designers responsibility to plan for such consequences of possible features the computers that are connected to network can be seperated as a result of failures but they might not stop running. The programs running on them may not know that about the failure or slowness of the network. Similarly, the failure or program termination of computer may not be known to the other computer with which communicates. Hence, the other systems can still running even if one fails

* Examples of Distributed Systems :-

(i) Internet :-

In simple terms, internet is a network of network that consists of millions of privat, public, academic busines and government networks connected to each other with the help of widely available internet working devices such as routes, gateways, bridges etc.,

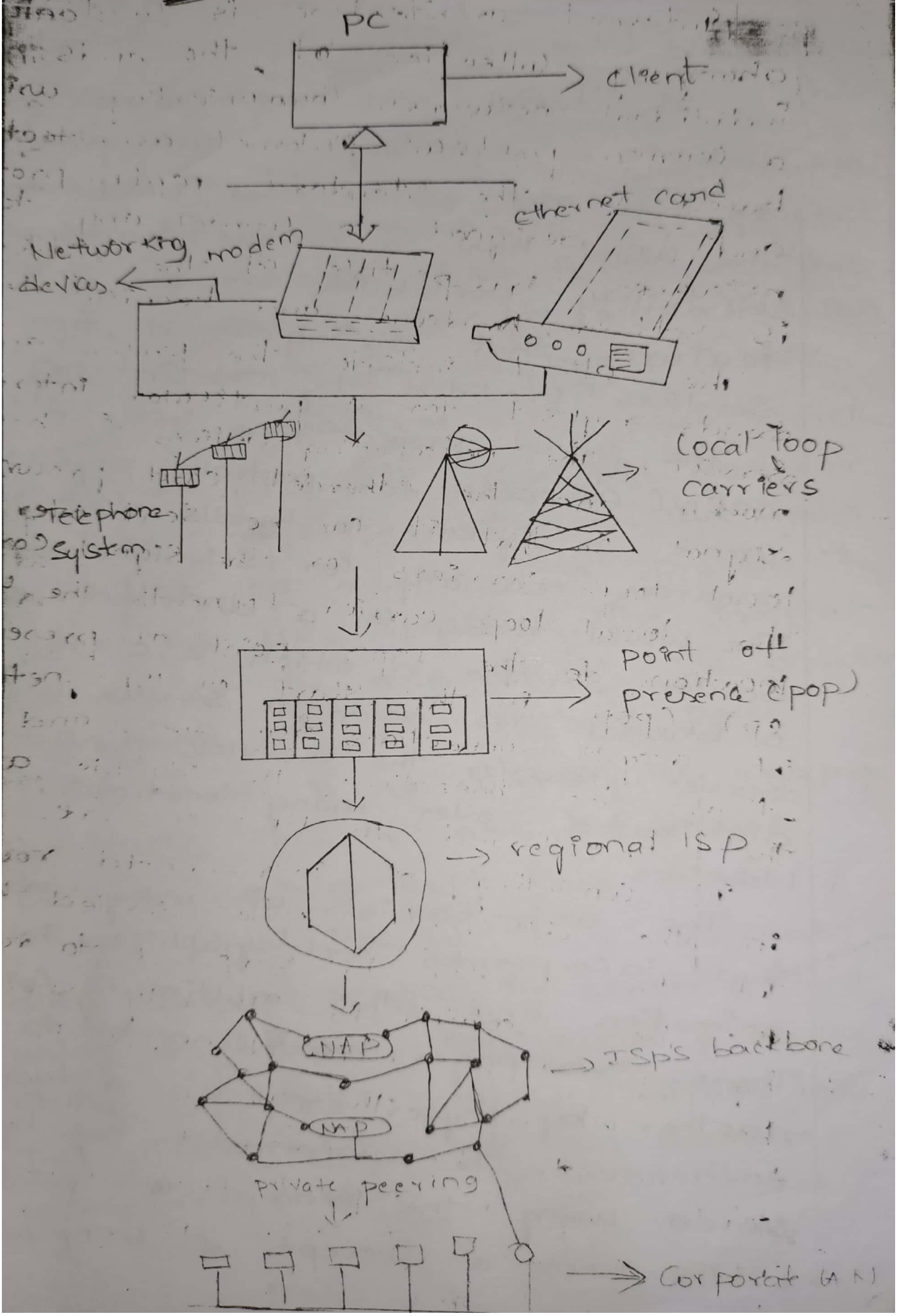
Architecture of Internet:-

Internet architecture is constanly changing collection of thousands of individual networks Communicating using a common protocol. Internet architecture bared on the standard TCP/IP protocol and is designed to connect any two networks, irrespective of the difference is software, hardware and technical design.

The client initiates the process and sends request for a particular internet service. The networking devices such as the modem and the ethernet card process, the signal so that it can be sent over the local loop carriers or the signal carriers. The local loop carrier connects the user location to the ISP's point of presence (POP), (POP is the start of isp's network) it accepts connections from users and authenticates the connections. The signals are then transferred to isp's network.

It consisting of interconnectd routers in the different cities served by ISP. If the packet is supposed to reach the host, which is directly served by ISP, then it is delivered to the host. otherwise the packet is forwaded to reach the host which is directly served by ISP, ISP backbone operator. It interconnects the ISP's, POP's and also to other ISP's.

fig :- Internet architecture



If the packet received by the bac-one is supposed to reach on ISP or company served by backbone then the packet is forward to the nearest router.

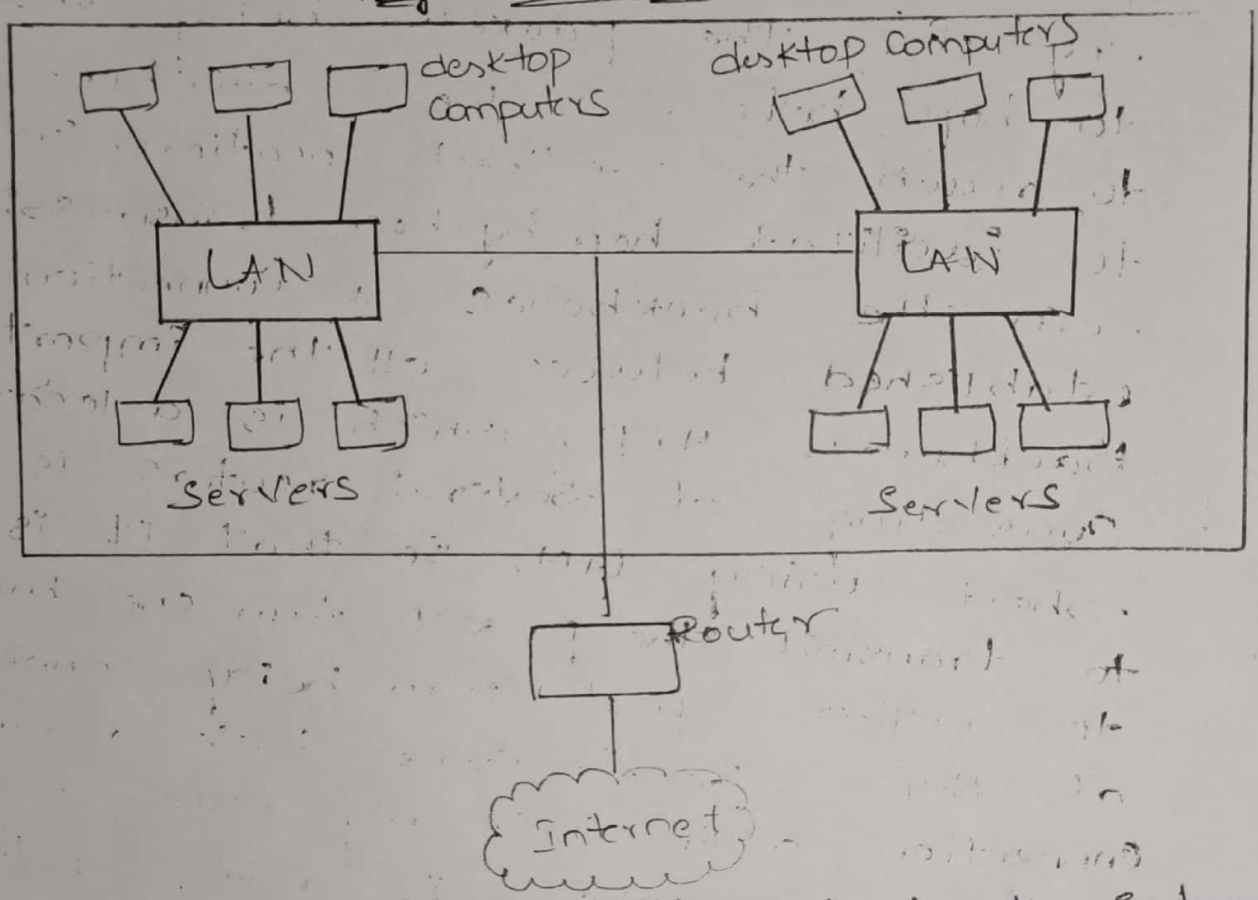
Since, there are many backbones present in the network, it is necessa-ry for the packets being transmitted to hop between these backbones so as to reach the destined location. In order to facilitate hop-by-hop transmission betw-een the backbones, a connection is established between all the important backbone's NAP, which is a location consisting of several routers is establi-shed using LAN, so that it is possible to transmit the packet from one backbone to another. Apart from being connected at NAP's, the backbones have a direct connection with their respective router(s). Such a connection is refferred to as "private peering".

(ii) Intranets:-

A portion of an internet that can be administered seperately and which has boundary that is configured to impose security policies is known as intranet. Several 'LANS' are linked in it backbone connections. The organization which

administers the intranet is responsible for its network configuration and also vary from LAN on a single site to a set of connected LAN's of several branches of companies across different countries.

fig :- Intranet



An Intranet is connected to the internet through a router that allows the users of the intranet to use the services such as web or email. The services that it provides can also be accessed by the users of other intranets.

* Resource Sharing and the web:-

World Wide Web (WWW):-

The "world wide web (www) or the web, is a repository of information, spread all over the world and linked together. The WWW has a unique combination of flexibility, portability and user friendly features that distinguish it from other services provided by the Internet.

The WWW project was initiated (by CERN (European for particle physics) to creat a system to handle distributed resources necessary for scientific research. The WWW today is a distributed client server service in which a client using a browser can access a service using a server. However the service provided is distributed over many locations called websites.

The web consists of many webpages that incorporates text, graphics, sound, animation and other multimedia components. These web pages are connected to one another by hyper text. In a hyper text environment the information is stored using the concept of pointers.

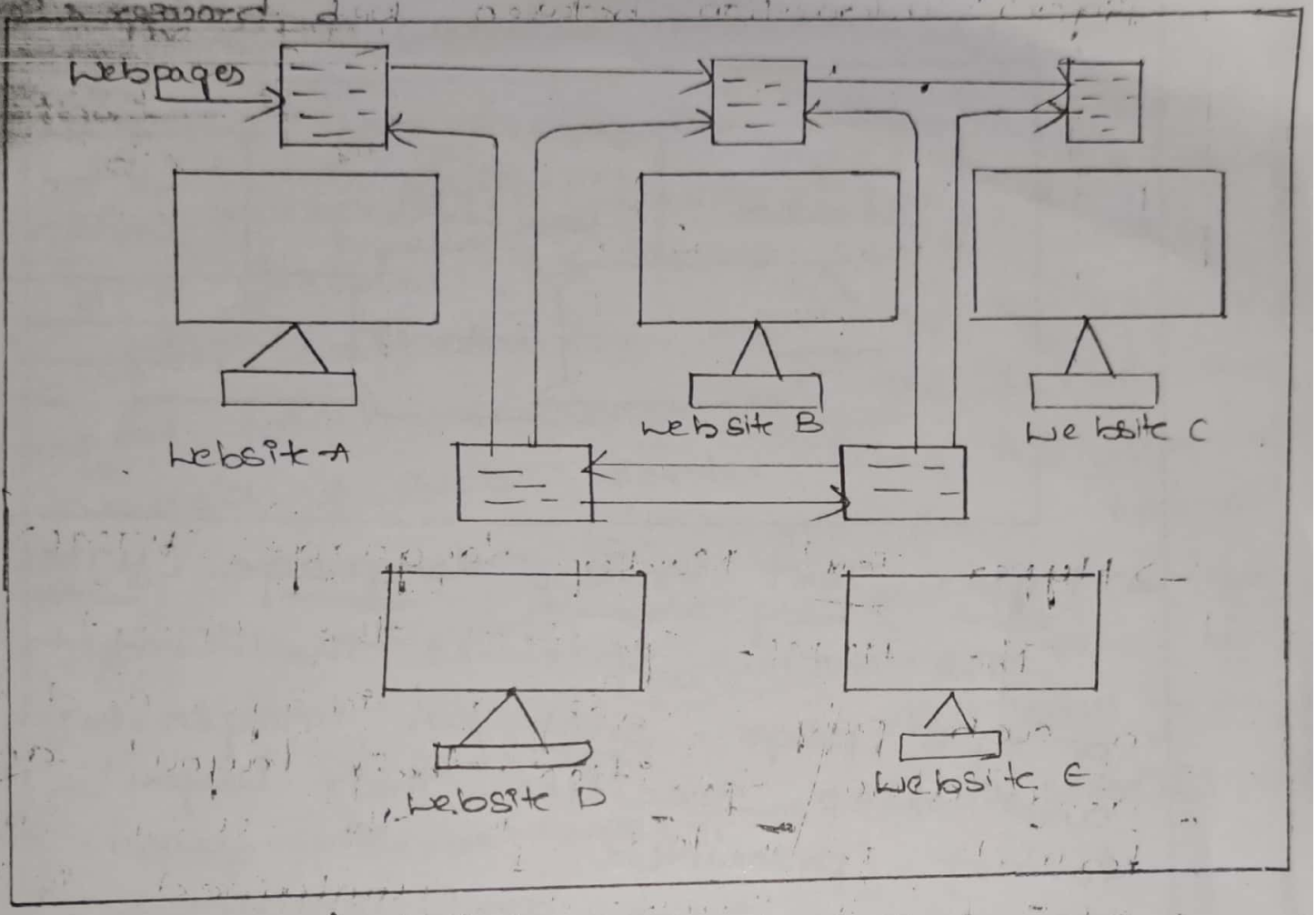
- WWW uses a concept of HTTP which

allows to communicate between web browser and webserver. The web pages can be created by using a HTML. This language has some commands which are used to inform the browser about the way of displaying the text, graphics and multimedia files. HTML also has some commands through which we can give links to the web pages.

If we want to get a page from the web, we have to type URL for our desired pages, or otherwise we have to click on a link that provides the URL. The URL specifies the internet address of the web server, the directory and name of our desired page.

If there is no directory or web page specified, then the web server will provide a default home page.

The WWW today is a distributed client-server service in which a client using a browser can access a service using a server. fig (1) illustrates how the different web site can communicate with each other.

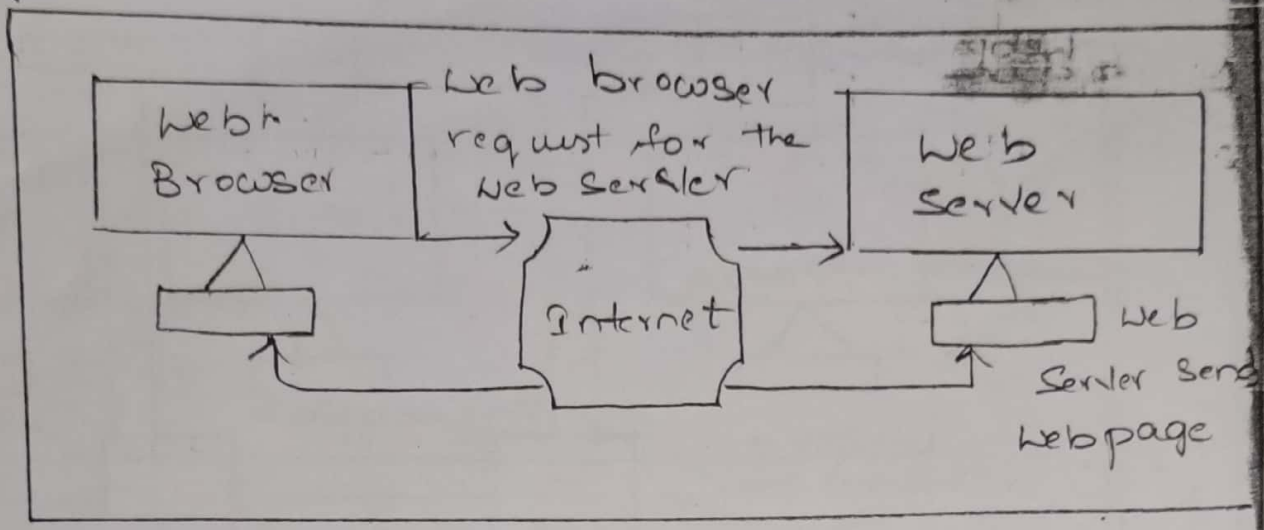


Working of Web :-

The web operates on a client/server model. A web browser acts as the client, in the WWW interaction. using this program, a user sends a request for a web page stored on a web server. The web server locate this web page and sends it back to the client computer. The web browser then interprets the web page written in the HTML language and then displays it on the client computer's screen.

The

figure: Interaction between web browser & server



-> Hyper Text Markup Language (HTML):-

The HTML specifies the contents of a web page such as images and text and also specifies their layout and format presented to the user. Such structured items are contained in the web pages as headings, paragraphs, tables and images. The links and its associated resources are also specified by the HTML.

The HTML can be produced by the users either by using standard text editor or by using an HTML-aware 'wysiwyg' editor that generates the HTML from the layout created by them. A typical HTML text would be as created by them. A typical HTML text would be as

```
<IMG SRC="http:
```

The HTML text is stored in a file that is accessed by the web server and later returned read and rendered by the browser into the formatted and images laid on a web page. Th only the browser can interpret the HTML text. The server only informs th browser about the type of context i is returning, by inferring from file name extension in order to distingu from other text files.

All the HTML directives known as t are enclosed by angle brackets. The text between these tags is as it presented on the web page of the link. The browser are also configure to show the text of the underlined links by default in paragraphs to th user:

if interested also have, a look o Company details, the URL contained in A

The association between the link displayed text and URL contained in <A HREF ...> tag. when clicked, on the underlined text. The resource which is identified by the URL is identified and presented to user

for example, a web page about the company details specified by the resource HTML file.

→ Uniform Resource Locator (URL):-

The URL is the address of the system on the internet. It identifies the resources and hence can be examined by the browsers to access the corresponding resources. When the user clicks on a link or select one of its 'book marks', the browser looks up the respective URL or fetches a resource embedded in a web page such as an image or text. The URL has two top-level components.

Scheme: Scheme-specific-identifier.

The first component 'scheme' declares the type of the URL such as user's email, file to be retreived using ftp, http and mid.

The URLs can be used of the form widget and the browsers must be given the capability of using the new "widget" protocol by adding plug-in when a widget is invented with its own addressing scheme and protocol to locate and

access them.

The resources can also be accessed by using HTTP URLs by using standard HTTP protocol. The HTTP URL's identify which web server maintains the resources and which resources at the server are needed. A web server files are maintained in a one or more sub trees of its file system. every resource can be identified by a path name related to the server. The HTTP URL will be as,

http://servername[:port][:/path Name][?query [#fragment]

The items which are enclosed in bracket are optional. A full HTTP URL begins with 'http://' and the server name follows it. The server's name is expressed as Domain name system and the "port" number that follows it is optional through on which the request are retained by the server's and is so by default Then followed by path name of the server's resource is optional and in its absence the servers default web page is needed.

Atlast, the URL ends with a query component such as requesting a query page or a fragment identifier.

for example, Consider the below URL,

http://www.google.com/search?q=html elements

DNS name — path name — query

http://www.cms.org/Description/main.html#def — fragment

The resources can even be published on the web by a simple method by placing the correspond file in a directory accessible by the web server. And then the URL can be constructed as http://ip, after knowing the server's and path name of file p. This URL is placed in a link from an existing document or distributed to the other users.

-> HTTP (Hypertext transfer protocol):-

It defines the ways in which browsers and other types of client interact with web servers.

features of http:-

(i) Request Reply Interactions:- http is

request-reply protocol, the URL of the resource required is sent as a messag by the client to the Server. The Se sends back the file contents as a reply to the client if the path na exists or else it sends an error response such as "404 Not found".

(ii) Content Types:-

Every Content type may not be handle by the browser. So, it adds the Content types it prefers in the r for example, the browser may be applicable of displaying images of typ 'png' format. The Servers consider it and adds the Content type in the reply to make the browser understa how to procss it. The Content types are indicated by the strings called MIME types. The set of actions taken by the browser for a given type are Configurable

(iii) One Resource per Request:-

One Resource per request is specifie by the client. Consider for example a web page contains nine images, then totally ten requests are issued

by the browser in order to access the whole content of the page. Several request are made concurrently by the browser so that the overall delay to the user can be reduced.

(iv) Simple Access Control:-

Any user who have network Connecti -vity, to a web Server can access any of its resources by default. In case, if the user need to restrict access to any resource, then the Server can be configured to issue a challenges to the client who request it. the client can access the resource for example by typing a password

* Distributed System challenges:-

The construction of distributed systems produce the challenges that are given below,

(a) Heterogeneity:- The distributed system must constructed from different networks operating systems, Computer hardware and programming languages. The differer -ce in networks is marked by the internet Communication protocol and the

other differences are dealed by middle -ware.

(b) openness:- They must be extensible by publishing the interfaces of the components first; but the real challenges would be the integration of component that is written by different programmers.

(c) Security:- Adequat protection of shared resources can be provided by using the encryption and also the sensitive informa -tion can be maintained confidential when transmitted over the network. The problem is still caused by denial of service attacks

(d) Scalability:- If the cost of adding a user is constant in trms of resources that need to be added, then the distribut -d system is said to be scalable. The performance bottleneck should be avoided by the algorithms that are used to acce the shared data and this data must be structured heirarchically to obtain best access times. The data that is frequently accused can be replicated.

(e) failure Handling:- Any computer or network may fail independently without effecting others. Hence each of the component must be aware of possible

ways the components on which it depends fail and also must be designed in a way to be able to deal those failures.

(f) Concurrency :- multiple users in distributed System send concurrent request to its resources. Hence, each of it must be designed to be safe in such environment;

(g) Transparency :- The aim of transparency is to hide some aspects of the distributed System from the application program to make it available to the design of their particular application.

-> Heterogeneity :-

Internet allows users to access services over a variety of different networks it also allows to run applications over heterogeneous collection of Computers and networks.

Heterogeneity is applied to different networks, programming languages, OS Computer hardware, implementation by different developers.

Integers are presented in different

forms on different hardware for exam
there are two ways for byte orderin
of integers. when messages are exchange
on different hardware these differenc
of representation must be addressed
for example calls for exchanging
messages in windows is different
from calls in unix.

For characters and data structures
different programming languages use
different representation. Hence in order
to communicate with each other, these
differences must be dealt.

for working in different environmen
of network communication programmin
language common standards has to be
adapted by developers so that they
can easily understand and can easily
Communicate with each other using pro
-ms they haile written.

(a) Middle ware :-

Middleware refers to a software laye
whose purpose is to make heterogenity
and provide programming abstraction. For
example Common object request broken
Architecture (CORBA). difference of under
networks, operating system, Computer h/w
is dealt by middle ware by implement

internet protocols which masks these differences.

Java remote method invocation is a middleware which support single programming language

(b) mobile code and heterogeneity:-

mobile code is a term which applies to code that is sent from source computer to destination and runs at destination for example Java applets. Executable programs are specific to instruction set and host operating system. hence code which is suitable for running on one computer is not suitable for other. for example when an executable file is sent via email attachment by windows, macintosh users will not run on macos or x86 users will not run on x86 computer running Linux operating system.

In virtual machine approach Compiler generates code for virtual machine instead of particular hardware. for example java Compiler generates code for virtual machine which is implemented once in each type of hardware to allow java programs to run. however this solution is not possible with other programming languages.

→ openness:-

openness can be defined as a feature which determines whether the system can be re-implemented and extended in different ways. openness of distributed system can be known by the level at which new resource sharing can be included and made accessible for use by client programs.

openness cannot be obtained until slw developers are provided with documentation and specification of the key interface of the ports of system. In other words, until the key is published. This process is similar to standardizations of key interfaces but it usually ignore those standardizations because they are slow and complicated.

The internet protocol designers introduced series of specification and documents, series of documents are called as "Request for Comments" each of them is determined by a number Specifications for internet protocols were published in the year 1980's which followed specifications for applications that run over them like email, telnet & file transfer

The process of documentation and specificat
-on has been continued and copies of
specification as well as discussion be obtain
-ed from www.ietf.org. for example

CORBA publishes documents and specifications of the interfaces of its services which can be obtained from www.omg.org.

System that are designed to serve resource Sharing in this way are called open distributed System which are extendable at hardware level they are extended by including more number of computers attached to network and at the network and at the software level they can be extended by re-implementation of old ones and introduction of new services and by allowing applications to share resources.

→ Security :

Information resources that are of high value to users is maintained in distributed Systems and their security is of high importance.

Security for information resources has three parts :- 1. Confidentiality
2. Availability
3. Integrity.

Security risks increases when one program in a Computer interacts with a program on another Computer because it allows sharing of resources freely in an intranet. firewall can be used to stop unauthorized

users from entering and gaining access to important data in internet.

But firewall do not ensure about the proper use of available resource in an intranet or internet.

Clients sends request to access data or is maintained by server in distributed systems and in reply server sends information via message over a network.

For example, a bank manager might request to access list of new customers from server. In this the difficult task is to send the details of customers like account number address in a message over a network in a secure manner. Hiding the contents of message is not just important even the identity of the bank manager should be checked and assured.

Some of the security challenges which has not been met fully are,

(i) Denial of Service
(ii) Security of mobile code.

(i) Denial of Service:- Some users try to disrupt service of target sites or services hosted on high profile web servers. Such as banks, offices by bombarding large number of pointless requests so that

Service becomes unavailable to the serious users of the site, that internet site stops functioning efficiently this is called denial of service.

(ii) Security of mobile code:- mobile code has to be tackled very carefully. If a users receives an email attachment after loading that email locally on the computer and when it is being executed effects may be unpredictable because from above it may look a beautiful picture but in real without realizing we are being a part of denial of service attack.

-> Scalability:-

Distributed systems works effectively and efficiently at different scales ranging from intranet to internet.

A distributed system can be defined as Scalable only if it works effectively and efficiently with an increase in the number of resources and users.

During the last 29 years there is a dramatic increase in the number of computers and services from 1993-2003 there is significant increases in the no. of web servers and computers.

Some of the challenges encountered while designing scalable distributed

Systems are,

1. Controlling physical resources Cost
2. Controlling the loss in performance
3. preventing running out of SLW resources
4. Avoiding performance delays

1. Controlling physical resources Cost:- The Cost of physical resources should be reasonable as the demand increases and should be possible to extend the system.

for example as number of user and Compu ter increases the frequency at which files are being accused also increases in an intranet Server Computer should be possibly added to avoid performance delays, when a single Server has to tackle n number of file system with N number of users to be scalable there should be atmost O(n) physical resource to support them.

for ex, if there are 20 users and a single server supports them, such 2 servers then should be capable of supporting 40 users.

(2) Controlling the loss in performance:-

Let assume a data set whose size is equal to number of users in the system. for example, the table with domain names of Computer and then IP addresses held

by domain name System, which is used for looking up domain name like www.siag-ups.com heirarchic structures scales better in algorithms than linear structures. But increased size in heirarchic structures will cause loss in performance. maximum performance loss should not be less than O(log)n.

(3) Preventing Running out of SW Resources:-

In 1970's 32 bits was used for assigning the numbers as Internet (IP) address but unavailability of numbers to assign as IP addresses have become the problem. To overcome this problem 128-bits addresses is used, there is no solution for this problem correctly as said by early designers lot of changes will be required on SW components in order to make them work in 128 bits.

(4) Avoiding performance Delays:-

To avoid performance delay, algorithms should be decentralized. for example, in domain name System in which the name table was kept in a single master file and can be downloaded whenever needed this was not a problems until there were only few hundred Computers that

needed it. (There was a delay in performance and administrative delays)

But problems close when there was a dramatic increase in the number of Computer which needed it. There was a delay in performance and administrative delays. This problem was solved by separating the name table between Servlers and admistered locally.

The resources which are frequently used by users causes loss in performance. To overcome this problem catching and replication is used which improves performance of resources that are heavily used by users. Thee need for modifications in system and software applications should be there even when the demand increases.

→ Techniques for dealing with failures:-

When a failure occurs in hardware or software proper work of systems fail which leads to wrong results and system may stop working before the completion of Computation.

There are different types of failure which occurs over the network and in the processes in distributed systems. In distributed systems, failures are partial

which means some parts continues to work while the others fail hence it is difficult to tackle failures

(1) failure detection:- Some failures can easily be detected. for example: By using checksum corrupted data in a file or message can be detected. Some of the failures are impossible to detect like remote crashed server.

(2) Hidden Failures:- Some detected failures can be hidden, for example (i) when a messages get fail to reach the destination it can be retransmitted

(ii) A copy of file data is created, on disks, if one gets, corrupted other will be correct.

In worst cases, hiding failures techniques are not guaranteed, which means copy of data on the other file may also get corrupted. the message may not reach the desired destination how often they may be transmitted

(3) Tolerating failures:- most of the internet services show failures and it will not be practical to detect and hide the failures in such a large network with so many parts

for example, when a web browser tries to connect to web server and it doesn't gets connected, the user has to tolerate the delay in service

(4) Recovery from failures:- In recovery, the data can be rolled back or recovered after a server has been crashed. In some programs, a fault occurs when computations performed are incomplete and not consistant.

(5) Redundancy:- Some services can be tolerated by using redundant components

Every name table in domain name system should be copied in at-least two server there should be at-least two routes between two router

To ensure that data is accessible even after a failure db should be copied in different servers, when a fault is detected in a server clients are redirected to other servers without loss in performance the data has to be updated by creating copies of the data

=> Constructing Distributed System:-

In distributed systems, clients can share the resources provide by services and

applications. Hence, many clients will try to access the shared resources at the same time. for example, when applying for online scholarship and when deadline comes close, many clients will access frequently at the same time.

A process allows a shared resources to take one client request at a time but this limits the throughput. hence services and applications allow many client requests concurrently. In order to make it more robust each resource is encapsulated as an object and are executed concurrently. In this case, many threads can be executed concurrently within an object

In other cases, executing several threads concurrently. operations on object may conflict and as a result it may produce inconsistent results.

for example, in a bank when two concurrent transaction is taking place with account number 502184, 502185 502184 is transferring 50,000 to some other account and 502185 has received 1 lakh rupees. if the corresponding

-ding operations are interleaved without any control then they will become inconsistent

In distributed system, a shared resour must ensure operations correctly in a Concurrent. Therefore, a programmer shou implement operations an objects and server in a safe way and should not be inconsistent

By using standard techniques like Semaphores, operations on objects in a Concurrent environment can be safe a Consistent data can be achieved

-> Transparency :- It refers to component Se -ration details of distributed system f the user and the application programm This is done in order to deploy the system as a whole rather than deployin it into individual components

There different eight forms of transp -arency are,

(1) performance Transparency :- This type of transparency allows system reconfigur -on so as to increase the performa with change in loads.

(2) Scalability Transparency :- This type of transparency allows extension of the System and application without effecting the System structure or the application algorithms

(3) Failure Transparency :- This type of transparency allows the hides the faults and failures of the System hardware and software components allowing the users and application programs to finish their tasks without any hindrance

SYSTEM MODELS

Architectural Models:-

The Structure is the architectural of a system. It is a Collection of individually defined Components the Complete objectiyle is to make sure that the structure will satisfy the current and future demands on it. The vital reponsibilities are to make the System cost effective, manageable, relia -ble and adaptable. The architectural design of building has some features like determin -ng architectural style as well as its genera -ted structure. The architectural styles gives a constant frame of reference for the design.

The central architectural models are Constructed almost on the nation of proces and object the functioning of seperate Comp -onents of a distributed system is first simplified and then abstracted in an archit -ctural model. This model later evlaluates the following things.

(a) The arrangement of Components throughout a network of computers i,e trying to specify beneficial patterns for the distribution of data and workload.

(b) The interrelationships between the compon -cnts i,e the Components functional roles and the pattern of Communication.

The processes are divided into server processor, peer processes and client processes in initial simplification. In the latter simplification, the process that invite, unite and communicate in a proportional way are involved to perform a task. Through this division, processes recognizes their concerns which is helpful in evaluating their workloads. It also makes easy to decide the effect of failures, in each process. To achieve the performance and reliability objectives for the resulting system, the arrangement of processes is important. This arrangement can be identified through the above analysis.

By altering client server model, one can construct new systems.

(a) A process can assign a task to another process since it is possible the code from one process to another.

Example:- The code from servers is downloaded by the client process and executed locally. To minimize access delays and communication traffic. The code and objects that accesses, the process can be transferred

Most of the distributed systems are designed to allow the computers and other mobile devices to be added or removed smoothly, enabling to find, obtain services and also to provide their services to others.

There are many patterns that are used fully for the allocation of work in a distributed system. These patterns have a major effect on the performance of the resulting system.

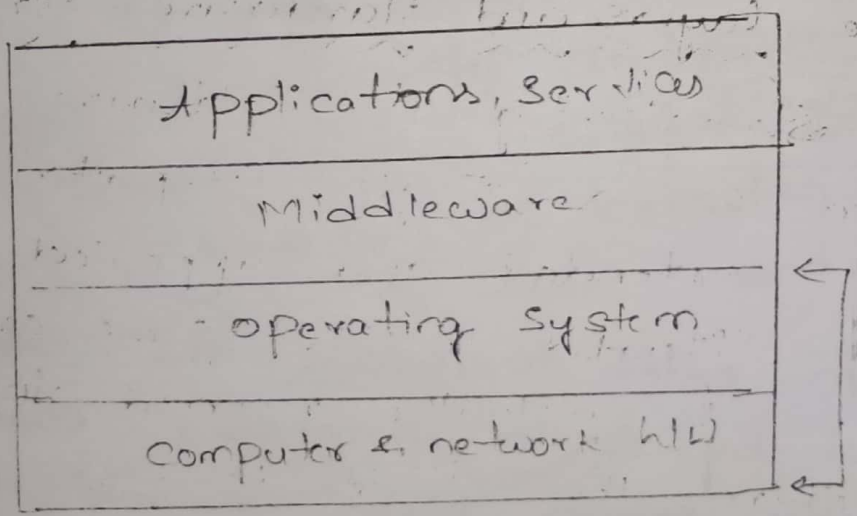
The particular issues of performance reliability cost and security influences the arrangement of the processes in a network of computers. The above discussed architectural models give only a simple view of major patterns of distribution.

* Software layers and hardware service layer

The structuring of software as modules or layers in a single computer and the services extended and appealed between processes situated in the same or different computers is actually referred to as software architecture. The disclosed form of this process and service oriented view is a layer of services in distributed systems. This layer of services included both software and hardware service. The process which accepts request from other processes is a

a server : one or more servers can supply a distributed service to support a consistent system wide view of the service resources. these servers can communicate with one another and also with client processes.

As an example, consider a network time service which is carried out by server processes on the internet located on the network time protocol (NTP). These server processes executing on hosts receives the request of client processes on the internet are provide current time to them. as a result of interaction with each other, these processes alter their version of current time.



Software & hardware service layers

1. Platform :-

The bottom layers (operating system, Compu
-ter and network hardware layers) serve
as platform for distributed applications. The
services provided by these bottom layers is
utilized by higher layers. In every computer,
operating system layer and computer network
layer are implemented separately. This implem
-entation facilities communication and co-ordina
-tion between process by transferring the
system's programming interface to that level.
Important examples of platform include Intel
x86/Linux, Intel x86/solaries, Intel x86/windows
power pc/macos.

2. Middle ware :-

A piece of software placed between
the application and operating system is called
"middleware". Its intension is to enclose
hetrogeneity and to supply an accessible
model to application programmers. The concept
of middleware can be easily explained in the
context of objects or process in a set of
computers that interact with each other to
carryout communication and resource sharing
support for distributed application. Middlewa
-re supplies helful building block in the constr
-uction of software components. These

Components can interact in distributed applications. In specific, it maintains abstractions to increase the level of communication activities in a distributed system. These abstractions include the transfer of multimedia data in real time, the partiting, interactions between a group of processes, placement and extraction of shared data objects amongst operation systems, Remote method invocation (RMI), making multiple copies of shared data objects and notification of events.

The earlist instances of middleware include remote procedure calling packages (eg: Sun RPC) and group communication systems (eg: ISIS) Object-oriented middleware products and standards are:

(a) CORBA
(b) java RMI
(c) microsoft's Distributed component object model (DCOM)
(d) Web Services
(e) The ISO/ITU-T's Reference model for open distributed processing (RM-ODP)

The services provided by middleware products and standards such as infrastructural services can be used by applications programs. As an example, consider the services offered by CORBA. They are security, presistent storage, transactions, naming and event

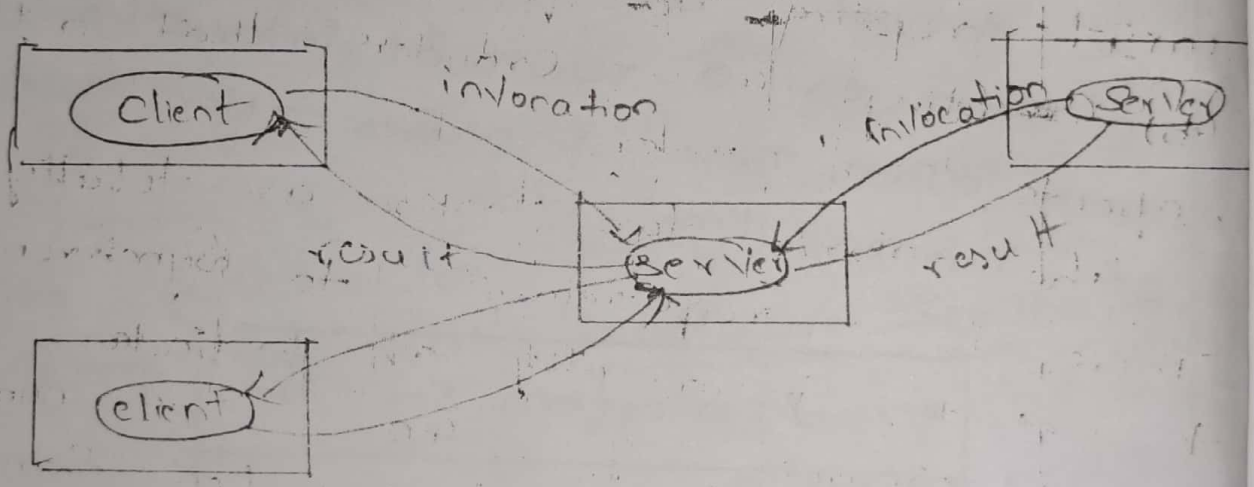
notification the upper layer provides domain specific services

System Architecture Models:-

(i) Client-Server Model:-

When dealing with distributed systems, the architecture very frequently referred to is client-server. This architecture is broadly utilized and significantly very important. The figure below shows the structure of client server model.

fig (i) Service provided by multiple servers.



In order to access the shared resources managed by server processes. Example, the local file server that controls the files consisting webpages has a common client called the files web servers and many others has many clients such as web servers and many other internet services search engines is other example needed to web

It allows users to search the sites all over the internet for synopsis of information accesible on web pages. These synopsis are created by programs called web crawlers. At search engine site, these web crawlers are executed in the background, using http requests to access web servers all over the internet. A search engine thus plays the role of both a server and a client.

It performs the following two tasks.

(a) It replies to queries from browser client

(b) Executing web crawlers that act as clients web server.

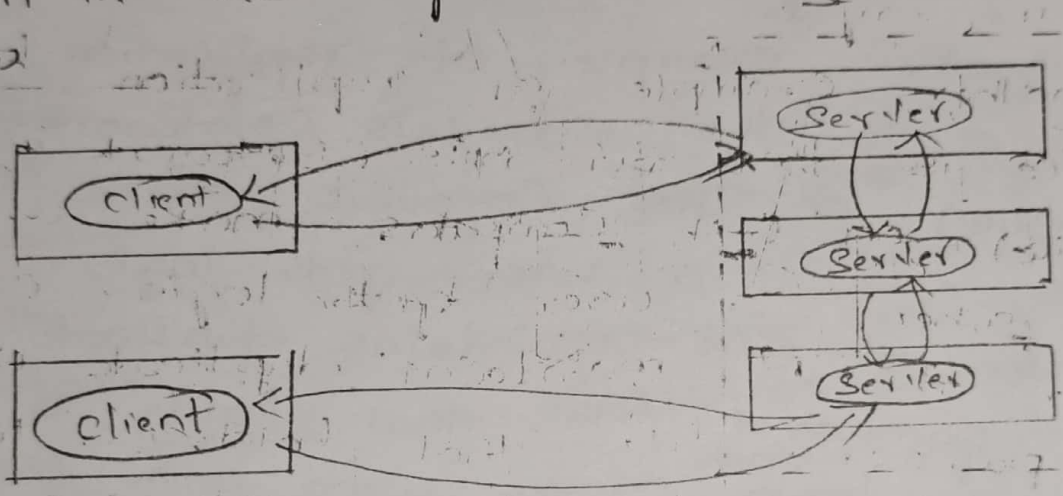
The above two tasks are totally separat There is a requirement to synchronize them to execute concurrently. In particular, a normal search engine consists of numerous concurrent threads of execution. Some threa provide servers to client and others execut web crawlers.

(ii) Multiple server model :-

In multiple server model, several server implement the services as its procuses in seperate host computers, after implementation these computers interact with each other if required, in order to provide a

Service to client processes. The Servers may also provide a service to client processes. The Servers may also provide the service either by partitioning the services into set of objects and distributing them among themselves or by maintaining replicate copies of these services on several hosts. The Scenario of multiple Server model is shown in the figure (2).

fig (2)



Clients Invoke Individual servers' multiple

The concepts of this model are illustrated in the following examples.

1. On web each Server maintains their own resources, called partitioned data. Therefore a user can access a resource of any one Server

2. Replication is an important Concept of increasing availability and performance

2. and for improving fault tolerance. the proces that run on different computers can have multiple consistent copies of data due to replication. for an instance, data avlailable at mittal, elec.com might also be provided by some other web services. It is due to mapping of service into several servers maintaining database replicated in memory.

3. Another example of replication based service is the Sun NIS (Network information service). on LAN, Computers make use of NIS when a user gets login. each server of NIS maintains duplicate copies of password file that holds a list of user's login names along with encrypted passwords.

→ Proxy Servlers and Caches :-

A cache is a repository. The newly used data objects are stored in a cache. Whenever a new object is Colleetd at a computer, it is intgrated to the cache store. Some items, this integration requires restoring already avlaible objects. when a client procus request an objeet, the caching service starts examing the cache for updated copy. if the updated copy is avlaible then caching

Service provides the object from the repository otherwise, it retrieves an updated copy.

In general, caches are situated in a proxy server that is shared by multiple clients or available with each client.

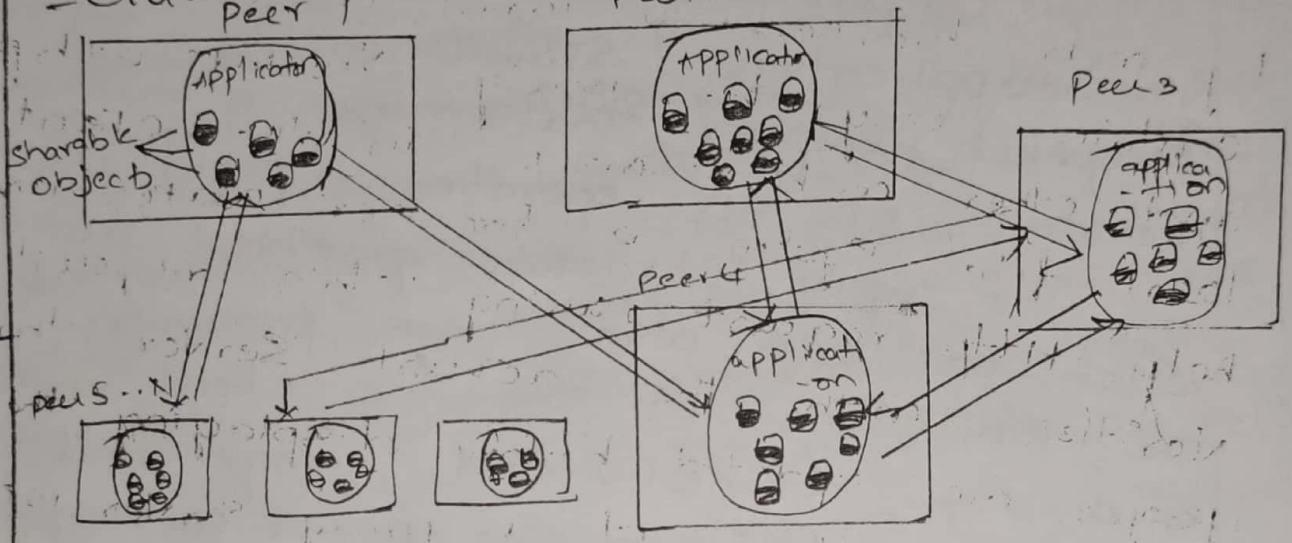
In reality, caches are employed widely by web browsers to maintain a cache that contains only the newly visited web pages and other web resources in the client's local file system, a browser makes use of help http request using this request it verifies with the actual server that pages are updated before displaying them.

The objective of a proxy server is to extend the performance and availability of the service by decreasing the load on wide-area network and web servers through a firewall. for a client machine on a site or over multiple sites, a shared cache is supplied by web proxy server.

→ peer processes:-

The peer processes architecture, all processes play akin roles and intract cooperatively with each other as peers, performing distributed activity or computation without making any differentiation among clients and servers.

In code in peer processes are responsible of maintaining consistency for application level resources and synchronizing application level resources actions based on the requirements. The scenario of peer processes architecture is shown in figure below.



Basically, n number of peer processes can take part in communication, whereas the communication pattern will be as per the need of application.

The elemination of servler processes leads in the reduction of delays in inter-process communication that involves local objects accessing.

for example, assume an application of a distributed 'white board' that allows users to view and modify a picture shared between several computers. This can be made possible by implementing

as an application process at each site relying on middleware layers. So that event notification and group communi -cation can be performed in order to notify the changes to the picture for all application processes. From this we can infer that users can have better interactive response from distributed shared objects than a server based architecture.

* Mobile Agents:-

A mobile agent is a program under execution that goes from one system to another in a network. A network performs a task such as gathering information, provi -ding results on behalf of someone. At each site visited by a mobile agent several calls are made to local resources. for exam -ple, to get single data base entries. This architecture when compared to static client that makes remote calls to some resources, perhaps carryout huge amount of data, with the substitution of remote calls with local ones there is a decrease in communication cost of time

Mobile agents are possibly used in cases such as,

(a) To install and maintain software on the computers with in an organization

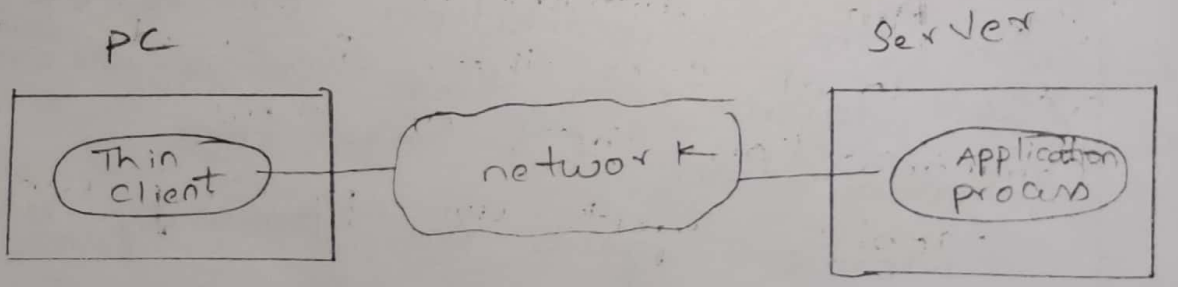
(b) To calculate the prices of products from different vendors by visiting the site of every vendor and carryout a sequence of database operations.

A recent example is worm program developed at xerox PARC. It is designed in such a way to transfer detailed computation by using of idle computers.

The environment accepting a mobile agent should determine which local resources are allowed to access. Because there is a potential security threat to the resources by mobile agents.

* Thin clients:-

A Window-based user interface on a computer local to the user while running applications on a remote computer is supported by a software layer referred to as thin client.



Similar to the Network Computer scheme, this architecture is has the same low management and hardware costs unlike network computer scheme, it runs applications on a Computer Server instead of downloading the code into the user's Computers.

A Computer Server has the capacity to run applications simultaneously. It is a powerful Computer that runs a multiprocessor version of an operating System (UNIX).

The highly interactive graphical activities such as CAD and image processing are the limitations of the thin client architecture since, the transfer of image and vector information between the client and application process include both network and operating system potentials, the delay experienced by users is increased

* The limited processing and Communication capacities of Computers and Networks:-

The performance issues arising from the limited processing and communication capacities of computers and networks are illustrated below.

1. Responsiveness:-

The interactive application users always expect fast and consistent response, but frequently client programs require accessing

of shared resources. In case of remote service, the response speed can be obtained not only with the load and performance of the server and network but also by means of delays in every involved slow components such as client and server OS involved in the communication, middleware services, and the proces code used for service implementation. This can be overcome if the systems contain relatively few software layers, and if small amount of data is transferred b/w the client & server.

In web browsing clients, the locally cached pages and images which are held by the client application can be accessed in faster response mode. However in case of graphical images that contain large volume of data, intractive response time is very slow.

2. Throughput:-

It can be defined as the rate of completing the computational task, It is the traditional performance measure for computer systems.

For distributed system, throughput determines the ability of performing work for all its users The factors such as processing speeds and data transfer rates

affect distributed systems.

Data present on remote server if needed to be passed from a server proces to a client proces, then it has to pass through all software layers present in both computer thus the throughput of network and softwa -re layer is needed to be determined.

* distributed system must be developed such that it should enable applications and service proces. This is done, for concurrent proceeding by making use of available computational resources instead of going ahead for the some resources. Ther loads is said to be eleminated from the web server easing service improvement.

* Quality of Service :-

In a distributed system, as soon as the users are provided with the functionality they are in need of services like file service Adaptability and resource availability are the important characteristics of service quality

Quality of service gets affected by some non-functional properties of system such as reliability security and performance.

Most of the computer system's design basically depends on the issues of reliability and security. From the perform

-nce point of view, quality of service was basically referred to as the response-ness, and computational throughput; according to the definition, a distributed system, from performance point of view is strongly related to the interaction model.

In certain applications, the transfer of data streams from one process to another is done at a fixed amount of time, this is called time-critical data. This signifies that a video clip should be retrieved from the server, and then it is presented on the screen of the user. If the process involves displaying of successive clip with in some specified time limits, then its, then its service quality is said to be satisfied.

Service quality can be achieved based on the availability of the necessary computing and resources of network at the particular instance-of-time. But the performance of today's network even with quite good performance characteristics is getting deteriorated due to heavily loaded traffic.

Web - caching protocol:-

The responses to request by client to web servers are cached by both web browsers and proxy servers. Because of this, the request a client can be satisfied with the response cashed by the browser or by the proxy server between it and the web server. But updation can be delayed or made to relax in order to allow per-mana, availability and disconnect operation.

To validat the response cached by browser or proxy, the updation web of original web server is checked and if it fails the test, then the state response is replaced with the fresh response returned by the web server. if the cashed response is found to be sufficiently fresh, then validation is not required. Even if the web server acknowledge with the updation of a resource then also it doesn't inform it the browsers and proxies.

The web server can inform if it maintains a record of interested browsers and proxies for each of its resources. Servers enable browsers and proxies orders to determine if the cached responses are state; it does this by assigning approximate; it does this by assigning approximate expiry times time.

The browsers and proxies along with their cached response stores the expiry time and server perform comparision of cached response age the expiry time in order to determine if a cached response is stale. Here response and the sum of the time taken to the cached the response and the server time. In this concept, the calculation is independent of computer clocks.

* Dependability Issues:-

The dependability for computer system can be defined as correctness, security and fault tolerance. In most application domain dependability is the crucial requirement. It is not only important in command and control activities but also play an important role in various commercial applications.

These applications may include internet commerce in which financial safety and soundness of the participants is dependent on the systems dependability on which they operate.

The issues arising in dependability are considered under the following sub-headings.

1. Fault Tolerance:

Dependable applications should have the ability to continue its functions even in the presence of faults that may arise in software, hardware, and net works. Reliability of an application can be obtained by means of redundancy. Where in redundancy can be achieved by employing several resources. with the help of these resources, the system and application software can reconfigure and proceed its function even after the occurrence of faults.

But redundancy is cost effective limiting its application at times, there by limiting the degree of fault tolerance

2. Security:

The requirement for security stresses on the consideration of confidential data and data from other resources that can be safeguarded against all the attacks for an instance, consider the database of a bank containing customer records with sensitive components and widely available components. The sensitive components must be visible to only on some specific bank authorities. Then it is unappropriate to delete -p a system that can read load the whole record of the patient into the desktop

Computer may not provide a secure environment

* System Model:-

System model is a model used to creat accurat assumptions regarding the systems that are to be modeled. It is used to create generalizations about the systems that are to be modeled based on the assumptions made. there, generalizations are consider as efficient general-purpose algorithm or the properties which are required and guaranteed.

Factors of fundamental models:-

The factors that are to be considered for the construction of fundamental models.

(a) Security:-

The distributed Systems are vunuable to attacks because of their openness and modular nature. The Security in fundamental model defines attacks from System design that are resistant to threats.

(b) Interaction:-

It is the Communication and Co-ordination between the processes with the exchange of message

The interaction model in distributed Syst must reflect the following facts.

* Communication involves delay of specific duration
* The delays limit the accuracy of procss Coordination
* Difficult to maintain same time Conven across the Components

(c) failure:-

The operation of a distributed Systm i Velnerable to failure with the occurance o failure in the network or in the Comput which is connected to it. fundamental m describes the faults and analyses their effects. It also designs fault tolerable Systems capable of running even in the presence of faults.

* Interacting procss:-

Intracting processes are responsible fo the execution of activities in the distri Systems. Each procss has a state that cannot be accused or updated by anothe procss. A stak Consists of Variables an a set of data that can be updated a accused by its state.

Pactors effect the interacting procss:-

In distributed System, two factors affe the intracting procss:

1. performance of Communication channels

9. Variation in Time

Performance of Communication Channels:-

The performance of Communication channel is restricted. The Communication in modern the place by implementation of streams or by the transmission of message over the computer network. There are three performance characteristics associated with the Communi-cation over a Computer Network.

(a) Latency :- Latency is the delay that occurs when the message is transmitted from one process to another. The delay includes the limit in between the initiation of message transmission of its reception. Latency consists of the following three factors:

(*) The variation in time Consumption by the operating system Communication at the source and destination processes is on the basis of current load on the operating system.

(*) The time Consumed for the transmission of the first stream of bits of string to reach the destination via network. This can be seen in radio signals that travel back and forth the satellite for the transmission of a message

(*) The increase in delay to access the network is due to heavy traffic in it.

(b) Jitter:-

It is the difference in time during the transmission of message. This difference in time can be seen when a series of audio data samples with variant time intervals are played resulting into distortion in time.

(c) Bandwidth:-

It is the amount of data that can be transmitted on network in a given time period. If same network is being used by many communication channels, then they have to share the bandwidth.

2. Variation in Time:-

In distributed system, every computer has an internal clock that gives the current time to the local processes. Hence the processes running on distinct computers can relate the time stamps with their events. But this doesnot guarantee the supply of accurate time. This is due to the clock drifts whose rate varies from one procm to another. A clock drift rate is the relative rate at which a computer clock drifts from an accurate clock that is to be referred. It is obvious that clock timings vary significantly unless some corrections have been applied on it.

The Correction of time on Computer clock can be done in many ways. Radio receivers can be used by the Computers to know the accurate time from the Global positioning System (GPS). It gives the time with an accuracy of about 1 microsecond. The cost of GPS receiver is high and also they can't be operated inside the buildings. A computer that possesses accurat time giving services like GPS can transmit the messages containing accurat time to other computers. However there may be delay in messages which ultimately affect the agreement made by the local computers.

Variants of Interaction model:

There are two variants of interaction model:- (i) Synchronous distributed systems (ii) Asynchronous distributed systems.

Synchronous distributed System is based on the assumptions of time, whereas asynchronous distributed system has no assumptions about time.

(i) Synchronous Distributed Systems:

A Synchronous distributed System is defined to be system which has

known upper and lower bounds associated with the execution of each and every phase of a process. The transmitted messages are received within a bounded time. The process has local clocks, the drift rate of which is associated with a known bound.

However, the values of upper and lower bounds chosen are not guaranteed.

The behaviour of a synchronous system in a real distributed systems, the process that performs the tasks should be familiar with the resource requirements. These requirements must guarantee the processor cycles and network capacity sufficient enough to perform the task process should also be supplied with clocks having bounded drift rate.

(ii) Asynchronous Distributed System :-

An asynchronous distributed system is a system in which there are no bounds on execution speed of a process, message transmission delays and drifts rates between the local clocks.

Asynchronous models do not have any timing conventions. This criteria models the internet in which a server or a network load neither has intrinsic bounds on the server nor on their duration

This can be seen in the transmission of file via FTP. However an email message can take too many days to deliver.

Some problems associated with the design can be dealt even coith out the time assumptions. This can be seen in web browsers. When web is unable to provide a response within a specific time. A solution reasonable for asynchronous distributed system is also effective for a synchronous distributed system.

In general, the distributed systems are asynchronous because it requires the sharing of procsors and networks among the proces and communication channels. respectively. This can be seen when too many procsses of unknocon character share a single procsor, then the performance of any one procss among them cannot be guaranteed

Event Handling in Distributed System

In distributed system, event ordering is essential to describe the execution of system on the basis of its events and their ordering.

Consider the events involving a group of users (user 1, user 2, user 3, user 4) who

exchange the email messages among themselves where user1 sends the email to which user2 and user3 reply back. In actual user1 has sent the message first to which user2 replies immediately then user3 leads user1's sent and user2's replied messages and replies accordingly.

Due to delay on the delivery of message and replies accordingly that they users may view these messages in aorong order for example, user4 may view the messages in the following order:

order of message delivery seen by user4

1st → user3

2nd → user1

3rd → user2

if user1, user2, and user3 have synchronized clocks on their computers, then the sent messages includes the time of the local computer's clock. This allows the reciption of messages to its users according to their time order.

However, in a distributed system, no clock is perfectly kept synchronized. So, a model of logical time was proposed by lamport to provide events ordering delivered messages can be inferred without

any alternative.

following are some of the aspects of logical ordering that can be applied to the problem of event ordering. Suppose, there are two users, user1, user2 send the messages msg1 and msg2, then it is obvious that:

*) A message is received only after it was sent. user1 sends msg1 before user receives msg2
user2 receives msg2 before user1 receives msg2

* A message is replied only after it has received any message.
user2 receives msg1 before sending msg2

These two aspects can be used further when there are too many users and messages to be transmitted and received. Then a number can be assigned to each event based on its logical ordering.

* Failure Model:

Failure model defines methods of failure occurances to facilitate an understanding of failure effects.

The taxonomy that distinguishes between faeilures of processes and communication channels can be understood by the

following concepts:

1. omission failures
2. Arbitary failures
3. Timing failures

Omission failures

The faults classified as omission failures refer to cases when a proces or communication channel fails to perform actions that are supposed to do.

Proces omission failures:-

The primary reason in a proces for omission failure is its crash. That means the proces has been terminated completely. The design services that can resist the failure can be simplified on the assumption that the services crash cleanly. In a clean crash a proces terminates or executes correctly. Due to crash, the proces does not respond to invocation messages repeatedly. This feature other processes in identifying the crash using timeouts. Allot a specific time period to the proces with which it should occure. In an asynchronous system a time out implies that a proces is not responding. The reason for this may be its crash or slow performance or undelivery of messages.

A crash that is identified by other process is called fail-stop. This can be introduced in a synchronous system as;

* if message delivery is correct
* if processes employ timeouts identify failure of other processes.

Consider an example, if the reply to a process is not delivered within the specified time limit, then the process that is waiting for the reply concludes that sender process has been failed.

(b) Communication omission failures:

Send and receive are two communication primitives. Consider the example where Send operation is performed by process A. It inserts a message in its outgoing message buffer. Communication channel transmits this message to process B's incoming message buffer. Receive operation is performed by process B that takes the message from its incoming message buffer and delivers it.

The communication channel raises an omission failure if the message from A's outgoing message buffer is not transmitted to the incoming message buffer. This is called

droping messages which is due to insufficient buffer space at the receiver site or at the intervening gateway or by error at network transmission. These consequences are identified by the checksum that is attached with the message content

following are the failures resulting due to loss of messages:

(i) Send-omission Failures:
occur in between sending process and outgoing message buffer.

(ii) Channel-omission Failures:
occur in between outgoing message buffer and incoming message buffer.

(iii) Receive-omission failures:
occur in between incoming message buffer and receiving process.

2. Arbitary Failures:-

Arbitary or Byzantine failure describes the semantics of a worst failure in which there is possibility of occurence of any type of error.

In a process an orbitary failure is caused when it skips any of its execution phase or adopts a new phase for execution. Hence, detection of arbitary failures in processes cannot be known

by checking the responses to invocations because there is a possibility for the process to skip the process reply. for example, a process may set incorrect values in its data items, or it may return an incorrect value.

In communication channels, arbitary failure can take the form of a corrupt message, delivery of a wrong message or delivery of a correct message twice or many times. however, there occurs very few arbitary failures in communication channels because the communication s/w can identify them and discard corrupt messages. for example, checksums determine corrupt messages and a message sequencer identifies wrong or duplicated messages

Omission & arbitrary failures

class of failures	Affects	Description
Fail-stop	process	process halts and remains halted. other processes may detect this state.
crash	process	process halts & remains halted. other processes may not be able to detect this state.

omission	channel	A message inserted in an outgoing message buffer ne arrives at the other end incoming message buffer.
Send-omiss -ion	procss	A procss completes a Sen but the message is no put in its outgoing me buffer
Receive	procss	A message is put in pro incoming message buffer, omission that procss does receive it
Arbitrary	procss (or) channel.	procss/channel exhibits -bitrary behaviour. it Send/transmit arbitra message at arbitrar times, commit omissio a procss may stop o take an incorrect st

3: Timing failures:-

It is applicable in Synchrono distributed system where as in asynch -nous distributed system, it is not gua

class of failure	Affects	Description
clock	procss	procss's local clock exc the bounds on its rat drift from real time
performance	procss	procss exceeds the bound the interval blw two st
performance	channel	A message transmission takes than the stated bound

The term reliable communication is defined in terms of validity and integrity as follows:

Validity: any message in the outgoing message buffer is eventually delivered to the incoming message buffer.

integrity: The message received is identical to one sent, and no messages are delivered twice.

(a) Synchronous Distributed System:- In this system, time limits are fixed on process execution time, message delivery time and clock drift rate. A process is effected if it exceeds its time bounds. A channel is effected if the time taken for the transmission exceeds the specified bounds. A clock failure effects its process if its local time exceeds its drift rate bounds from real time. There three failures fail to respond within the specified bounds.

(b) Asynchronous Distributed System:- In asynchronous distributed system, timing failure cannot be guaranteed, and there is a possibility of a heavily loaded server to respond late

Threats to integrity is derived from two independent sources e.g. protocols and intruders.

Protocols:- It is the protocol that retransmits the messages without discarding the message that is received twice. Such protocols assign sequence number to the messages to identify those messages that are delivered twice.

Intruders:- It may send take messages or interface with old messages. In such cases, measures should be taken to preserve integrity.

* Security model:-

A Security model is constructed on the basis of distributed system architecture where the processes, objects and interaction channel have to be protected against threats. This model provides the analysis of threats, different forms of attacks by threats, evalution of risks and consequences for each at minimum cost.

Management of objects by Server:- The distributed system architecture consists of Server processes that encapsulate objects. These objects are accessed by different clients through interactions. The

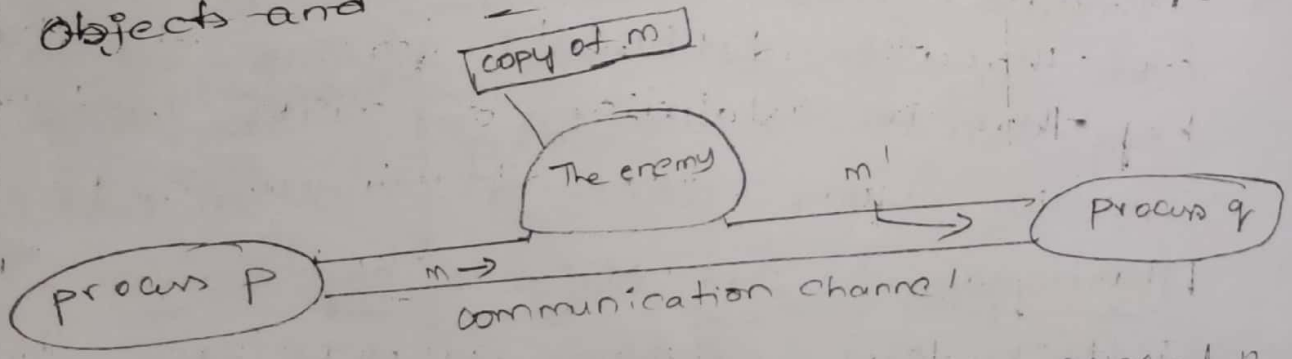
user/client process invokes the server to perform operation on the objects. Server performs the operation for each invocation and sends results to client.

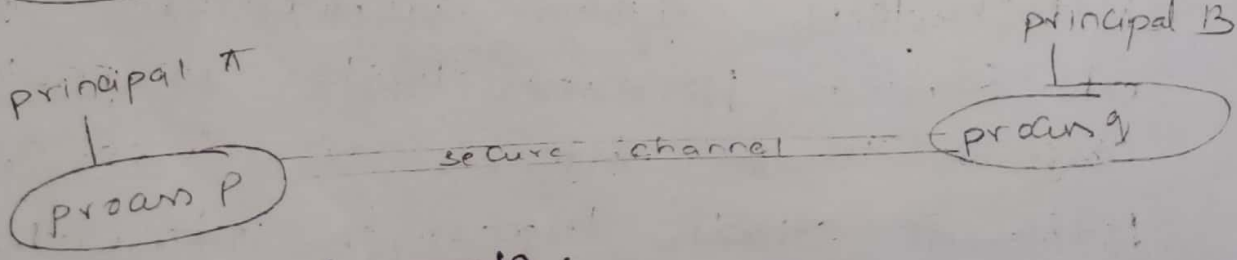
The whole process requires the association of authority to both the processes (client & server) such authorized processes are known as principal.

The server's authority is to verify the identity of client principal. It checks whether the client principal has sufficient access rights to perform operation on the particular invoked object or not.

On the other side, the client verifies the identity of server principal and confirms whether the result received is from required server or not.

Objects and The enemy





Secure channels.

→ Threats generated from potential enem

The threats that can be generated from a potential enemy are,

1. Threats to processes
2. Threats to communication channels
3. Denial of Service

1. Threats to processes:-

In a distributed system, a process reserved for handling incoming requst This process might receive any message from another process, it is not necessa that it can identify the identity of process, the sender. Although communicat protocol like IP of each message contai the source computer address, there is possibility for an enemy to generate a message with a fake server address. Thus it leads to a threat for both cli and servers to function correctly.

(i) Servers:- A server receives invocation from several clients which makes it difficult to identify the identity of an invocation.

Actually the server can identify the identity of any invocation. But, there is a possibility of generating false identity by an enemy due to which the server cannot make a decision of

performing the operation or rejecting it

(ii) Client:- The client that recieves the message from the server is unable to determine whether the message is from the desired server or from an enemy. This received message might not match the original invocation.

2. Threats to Communication channels

Across communication channels an enemy can attack over information by means of altering, copying or injecting. This leads to a threat for privacy and integrity of the information. An enemy can attempt to serve messages copy in order to reply them at later time.

3. Denial of Service

It is an attack where the enemy overloads the physical resources of authentic users. The enemy does this by making several unnecessary invocations on service or transmitting the messages in a network. The main reason behind this attack is to delay or prevent the actions of other users.

UNIT-II

INTERPROCESS COMMUNICATION

* API for Internet protocols:-

→ characteristics of Interprocess communication:-

Message passing between a pair of processes can be supported by two message communication operations: send and receive. In order for one process to communicate with another, one will sends a message to destination and other process at the destination receives the message. This activity involves the communication of data from the sending process to the receiving process and may involve the synchronization of the two processes.

Synchronous and Asynchronous Communication:-

→ In Synchronous inter process communication both send and receive operations are blocking system calls. It means that sending process stops execution untill it receives a reply from receiver and receiving process stops execution untill it receives from sender which ensures both are Synchronized.

→ In asynchronous inter process Communication, in this sending process uses a non blocking send operation. As a result

Sending proces doesn't wait after sending message instead it continous execution. However the reciving proces can use either blocking or non-blocking reciving operation. if blocking receive is used, proces waits untill it recives message else proces doesn't wait and continous its execution.

Characteris

Message Destination: In a distributed system, messages are specified using internet address, port address pairs. The port address refers to proccess on destination system which is supposed to receive a message. This mechanism doesn't provide transparency following approaches are used.

a) program can refer services by specific names and there are special name into Server location

b) most operating system uses location independent identifiers which are later mapped to low level addresses.

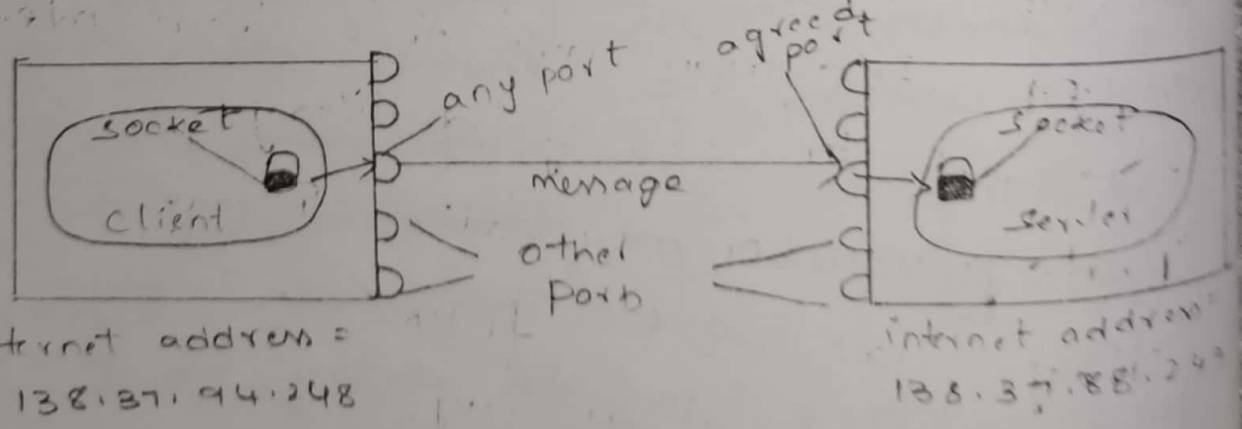
Reliability:- Validity and integrity are two important integredients of reliable communications. Reliability refers to

guarantee that messages are delivered to destination without being dropped and integrity refers to packets are received with damage and duplication

Ordering:- messages are said to be in order if they arrive at receiver side in the same order that they were sent by sender some application will not accepted packets if they are not rec -d in order.

→ Sockets:-

A Socket acts as endpoint in connection between a client and a server present in a network using a Socket, a process running on one Com -ter can communicate with other proc running on distinct computer.



Sockets & ports-

→ UDP Datagram Communication:-

A datagram sent by UDP is transmitted from a sending process to a receiving process without acknowledgment or retries If failure occurs, the message may not arrive. A datagram transmitted between processes when one process sends it and another receives it. To send or receive messages a process must first create a socket bound to an Internet address of the local host and local port. A server will bind its socket to a server port.

→ Uses of UDP:-

The uses of UDP are listed below.

(1) UDP implements DNS that looks up the DNS names in the internet

(2) UDP allows VoIP to run over it

(3) UDP datagram are free from overhead released to message delivery

→ Datagram Communication Issues:-

They are as follows:-

1. Size of message
2. Blocking
3. Receive message from any source

Size of Message:- Size of packet allowed by IP protocol ranges upto 216 bytes along with message packet holds header also. → Progm that receives the message header specifies array of bytes to hold message. If message exceeds the allowable range then it is truncated after message has been received.

Blocking:- usually sockets provide non blocking send operation and blocking receive operation for datagram communication

(a) Send operation sends message to relevant UDP and IP protocols. These protocols transfer message to desired destination. The received message is placed in a queue for that socket which is intended for destination port and if progm doesn't have socket message is discarded.

(b) The receive operation is blocked till a datagram received and time out has been set on socket. In case progm that invokes receive operation is found to be busy, it must assign new thread to use.

Receive messages from any source:- The receive operation does not mention

Source of message. Its invocation will address a message from any source. It enables recipient to check the source of message by returning the internet address and local port of the sender and also receive operation. enables connection from socket to remote port

→ Java API for UDP Datagram!

Java API provides two types of classes namely Datagram packet and datagram socket.

Datagram packet!- Datagram packet class provides a constructor that creates an instance which can be transmitted from one procss to another such that one procss sends and other procss receives it. The instances are created by array of bytes consisting of message length of message, internet address and local port number of destination socket.

Datagram Socket!- datagram socket class provides support to socket for sending and receiving UDP datagrams. It provides a constructor which accepts

port number as arguments. This typ
of Constructor is used by proces
deal with the specific port

=> TCP Stream Communication :-

The API to the TCP protocol, whic
originates from BSD 4.x UNIX, provides
abstraction of a stream of bytes to
which data may be written and fro
which data may be read. The follow
characteristics of the network are hidd
by the stream abstraction:

Message sizes :- The application can cho
how much data it writes to a stre
or reads from it. It may deal in ver
small or very large sets of data. Th
underlying implementation of a TCP stre
decides how much data to collect be
transmitting it as one or more IP pac
On arrival, the data is handed to t
application as requested. Applications ca
if necessary, force data to be sent im
diately.

Lost messages :- The TCP protocol uses an
acknowledgment scheme. As an example
a simple scheme, the sending end ke
a record of each IP packet sent an
the receiving end acknowledgment all
arrivals. If the sender does not rec
an acknowledgment within a time out,
it retransmits the message. The more

Sophisticated sliding window scheme, cuts down on the number of acknowledgement messages required.

Flow control:- The TCP protocol attempts to match the speeds of the processes that read from and write to a stream. If the writer is too fast for the reader, then it is blocked until the reader has consumed sufficient data.

Message duplication & ordering:- message identifiers are associated with each IP packet which enables the receipt to detect and reject duplicates, for to reorder messages that do not arrive in sender order.

Message destinations:- A pair of Communicating processes establish a connection before they can communicate over a stream. Once a connection is established the processes simply read from and write to the stream without needing to use Internet addresses and ports. Establishing a connection involves a connect request from client to server followed by an acapt request from server to client before any communication can take place. This could be considerable overhead for a single client-server

request and reply.

Uses of TCP :-

The services that run across TCP connection with reserved port number are as follows:

HTTP :- (Hyper Text transfer protocol):-

This service is utilized inorder to establish communication between web browsers and web services.

FTP (File Transfer protocol):- This service is utilized inorder to encorporate directories running on remote computer to be browsed on other computer.

Telnet:- This service is utilized by trminal sesion inorder to gain accss to remote computer.

SMTP:- It used for sending mails between computer.

⇒ Issues Related to Stream Communication:-

The issues are:

(i) Matching of Data Items:- In Tcp stream communication the two processes must agree on the contents of data transmissi -on across stream. It is essential beca -use the contents written by one proces must be read in order by other proces.

(2) Blocking:-

In this Communication data written to stream is stored in the form of queue across destination socket When data is available at destination tries to read data from channel else if unavailable it blocks itself until data is available.

(3) Threads:-

In this communication when client establishes connection with server. It accept request and forms separate thread during this it blocks other client threads with which it already established

* External data Representation & marshalling:-

The information stored in running programs is represented as data structures - for example by sets of interconnected objects - whereas the information in messages consists of sequences of bytes Irrepective of the form of Communication used, the data structures must be flattened before transmission and rebuilt on arrival. The individual primitive data items transmitted in messages can be

data values of many different types, and not all computers store primitive values such as integers in the same order. The representation of floating-point numbers also differs between architectures. There are two variants for the ordering of integers. The so-called big-endian order, in which the most significant byte comes first, and little-endian order, in which it comes last. Another issue is the set of codes used to represent characters. For example, the majority of applications on systems such as UNIX use ASCII character coding, taking one byte per character, whereas the Unicode standard allows for the representation of texts in many different languages and takes two bytes per character.

One of the following methods can be used to enable any two computers to exchange binary data values:

* The values are converted to an agreed external format before transmission and converted to the local form on receipt; if the two computers are known to be the same type, the conversion to external form can be omitted.
* The values are transmitted in the sender format together with an indication of the format used, and the recipient converts the values if necessary.

Note, however that bytes themselves are never altered during transmission. To support RMI or RPC, any data type that can be passed as an argument or returned as a result must be able to be flattened and the individual primitive data values represented in an agreed form. An agreed standard for the representation of data structures and primitive values is called an external data representation.

Marshalling: It is the process of taking a collection of data items and assembling them into a form suitable for transmission in a message. unmarshalling is the process of disassembling them on arrival to produce an equivalent collection of data item at the destination. Thus marshalling consists of the translation of structured data items and primitive values into an external data representation.

* CORBA's common data representation which is concerned with an external representation for the structured and primitive types that can be passed as arguments and results of RMI in CORBA. It can be used by a variety of programming languages.

* XML or extensible markup language, which defines a textual format for representation of structured data.

In first two cases, the marshalling and unmarshalling activities are intended to be carried out by a middleware layer without any involvement on the part of the application programmer. Then the case of xml, which is textual and therefore more accessible to hand encoding, software for marshalling and unmarshalling is available for all commonly used platforms and programming environment.

CORBA's Common Data Representation (CDR):-

CORBA CDR is the external data representation defined with CORBA 2.0. CDR can represent all of data types that can be used as arguments and return values in remote invocations in CORBA. There Consist of 15 primitive types, which include Short (16-bit), long (32-bit), unsigned short, unsigned long, float (32-bit), double (64-bit), char, boolean (TRUE-FALSE), octet (8-bit) and any together with a range of composite types, each argument or result in remote invocation is represented by a sequence of bytes in the invocation or result message.

Primitive type :- CDR defines a representation for both big-endian and little-endian orderings. The values are transmitted in the sender's ordering, which is specified in each message. The recipient translates it if requires a different orderings. For example, a 16-bit short occupies two bytes in the message and for big-endian ordering, the most significant bits occupy the first byte and the least significant bits occupy the second bytes.

CORBA CDR for Constructed types:

Type	representation
Sequence	Specifies length followed by elements in order
String	Specifies length followed by character in order
Array	Array elements in order
Struct	order of declaration of components
enumerated	Specifies unsigned long

CORBA CDR message :-

Each primitive value that contains constructed type are added to the sequence of bytes in order

Index in sequence of bytes

0 - 3	4 bytes
4 - 7	7 -> length of string
8 - 11	masa -> marsarat
⋮	rat
16-19	9 -> length of string
20-23	"Hyde" -> Hyderabad.
24-27	"raba"
32-35	"d"
⋮	

Constructed types:- The primitive values that comprise each constructed type are added to a sequence of bytes in a particular order.

2. Java object Serialization:- An object instantiated from Java class. The Both objects and primitive data values existing in Java RMI is passed as arguments and outcomes of method invocations. For instance, Java class equivalent to stud struct in CORBA IDL is

```
Public class stud implements Serializatio
{
    privat string name;
    privat string location;
    privat int year;
    public stud (string bname, string blocati
                 int byear)
```

```
{
    name = bname;
    location = blocation;
    year = byear;
  }
}
```

The above stated class implements Serializable interfaces and does not include methods. As the class implements Serializable interface located in java.io.package. It allows serialization of instances serialization in java refers to an act of flattering an object or connected set of objects at equals levels in order to achieve serial formation

3. XML:-

It is a simple markup language for describing structure of document It has been derived from standard generalized markup Language for large Scale electronic publishing and for exchange of data in web

xml has a set of rules for creating other markup languages called as "meta markup" language.

xml schema:-

It is for constructing xml docume

-nts.

1. xml schema depends on xml syntaxes for their documentation:-

We can directly make use of xml editors as well as parsers for generating xml schemas and parsing.

2. xml schema documents support usage of data types:-

We can impose restrictions on data and define the data patterns. Data belonging to one data type can be easily converted to other

3. using xml schema we can ensure secure data communication between various entities:-

Whenever any of two distinct entities resort to data communication, they both have to agree on same patterns to ensure error free communication

4. As xml schemas are written in xml, they are extensible in nature:-

Basing on one standard type, we can derive numerous data types are required

A given Schema can be implemented or reuse other Schema etc.

5. xml Schemas can be taken as Successors to DTD:-

xml Schemas extensively support namespaces as well as data types. They have adverse sets of provisions which make xml Schemas more powerful and riches than DTD.

Creation of xml Schema:-

<xs:element name="d.o.b", type="xs:date">
</xs:element>

For attributes declaration.

<xs:attribut name="name of attribute"
type="format of attribute">

Building blocks of xml:-

Building blocks of xml are as follows

1. Tags:- Tags usually mark up elements embedded between them. There are two tags namely opening and closing tags.

eg: <heading> final notice </heading>

2. Elements:- There are major entities to represent or to structure data in these documents

Syntax:-

<!ELEMENT name-of-element (Content)>
<!ELEMENT name-of-element (EMPTY)>
<!ELEMENT name-of-element (#PCDATA)>
<!ELEMENT name-of-element (#CDData)>

3. Attribute:-

Attributes are defined after element. Attributes are followed by their value.

Syn:

<ATTLIST name-of-element, name-of-attribute attribute-type, default-value>

4. PCDATA:-

It is a data which will Suitable parsed by a given parser. They may carry certain tags which are nothing but mark up elements and finally the entities.

5. CDATA:- This is just opposite to PCDATA there is no entity expansion. tags are not treated as mark up element and finally CDATA is not parsed at all.

6. Entities:-

We use few variables in xml instead of some frequently used text such variables are nothing but entities there are two types of entities. Internal entities and external entities.

→ Client-Server Communication :-

This form of Communication is designed to support the roles and message exchanges in typical client-server interactions.

A protocol built over datagrams avoids unnecessary overheads associated with the TCP stream protocol. In particular:

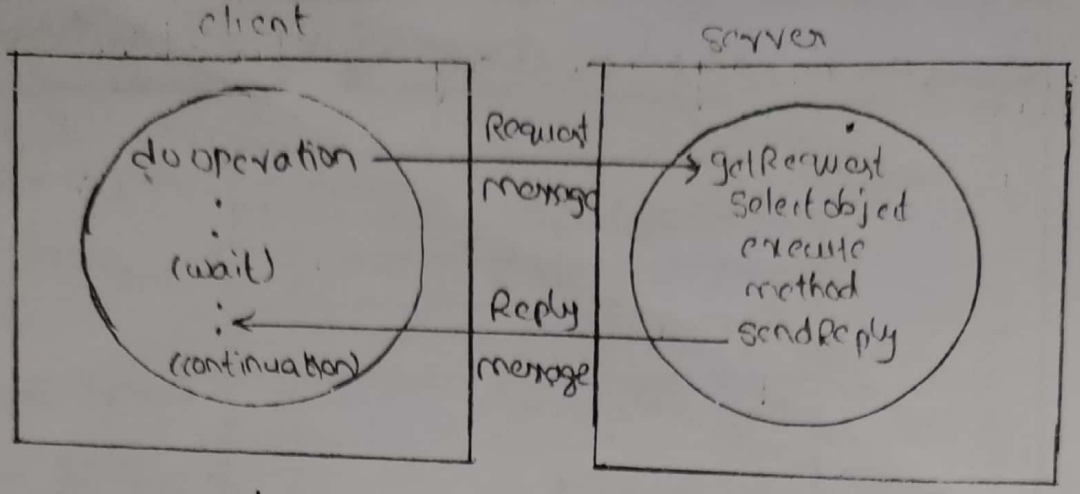
→ acknowledgements are redundant, since requests are followed by replies;

→ establishing a connection involves two extra pairs of messages in addition to the pair required for a request and a reply

→ flow control is redundant for the majority of invocations, which pass only small arguments and results

Request-Reply protocol :-

The following protocol is based on trio of communication primitives: doOperation, getRequest and sendReply. RMI that it passes a remote object reference for the object whose method be invoked in the request message..

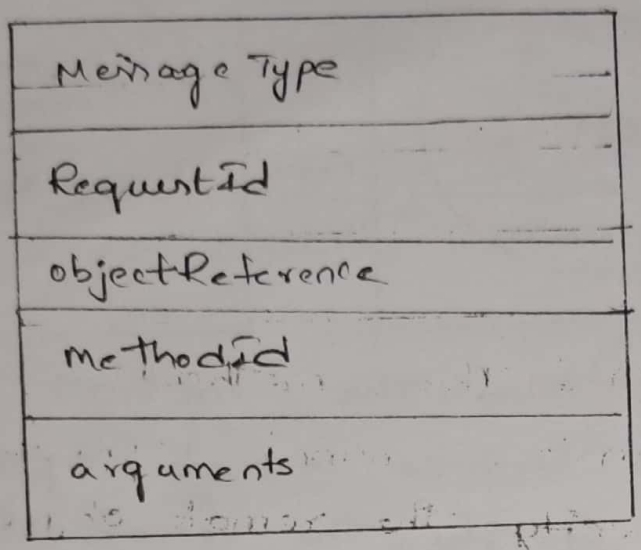


The do operation :- The method is used by clients to invoke remote operations. Its arguments specify the remote object and which method to invoke, together with additional information required by the method. The first argument of do operation, is an instance of the class RemoteObjectRef.

GetRequest :- It is used by a server process to acquire service requests. When the server has invoked the method in the specified object it then uses sendReply to send the reply message to the client. When the reply message is received by the client the original dooperation is unblocked and execution of the client program continues.

The information to be transmitted in a request message or a reply message. The first field indicates whether the message is a Request or Reply message. The second field requestId contains identifier

Request reply message Structure



→ RPC exchange protocols :-

Three protocols, which produce different behaviours in the presence of communication failures, are used for implementing various types of RPC. They were originally identified by Spector [1982] :

- The request (R) protocol
- The request-reply (RR) protocol
- The request-reply-acknowledge reply (RRA) protocol.

Request-protocol :-

In this protocol, client sends single request to Server. In this client sends next request immediately after first request is sent as client does not need any confirmation that request has passed to Server

2. Request - Reply protocol:-

Here client sends request to Serv and Server sends reply message to client. It can be used when there is a value to be returned from remo method.

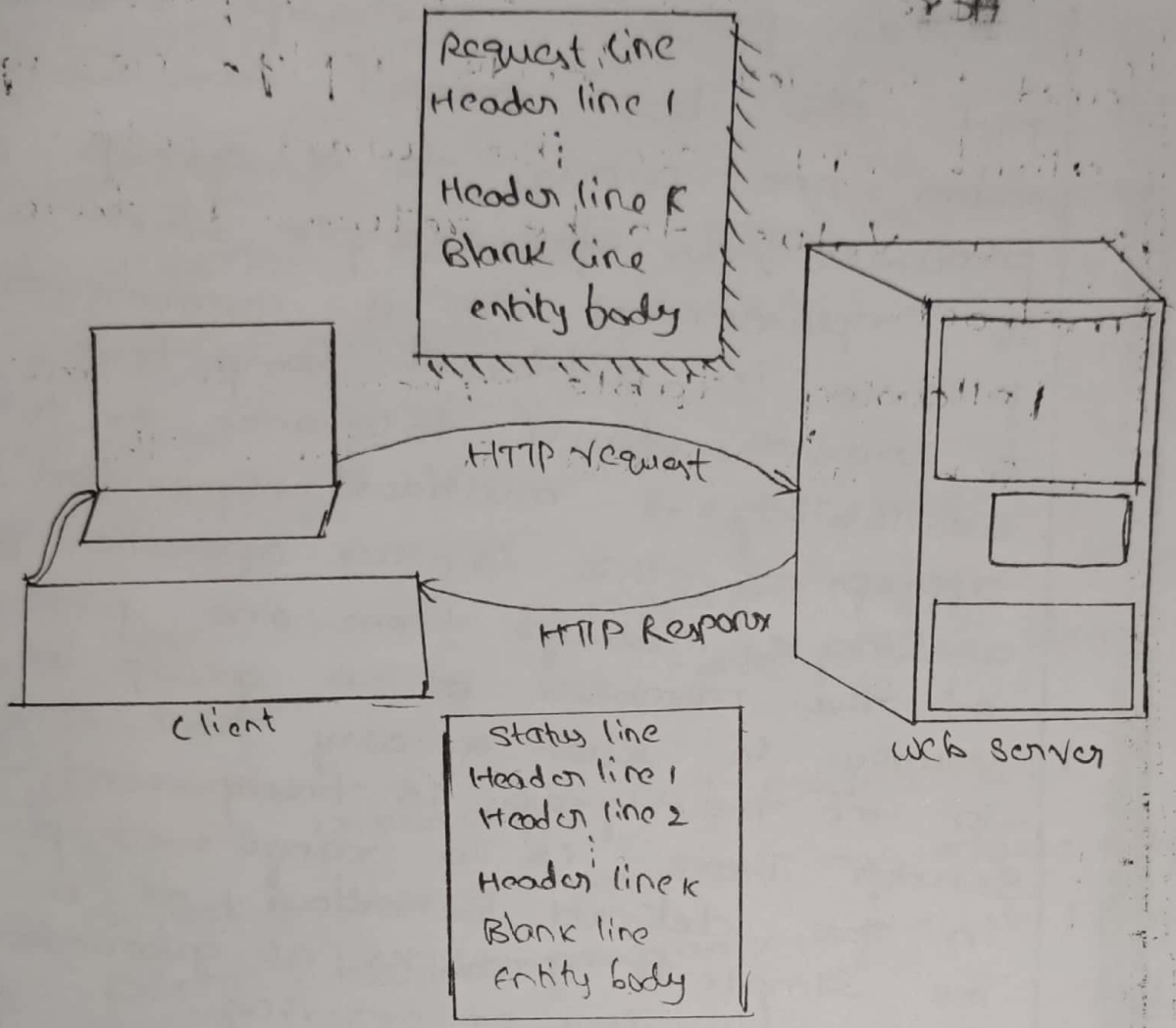
Following table shows the message which are used in these three prot

Message passed by	Name of the protocol		
	R	RR	RRA
client	Request	Request	Request
server		Reply	Reply
client			Acknowledge reply

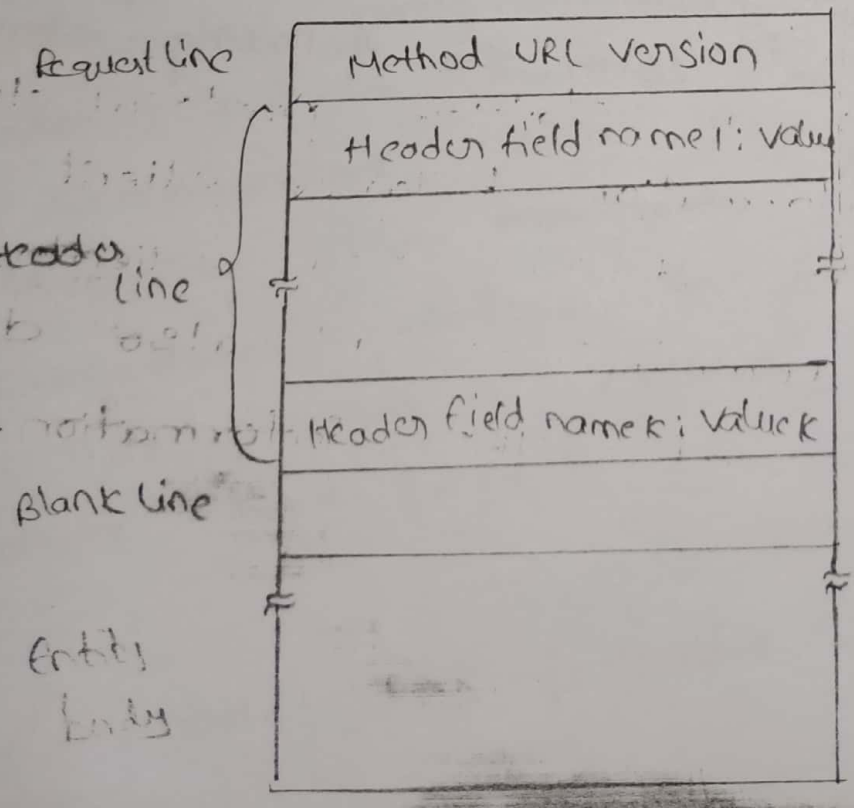
Http protocol:-

It is a stateless protocol to trans-mit information between client and Server. It defines how client and Server communicate. It also defines structure of this information.

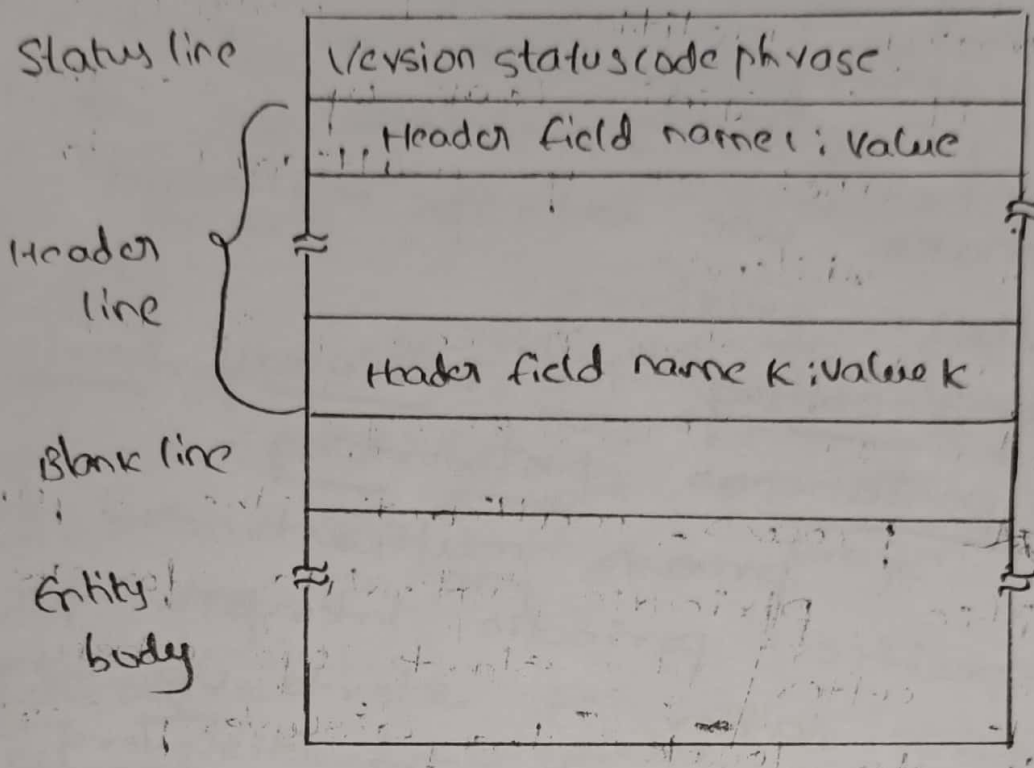
Basic HTTP Model :-



HTTP Request :-



HTTP Response:-



→ Reliability and ordering of multicast:-

Effects of reliability and ordering are as follows.

1. Fault tolerance depending upon replicate services:-

Consider a replicated service which consists of a group of servers. All the servers start at same initial state and perform the same operation in the same order in order to be consis-tent with each other.

2. Spreading the Event Notification:-

A specific application defines the necessary qualities of multicast. Example of it is Jini look up service which make use of ip multicast to notify their existence.

3. Locating the Discovery Servers in Spontaneous Networking:-

A process multicasts the requests at specific periodic intervals for the same time after it starts up when it wants to locate the discovery servers. Any request which is lost is not considered

4. Enhanced performance through Replicated Data:-

Consider such a case where replicate data by itself distributed by means of multicast messages instead of operations on data. The loss of messages, and inconsistent sequence have an effect which is based on technique of replicating and importance of updating all replicas

Group Communication:-

The pairwise exchange of messages is not the best model for Communication from one process to a group of other processes, as for example when a service is implemented as a number of different processes in different Computers, perhaps to provide fault tolerance or to enhance availability. A multicast operation is more appropriate - this is an operation that sends a single message from one process to each of the members of a group of processes usually in such a way that the membership of the group is transparent to the sender. There is a range of possibilities in the desired behaviour of a multicast. The simplest provides no guarantees about message delivery or ordering.

Multicast messages provide a useful infrastructure for constructing distributed systems with the following characteristics:

Fault

UNIT-III

DISTRIBUTED OBJECTS & REMOTE INVOCATION

Introduction:-

Application consists of Cooperative programs (distributed programs) running in different processes. These processes may generally related to different Comp-uters. The programs might have a need to invoke operations from other processes. In this we have three design models.

1. Remote procedure call model (RPC)
2. object based programming model (OBP)
3. Event based programming model (EBP)

RPC :-

This model is followed to develop client/server programs in different processes, where the client program can communicat with server programs by invoking the methods of objects ti procedures define in server progr-ams. This proass is called remote procedure call (RPC)

OBC:

This model is followed to develop

objects in different procas where the objects can communicate one another by invoking the methods of objects living in any other procas. This procas is called Remote method Invocation (RMI)

EBP :-

This model is used to develop the programs where the objects can receive notification of events occured at other objects living in any other procas.

Middleware :-

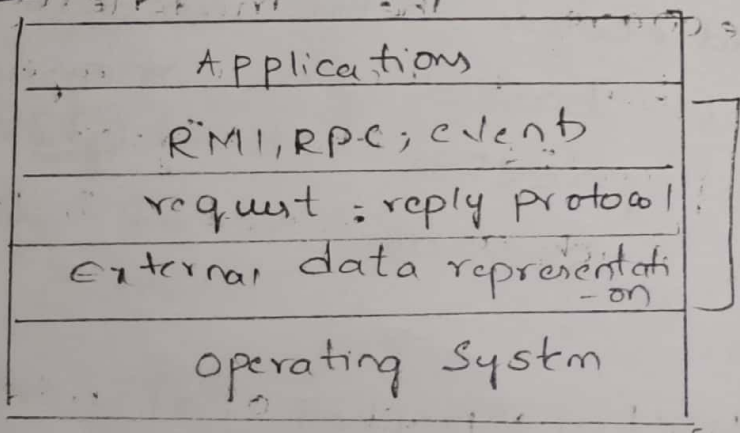
Software that provides a programming model above the basic building blocks of procases and message passing is called middleware. The middleware layer uses protocols based on message between procases to provide its higher-level abstractions such as remote invocations and events.

An important aspect of middleware is the provision of location transparency and independence from the details of Communication protocols, operating systems and computer hardware. Some forms of middleware allow the separate components to be written in different programming

languages.

Location transparency:- In rpc the client that calls a procedure cannot tell whether the procedure runs in the same process or in a different process, possibly on a different computer. Similarly in RMI the object making the invocation cannot tell whether the object it invokes is local or not and doesnot need to know its location. In distributed event-based programs the objects generating events and the objects that receive notifications of those events need not be aware of one anothers locations.

Middleware layer:-



Communication protocols:-

The protocols that support the middleware abstractions are independent of the underlying transport protocols.

for example, the request-reply protocol can be implemented over either UDP or TCP.

Computer hardware:-

Three agreed standards for external data representation are described above. These are used when marshalling and unmarshalling messages. They hide the differences due to hardware architectures such as byte ordering.

Operating Systems:-

The higher-level abstractions provided by the middleware layer are independent of the underlying operating systems.

Use of several programming languages:-

Some middleware is designed to allow distributed applications to use more than one programming language. In particular, CORBA allows client written in one language to invoke methods in objects that live in server programs written in another language. This is achieved by using an interface definition language or IDL to define interfaces.

Interfaces:-

The interface of a module specifies the procedures and the variables that can be accessed from other modules. Modules are implemented so as to hide all the information about them except that which is available through its interface

Interface in distributed Systems:-

The CORBA IDL interfaces can specify attributes, which seems to break this rule. However, the attributes are not accessed directly but by means of some getter and setter procedures added automatically to the interface

Service interfaces:-

In the client-server model, each server provides a set of procedures that are available for use by clients. For example a file server would provide procedures for reading and writing files. The term Service interface is used to refer to the specification of the procedures offered by a server, defining the types of the input and output arguments of each of the procedures.

Remote interface:-

In the distributed object model, a remote interface specifies the methods of an object that are available for invocation by objects in other processes, defining the type of input and output arguments of each of them.

Interface definition languages:-

IDL provides the notations for defining interfaces. These interfaces allow the objects written in different programming languages to communicate to another. The following example uses CORBA as IDL

```
// In file employee.idl
struct employee{
  string ename;
  string eloc;
  long age;
};
interface Employeelist{
readonly attribut String listname;
void addemployee(in employeee);
void getEmployee(in string ename, out employee
                                        e);
long number();
};
```

The above example shows an interface

"EmployeeList". It specifies the method implemented by remote objects. They are addEmployee(); getEmployee(), number();

"addEmployee" method has one argument which specified as "in" it means that it is an input argument

"getEmployee" method has two arguments The first argument is specified as 'in' and the second argument is specified as 'out' The second argument shows that it is an o/p argument

* Communication between distributed objects

The object-based model for a distributed system extends the model supported by object-oriented programming languages to make it apply to distributed objects.

-> Object model:- A brief

An object-oriented program, for example in java or c++, consists of a collection of interacting objects, each of which consists of set of data and set of methods. An object communicates with other objects by invoking their methods generally passing arguments and receiving results. objects can encapsulate their

data and the code of their methods. Some languages, for example java and c++, allow programmers to define objects whose instance variables can be accessed directly. But for use in a distributed object system an object's data should be accessible only via its methods.

Object references:- objects can be accessed via object references. For example in java, a variable that appears to hold an object actually holds a reference to that object. To invoke a method on a object, the object reference and method name are given, together with any necessary arguments. The object whose method is invoked is sometimes called the 'target' and sometimes the 'receiver'.

Interfaces:- An interface provides a definition of the signatures of a set of methods, without specifying their implementation. An object will provide a particular interface if its class contains code that implements the methods of that interface.

Actions:- The receiver executes the appropriate method and then returns

control to the invoking object, sometimes supplying a result. An invocation of a method can have three effects:
1. the state of the receiver may be changed.
2. A new object may be instantiated for example, by using a constructor in Java or C++; and
3. Further invocations on methods in other objects may take place.

Exceptions:- programs can encounter many sorts of errors and unexpected conditions of varying seriousness. During the execution of a method, many different problems may be discovered. A block of code may define to throw an exception whenever a particular unexpected condition or errors arise. This means that control passes to another block of code that catches the exception. Control does not return to the place where the exception was thrown.

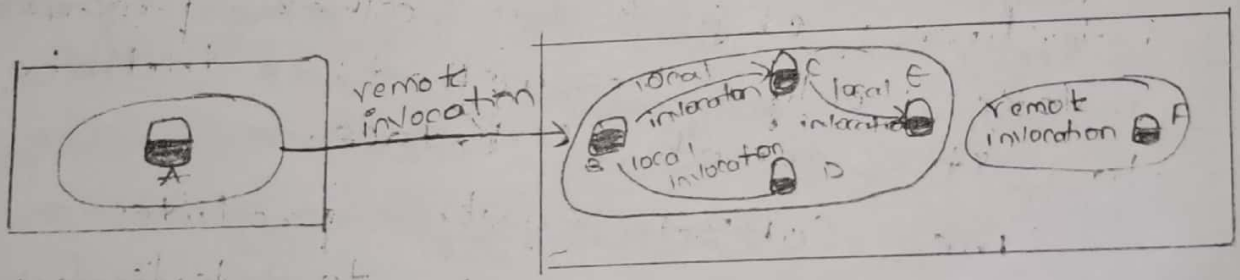
Garbage Collection:- It is necessary to provide a means of freeing the space occupied by objects when they are no longer needed. A language, for example Java that can detect automatically when an object is no longer accessible recovers the space & makes its available

for allocation to other objects. This process is called garbage collection.

→ Distributed object model:-

Each process contains a collection of objects, some of which can receive both local and remote invocations, whereas the other objects can receive only local invocations. Method invocations between objects in different processes, where in the same computer or not, are known as remote method invocations. Method invocations between objects in the same are local method invocations.

Remote and local method invocations:-



All objects can receives local invocations, although they can receive from other objects only for them that hold references to them.

References:- The notion of object reference is extended to allow any object that can receive an RMI to have a remote object reference. A remote object reference is an identifier that can be used throughout a distributed system to refer to a particular unique remote object.

1. The remote object to receive a remote method invocation is specified by the invoker as a remote object reference.

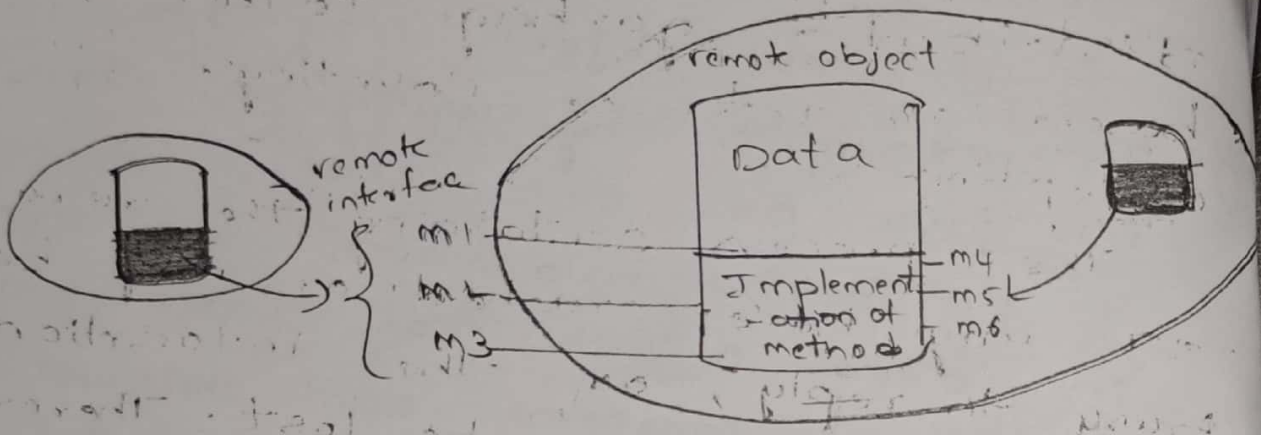
2. remote object references may be passed as arguments and results of remote method invocations.

Interfaces:- The class of a remote object implements the methods of its remote interface. objects in other progams can invoke only the methods that belong to its remote interface.

The CORBA system provides an IDL, which is used to defining remote interfaces. for example of a remote interface defined in CORBA IDL. The classes of remote objects and the client programs may be implemented in any language such as c++, java or python for which an IDL Compiler

is available. CORBA clients need not use the same language as the remote objects in order to invoke its methods remotely.

remote object and its remote interface



Actions :-

As in the non-distributed case, an action is initiated by a method invocation, which may result in further invocations on methods in other objects. But in distributed case, the objects involved in a chain of related invocations may be located in different process or different computers. When an invocation crosses the boundary of a process or computer, RMI is used and the remote reference of the object must be available to the invoker.

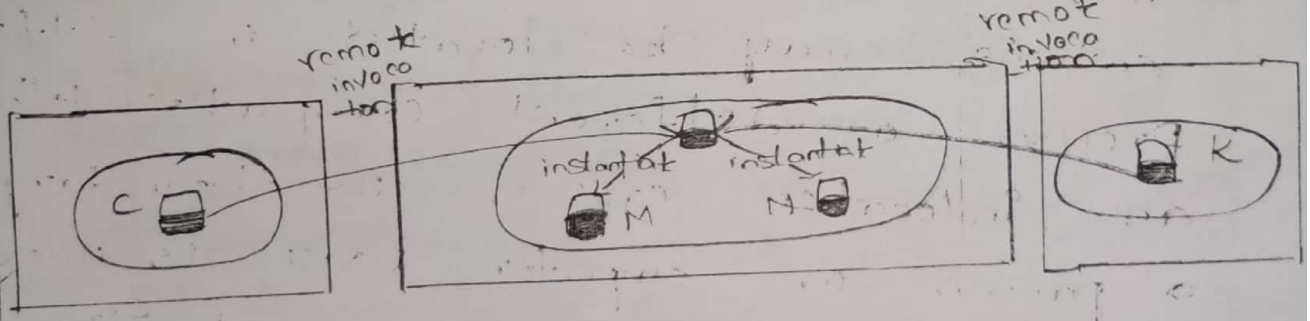
Garbage Collection :- If a language, for example Java, supports distributed garbage collection, then any associated RMI

System should allow garbage Collection of remote objects. It is generally achieved by Cooperation between the existing local garbage Collector and an added module that carries out a form of distributed garbage collection, usually based on reference Counting.

exceptions :-

The process containing the remote object may have crashed or may be too busy to reply, or the invocation or result message may be lost. Therefore, rmi should be able to raise exceptions such as time outs that are due to distribution as well as those raised during the execution of the method invoked.

Instantiation of remote objects :-



Design Issues for RMI :-

We have two design issues :-

1. The choice of invocation semantics - although local invocations are executed exactly once, this cannot always be the case for remote method invocations.
2. The level of transparency that is desirable for RMI.

RMI Invocation Semantics :-

The main choices are :-

Retry request message :- Whether to retransmit the request message until either a reply is received or the server is assumed to have failed.

Duplicate filtering :- When retransmissions are used, whether to filter out duplicate requests at the server.

Retransmission of results :- Whether to keep a history of result messages to enable lost results to be retransmitted without re-executing the operations at the server.

→ Local method invocations the semantics are exactly once, that means every method is executed exactly once.

The choices of RMI invocation Semantics are defined as follows :-

Maybe Invocation Semantics:- With maybe invocation semantics, the remote method may be executed once or not at all. may be semantics arises when none of the fault tolerance measure is applied. This can suffer from the following types of failure:-

-> Omission failures if the invocation or result message is lost;

-> Crash failures when the server contains the remote object fails

Invocation Semantics:-

fault tolerance measures			Invocation Semantics
retransmit request message	Duplicate filtering	Re-execute procedure	
NO	Not applicable	Not applicable	maybe
Yes	NO	re-execute procedure	At-least-once
Yes	Yes	Retransmit reply	At-most-once.

If the invocation message was lost, then the method will not have been executed. on the other hand, the method may have been executed and the result message lost. -> crash

failure may occur either before or after the method is executed. moreover in an asynchronous system, the result of executing the method may arrive after the timeout.

At-least-once invocation Semantics:-

With at-least-once invocation Semantics the invoker receives either a result, in which case the invoker knows that the method was executed at least once or an exception informing it that no result was received. At-least-once invocation semantics can be achieved by retransmission of request messages. At-least-once invocation semantics can suffer from the following types of failure

Crash failures When the server containing the remote object fails!

arbitrary failures. In cases when the invocation message is retransmitted, the remote object may receive it and execute the method more than once.

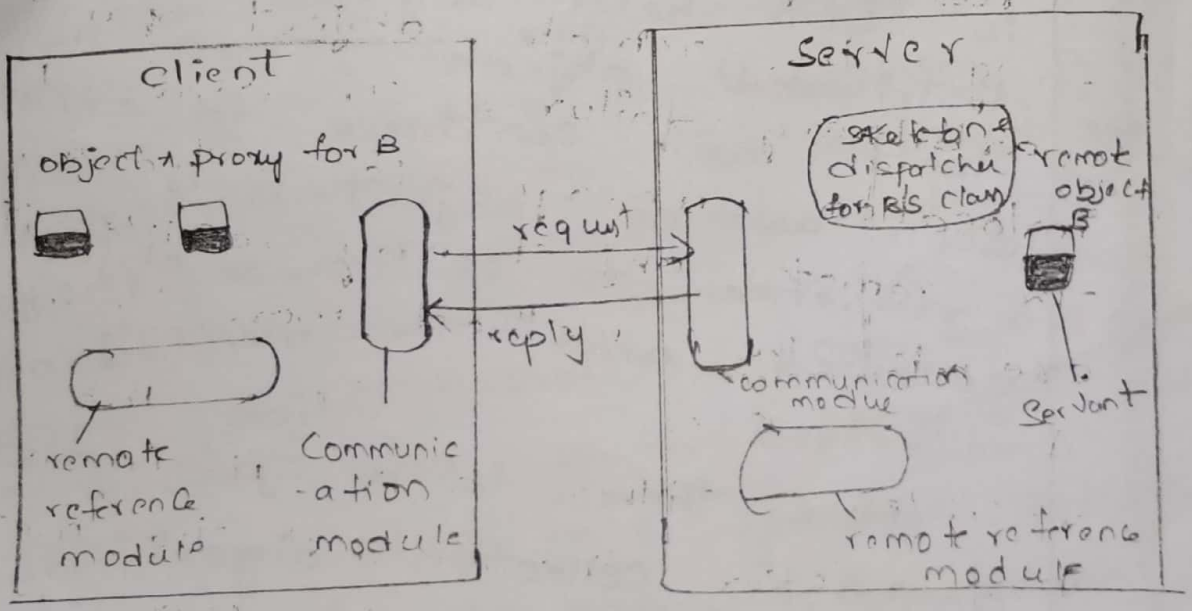
At-most-Once Semantics:- The invoker receives either a result in which case the invoker knows that the method was executed exactly once, or an exception informing if that no result was received, in which case the method will have been executed either

once or not at all.

Implementation of RMI :-

Several objects and modules are involved in achieving a remote method invocation in which an application level object A invokes a method in remote application-level object B for which it holds a remote object reference. This section discusses the roles of each of the components, dealing first with the communication and remote reference modules and then with the RMI software that runs over them.

role of proxy and skeleton in rmi :-



Communication module:- The two cooperating Communication modules carry out the request reply protocol, which transmits request and reply messages between client and server. The communication module uses only the first three items which specify the message type, its requestId and the remote reference of the object to be invoked. The methodId and all the marshalling and unmarshalling is the concern of RMI software.

The communication module in the server the dispatcher for the class of the object to be invoked, passing on its local reference, which it gets from the remote reference module in return for the remote object identifier in the request message

Remote reference module:- It is responsible for translating between local and remote object references for creating remote object references. The remote reference module in each process has a remote object table. The table includes:-

→ An entry for all remote objects held by the process

→ An entry for each local proxy.

Servants:- A servant is an instance of a class which provides the body of

of remote object. It is the servant that eventually handles the remote requ passed on by the corresponding skeleto

→ The RMI software:-

This consists of a layer of softwa between the application-level objects and the communication and remote reference modules. The roles of middlew are objects are as follows:-

Proxy:- The role of proxy is to make rmi transparent to clients by behavin like a local object to the invoker; but instead of executing an invocation forwards it in a message to a remot object. It hides the details of the remot object reference, the marshalling of arguments, unmarshalling of results and sending & receiving of messages from the client.

Dispatcher:- A server has one dispatcher and skeleton for each class represen -ting a remote object. In our exampl the server has a dispatcher and skeleto for the class of remote object B. The dispatcher receives the request message from the communication module. It uses the methodId to select the appropriate method in the skeleton, passing on the request message. The

dispatcher and proxy use the same allocation of method ids to the methods in the remote interface.

Skeleton:- The class of a remote object has a skeleton, which implements the methods in remote interface. A skeleton method unmarshals the arguments in the request message and invokes the corresponding method in the servant. It waits for the invocation to complete and then marshals the result together with any exceptions in a reply message to the sending proxy's method.

Distributed Garbage Collection:

The aim of a distributed garbage collector is to ensure that if a local or remote reference to an object is still held anywhere in a set of distributed objects, then the object itself will continue to exist, but as soon as no object any longer holds a reference to it, the object will be collected and the memory it uses recovered.

Here there is a Java distribute garbage collection algorithm, which is similar to the one described by Birrell et al. [1995]. It is based on reference counting. Whenever a remote

object reference enters a process, a pro will be created and will stay there for as long as it is needed. The process when there is no longer a object lives should be informed of the new proxy at the client. Then later when there is no longer a proxy at the client, the server should be informed. The distributed garbage collector works in cooperation with the local garbage collector as follows:-

* each server process maintains a set of the names of the processes that hold remote object reference for each of its remote objects. for example, B.holders is the set of client process that have proxies for object B. This set can be held in an additional column in the remote object table.
* When a client c first receives a remote reference to a particular remote object, B, it makes an addRef(B) invocation to the server of that remote object and then creates a proxy the server adds c to B.holders.
* When a client c's garbage collector notices that a proxy for remote object B is no longer reachable, it makes a removeRef(B) invocation to the corresponding server and then

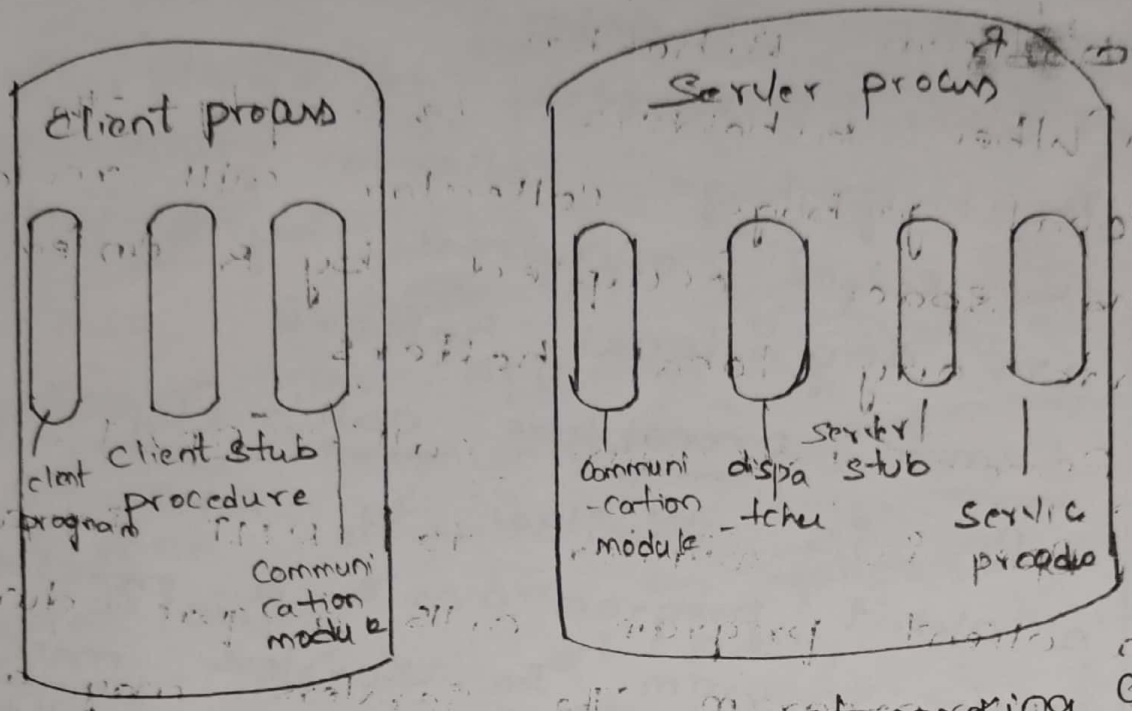
deletes the proxy, the server removes c from B.holders

* When B.holders is empty, the server's local garbage collector will reclaim the space occupied by B unless there are any local holders.

* Remote procedure call :- [RPC] :-

Rpc is similar to RMI in which a client program calls a procedure of server program. The servers may act as clients of other servers to allow chains of Rpc's. The procedures available for calling remotely are defined by the server process in its service interface. This type of service is like a single remote object containing stat and methods. It does not support remote object interfaces because it doesnot have the ability to create new instances of objects.

like RMI, Rpc can also be implement -ed to have one of the invocation semantics choices such as at-least-once or at-most-once that are generally chosen. It is implemented generally over a request-reply protocol that the omission of object references simplify from request messages.



Sun Rpc :- It is open networking Compu -ng Rpc, As described by the RPC 183 Srinivasan (1995). It was developed for client server communication in the Sun network File system. It is available with NFS installation and provides as a part of the various Sun and other unix operating systems. Their is liberty for implementors over using Rpc over either UDP or TCP when oper -ted with UDP the length of request and reply messages is limited in length theoritically upto 64 kb, but practically to 8 or 9 kb. This Sun Rpc with UDP make use of atleast once call Semanti Interface Definition language :- XDR, the Sun interface language was developed basically for specifying representation of data and later it was extended to become an interface detinition

language. The ... language defines
services of sun RPC by elucidating a
set of procedure definitions together
with supporting type definitions.
Input-output parameter:- only input-output
parameter is permitted. Hence, procedures
utilizing multiple parameters have to them
as parts of single structure. Single result
returns the output parameter.
Procedure Signature:- It includes the result
type, the name of the procedure and
the input parameter type. The type of
both input-output parameter may indicate
either a single value or a structure
consisting several values.
Interface Compiler:- The rpcgen generates
the following
1. client stub procedures
2. The main server procedure, dispatcher
and server stub procedure
3. for use by dispatcher and client and
server stub procedures XDR marshalling
and unmarshalling procedures.

* Events & Notifications:-

The idea behind the use of events
is that one object can react to a
change occuring in another object.
Notifications of events are essentially
asynchronous and determined by their

receivers

Events and notifications can be used in a wide variety of different applications, for example, to communicate that a shape added to a drawing, a modification to a document, the fact that person has entered, or left a room, or that a piece of equipment or an electronically tagged book is at a new location. Distributed event-based systems have two main characteristics. Heterogeneous: When event notifications are used as a means of communication between objects, components in a distributed system that were not designed to interoperate, can be made to work together. All that is required is that event-generating objects publish the types of events they offer, and that other objects subscribe to events and provide an interface for receiving notifications.

Asynchronous: Notifications are sent asynchronously by event generating objects to all the objects that have subscribed to them to prevent publishers needing to synchronize with subscribers - publishers and subscribers need to be decoupled

Mushroom is a distributed event based system designed to support collaborative work, in which the user interface displays objects representing users and information objects such as document and notepads within shared workspaces called network places.

Participants in distributed event notification:

The architecture is designed to decouple the publishers from the subscribers, allowing publishers to be developed independently of their subscribers, an as far as possible limiting the work imposed on publishers by subscribers. The main component is an event service that maintains a database of published events and of subscribers interests.

The role of participating objects are as follows:-

The object of interest:- This is an object that experiences changes of state, as a result of its operation being invoked. Its changes of state might be of interest to other objects. This description allows for events such as a person wearing an active badge entering a room, in which case the room is the object of interest and operation consists of adding

information about the new person in its record of who is in the room. The object of interest is considered as part of event service if it transmits notifications.

Event:- An event occur at an object of interest as the result of the completion of a method execution.

Notification:- It is an object that contains information about an event.

Subscriber:- It is an object that has subscribed to some type of events in another object.

publisher:- It is an object that declares that it will generate notifications of particular type of event. A publisher may be an object of interest or an observer.

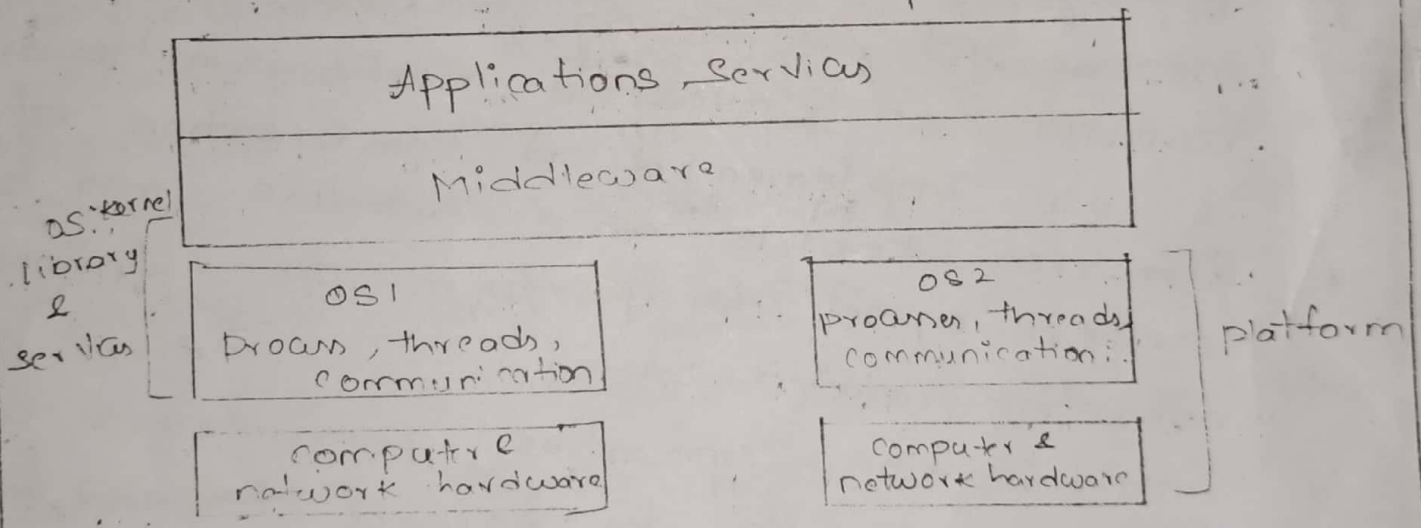
Observer objects:- The main purpose of an observer is to decouple an object of interest from its subscribers. An object of interest can have many different subscribers with different interests

Operating System Support

Operating System layer:-

Users will only be satisfied if their middleware - os combination has good performance Middleware runs on a variety of os - hardware combinations at the nodes of a distributed system. The os running at a node - a kernel and associated user-level services. Middleware utilizes a combination of these local resources, to implement its mechanism for remote invocations between objects or processes at the nodes

System Layers



Encapsulation:- They should provide a useful service interface to their resources that is a set of operations that meet their clients needs. Details such as management of memory and devices used to

implement resources should be hidden from clients.

Protection :- Resources require protection from illegitimate acesses - for example, files are protected from being read by users without read permissions and device registers are protected from application processes.

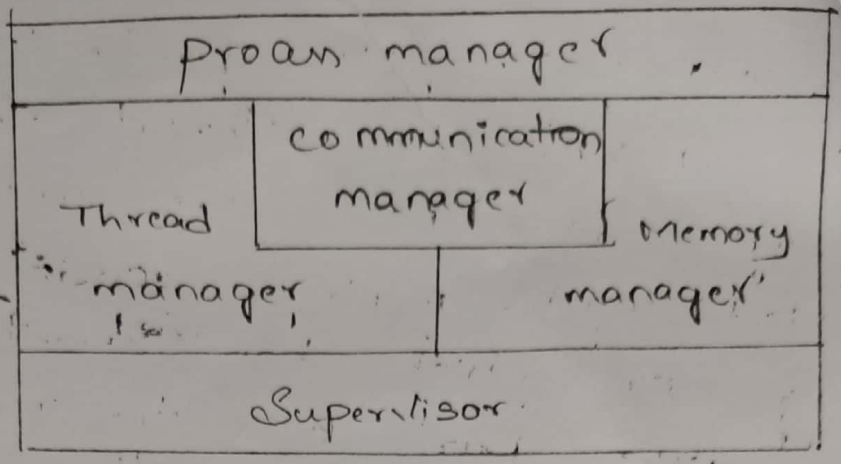
Concurrent procesing :- clients may share resources and access them concurrently resource managers are responsible for achieving concurrent transparency.

clients access resources by making for example rmi to a server object or system calls to a kernel. We call a means of accesing an encapsulated resource an invocation mechanism, however it is implemented. A combination of libraries, kernels and servers may be called upon to perform the following invocation-related tasks.

Communication :- operation parametrs and results have to be passed to and from resource managers over a network or within a computer

Scheduling :- When an operation is invoked, its procesing must be schedulded with the kernel or server.

Core OS functionality:-



The core OS Components are as follows:-

Process manager:- Handles the creation of and operations upon proamses. A proam is a unit of resource management, including an address space and one or more threads.

Thread Manager:- Thread creation, Synchronization and scheduling. Threads are schedulable activities attached to proamses.

Communication manager:- communication blw threads attached to different proamses on the same Computer. Some kernels also support Communication blw threads in remote proamses.

Memory manager:- management of physical and virtual memory.

Supervisor:- dispatching of interrupts, system call traps and other exceptions, control memory management unit and hardware caches

Protection:-

Resources require protection from illegit-mate accesses. The threat to a system's integrity does not come only from maliciously contrived code. Benign code that contains a bug or which has unanticipated behaviour may cause part of the rest of the system to behave incorrectly.

We can also employ hardware support to protect modules from one another at the level of individual invocations, regardless of the language in which they are written. To operate the scheme on a general-purpose computer, we require a kernel.

Kernels and protection

Kernel is a program that is distinguished by the facts that it always runs and its code is executed with complete access privileges for the physical resources on its host computer. It can control the memory management unit and set the processor registers so that no other code may access the machine's physical resources except in acceptable ways.

A kernel process executes with the processor in supervisor (privileged) mode; the kernel arranges that the other

processes execute in user (unprivileged) mode. The kernel also sets up address spaces to protect itself and other processes from the accesses of an aberrant process.

When a process executes application code it executes in a distinct user-level address space for that application. When the same process executes kernel code it executes in kernel's address space. The process can safely transfer from a user-level address space to the kernel's address space via an exception such as an interrupt or system call trap. It is implemented by a machine-level TRAP instruction, which puts the processor into Supervisor mode and switches to the kernel address space.

* Processes and Threads:

Process consists of an execution environment together with one or more threads → thread is the operating system abstraction of an activity. An execution environment is the unit of resource management: a collection of local kernel-managed resources to which its threads have access. An execution environment primarily consists of:

→ an address space;

→ Thread Synchronization and communicatio resources such as semaphores and communication interface.
→ higher-level resources such as open files and windows.

Execution environment are normally expensive to create and manage, but several threads can share them. Threads can be created and destroyed dynamically as needed.

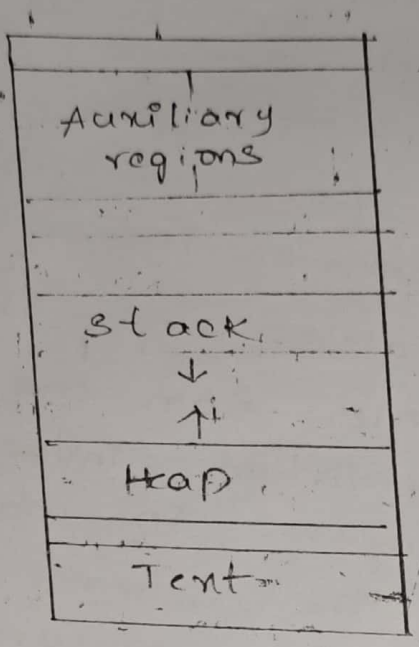
An execution enviroment provides protection from threads outside it, But some kernels allow the controlled sharing of resources such as physical memory between execution environments residing at the same computer.

Sec-2

1. Address Spaces:-

It is a unit of management of a process's virtual memory. It is large (upto 2^{32} bytes and sometimes upto 2^{64} bytes) and consists of one or more regions seperated by inaccessible areas of virtual memory. A region is an area of contiguous virtual memory that is accessible by the threads of the owning process. regions do not overlap. Each region is specified by the following properties:-

-> its extent (lowest virtual addres
-> read/ write/ execut permissions for
proam's threads;
-> Whether it can be grown upwards
doonwards



The model is page-oriented. There is need to support a separate stack for each thread. Allocating a separate stack region to each thread makes it possible to detect attempts to exceed the stack limits and to control each stack's grow-th. unallocated virtual memory lies beyond each stack region and attempts to acass this will cause an exception. The alternative is to allocate stacks for threads on the heap, but then it is difficult to detect when a thread has exceeded its stack limit.

Another motivation is to enable files in general - and not just the text

and data sections of binary files to be mapped into the address space. A mapped file is one that is accessed as an array of bytes in memory. The virtual memory system ensures that accesses made in memory are reflected in the underlying file storage.

The need to share memory between processes and the kernel is another factor leading to extra regions in address space. A shared memory region is one that is backed by the same physical memory as one or more regions belonging to other address space

Creation of a new process :-

The creation of a new process has traditionally been an indivisible operation provided by the operating system. The design of the process creation mechanism is to take account of the utilization of multiple computers. Consequently, the process support infrastructure is divided into separate system services.

The creation of new process can be seperated into two independent aspects:

-> The choice of a target host for example, the host may be choosen from among the nodes in a cluster of

Computers acting as a Comput server.
→ The creation of an execution environment

↳ Choice of proces host :-

The choice of node at which the new proces will reside - the proces allocation decision - is a matter of policy. In general, proces allocation policies ranges from always running new processes at their originator's workstation to sharing the processing load between a set of computers.

The transfer policy determines whether to situat a new proces locally or remotly this may depent.

The location policy determines which node should host a new proces selected for transfer. This decision may depend on the relative loads of nodes, on their machine architectures.

proces location policies may be static or adaptive. There are two cases, in first case one load manager component and in the second there are several organized in tree structure. Load manager collect information about the nodes and use it to allocate new processes to nodes.

↳ Creation of a new execution environment :

Once the host Computer has been selected, a new process requires an execution environment consisting of an address space with initialized contents.

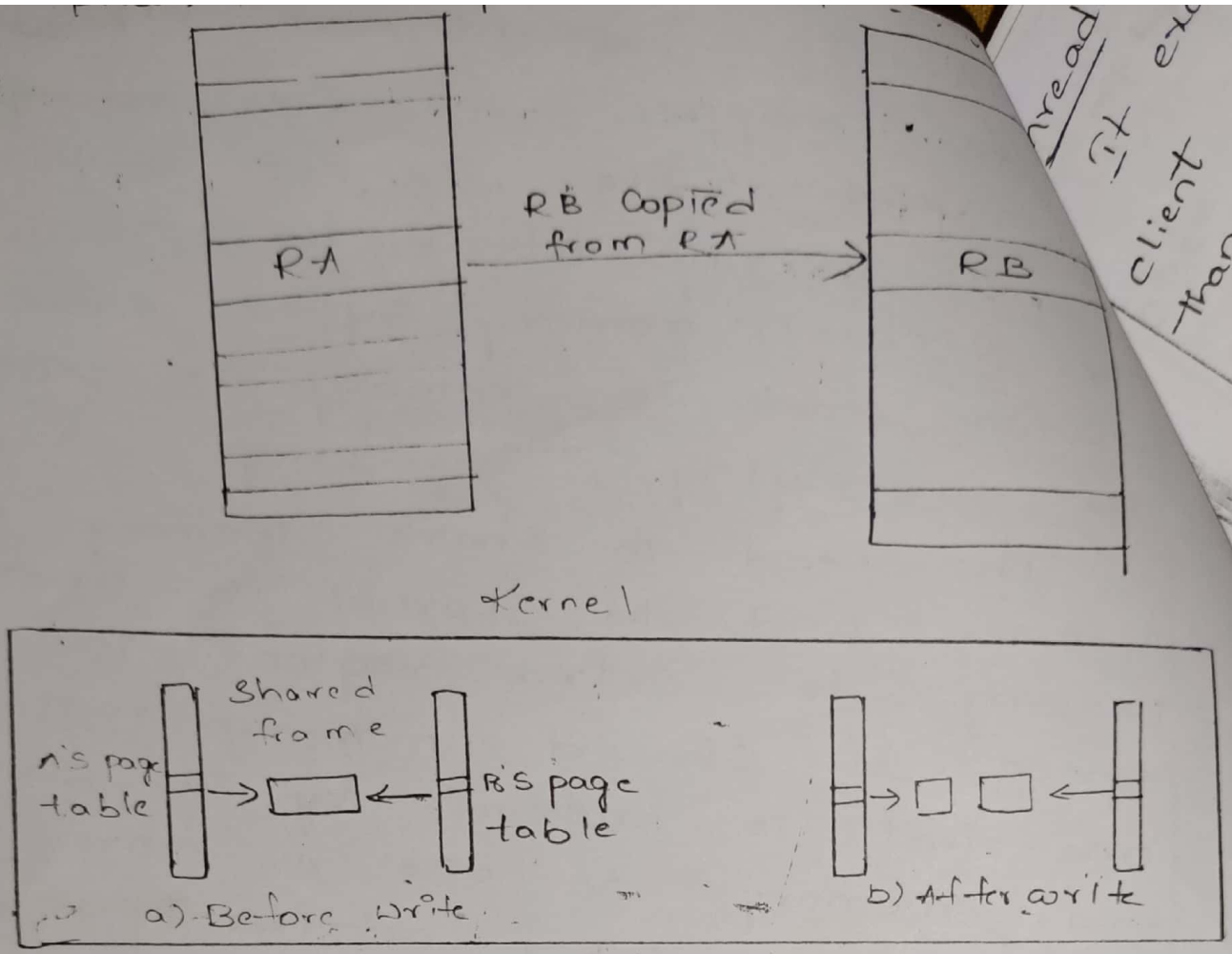
There are two approaches to defining and initializing the address space of newly created process. The first approach is used where the address space is of statically define format.

Alternatively, the address space can be defined with respect to an existing execution environment. for ex, the newly created child process physically shares the parent's text region and has heap and stack regions that are copies of the parents in extent. This scheme has been generalized so that each region of parent process may be inherited by the child process. An inherited region may either be shared with or logically copied from the parent's region. When parent and child share a region, the page frames belonging to the parent's region are mapped simultaneously into the corresponding child region.

There are some techniques to determine this environment:

→ Copy-on-write.

client and

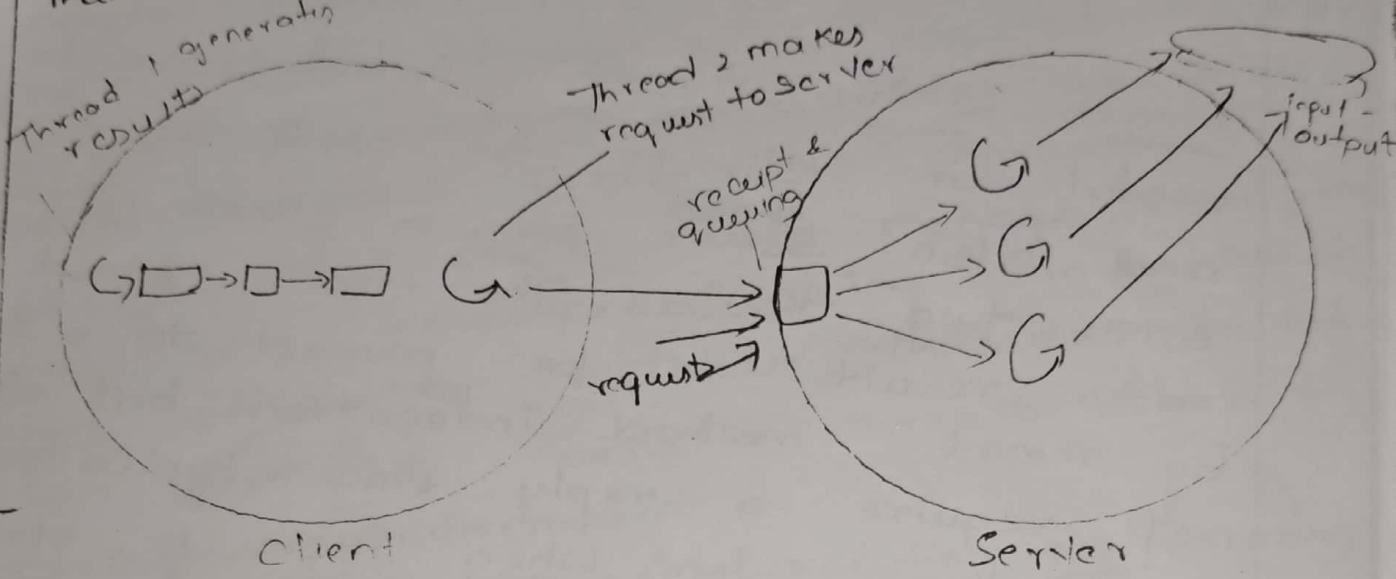


Copy-on-write is a general technique for example, it is also used in copying large messages, so we take some time to explain its operation. for example of regions RA & RB whose memory is shared copy-on-write between two pro cess A and B

—

Threads :-

It examines the advantage of enabling client and Server processes to possess more than one thread.



The Server has a pool of one or more threads, each of which repeatedly removes a request from a queue of received requests and processes it. We shall not concern ourselves for the moment with how the request and processes it. We donot concern for the moment with how the requests are received and queued up for the threads. Also, for the sake of Simplicity, we assume that each thread applies the Same procedure to process the requests

consider the maximum Server throughput measured in client requests handled per second, for different numbers of threads.

If a single thread has to perform all processing, then the turnaround time for handling any request is on average 2 + 8 = 10 milliseconds. So this server can handle 100 client request per second. Any new request messages that arrive while the server is handling a request are queued at the server port

SET-3 ① **Architectures for multi-threaded servers:-**

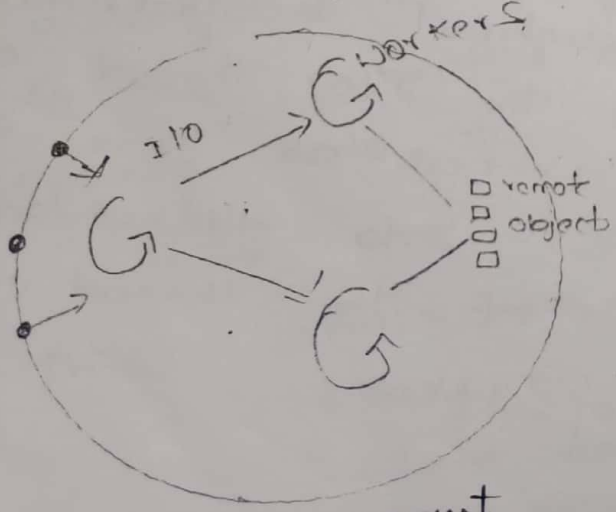
To describe the various ways of mapping requests to threads within a server, we summarize Schmidt [1998], who describes the threads architecture of various implementations of CORBA object Request Broker (ORB).

Thread-per-request architecture:-

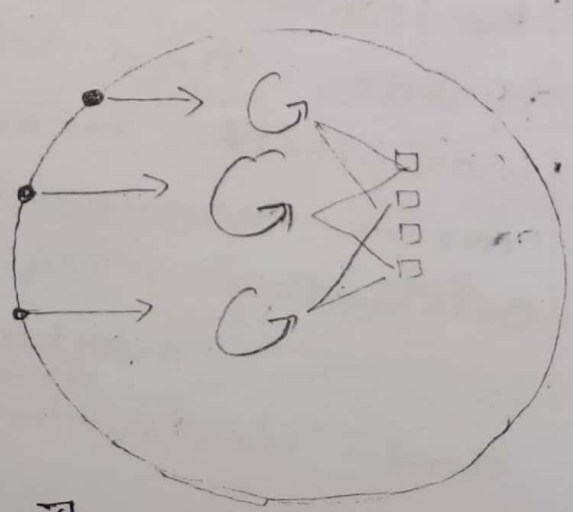
The I/O thread spawns a new worker thread for each request, and that worker thread destroys itself when it has processed the request against its designated remote object. This architecture has the advantage that the threads do not contend for a shared queue, and throughput is potentially maximized because the I/O thread can create as many workers as there are outstanding requests. Its disadvantage is the overhead of the thread creation and destruction operations.

Thread-per-Connection architecture: It associates a thread with each connection. The server creates a new worker thread when a client makes a connection and destroys the thread when the client closes the connection. In between, the client may make many requests over the connection targeted at one or more remote objects. The thread-per-object architecture associates a thread with each remote object. An I/O thread receives request and queues them for the workers, but this time there is a per-object-queue

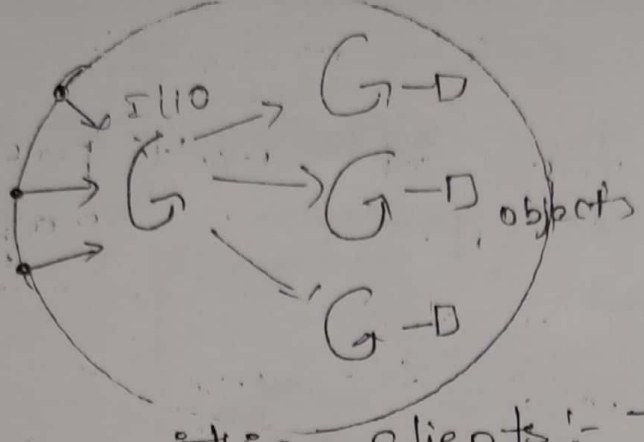
In each of these last two architectures the Server benefits from lowered thread-management overheads compared with the thread-per-request architecture. Their disadvantage is that clients may be delayed while a worker thread has several outstanding requests but another thread has no work to perform.



Thread-per-request



Thread-per-Connection



Thread per-

Threads within clients :- Threads can be useful for clients as well as servers and also shows server by a client process with two threads. The first thread gene-rates results to be passed to a server by remote method invocation, but does not require a reply. RMI typically block the caller, even when there is strictly no need to wait. This client process can incorporate a second thread, which performs the RMI and blocks while the first thread is able to continue computing further results. The first thread places its results in buffers, which are emptied by the second thread. It is only blocked when all the buffers are full.

The case for multithreaded clients is also evident in the example of web browsers, users experience substantial delays while pages fetched ; it is essential, therefore, for browsers to handle multiple concurrent requests for web pages.

Threads versus multiple processes:-

In the case of multi processor, other computation. The reader may have noted, however, that the same overlap could be achieved through the use of multiple single-threaded processes. Why we prefer multi-threaded process is threads are cheaper to create and manage than processes and resource sharing can be achieved more effectively between threads than between processes because threads share an execution environment

State associated with execution environments and threads:-

Execution Environment	Thread
Address space tables	Saved processor registers
communication interfaces, open files	priority and execution state (such as Blocked)
Semaphores, other Synchro-nization objects	Software interrupt handling information
List of thread identifiers	execution environment identifier.

The table shows that an execution environment and the threads belonging to it are both associated with pages belonging to the address space held in main memory and data and

1. → Schedule another thread to run
instructions held in hardware caches

We can summarize a comparison of processes and threads as follows;

* Creating a new thread within an existing process is cheaper than creating a process.
* More importantly, switching to a different thread within the same process is cheaper than switching between threads belonging to different processes.
* Threads within a process may share data and other resources conveniently efficiently compared with seperate processes
* But, by the same token, threads within a process are not protcted from one another.

Threads programming:-

Thread programming is concurrent programming, as traditionally studied in, for example, the field of operating systems. It refers the concurrent pro-gramming concepts, which are explained by Bacon [2002]: race condition, critical section, monitor, condition, Variable, Semaphore.

Much threads programming is done in a conventional language such as C, which has been augemented with a threads library.

Java Thread Constructor & management methods

Thread (ThreadGroup group, Runnable target, String name)
Creates a new thread in the SUSPENDED stat, which will belong to group and be identified as name; the thread will execute the run() method of target

setpriority(int newpriority), getpriority()
Set and return the threads priority.

run()
A thread executes the run() method of its target object, if it has one, and otherwise its own run() method (Thread implements Runnable).

start()
change the stat of the thread from SUSPENDED to RUNNABLE

sleep(int milliSecs)
Cause the thread to enter the SUSPEND -ED stat for the specified time

yield()
entr the READY stat and invoke the scheduler

destroy()
Destroy the thread

Thread lifetimes :- A new thread on the same Java virtual mach its creator, in the SUSPENDED & After it is made RUNNABLE with start() metho, it executes the run method of an object designated its constructor. The JVM and the threads on top of it all executes in a process on top of the underlying operating system. Threads can be assigne a priority, so that a Java implementatio that supports priorities will run a partic -lar thread in preferena to any thread with lower priority. A thread ends its life when it returns from the run() method or when it destroy() method is called.

Thread Synchronization :-

Java provides the Synchronized keyword for programmers to designat the well-known monitor construct for thread coordination. programmers designat either entire methods or arbitrary blocks of code as belonging to a monitor associated with an individual object. Ide could serialize the actions of I/O and worker threads by designating addTo() and removeFrom() methods in the Queue clan as Synchronized methods.

Java Thread Synchronization calls.

Thread.Join(int millisecs)
Blocks the calling thread for upto the specified time untill thread has terminated.

Thread.interrupt()
interrupts thread: causes it to return from a blocking method call such as sleep()

object.wait(long millisecs, int nanosecs)
Blocks the calling thread until a call made to notify() or notifyAll() on object wakes the thread, or the thread is interrupted or the specified time has elapsed

object.notify(), object.notifyAll()
wakes, respectively, one or all of any threads that have called wait() an object.

Threads Scheduling:

It is between preemptive and non-preemptive Scheduling of threads. In preemptive Scheduling, a thread may be Suspended at any point to make way for another thread. In non-preemptive Scheduling, a thread runs until it makes a call to the threading system when the System may de-schedule

it and schedule another thread to run.

The advantage of non-preemptive scheduling is that any section of code that does not contain a call to the threading system is automatically a critical section. On the other hand, non preemptively scheduled threads cannot take advantage of a multiprocessor, since they run exclusively. care must be taken over long running sections of code that do not contain calls to the threading system.

=

V

UNIT-V

DISTRIBUTED FILE SYSTEMS

Introduction:

File Systems were originally developed for centralized computer systems and desktop computers as an os facility providing a convenient programming interface to disk storage. They subsequently acquired features such as access control and file-locking mechanisms that made them useful for the sharing of data and programs. Distributed file systems support the sharing of information in the form of files and hardware resources in the form of persistent storage throughout an intranet. A well-designed file service provides access to files stored at a server. with performance and reliability Similar to, and in some cases better than, files stored on local disks. Their design is adapted to the performance and reliability Similar to, and characteristics of local networks and hence they are most effective in providing shared persistent storage for use in intranets.

file System modules:-

Directory module relates file names to file IDs

File module relates file IDs to particular files

Access control module checks permission for operation requested.

file access module reads or writes file attributes.

Block module accesses and allocates blocks

Device module disk I/O and buffering

Characteristics of file Systems:

File Systems are responsible for the organization, storage, retrieval, naming, sharing and protection of files. They provide a programming interface that characterizes the file abstraction, freeing programmers from concern with the details of storage allocation and layout. files are stored on disks or other non-volatile storage media.

Files contain both data and attributes. The data consist of sequence of data items (8-bit bytes), accessible by operations to read and write any portion of the sequence. The attributes are held as a single record contains information such as length of the file, timestamps, file type, owner's identify and access-control lists. file systems are designed to store and manage large numbers of files, with facilities for creating, naming and deleting files.

File attribut record structure.

- File length
- creation timestamp
- Read timestamp
- write timestamp
- Attribute timestamp
- Reference count
- owner
- File type
- Access Control list

2b File System operations :-

The main operations on files that are avlailable to applications in UNIX systems. These are the system calls implemented by the kernel; application programmers usuasly access them through library procedures.

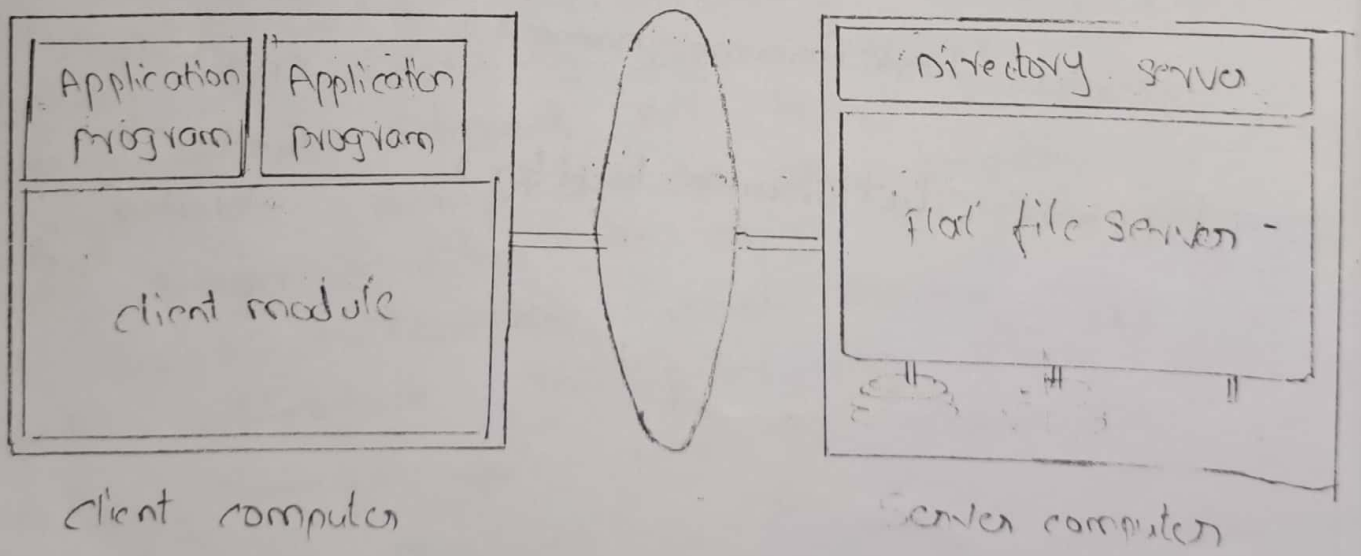
The file system is responsible for applying access control for files. In local file systems such as UNIX, it does so when each file is opened checking the rights allowed for the user's identity in the access control list against the mode of access requested in the open system call.

UNIX file System operations:-

filedes = open(name, mode)	opens an existing file with given name
filedes = create(name, mode)	creates a new file
Status = close(filedes)	closes the open file
count = read(filedes, buffer, n)	transfers n bytes from the file referenced by filedes to buffer
count = write(filedes, buffer, n)	transfers n bytes to the file referenced by filedes from buffer
pos = lseek(filedes, offset, where)	moves the read-write pointer to offset.
status = unlink(name)	removes the file name from directory
status = link(name1, name2)	adds a new name (n2) for file (n1)
Status = stat(name, buffer)	Gets the file attributes for file name into buffer

File Service Architecture :-

SE 1-2
2 An architecture that offers a clear separation of main concerns in providing access to files is obtained by structuring the file service as three components - a flat file service, a directory service, a client module. A flat file service and directory service each export an interface for use by client programs and their RPC interfaces, taken together, provide a comprehensive set of operations for access to files. The client module provide single programming interface with operations on files.



2a Flat file Servlice:- The flat file
Concerned with implementing. operati
the contents of files unique file
Ers (UFIDS) are used to refer to file
all requests for flat file Servlice op
-ons. The division of responsibilities
the file Servlice and directory Servlic
based upon the use of UFIDS. UFIDS
are long sequenas of bits choosen
that each file has UFID that is unique
among all of these files in distribut
system.

2b Flat file Servlice operations:-

Read (Field, i, n) → Data throws BadPosition	If $1 \le i \le$ length (File): Read Sequence of upto n items from a file starting at Item i & returns it in Dat
curit (Field, i, Data) throws Bad.position	If $1 \le i \le$ length (File) + 1: curit sequence of Data to a file
create() → Field	creates a new file of length 0 & delivles a UFID
Delete (Field)	removes file from file store.
GetAttributes (Field) → attr	returns the file attribut for the file
SetAttributes (Field, attr)	sets the file attributes

2) Directory Service :- The directory service provides a mapping between text names for files and their UFIDs. Clients may obtain the UFID of a file by quoting its text name to the directory service. The directory service provides the functions needed to generate directories, to add new file names to directories and to obtain UFIDs from directories. It is a client of the flat file service; its directory files are stored in files of flat file service. When a hierarchic file naming scheme is adopted, as in UNIX directories hold references to other

Directory Service operations:-

Lookup (Dir, Name) -> Field
-throws NotFound

locates the text name in directory & returns the relevant UFID. If name not found it throws an exception

AddName (Dir, Name, Field)
-throws NameDuplicate.

If Name is not in the directory, adds (Name, File) to the directory & updates the file's attribute record

UnName (Dir, Name)
-throws NotFound

if Name is in the directory: the entry containing Name removed from the directory.

GetNames (Dir, Pattern) -> NameSeq

if names returns all text names in directory

Client module :- A client module runs in each client computer, integrating and extending the operations of the flat file service and the directory service under a single application programming interface that is available to user-level programs in client computers.

FSA :- It is an abstract architectural model that underpins both NFS and AFS. It is based upon a division of responsibilities between threen modules. The architecture is designed to enable a stateless implementation of server module.

SUN NFS :-

SUN Microsystm's Network File System has been widely adopted in industry and in academic environment since its introduction in 1985. The design and development of NFS were undertaken by staff at sun microsystems in 1984. Although several distributed file services had already been developed and used in universities and research laboratories. NFS was the first file service that was designed as a product. The design and implementation of NFS have achieved success both technically and commercially.

To encourage its adoption as a standard, the definitions of the key

interfaces were placed in the public domain, enabling other vendors to produce implementations, and the source code for a reference implementation was made available to other computer vendors under licence. It is now supported by many vendors, and the NFS protocol is an Internet standard, defined in RFC 1813.

NFS provides transparent access to remote files for client programs running on UNIX and other systems. The client-server relationship is symmetrical, each computer in an NFS network can act as both client and server.

An important goal of NFS is to achieve a high level of support for hardware and OS heterogeneity. The design is OS independent. Implementation of NFS on high-performance multiprocessor hosts have been developed by several vendors and these are widely used to meet storage requirements in intranets with many concurrent users.

Andrew File System:-

Andrew is a distributed computing environment developed at carnegie mellon university for use as a campus computing and information system. The design of AFS reflects an intention

to support information sharing
large scale by minimizing client
communication. This was achieved
transferring whole files between ser
and client computers and caching th
at clients until the server receives
a more up-to-dat version. ∞.

* *

Peer-to-peer Systems: introduction:-

Peer-to-peer Systems aim to support useful distributed services and applications using data and computing resources available in the personal computers and workstations that are present on the Internet and other networks in ever increasing numbers. This is increasingly attractive as the performance difference between desktop and server machines narrows, and broadband network connection proliferate. Peer-to-peer applications enhance their scalability, reliability and security. These systems provide access to information resources located on computers throughout a network whether it be the Internet or a corporate network. Algorithms for the placement and subsequent retrieval of information objects are a key aspect of system design. Their design aims to deliver a service that is fully decentralized and self-organizing, dynamically balancing the storage.

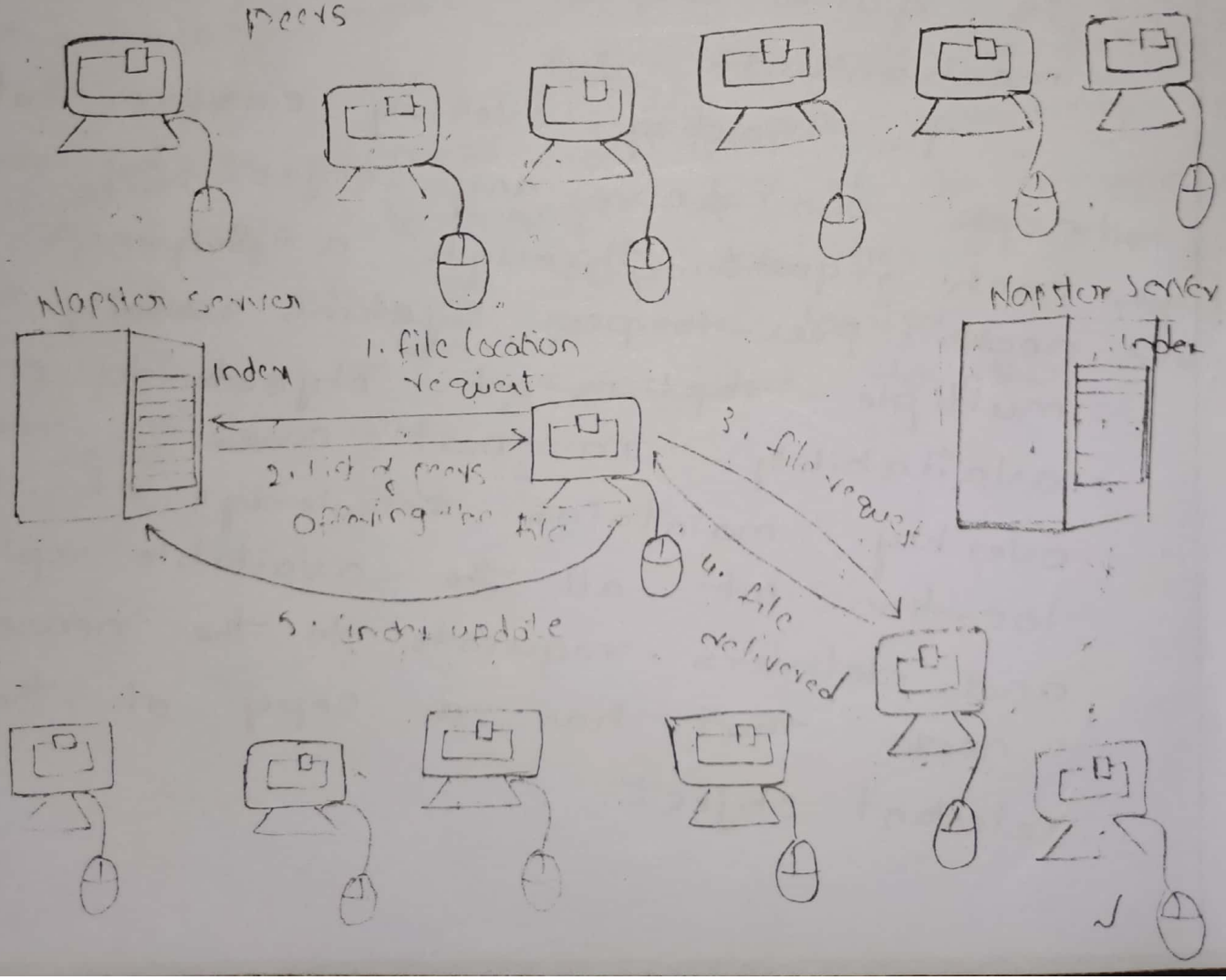
Charactristics:-

* Their design ensures that each other contributes resources to the system
* Although they may differ in the resources that they contribute, all the nodes in a peer-to-peer system have the same functi

* Napster took advantage of the special characteristics of the application for which it was designed in other ways

-> Music files are never updated; avoiding any need to make all the replicas of files consistent after updates

-> No guarantees are required concerning the availability of individual files - if a music file is temporarily unavailable it can be downloaded later. This reduces the requirement for dependability of individual computers and their connections to the internet



* Peer-to-peer middleware :-

Peer-to-peer middleware systems are designed specifically to meet the need for the automatic placement and subsequent location of the distributed objects managed by peer-to-peer systems and applications.

Functional requirements :- The function of peer-to-peer middleware is to simplify the construction of services that are implement across many hosts in a widely distributed network. To achieve this it must enable clients to locate and communicate with any individual resource made available to a service, even though the resources are widely distributed amongst the hosts.

Non-functional requirements :- To perform effectively, peer-to-peer middleware must also address the following.

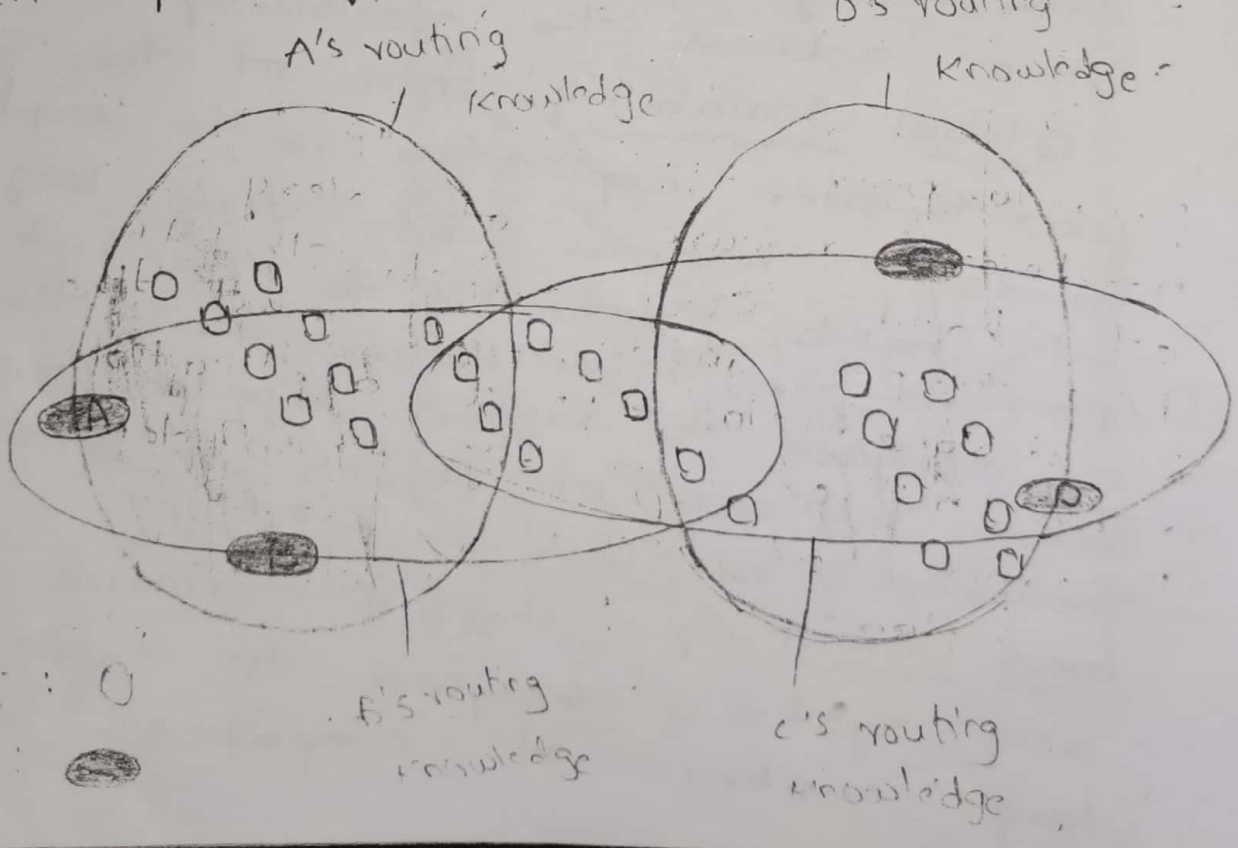
Global scalability :- one of the aims of peer-to-peer application is to exploit the hardware resources of very large number of hosts connected to the Internet. Peer-to-peer middleware must therefore be designed to support applications that access millions of objects on tens of thousands of hosts.

load balancing :- The performance of any system designed to exploit a large number of computers depends

upon the balanced distribution of workload across them. for the systems we are considering, this will be achieved by a random placement of resources together with the of replicas of heavily-used resources.

Optimization for local interaction between neighbouring peers :- The network distance between nodes that interact has a substantial impact on the latency of individual interactions, such as client requests for access to resources. Network traffic loading are also impacted by it

Security of data :- In global-scale systems with participating hosts of diverse ownership, trust must be built up by the use of authentication and encryption mechanisms to ensure the integrity and privacy of information.



2. routing overlays:-

A distributed algorithm of a routing overlay takes known as responsibility for locating nodes and obje
The name denotes the fact that middl
-ware takes the form of a layer that is responsible for routing requests from any client to a host that holds the object to which the request is addressed. The objects of interest may be placed and subsequently relocated to any node in the network without client involvement. It is termed an overlay since it implements a routing mechanism in the application layer that is quite seperate from any other routing mechanisms.

The routing overlay ensures that any node can acces any object by routing each request through a sequence of nodes. peer-to-peer systems usually store multiple replicas of objects to ensures availability. In that case, the routing overlay maintains knowledge of the location of all the available replicas and delivers requests to the nearest node that has a copy of the relevant object

3. The main task of a routing overlay is the following :-

1. A client wishing to invoke an operation on an object submits a request including the object's GUID to the routing overlay which routes the request to a node at which a replica of the object resides.

2. A node wishing to make a new object available to a peer-to-peer service computes a GUID for the object and announces it to the routing overlay which then ensures that the object is reachable by all other clients.

3. When clients request the removal of objects from the service the routing overlay must make them unavailable

Nodes may join & leave the service. When a node joins, the routing overlays arranges for it to assume some of the responsibilities of other nodes. When a node leaves, the responsibilities are distributed to other nodes.

COORDINATION & AGREEMENT

Introduction:-

failure assumptions:-

A various failure assumptions' include!-

-> The connection between each pair of process is made over reliable channe -ls because these channels mark the failures of network components.

-> The process do not depend on each other for forwarding the messages.

-> Hardware redundancy exists at certain points in the synchronous system which results in delivering the message on time without failure.

-> It is also assumed that procus can fail only by crashing.

Failure Detectors:-

It is a fundamental abstraction in distributed computing. failure detectors are used as buiding blocks to simplify the design of reliable distributed algorithms. In particular, we describe how failure detectors can factor out timing assumpti -ons to detect failures in distributed algorithms.

failure detectors can be defined as service which can be used to find whether the process has failed for detecting failures locally, each computer

runs algorithms that detect failures of each local process which is called as local failure detector.

The concept of unreliable failure detectors characterize two properties completeness and accuracy. It results Suspected and unsuspected. The term Suspected suggests that process has failed because it has been long time that it has received any message. In this method, suspicion may be wrong because the process may be running slowly because of some reason. The term unsuspected suggests that the process has not failed due to any reason and it is working correctly.

A reliable failure detector is accurate because it detects whether the process has failed or crashed, or it is getting executed correctly very accurated. If a process is failed it queried & if it does not receive any message for long time than it is declared as failed, or crashed.

unreliable failure detector can be implemented using algorithm. In this algorithm every process sends message. 'p is here' and maintain time record and if the process is taking more time to transmit

a message than usual and get message with T+D second the last message which it received
A reprot is sent to q by p that p is suspecters or else (p is here then it reports to q that p is (fine)

* Distributed mutual exclusion!-

If a collection of proam shared collection of resouras then often mutual exclusion is required to prevent interference and ensure consistency when accessing the resours. This is the critical section problem, familiar in operating system. In some cases shared resourse are manage by servers that also provide mechanisms for mutual exclusion

Algorithms for mutual exclusion!-

The performance of mutual exclusi -on algorithm can be calculated with following measures!-

1. Bandwidth!- The no. of messages transmi -tted in each entry and exit operati -on is directly proportion to the band width utilized

2. client Delay!- The delay that occur due to proames at each entry and exit operation of critical Section.

System throughput:- The number of critical section request fulfilled per unit time

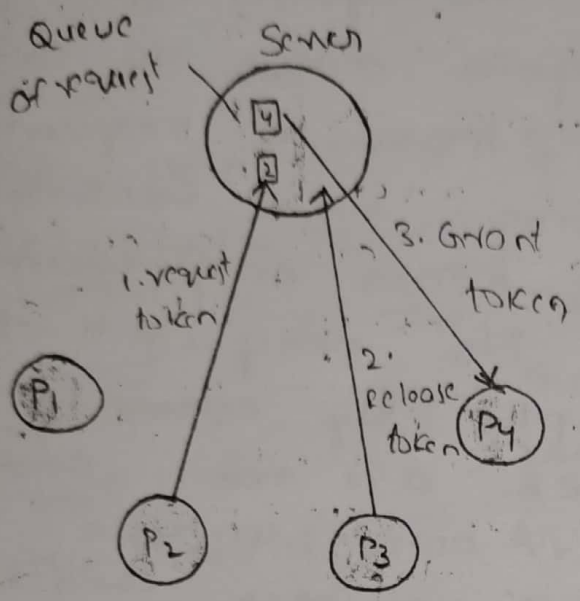
4. Synchronization Delay:- The time elapsed between when some proces leaves its critical section. System's throughput increases as their is decrease in Synchronization delay

Algorithms:

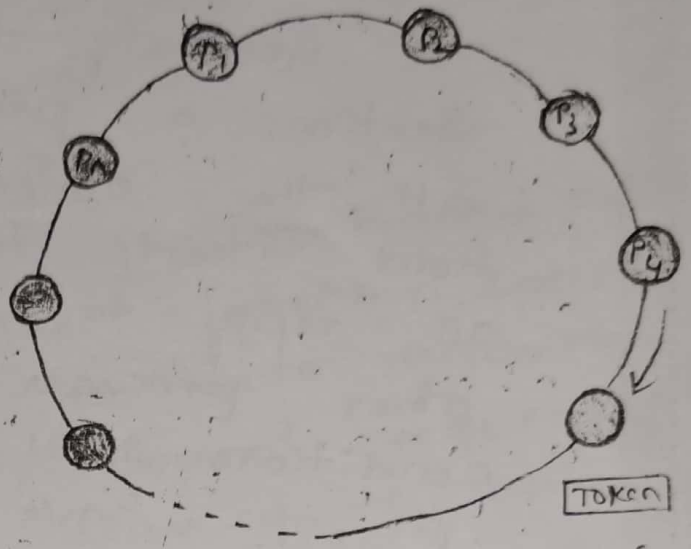
(1) Central Server algorithm:-

It can be seen as a token based algorithm. In this, the system maintains a token and makes sure that one proces at a time gets that token. The proces of granting a token is done by a server that grants the permission to enter the critical section on the request of client proces. If a proces does not have a token then it cannot enter the critical section. In order to enter the critical section, a proces requests the central server by sending a request menage and if no other proces is in critical section, then the central server immediately replies by granting the tokens. if other proces is in critical section then the server adds the requests in the queue list and waits till

the process exists, as the process exits it asks the server to return its token. Then the server grants the token to the next process.



Server managing a mutual-exclusion token for a set of processes



A ring of processes transferring a mutual exclusion token

The process P3 which is added to the queue, that already consists of P2 requests for token other hand where P1 exists the critical section, the server grants the permission to P2. process P4 is Constant as it doesn't need entry in the critical section. Although this algorithm is simple to implement but it can become a bottleneck. Inorder to overcom this problem, shared data can be distributed among several servers.

Ricart-Agrawala Algorithm:-

Ricart and Agrawala developed a distributed algorithm using multicast and logical clocks in the year 1981.

Generally to gain access to critical section, a process makes a request to all the other processes and considers to hold itself from access only when they all reply to say its okay. All the other processes reply by ensuring that conditions ME1 & ME3 are met each process P1, P2 have their own unique identifiers through which they send requests use lamport clocks and their ID's

The main steps involved in this algorithm is

On Initialization
    state := RELEASED;

To enter the section
    state := WANTED
    multicast request to all processes;
    T = request's timestamp;
    wait until (number of replies received = (N-1));
    state := HELD;

On receipt of request <Ti, Pi> at Pj (i≠j)
if (state = HELD or (state = WANTED and (T, Pj) < (Ti, Pi)))
then
    queue request from Pi without replying
else
    reply immediately to Pi;

end if'

To exit critical section.

state := RELEASED;

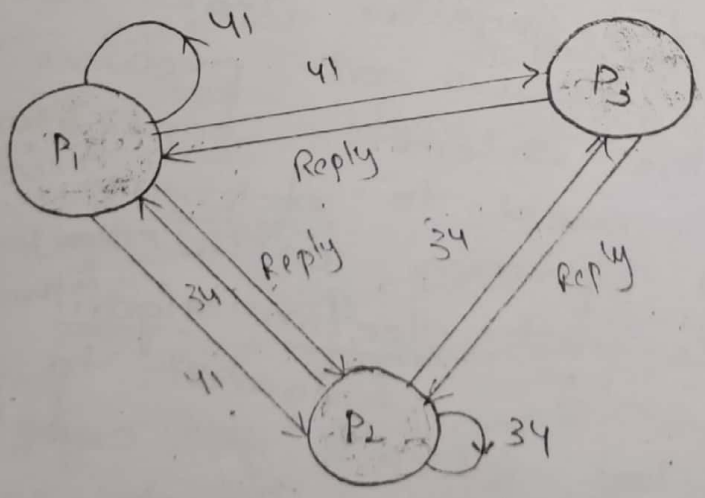
reply to any queued requests.

All the process send request to enter critical section including the timestamp and identity Pi. The process record their states when they are out of critical section as RELEASED and when they want to enter critical section as WANTED. & the process which is in critical section will record its state as HELD.

Whenever a process request entry is the critical section and all the processes reply at once then the process can gain entry at that time only. if processes do not reply to the request because process is already in the critical section then the process who requested entry in the critical section has to wait till the other process exists the critical section.

When 2 or more processes request entry in critical section it will not acapt from other processes requesting, un till it has recorded its timestamp. This multicast and logical locks algorithms meets the conditions of ME1, ME2, ME3.

t us consider an example of 3 roars P1, P2, P3. Whereas P1 & P2 requ to enter critical section simultaneously and P3 is not willing to entr. P1 recorded its timestamp as 19 and P2 recorded its timestamp as 23. When P3 receives requests from P1 and P2, it replies to them immediatly whereas whe P1 receives P2's request it finds that its own recorded timestamp is lower than P2 and it doesnot reply. When P2 receives P1's request it finds that P1's timestamp is lower compared to its own and replies thus P1 gains entry in critical section. When P1 exits it grants request to P2 to enter.



P1 & P2 request entering the critical Section Simultaneously.

This algorithm is expensive because it uses more bandwidth. performance can be improved by deciding that the proces which enters last in CS must again sends request to all the proceses again.

Maekawa's Voting algorithm :-

It is the first quorum-based mutual exclusion algorithm. The algorithm specifies that for a proces to have acces to the critical section, doesnot require its peer to grant acces to the critical section. Therefore, a single proces can get acces to the critical section. Here the permission is granted to the proces to enter subsets of their respective peers and the permission is given untill the subsets of any two proceses intersect the proceses make use of voting concept. i,e each proces votes for one another for gaining acces to the critical section. An adequate amount of votes must be collected by a candidat proces so as to enter the critical section. When the proceses under the intersection about two sets of votes occurs, a Safety property mei is established.

The algorithm defines the voting set VSi with each proces Pi (i = 1, 2, 3, 4, ... N) & VSi ⊆ {P1, P1, ... PN}. The voting set VSi are chosen such that ∀ i, j = 1, 2, 3, 4 ... N is,

1. Pi ∈ VSi
2. There exist at most one member common to any two voting sets.
   VSi ∩ VSj ≠ ∅
3. each and every process must hold a voting set of equal size
   |VSi| = K
4. each and every proces Pi, belongs to m of voting sets VSi.

The main steps involved in algorithm is

initialization
state = RELEASED;
voted = FALSE;
proces Pi request entring the C.S
state = WANTED;
making a multicast request to all processes present in VSi;
wait until (No. of replies received is = K);
state = HELD;
upon receiving the request from Pi at Pj
if (state = HELD) OR voted = TRUE

Then queue the request from Pi
else send a reply to Pi;
Voted = TRUE;
end if.
Inorder for Pi to exit C.S
Stat = RELEASED
if (queue of requests is non-empty) then
Remove the head of queue - from Pk, say;
Send reply to Pk;
Voted = TRUE;
else
Voted = FALSE;
End if

* Elections:-

* ELECTIONS:-

Election algorithm:- It is algorithm used for solving the problem of choosing an election coordinator.

Goal:- It ensure that each time an election occurs, all processes must agree upon the selected coordinator. Moreover, it is used to determine the non-crashed process with the highest ID.

Features of election Algorithm:- election algorithm is designed to choose a coordinator which is capable of choosing a process that can play the role of server. When the process which is playing the role of server retires

another election is needed to select the new Server or new Coordin for.

Any process calls the election to choos the new coordinator or Server. A Single process cannot call the election more than once at a time. Whereas, in principle the N processes can call N Concurrent elections.

A prominent requirement is elected process should be unique. The requirement for any specific run of the algorithm are as follows,

(i) El (Safely) :- All processes agree on the same elected process with the largest identifier. At the end of a run and the elected one should be a process that didnot crash yet

(ii) E2 (Live) :- All processes participated or crash during the process. The performance of an election algorithm can be measured by the total bandwidth utiliz and turn around time.

Bully Algorithm :-

It is a method in distributed computing for dynamically selecting a coordinator by process ID number.

This algorithm was devised by Garcia-Molina in 1982. Some of the assumption of this algorithm are,

1. each process knows the ID and address of every other process.
2. The system is synchronous and users timeout for identifying process failure/crash.

There are three types of messages in this algorithm. They are as follows,

(a) Election Message:- It is sent to announce the election.

(b) Answer Message:- This respond to the election message

(c) Coordinator message:- It is sent to announce the identity of the elected process

When a process notices that, the coordinator is no longer responding to request because of message timeout or failure of the Coordinator, It initiates the election process P holds an elections as follows,

(i) P sends an election message to all other processes with higher process IDs.

(ii) if no one responds with a higher process ID that it, wins the election and become coordinator.

(iii) if one of the process with a higher ID responds, P waits a certain amount of time for that process to broadcast itself as the leader. if it does not receive this message in time, it rebroad casts the election message

(iv) It gets an election message from

→ Multicast Communication is of various types that are as follows:-

Reliable Multicast:-

It should satisfy the following properties:-

1. Integrity:- A correct proam p must deliver a message at most once, p should belong to group(m) and m was send by a multicast operation by sender(m)
2. Validity: If a correct proam multicasts the message m then it will eventually transport m.
3. Agreement: If a correct proam transport message m, then all other correct proams in group(m) will eventually transport m.

Implementation of Reliable multicast:-

The two implementations are:-

1. B-multicast
2. Ip multicast

B-multicast: It can be explained with the help of following algorithm, that contains the primitives R-multicast and R-deliver.

On initialization

Received := {};

for proam p to R-multicast message m to group g B-multicast(g, m); // p∈g is entred as destination on B-deliver(m)

at process q with g = group(m) if (m
Then
Received := Received ∪ {m};
if (q ≠ p) then B-multicast (g, m); end if
R-deliver m;
end if.

IP multicast :- IP multicast is a method of sending Internet protocol (IP) datagrams to a group of interested receivers in a single transmission. It is a form of point-to-multipoint communication employed for streaming media & other applications.

FIFO Ordering :-

When a true process issues multicast (g, m) and multicast (g, m') then each true process delivering m' will deliver m before m'.

Implementation of FIFO ordering :- In FIFO implementation the multicast we achieve using sequence number is similar to what we achieve in one-to-one communication.

In this implementation consider only non-overlaping groups. The FIFO ordering is guaranted by the reliable multicast protocol that is defined on the top of IP multicast. A FIFO-ordered multicast can be constructed on top of any given basic multicast, for this, the

Variables such as sg^p & R_q^p can be used. these variables are held at process p from the reliable multicast protocol: the sg^p refers to the count of messages sent by p to q. where as R_q^q refers to the recorded sequence number of the latest message that has been delivered from q by p to group g.

Fault tolerance :-

It is an important feature that plays a key role in distributed systems. In ds, If one of the critical component of a system fail to function properly, it results in a partial failure. To overcome such partial failures, ds incorporate a design which tolerate faults and continues to operate even in the existence of faults. Such characteristics feature in ds is referred to as fault tolerance.

Due to some faults in hlw or slw programs generat inappropriate results and sometime may stop before the completion of the computation needed. So, the ability of a system to continue function -ing in the event of partial system failure. There are two approaches:

1. Hardware Redundancy
2. Software Redundancy

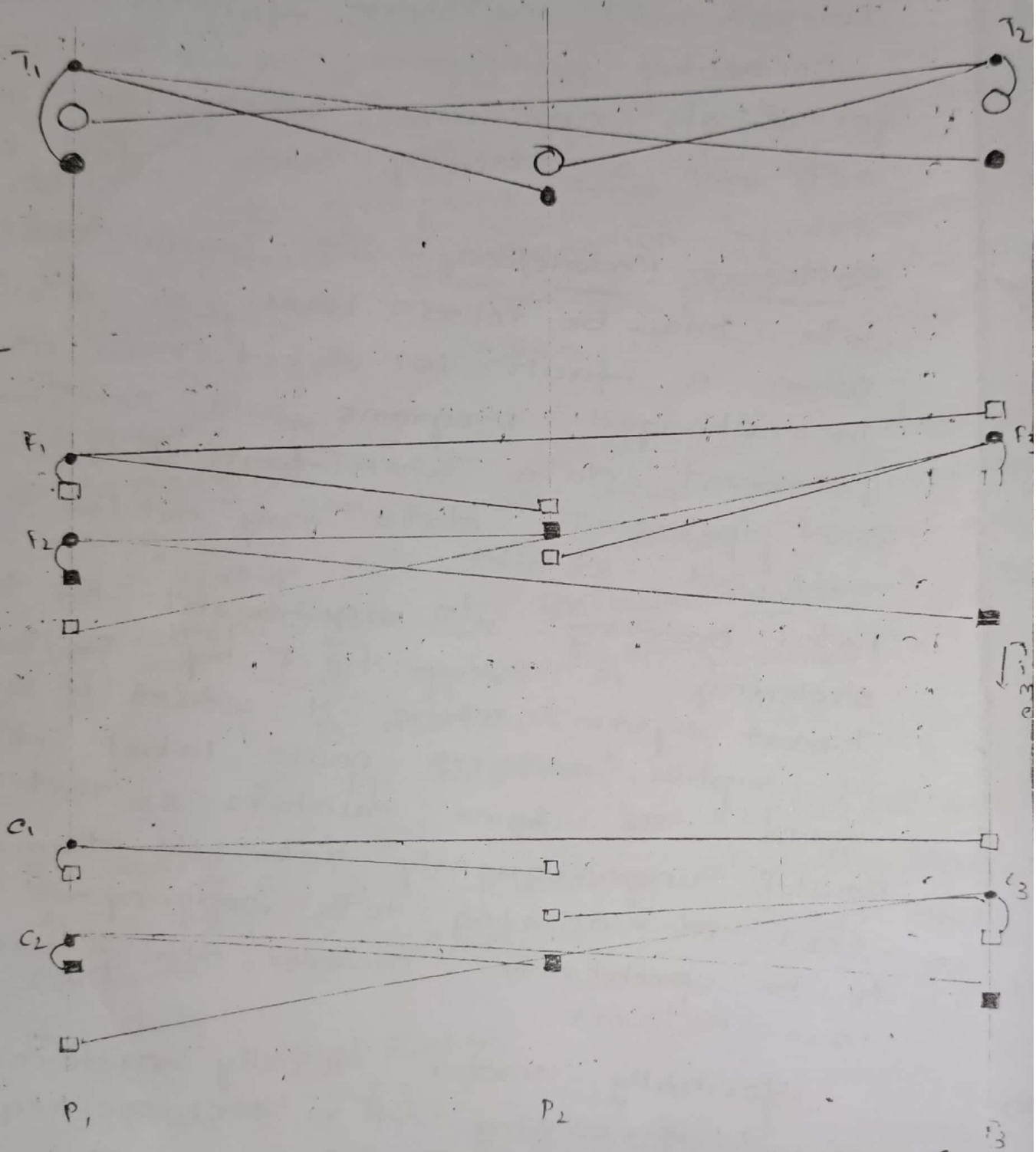
Hardware Redundancy :- It uses redundant components to produce systems that are tolerant of network failures, two inter-connected computers are often employed for single application where one of them acts as a standby machine for other i/e ATM.

Software Redundancy :- It implies that the data can be rolled back or recovered when a fault is detected. In other words, the changed programs will set back to permanent data when fault occurs and permanent data may not be in consistent state.

Total ordering :- A requirement for total ordering is exemplified by the bulletin board system, where it would be convineint if replica managers could label items with the same numbers so that users could unambiguously refer to them. The cost of achieving total ordering is liable to be prohibitive, however, over a wide are network.

formally, under totally ordered request processing, if r1 & r2 are requests then either r1 is processed before r2 at all replica managers or r2 is processed

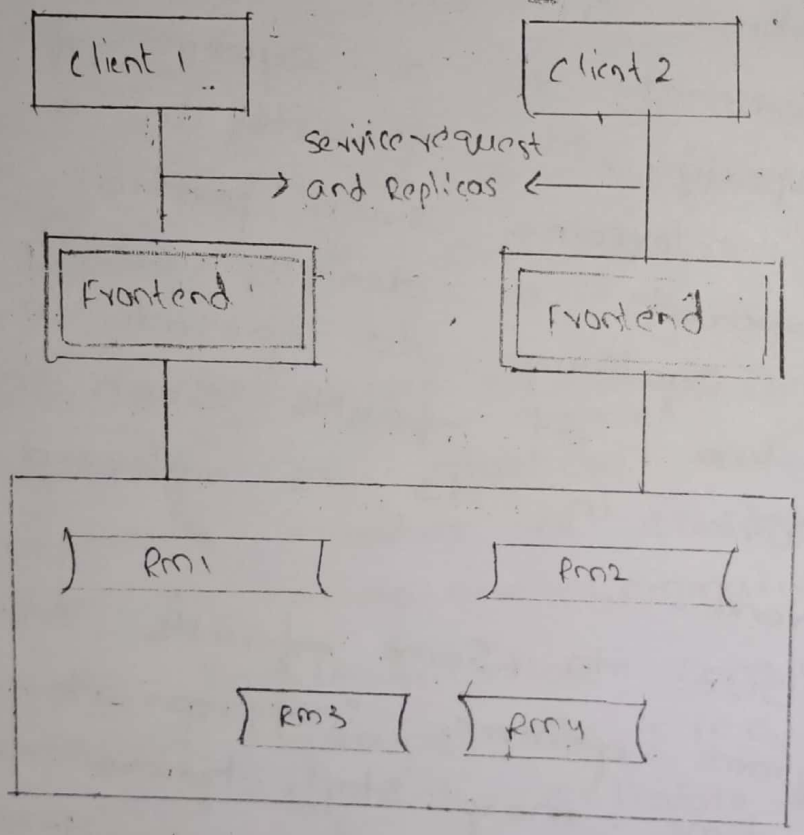
before r_1 at all replica managers. Total ordering is a general relation over event in distributed system.



6. TRANSACTIONS & REPLICATIONS.

* System model & Group Communication.

(3) System model :- In this model, a l
object is implemented using multiple
physical copies called replicas. These replica
are physical copies stored in Computer.
They may not be consistnt at all
times because one replica can get infor
-mation that other replicas did not get.
The data is replicated and held by
unique replica manager.



Different clients request for service
through front end. The frontend replica
managers apply operations and coben a
crash happens, they do not leave inconsistnt

esult:

Generally, every replica manager maint
-ins replicas of every object. However
different object's replica can also be
maintained by different set of replica
managers. Moreover, the managers can be
either dynamic or static. In dynamic
system, new replica manager can appear
and it can crash and leave the system
whereas in static system, managers cannot
- appear and it cannot crash although
it can leave the system for a time
period.

There are five phases involved in
the performance of single request
upon the replicated objects.
Example: (A single request upon replicated
object is -per) A service that
offers fault-tolerance services will obtain
from a service that offers disconnected
operations.

Phase-1 Issue Request: In this phase, the
front end request is a single replica
manager and this manager passes the
request to other replica manager. It
also multicast request to replica mana
-ger.

Phase-2 Coordination:- In this phase, replica managers decide whether to apply the request or not. They also decide the order of a request in relation to other request in one of following orders:

FIFO order:- In this order, a front-end request r1 and then request r2 then replica manager will handle r1 first and then handle r2.

casual order:- In this order, when a requst r1 happend before request r2. then replica manager will handle r1 first and then r2.

Total order:- In this phase, when, a replica manager handle request r1 and then r2

Phase-3 Execution:- In this phase, replica manager executes agrees on the request, effect.

Phase-4 Agreement:- In this phase replica manager agrees on the request effect.

Phase-5 Response:- In this phase, front-end receives request from one or more replica managers

SET-3

(3) Group Communication:-

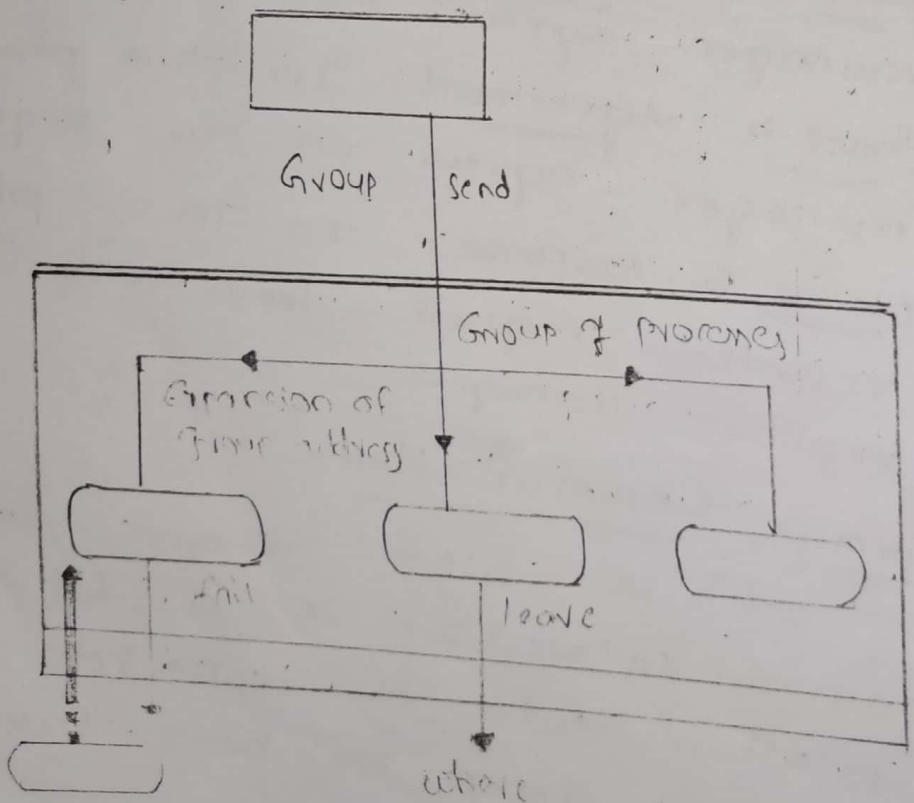
It is also known as multicast communication. It is useful specially,

1. When data is replicated
2. When grop members take common

message stream and
3. When the system process that cooperate to achieve common target through messages are managed by groups.

In general, a process may join or leave a group and the membership of a group can be either static or dynamic. Service. The dynamic membership is managed by group membership service for example, in replicated data service uses may add or withdraw a replica manager or the manager may crash and need to be replaced.

Moreover group membership management multicast are connected to each other or depicted as below.



→ four main tasks performed by group membership service.

1. Interface to Add/Remove members:- The membership service offers operations through which process groups are created and destroyed. This service also allows to add and remove a process from a group.

2. Incorporate a Failure Detector:- The membership service includes a failure detector. This detector checks for failure when a crash occur and also when a group member becomes unreachable due to connection failure.

3. Communicate Group member:- When a process is added to a group or removed from a group, the service communicates it to other group members

4. Expand Group Address:- A group identifier is supplied in the group when a process multicast message. This identifier is expanded by member management service into current group members. The service can also coordinate delivery by controlling address expansion. In other words, the service is capable to decide about the message delivery even when the membership is modified during its delivery.

View delivery! The group management service in a group provides any member process $pr \in gr$ set of view-views $view_0(gr), view_1(gr), \ldots$ fundamental requirement are:-

Order of view:- when a process pr delivers a $view(gr)$ and again delivers $view'(g)$ then other process $* \neq p$ cannot deliver $view'(gr)$ before dealing $view(gr)$. This order of view is always change in same order for different process.

Integrity! When process pr delivers a $view(gr)$ then $pr \in view(g)$. This is to perform sanity checking.

Non-triviality:- when a process pr_1 joins a group and if it becomes unreachable from process $\neq pr_1$ then pr_1 is always in view that $process_2$ will deliver. Moreover, when a group is partitioned, the view delivered in one partition will not contain any process in other partition of a group. It prevents agains trivial solutions for process

→ View - Synchronous group communication

This system ensures delivery of multicast messages view notification thrag agreement. At a given time same sequence of views are delivered by process along with same set of messages

Integrity! This ensures that when a process p_1 delivers a message m_1,

then m1 will not be delivered twice. It also ensures that the sender will be in same view that delivered m1.

Validity: In general, a correct process always detects messages, however if a system fails to detect a message for a process p, then the system informs the remaining processes through a new view delivery and this message will not be delivered to p.

* Concurrency control in Distributed Transactions:-

Locks in Distributed Transaction:- The locks in a distributed transaction on an object are maintained in the same server by a lock manager on one hand, this lock manager decides to make the lock by requesting transaction to wait or grant a lock to it. on the other hand, it cannot unlock the object held by transaction until it gets information that transaction is either aborted to or committed at all servers.

In locking mechanism, when an object is locked by a transaction it is inaccessible by other transaction. however, when a transaction is aborted its locks are unlocked after phase 1 of locking protocol.

Optimistic Concurrency Control:

Every transaction is validated before being committed * distributed transaction is validated by many independent servers that are involved on a transaction. However each server validate its own objects and operation This validation is performed in phase-1 of the two-phase protocol.

In general, transactions suffer from committed deadlock. Consider table which shows interleaving of transactions that suffer from a committed deadlock

Transaction P		Transaction Q	
operation	server	operation	Server
Read(L)	at M	Read(K)	at N
write(L)	at M	write(K)	at N
Read(K)	at N	Read(L)	at M
write(K)	at N	write(L)	at M

This table shows that transaction P access object before transaction Q on M servers whereas transaction Q access object before transation P on N server. However Validation protocol states that only one transaction is allowed to validate and update at a time, the other need to wait. This situation is called committed deadlock. Therefore a new method called parallel validation

Should be used to overcome commit deadlock problem.

Parallel Validation protocol:-

If it is used, transactions will suffer from commitment deadlock. However, if servers simply perform independent validations, it is possible that time different servers of a distributed transaction may serialize the same set of transactions in different orders.

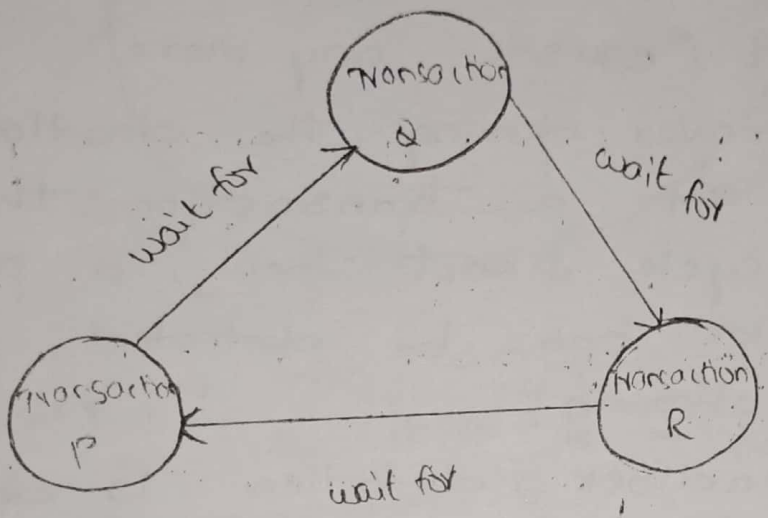
(b) Timestamp ordering Concurrency Control:-

every transaction coordinator must issue a timestamp which should be unique globally. when a transaction acces a cordinator, the client is issued a globally unique transaction timestamp by the first coordinator. This timestamp is supplied to the transaction coordinator when a transaction is performed in a server.

Moreover, when a distributed transaction is performed the servers which perform are combinely responsible in a sequential manner. This can be achieved by timestamp maintainance. It is made up of <local timestamp, server id> pair. Moreover, maintaining transtractions of same order is possible in all the servers in spite of local clocks.

Distributed Deadlocks:-

Distributed transactions can lead to distributed deadlocks. A global wait for graph can be formed by local wait for graphs. This global graph can contain a cycle which does not belong to any single local wait for graphs. This cycle is called as distributed deadlock



From above server M, transaction R wait for transaction P. finally, transaction P wait for transaction R at server L. These local wait for graphs are built by each servers lock manager.

In order to solve this they should be detected by finding a cycle in global transaction wait for graph This cycle can be found by communicating different servers that were

involved in transaction. Distributed deadlock can be solved by using centralized deadlock detection method. In this method, one server is assigned the responsibility of global detector.

Some advantages of centralized servers depend on single server is:

1. Lack of fault tolerance
2. Lack of scalability
3. poor availability
4. Expensive communication among different servers
5. Deadlock detection may consume more time; if global graph is not collected frequently.

Preventing Deadlocks in DS!-

It can be done in two ways!-

Deadlock can be prevented if for a transaction all the required objects are locked before the transaction is started. This must be done in a single operation in order to avoid deadlock with other transactions.

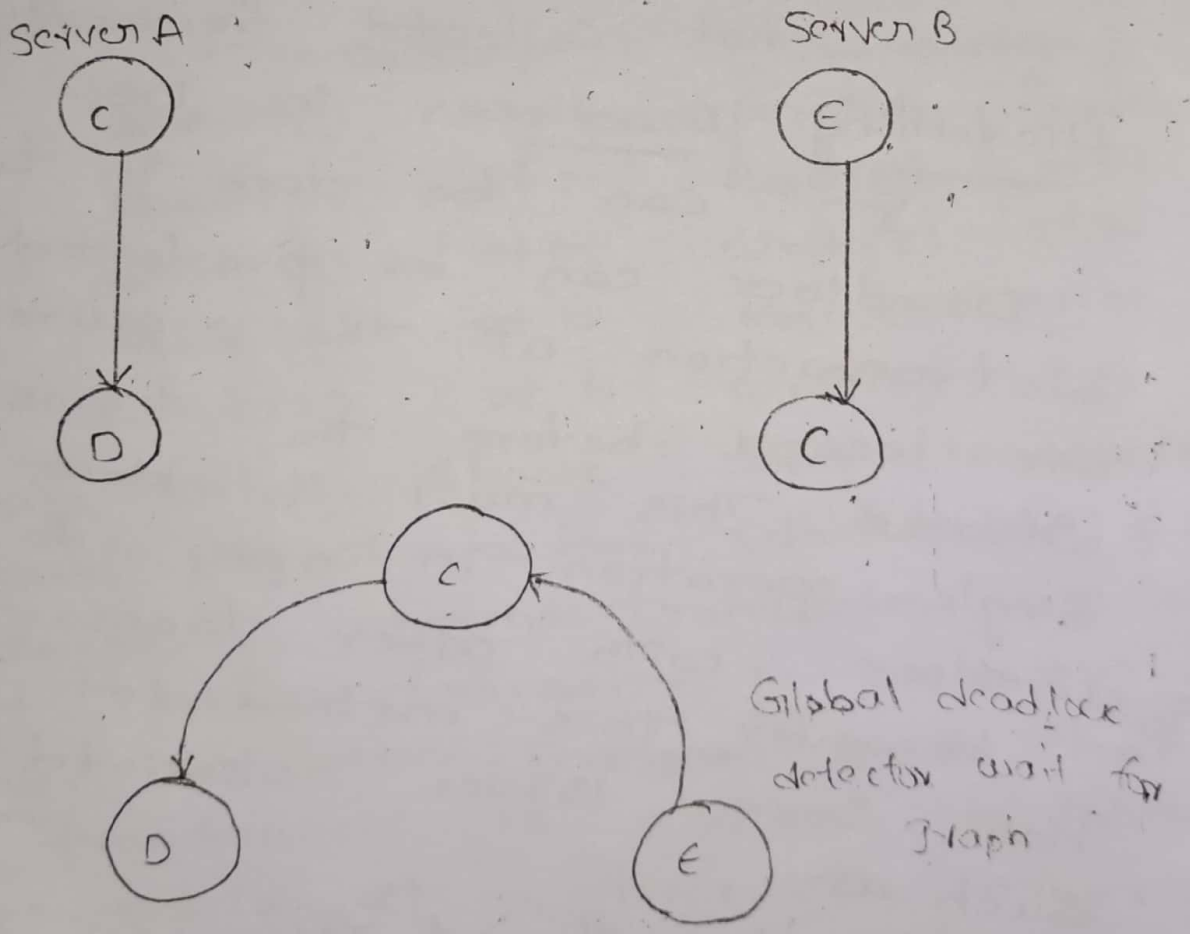
However this method of locking has some issues associated with it such as,

→ Access to shared resource is restricted if it is locked

→ Sometime the objects required to perform transaction are not known before starting.

→ Phantom Deadlock:-

It is a type of deadlock that is detected but it is not a real deadlock. The mechanism of distributed deadlock detection involve transmission of wait for relation information between the servers. During this transmission, if there is a deadlock then it about it is collected and a cycle is detected. However this method consume sometime during which a lock can be released by a transaction and deadlock does not exit



The above figure shows global and local wait for graphs. Suppose that

->At Server A transaction C unlocks object and request for the object B Server locked by transaction E and then

->The global detector get B Server's local graph before receiving Server A's local graph.

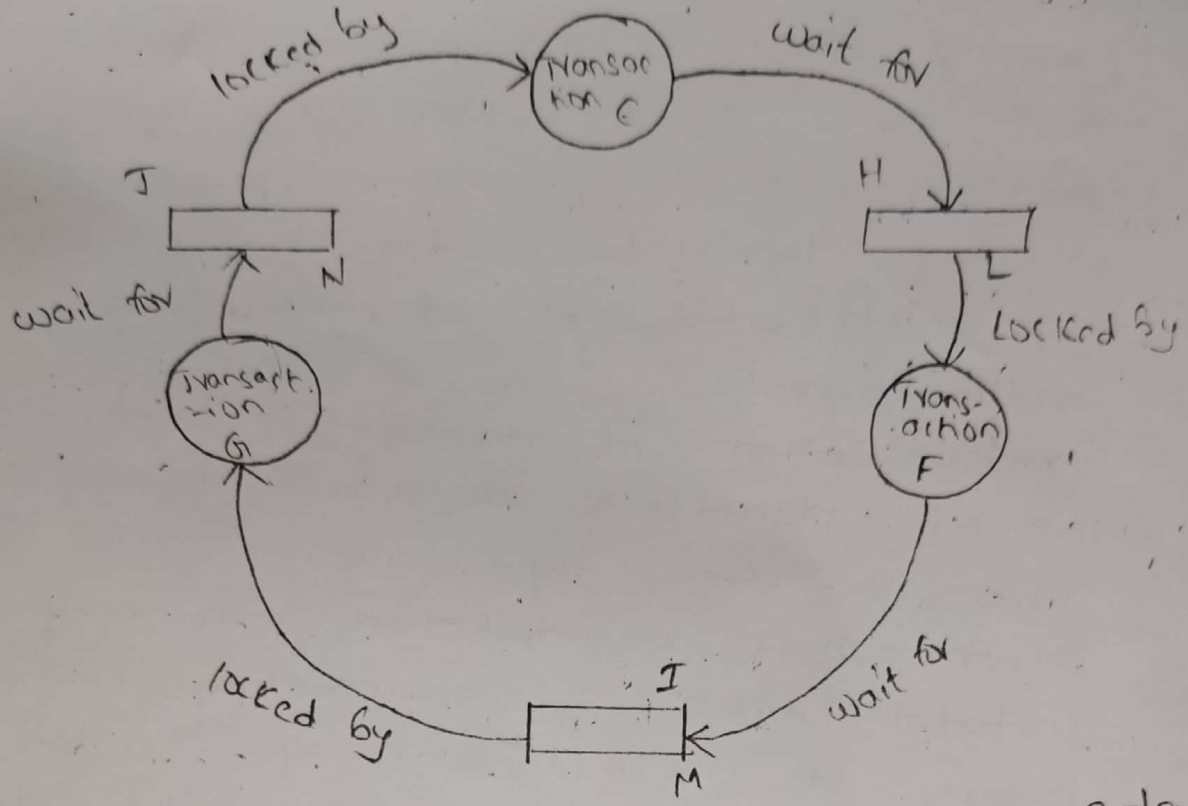
The global detector will detect the cycle C->D->E->C in which C->D cycle does not exists any more.

Moreover during the deadlock detection method, if a transaction that wait in a cycle abort then a phantom deadlock can be detected

-> Edge chasing:-

Deadlock detection is done by using a technique known as edge chasing. This technique is also known as path pushing. In this technique, the global wait for graph is not constructed instead every Server involved in transacti -on has some information about its edges. These Servers find cycles by using a message known as probes. These probes proceed the graph edges all the way in distributed systems These message Contain wait for relationship that request a path in the global wait-for graph. However time to Send a probe is

complicated task.



1. Server N has added a new edges $G \rightarrow E$ in its local graph. At this point transaction E is not waiting for an object. Therefore no need to send a probe.

2. Server L has added a new $E \rightarrow F$ edge in its local graph at the same time transaction F wait for I object. This I object is locked at M. Server by transaction G. This forms a cycle $G \rightarrow T_1 \rightarrow T_2 \cdots E \rightarrow F \rightarrow G$ involving many servers.

Edge chasing algorithm has 3 phases

1. Initiation
2. Detection
3. Resolution

Initiation:- In this phase, a server initiate a phase when it see that, for ex transaction x wait for transaction y and y wait for z transaction. The server add an edge x → y in a probe and send to the k server when transaction y is locked an object. In case, if lock is shared by other transaction then all the transactions that share this lock are also sent the probe.

Detection:- In detection phase, server k receive a probe A → B. for example stating that transaction A is waiting for an object locked by transaction B. This server k also verifies if transactions B is waiting for transaction c then it is also added in the probe and form a cycle A → B → C. Similarly, if transaction c wait for some other transactions then other transactions are also added into the probe.

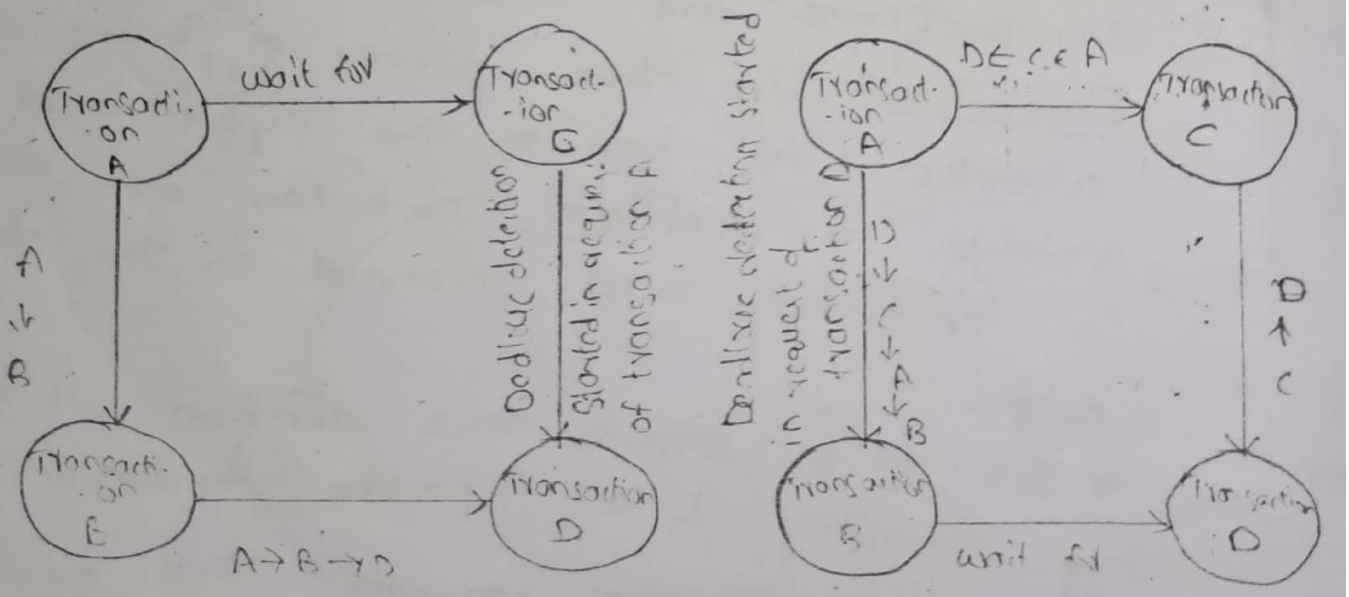
Resolution:- In this phase, a transaction is aborted to break the cycle and to resolve a deadlock. Detection of deadlock is initiated and probed in the following steps.

* Server P starts this dection of deadlock by sending L → J probe to server Q.
* Once Q server receives this probe it check that object N is locked by

transaction K. It further add to transact
-on K and produce L → J → K probe
* finally, R' Server receive L → J → K probe
and it see that object O is locked
by the transaction L and create
L → J → K → L probe. this probe form a
cycle and a deadlock is detected.

→ Prioritization of Transaction :-

transactions are assigned priority
to ensure that in a cycle only one
- transaction is aborted in case of a
deadlock. If priority is not assigned
to a transaction then there are chances
that multiple transactions are aborted in
a cycle. A transaction can be prioritize
based on the timestamp and a transaction
that has lowest priority in a cycle
is aborted and a deadlock is detected.

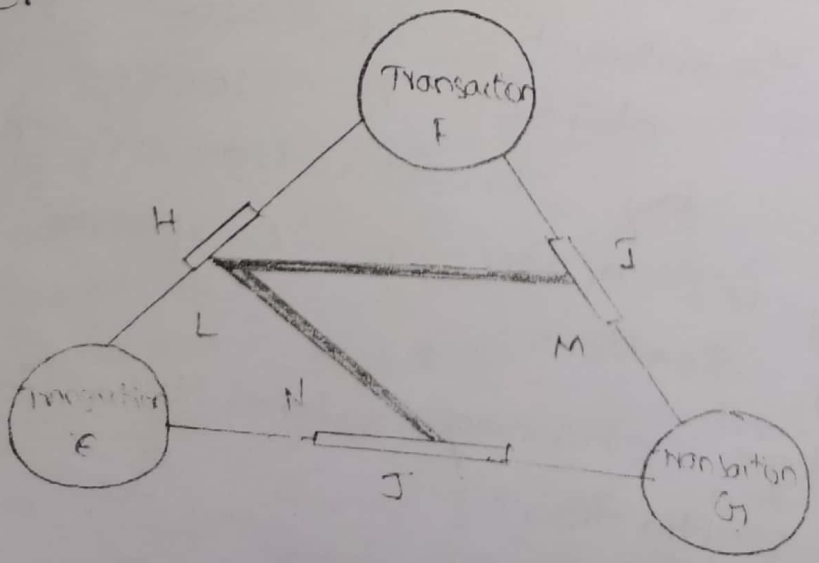


from the above it shows, for example as A -> C, where A has high priority than C. Suppose that transaction A -> B -> C -> D and when any cycle either A -> B -> D -> C -> A or D -> C -> A -> D -> W is detected then transaction D is aborted.

prioritization of transactions offer some advantages such as,

1. prioritization of transaction reduce the no. of deadlock detection to be initiated. This is done by using a rule which stat that a detection can only be initiated when a transaction has higher priority to start, wait for a low priority transaction. This rule reduce the detection intention by approximately 50 percent

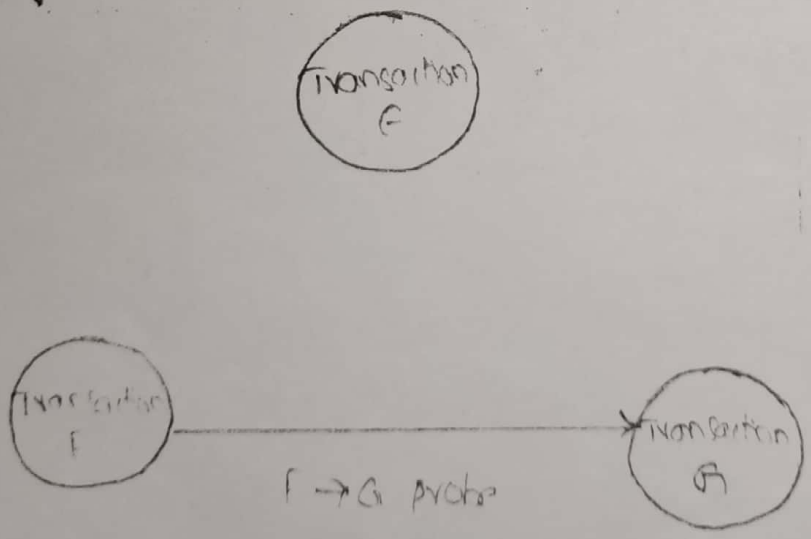
2. prioritization of transactions also reduce the no. of probes to be forwarded. This is done by applying a rule which stat that a probe should be sent from higher priority transaction to lower



transaction G start to wait for F where E is already waiting to G and return transaction G wait for E. In this circumstance, if transactions are not prioritized then a probe E→F can be sent and cycle can be detected. If transactions are prioritized then this probe cannot be sent as E<F as a result deadlock cannot be detected.

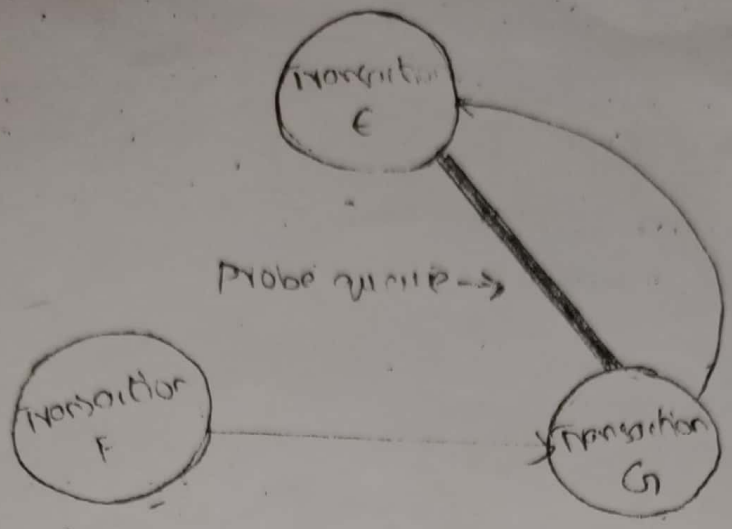
To overcome this problem, transaction coordinator showed save the probe copies sent by transaction in a queue known as probe queue and it should be forwaded is the server that wait for an object.

Example: where transaction E starts to wait for transaction G, the probe F→G is saved by transaction coordinator of G.



After this G transaction start to wait for transaction E, this forms a

queue F → G and G → E.



when transaction E start to wait for object P then F → G → E probe queue is passed to P server. This P check probe F → E and add this is in for to the queue to form F → G → E → F & this way deadlock is detected.

* Transaction Recovery :-

transaction recovery def :-

It ensures that the server objects are durable which requires that objects must be saved in permanent storage. It also ensures that failure of transaction has atomic effect. This recovery is done by a mechanism called recovery manager.

The recovery manager performs the following tasks.

1. It Save objects in permanent storage for the transactions that are committed
2. It rearrange the recovery file in orde to enhance the recovery performance
3. It restore the objects of server after a crash occur
4. It reclaim storage space in recovery file.

* Replication :- It is a proces through which multiple copies of data are maintained in different locations.

Motivations for Replication:- This technique is employed to

1. Improving performance
2. To keep system available at most of the time
3. To make system that can handle fault tolerance.

Improving performance:- A server performance can be improved by using replication. for example, resources are cached in web browsers and proxy servers from web servers. Moreover workload is shared through binding all Ip addreses of the server to a websit DNS. This DNS lookup will results in better performance with minor cost. However, there is a limit of improving performan through this technique. for example, if

dynamic data is replicated it needs processing overheads to handle the changing data.

Enhanced Availability:- In general, a server suffers from problems such as failure, network partition, unplanned disconnection and data locking. Due to these problems a server cannot be active and accessed 100% of time. These problems can be solved by replication of data.

In replication technique, data is automatically maintained in two or more locations on different servers that cannot be failed. These servers can be accessed when a default server fails.

$$1 - \text{prob (when a server fail)}$$
$$= 1 - p^{n}_{\text{serfail}}$$

Where
n = number of servers
prob = probability of server failure.

Fault Tolerance:- Replication of data guarantees that accuracy of data fault tolerance service although few faults can occur. The accuracy is maintained on client's data. Sometimes accuracy is also related to timeliness of response from service.

Code No: **R1642051** **R16** **Set No. 1**

IV B.Tech II Semester Regular/Supplementary Examinations, July - 2021

DISTRIBUTED SYSTEMS

(Common to Computer Science and Engineering and Information Technology)

Time: 3 hours **Max. Marks: 70**

Question paper consists of Part-A and Part-B
Answer ALL sub questions from Part-A
Answer any FOUR questions from Part-B
*****

PART–A *(14 Marks)*

1. a) Define and Give examples of Distributed systems. [3]
   b) Write about IGMP. [3]
   c) Why distributed garbage collection is important? [2]
   d) What is the role of kernel in OS? [2]
   e) What are the goals of distributed file system? [2]
   f) List the advantages of Data replication. [2]

PART–B *(4x14 = 56 Marks)*

2. a) Discuss various issues and challenges involved in the implementation of Distributed Systems. [7]
   b) How the security model ensures security to the interacting processes in a Distributed System? Explain. [7]

3. a) Draw the structure of UDP datagram and explain about various structures available in JAVA API for UDP transmission. [7]
   b) What is meant by Multicast transmission in Distributed Systems? Explain some of the important applications of Multicast Transmission in Distributed systems. [7]

4. a) With a neat sketch, Explain the implementation of Remote Method Invocation. [7]
   b) Discuss the issues in design and implementation of RMI in Distributed Systems. [7]

5. a) What are the design issues of distributed operating system? [7]
   b) Explain any five advantages of creating Threads over multiple execution environments. [7]

6. a) Write the differences between Overlay networks and IP routing. [7]
   b) What are the requirements for mutual exclusion in Distributed systems? Explain about various metrics used for evaluating the performance of mutual exclusion algorithms in Distributed systems. [7]

7. a) Write about the Local and Global Wait-for graphs. [7]
   b) Explain the passive replication model for fault tolerance in distributed systems. [7]

Code No: **R1642051** **R16** **Set No. 1**

IV B.Tech II Semester Regular Examinations, September - 2020

DISTRIBUTED SYSTEMS

(Common to Computer Science and Engineering and Information Technology)

Time: 3 hours **Max. Marks: 70**

Question paper consists of Part-A and Part-B
Answer ALL sub questions from Part-A
Answer any FOUR questions from Part-B
*****

PART–A *(14 Marks)*

1.	a)	Discuss about client server resource sharing.	[3]
	b)	What is Multicast Transmission in Distributed systems? Discuss.	[2]
	c)	Discuss about Remote Procedure Calls.	[2]
	d)	What is meant by Address space? Discuss.	[2]
	e)	What is election process? Discuss about its goal?	[2]
	f)	What is replication? Differentiate between Active and passive replication.	[3]

PART–B *(4x14 = 56 Marks)*

2.		Explain the architectural and fundamental models of distributed systems?	[14]
3.	a)	Explain the client server communication model. Also Discuss about marshaling in detail.	[7]
	b)	Discuss the issues relating to datagram communication.	[7]
4.	a)	Explain the features of distributed object model	[7]
	b)	Explain the design issues of RMI.	[7]
5.	a)	Briefly explain architecture for multi threaded servers.	[7]
	b)	What is the need for protection? Explain various protection mechanisms supported by operating systems.	[7]
6.	a)	Discuss the mounting issues of remote file systems on NFS client.	[7]
	b)	Explain about overlay routing? Explain how it useful in peer communication.	[7]
7.	a)	Describe various deadlock handling techniques.	[7]
	b)	Explain about concurrency control in distributed transactions.	[7]

Code No: **R1642051** **R16** **Set No. 2**

IV B.Tech II Semester Regular Examinations, September - 2020

DISTRIBUTED SYSTEMS

(Common to Computer Science and Engineering and Information Technology)

Time: 3 hours **Max. Marks: 70**

Question paper consists of Part-A and Part-B
Answer ALL sub questions from Part-A
Answer any FOUR questions from Part-B

*****

PART–A *(14 Marks)*

1. a) What is meant by distributed system? Give any two examples. [2]
   b) Discuss about any three applications of Multicast Transmission in Distributed systems. [3]
   c) What is an event and notifications? [2]
   d) What is meant by multi threaded model. Discuss. [2]
   e) Define overlay routing? What is its importance? [2]
   f) What is dead lock? How deadlock can be handled. [3]

PART–B *(4x14 = 56 Marks)*

2. a) What do you mean by Scalability of a distributed system? Explain the principles for designing scalable distributed systems. [10]
   b) Explain the security challenges of distributed systems. [4]

3. a) Explain the different methods for inter-process communication. [7]
   b) Discuss the issues relating to datagram communication. [7]

4. Discuss the design and implementation issues in Remote Method Invocation. [14]

5. a) Explain the general architecture of operating systems for Distributed Systems [7]
   b) What is thread? Explain the life cycle of the thread, with neat state diagram. [7]

6. a) Explain how mutual exclusion is handled in distributed system. [7]
   b) Discuss the Napster and its legacy with respect to distributed file systems. [7]

7. a) Explain the basic architectural model for the management of Replicated data. [7]
   b) What is transaction? Briefly explain about flat and nested distributed transactions. [7]

Code No: **R1642051** **R16** **Set No. 3**

IV B.Tech II Semester Regular Examinations, September - 2020

DISTRIBUTED SYSTEMS

(Common to Computer Science and Engineering and Information Technology)

Time: 3 hours **Max. Marks: 70**

Question paper consists of Part-A and Part-B
Answer ALL sub questions from Part-A
Answer any FOUR questions from Part-B
*****

PART–A *(14 Marks)*

1.	a)	What is meant by resource sharing? Discuss with an example.	[2]
	b)	Discuss about the characteristics of the IPC.	[3]
	c)	Differentiate static and dynamic invocation methods.	[2]
	d)	Differentiate between process and threads.	[3]
	e)	What is mutual exclusion? List its requirements.	[2]
	f)	Define replication? What is the importance of it?	[2]

PART–B *(4x14 = 56 Marks)*

2.		What is distributed systems? Explain its key characteristics of distributed system in detail.	[14]
3.	a)	What is marshaling? Explain marshaling operations in detail.	[7]
	b)	Explain Multicast transmission in Distributed Systems? Discuss about important applications of Multicast Transmission in Distributed systems.	[7]
4.	a)	What is the importance of distributed garbage collection? Explain the Distributed garbage collector algorithm.	[7]
	b)	Discuss about various Remote Procedure Calls.	[7]
5.	a)	What is an Execution environment? Explain in detail about the process execution environment .	[7]
	b)	Describe the architecture for multi-threaded servers.	[7]
6.	a)	What is distributed file system? Briefly explain the file service architecture.	[7]
	b)	What is the goal of an election algorithm? Explain it detail.	[7]
7.	a)	What is concurrency? Write the importance of concurrency control in distributed systems.	[7]
	b)	What is distributed deadlock? Explain with example.	[7]

Code No: **R1642051** **R16** **Set No. 4**

IV B.Tech II Semester Regular Examinations, September - 2020

DISTRIBUTED SYSTEMS

(Common to Computer Science and Engineering and Information Technology)

Time: 3 hours **Max. Marks: 70**

Question paper consists of Part-A and Part-B
Answer ALL sub questions from Part-A
Answer any FOUR questions from Part-B
****

PART–A *(14 Marks)*

1. a) What is meant by failure handling? [2]
   b) Discuss about marshaling. [3]
   c) What is the importance of distributed garbage collection? [2]
   d) Discuss about protection mechanisms supported by operating systems. [3]
   e) What are the goals of election algorithm? [2]
   f) What is transaction? List the different types of transactions. [2]

PART–B *(4x14 = 56 Marks)*

2. a) Explain the design requirements and challenges for distributed systems. [10]
   b) Explain the client server resource sharing system. [4]

3. a) List and Explain the various socket primitives used in TCP stream communication. [7]
   b) Describe IP Multicast communication. [7]

4. a) With a neat sketch, Explain the implementation of Remote Method Invocation. [7]
   b) Explain communication between distributed objects, With a neat diagram. [7]

5. What is thread? Explain the issues related to thread programming, thread lifecycle, and thread synchronization. [14]

6. a) Explain the techniques to achieve high performance in distributed file systems. [7]
   b) Explain the main tasks of Routing Overlays. [7]

7. a) What is replication? Explain about Active and Passive replications [7]
   b) Compare and contrast the various methods of concurrency control. [7]