MODEL ANSWER

SUMMER– 17 EXAMINATION

Subject Title: DISTRIBUTED OPERATING SYSTEM **Subject Code: 17635**

Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

Q. No.	Sub Q. N.	Answer	Marking Scheme
1.		**Attempt any FIVE :**	**20 Marks**
	(a)	**State the basic concept of Remote procedure call.**	**4M**
	Ans:	**What Is RPC** RPC is a powerful technique for constructing distributed, client-server based applications. It is based on extending the notion of conventional or local procedure calling, so that the called procedure need not exist in the same address space as the calling procedure. The two processes may be on the same system, or they may be on different systems with a network connecting them. By using RPC, programmers of distributed applications avoid the details of the interface with the network. The transport independence of RPC isolates the application from the physical and logical elements of the data communications mechanism and allows the application to use a variety of transports. **How RPC Works** An RPC is analogous to a function call. Like a function call, when an RPC is made, the calling arguments are passed to the remote procedure and the caller waits for a response to be returned from the remote procedure. Figure below shows the flow of activity that takes place during an RPC call between two networked systems. The client makes a procedure call that sends a request to the server and waits. The thread is blocked from processing until either a reply is received, or it times out. When the request arrives, the server calls a dispatch routine that performs the requested service, and sends the reply to the client. After the RPC call is completed, the client program continues. RPC specifically supports network applications.	**(Definition: 2 Marks, Explanation : 2 Marks)**

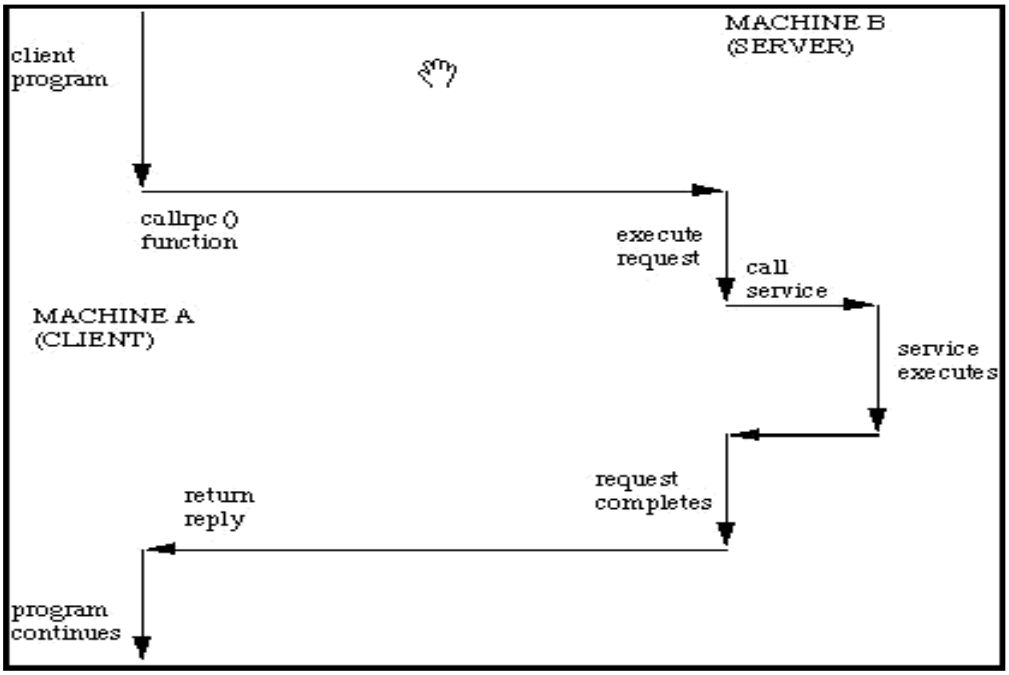


Fig: Remote Procedure Calling Mechanism

A remote procedure is uniquely identified by the triple: (program number, version number, procedure number) The program number identifies a group of related remote procedures, each of which has a unique procedure number. A program may consist of one or more versions. Each version consists of a collection of procedures which are available to be called remotely. Version numbers enable multiple versions of an RPC protocol to be available simultaneously. Each version contains a number of procedures that can be called remotely. Each procedure has a procedure number.

	(b)	**Differentiate between Static v/s dynamic remote invocation.**	**4M**
	Ans:		**(Any 4 Points: 1 Mark)**

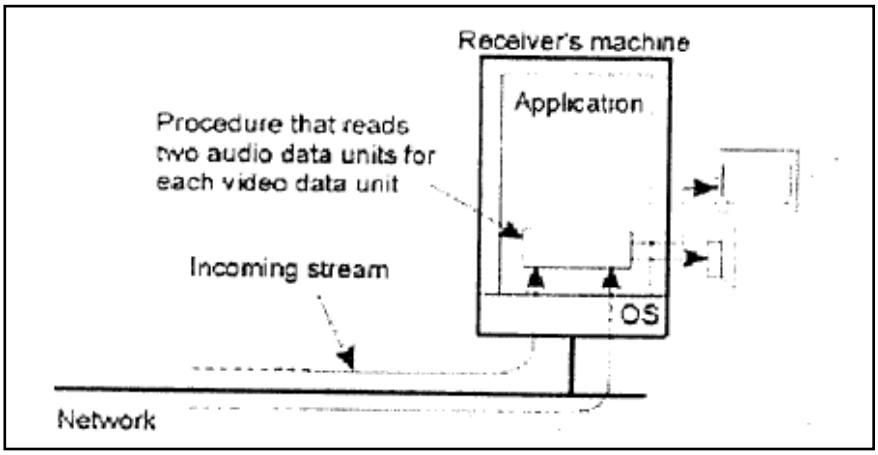
Static remote invocation	**Dynamic remote invocation**
1.Addition of new node without updating client code is not possible	1. Addition of client code is possible
2. Programming Made Easy	2.Relevitely,Programming is not easy
3.Type Checking is robust	3.Type checking is not robust
4.Code is self-explanatory	4. Code is not self-explanatory
5.Good Performance in term of flow control	5. Relatively, Performance is not Good in term of flow control

	(c)	**What is stream synchronization?**	**4M**
	Ans:	**Stream Synchronization**	**(Definition : 2 Marks, Explanation: 2 Marks)**

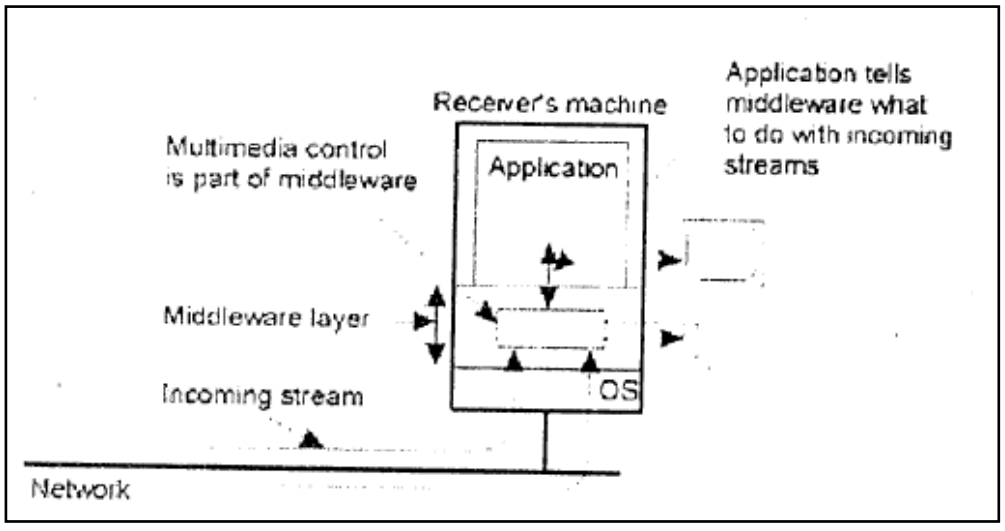
An important issue in multimedia systems is that different streams, possibly in the form of complex stream, are mutually synchronized. Synchronization of streams deals with maintaining temporal relation between steams. Two types of synchronization occur.

The **simplest form of synchronization** is that between a discrete data stream and a continuous data stream. Consider, for example, a slide show on the Web that has been enhanced with audio. Each slide is transferred from the server to the client in the form of a discrete data stream. At the same time, the client should play out a specific (part of an) audio stream that matches the current slide that is also fetched from the server. In this case, the audio stream is to be synchronized with the presentation of slides.

A more demanding type of synchronization is that between continuous data streams. A daily example is playing a movie in which the video stream needs to be synchronized with the audio, commonly referred to as lip synchronization. Another example of synchronization is playing a stereo audio stream consisting of sub streams, one for each channel.



The principle of explicit synchronization on the level data units.



The principle of synchronization as supported by high-level interfaces.

	(d)	**What are threads? Give the basic concepts of threads in distributed system.**	**4M**
	Ans:	**1. Threads in distributed system:** Property of threads - they can provide an easy way of allowing blocking system calls without blocking the entire process in which the thread is running. This property makes threads easier to express communication in the form of maintaining multiple logical connections at the same time. We illustrate this by **multithreaded clients and multithreaded servers**. **Multithreaded Client:** • **Multithreaded clients:** Main issue is hiding network Latency • A typical example where this happens is in Web browsers. • **Multithreaded Web client:** • Web browser scans an incoming HTML page, and finds that more files need to be fetched • Each file is fetched by a separate thread, each doing a (blocking) HTTP request • As files come in, the browser displays them • **Multiple RPCs:** • A client does several RPCs at the same time, each one by a different thread • It then waits until all results have been returned • If RPCs are to different servers, we may have a linear speed-up compared to doing RPCs one after the other. **Multithreaded Server:** • Main issue is improved performance and better structure. • To solve this issue following popular organizational structure developed.	**(Definition :2 Marks, Concepts: 2 Marks)**

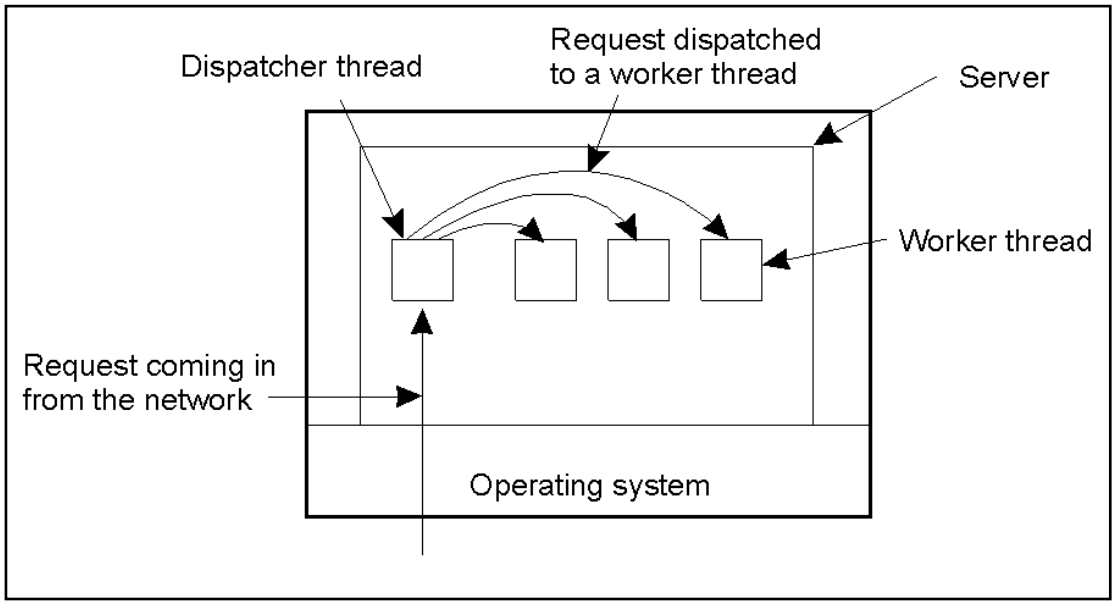


Fig: Multithreaded server

- **Dispatcher,** reads incoming requests for a file operation.
- The requests are sent by clients to a well-known end point for this server.
- After examining the request, the server chooses an idle (i.e., blocked) **worker thread** and hands it the request.
- The worker proceeds by performing a blocking read on the *local* file system, which may cause the thread to be suspended until the data are fetched from disk.
- If the thread is suspended, another thread is selected to be executed.
  For example,
- the dispatcher may be selected to acquire more work.
- Alternatively, another worker thread can be selected that is now ready to run.

There are **two possible designs**:
a **multithreaded file server and**
a single-threaded file server.

	(e)	**Mention the approaches to code migration.**	**4M**
	Ans:	**Code Migrations Approaches to code migration:** Reasons for migrating code: • Load Distribution algorithms: Load distribution algorithms by which decisions are made concerning the allocation and redistribution of tasks with respect to a set	**(Approach : 2 Marks, Explanation: 2 Marks)**

of processors, play an important role in compute-intensive systems.

- Minimize communication
- Becomes possible to dynamically configure distributed system

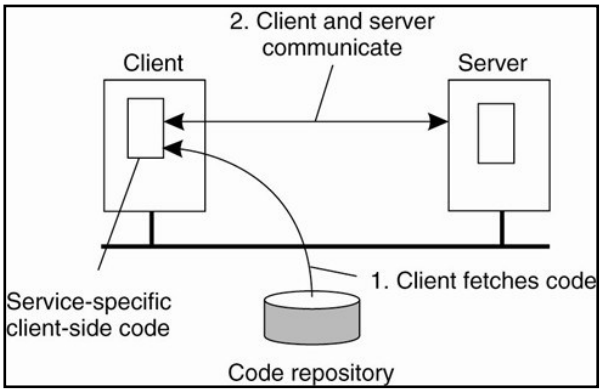


Figure above, The principle of dynamically configuring a client to communicate to a server. The client first fetches the necessary software, and then invokes the server.

- This model of dynamically moving code from a remote site.
- It does require that the protocol for downloading and initializing code is standardized.
- Also, it is necessary that the downloaded code can be executed on the client's machine.
- The important advantage of this model of dynamically downloading client side software is that clients need not have all the software preinstalled to talk to servers.
- Instead, the software can be moved in as necessary, and likewise, discarded when no longer needed.
- Another advantage is that as long as interfaces are standardized, we can change the client-server protocol and its implementation as often as we like.
- Changes will not affect existing client applications that rely on the server.
- There are, of course, also disadvantages- Blindly trusting that the downloaded code implements only the advertised interface while accessing your unprotected hard disk and does not send the juiciest parts to heaven-knows-who may not always be such a good idea.

Models for Code Migration

- A process consists of three segments:
  - Code segment
  - Resource segment
  - Execution segment
- The ***code segment*** is the part that contains the set of instructions that make up the

program that is being executed.

- The ***resource segment*** contains references to external resources needed. by the process, such as files, printers, devices, other processes, and so on.
- An ***execution segment*** is used to store the current execution state of a process, consisting of private data, the stack, and, of course, the program counter.
- The bare minimum for code migration is to provide only weak mobility.
- In this model, it is possible to transfer only the code segment, along with perhaps some initialization data.
- **A characteristic feature of weak mobility** - a transferred program is always started from one of several predefined starting positions.
- The benefit of this approach is its simplicity.
- **Weak mobility** requires only the target machine can execute that code,
- In contrast to weak mobility, in systems that support strong mobility the execution segment can be transferred as well.
- **The characteristic feature of strong mobility** is that a running process can be stopped, subsequently moved to another machine, and then resume execution where it left off.
- Strong mobility is much more general than weak mobility, but also much harder to implement.
- Irrespective of whether mobility is weak or strong, a further distinction can be made between sender-initiated and receiver-initiated migration. In sender initiated migration.
- Migration is initiated at the machine where the code currently resides or is being executed.
- Sender-initiated migration is done when uploading programs to a compute server.
- Another example is sending a search program across the Internet to a Web database server to perform the queries at that server.
- In receiver-initiated migration, the initiative for code migration is taken by the target machine.
- Java applets are an example of this approach.
- Receiver-initiated migration is simpler than sender-initiated migration.
- In many cases, code migration occurs between a client and a server, where the client takes the initiative for migration.
- Securely uploading code to a server, as is done in sender-initiated migration, often requires that the client has previously been registered and authenticated at that server. In other words, the server is required to know all its clients, the reason being is that the client will presumably want access to the server's resources such as its disk. Protecting such resources is essential.
- In contrast, downloading code as in the receiver-initiated case, can often be done anonymously.
- Moreover, the server is generally not interested in the client's resources.
- Instead, code migration to the client is done only for improving client side performance.
- To that end, only a limited number of resources need to be protected, such as

memory and network connections

- In the case of weak mobility, it also makes a difference if the migrated code is executed by the target process, or whether a separate process is started.
- For example, Java applets are simply downloaded by a Web browser and are executed in the browser's address space.
- The benefit of this approach is that there is no need to start a separate process, thereby avoiding communication at the target machine.
- The main drawback is that the target process needs to be protected against malicious or inadvertent code executions.
- A simple solution is to let the operating system take care of that by creating a separate process to execute the migrated code.
- As shown in figure (l)

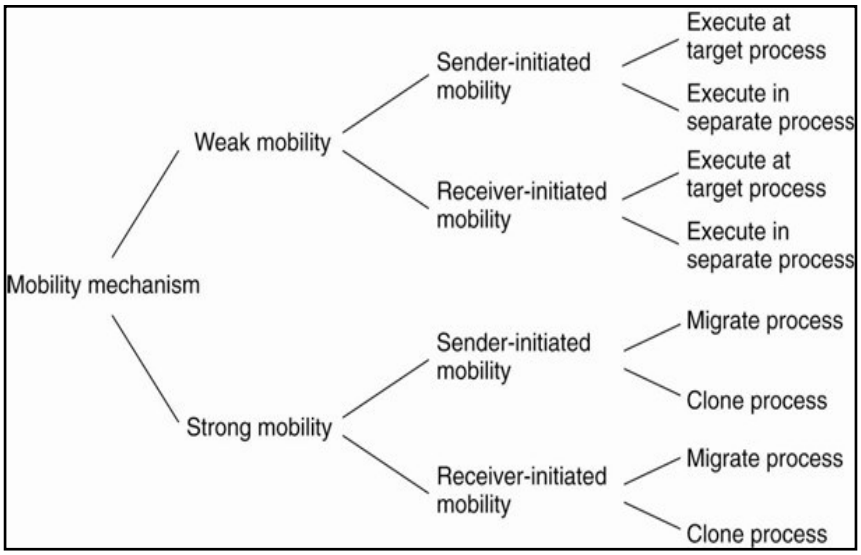


Fig: (l) Alternatives for code migration.

(f) What is DNS? Give example.

Ans:

1. Example DNS:

Internet Domain Name Server (DNS): the largest distributed name service in use Primarily used to lookup host addresses and mail servers The DNS name space is hierarchically organized as a rooted tree Path name root: represented as flits.cs.vu.nl. (rightmost dot indicates the root) Name of the node: label of the incoming edge Domain: subtree Domain name: path name to its root node Contents of a node: a collection of resource records A domain may be implemented by several zones The DNS database is distributed across a logical network of servers (each primarily data for the local domain

- The Domain Name System (**DNS**) is a central part of the Internet, providing a way to match names (a website you're seeking) to numbers (the address for the website). Anything connected to the Internet - laptops, tablets, mobile phones, websites - has an Internet Protocol (IP) address made up of

(DNS: 2 Marks, Example: 2 Marks)

numbers.

- DNS is primarily used for looking up IP addresses of hosts and mail servers.

The DNS Name Space:

- The DNS name space is hierarchically organized as a rooted tree.
- A label is acase-insensitive string made up of alphanumeric characters.
- A label has a maximum length of 63 characters; the length of a complete path name is restricted to255 characters.
- The string representation of a path name consists of listing its labels, starting with the rightmost one, and separating the labels by a dot (H. "). The root is represented by a dot.
- So, for example, the path name root: <nl, VU, cs,flits>, is represented by the string flits.cs. vu.nl., which includes the rightmost dot to indicate the root node.
- We generally omit this dot for readability.
- The contents of a node are formed by a collection of resource records.
- There are different types of resource records. As shown bellow:

Type	Meaning	Value
SOA	Start of Authority	Parameters for this zone
A	IP address of a host	32-Bit integer
MX	Mail exchange	Priority, domain willing to accept e-mail
NS	Name Server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
HINFO	Host description	CPU and OS in ASCII
TXT	Text	Uninterpreted ASCII text

The DNS Name Space

- The most important types of resource records forming the contents of nodes in the DNS name space.

Type of record	Associated entity	Description
SOA	Zone	Holds information on the represented zone
A	Host	Contains an IP address of the host this node represents
MX	Domain	Refers to a mail server to handle mail addressed to this node
SRV	Domain	Refers to a server handling a specific service
NS	Zone	Refers to a name server that implements the represented zone
CNAME	Node	Symbolic link with the primary name of the represented node
PTR	Host	Contains the canonical name of a host
HINFO	Host	Holds information on the host this node represents
TXT	Any kind	Contains any entity-specific information considered useful

(g) State the concept of Grid computing. 4M

Ans:

1. Grid Computing: **(Concept: 2 Marks, Need: 2 Marks)**

- **Grid computing** is the collection of computer resources from multiple locations to reach a common goal.
- Grids are a form of distributed computing.
- Grid is a special type of parallel computing that relies on complete computers connected to a network.
- The ideas of the grid were brought together by Ian Foster, Carl Kesselman, and Steve Tuecke, widely regarded as the "fathers of the grid".
- Grid computing is a distributed architecture of large numbers of computers connected to solve a complex problem.
- In the grid computing model, servers or personal computers run independent tasks and are loosely linked by the Internet or low-speed networks.
- Computers may connect directly or via scheduling systems.

Need Of Grid computing:

- It is a Core network technology.
- It exploits underutilized resources.
- It uses parallel CPU capacity.

- It provides access to additional resources.
- It can also be used in Government, Business, Education, and Industrial Designs

Working of Grid computing:

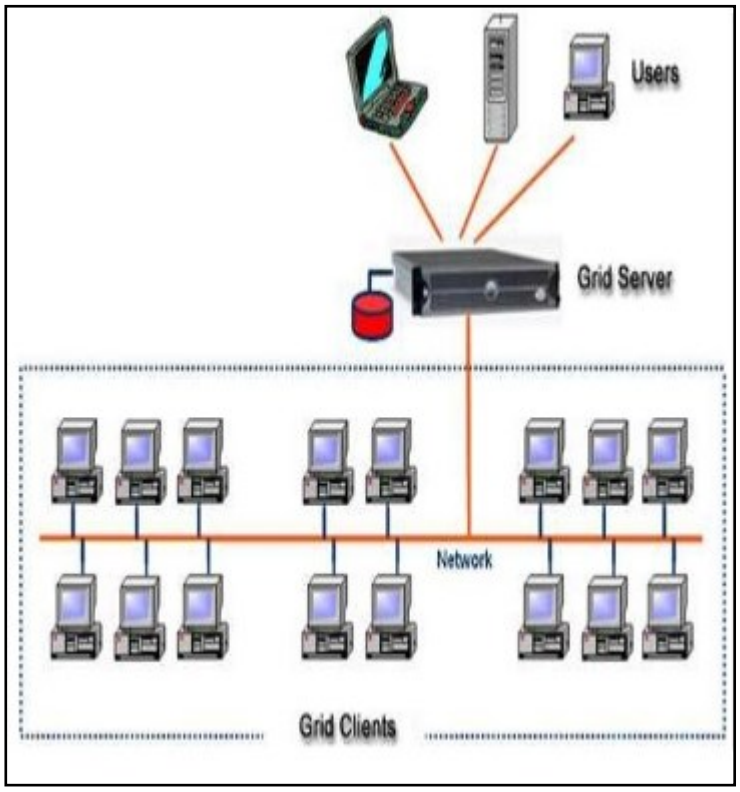


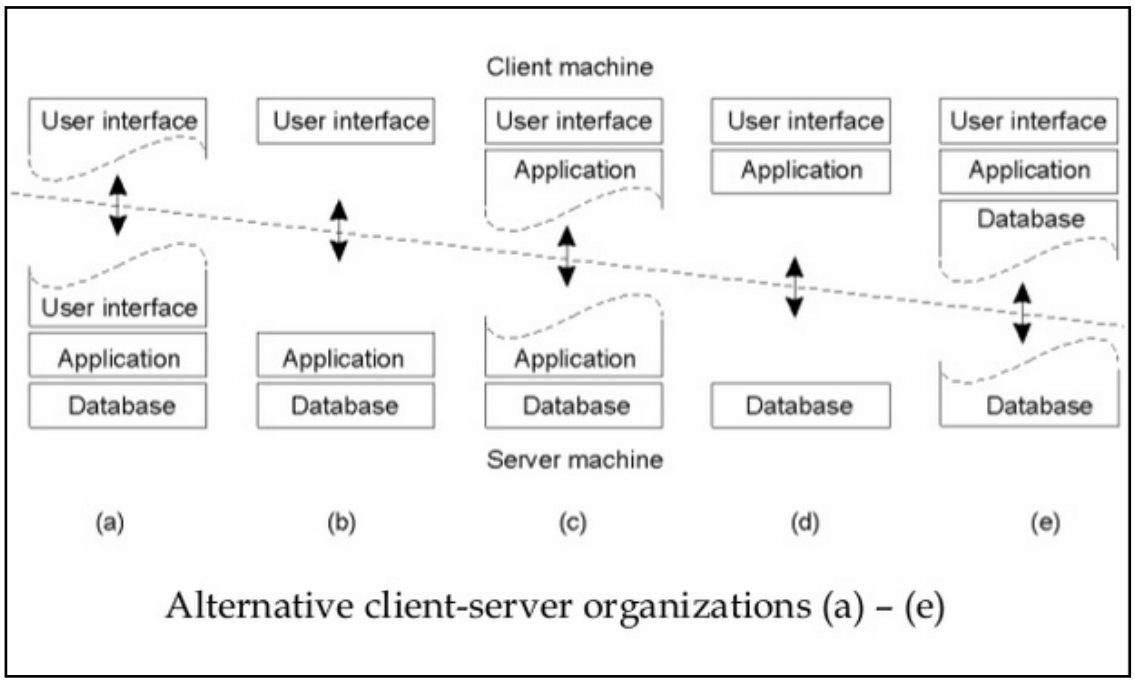
Fig: Working of Grid computing

- If a machine on a computing grid has a large task to be performed, the program must first be parallelized.
- The flow of the program needs to be analyzed and each module is separated.
- The modules are then arranged to illustrate which ones can be executed independently.
- Those modules are then sent to different machines for execution.
- The results are then resent to the original machine, where they are amalgamated into one whole.
- At its most basic level, grid computing is a computer network in which each computer's resources are shared with every other computer in the system.

		• Processing power, memory and data storage are all community resources that authorized users can tap into and leverage for specific tasks. • A grid computing system can be as simple as a collection of similar computers running on the same operating system. • The grid computing concept isn't a new one. • It's a special kind of distributed computing. • In distributed computing, different computers within the same network share one or more resources. • In the ideal grid computing system, every resource is shared, turning a computer network into a powerful supercomputer. • With the right user interface, accessing a grid computing system would look no different than accessing a local machine's resources. • Every authorized computer would have access to enormous processing power and storage capacity.	
2.		**Attempt any TWO :**	**16 Marks**
	(a)	**Draw and explain the client-server architecture.**	**8M**
	Ans:	**Client-Server Architectures:** The distinction into three logical levels as discussed in the previous section, suggests a number of possibilities for physically distributing a client-server application across several machines. The simples organization is to have only two types of machines 1. A client machine containing only the programs implementing (part of) the user interface level. 2. A server machine containing the rest, that is the programs implementing the processing and data level. The problem with this organization is that it is not really distributed everything is handled by the server, while the client is essentially no more than a dumb terminal.	**(Diagram: 4 Marks,Explanation: 4 Marks)**

Multitiered Architectures:

One approach for organizing clients and servers it to distribute the programs un the application layers of the previous section across different machines, as shown in Fig. as a first step, we make a distinction between only two kinds of machines: clients and servers leading to what is also referred to as a (physically) two tiered architecture.



Alternative client-server organizations (a) – (e)

One possible organization is to have only the terminal-dependent part of the user interface on the client machine, as shown if fig. (a) and give the applications remote control over the presentation of their data. An alternative is to place the entire user interface software on the client side, as shown in fig.(b) in such cases we essentially divide the application into a graphical front end, which communicates with the rest of the application (residing at the server) through an application-specific protocol. Continuing along this line of reasoning, we may also move part of the application to the front end, as shown if fig. (c) An example where this makes sense is where the application makes use of a form that needs to be filled in entirely before it can be processes. These organizations are used in the case where the client machine is a PC or workstation, connected through a network to a distributed file system or database fig. (e). represents the situation where the clients local disk contains part of the data. For example, when browsing the Web, a client can gradually build a huge cache on local disk of most recent inspected Web pages.

When distinguishing only clients and servers, we miss the point that a server may sometimes need to act as a client, as shown in fig below. leading to a (physically) three tiered architecture.

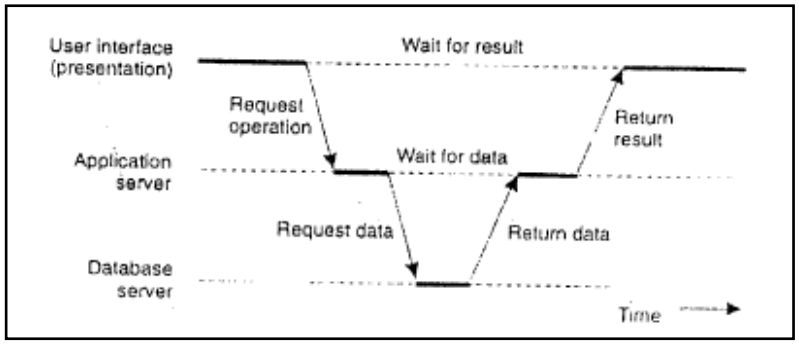


Fig: An example of a server acting as client

Modern Architectures:

Multitiered client-server architectures are a direct consequence of dividing applications into a user-interface, processing components, and a data level. The characteristic feature of vertical distribution.

However, vertical distribution is only one way of organizing client-server applications, and in many cases the least interesting one. In modern architectures, it is often the distribution of the clients and the servers that counts which we refer to as horizontal distribution. In this type of distribution, a client or server may be physically split up into logically equivalent parts, but each part is operating on its own share of the complete data set, thus balancing the load.

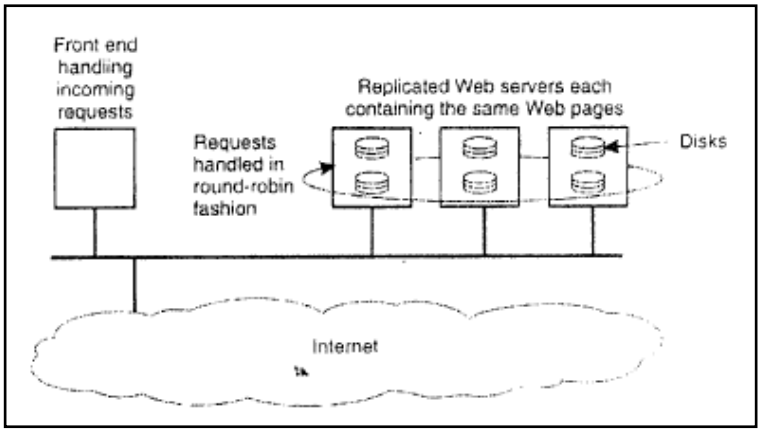
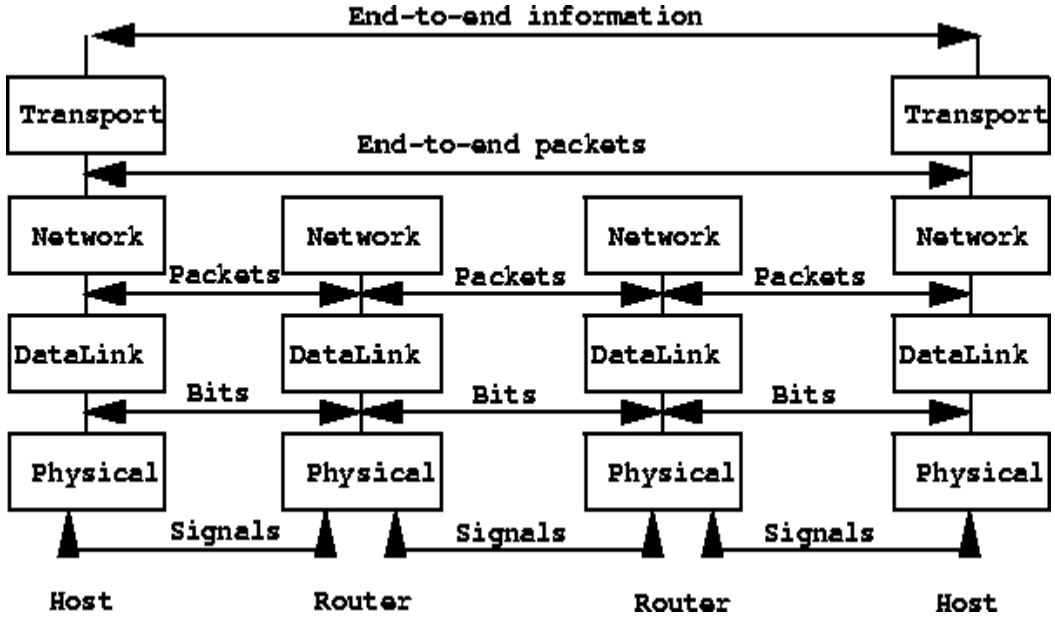
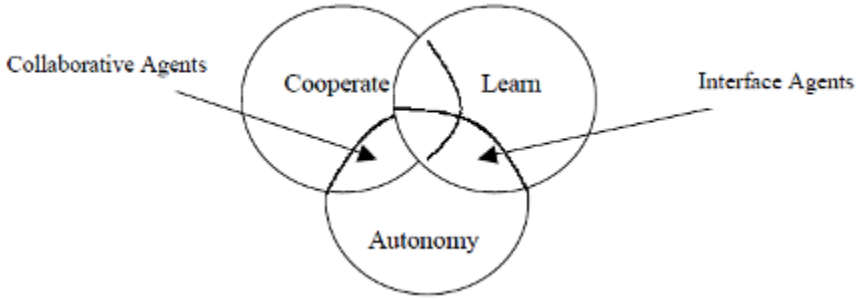


Fig. An example of horizontal distribution of a Web server

	(b)	**Mention and describe different layered protocols.**	**8M**
	Ans:	**Layered Protocols:** • A layered protocol architecture provides a conceptual framework for dividing the complex task of exchanging information between remote hosts into simpler tasks. • Each protocol layer has a narrowly defined responsibility. • A protocol layer provides a standard interface to the next higher protocol layer. • Consequently, it hides the details of the underlying physical network infrastructure. • **Benefit:** The same user-level (application) program can be used over very diverse communication networks. • **Example:** The same WWW browser can be used when you are connected to the internet via a LAN or a dial-up line.  **Lower-level protocols** • Physical –deals with mechanical and electrical details • Data link –groups bits into frames & ensure are correctly received • Network –describes how packet are routed, lowest i/f for most distributed systems (IP) **Transport protocols** • Transfer messages between clients, including breaking them into packets, controlling low, etc (TCP & connectionless UDP)	**(Concept & Diagram: 4 Marks, Types: 2 Marks Each)**

		High-level protocols • Session –provides dialog control and synchronization • Presentation –resolves differences in formats among sites • Application –originally to contain a set of standard apps	
	(c)	**Mention and describe about the software agents in distributed system.**	**8M**
	Ans:	**1. Software agents:** • "An agent is an entity that: · acts on behalf of others in an autonomous fashion · performs its actions in some level of proactivity and reactivity · exhibits some levels of the key attributes of learning, co-operation, and mobility." • There are several dimensions to classify existing software agents. • They can be classified according to: the tasks they perform; their control architecture; the range and effectiveness of their actions; the range of sensitivity of their senses; or how much internal state they possess. • three characteristics: autonomy, learning, and cooperation • Autonomy refers to the characteristic that an agent can operate on its own without the need for human guidance. In other words, an agent has a set of internal states and goals, it acts in such a manner to meet its goals on behalf of the user. • These three characteristics of agents are used to derive some types of agents to include in our classification as shown in Figure (n)  **Fig: (n) A partial view of agent classification**	**(Definition : 2 Marks, Any 3 Type: 2 Marks Each)**

- **Interface Agents:**
  - Interface agents perform tasks for their owners by emphasizing autonomy and learning.
  - They support and provide assistance to a user learning to use a particular application such as a spreadsheet.
  - The agent here observes the actions being carried out by the user and tries to learn new short cuts, then it will try to suggest better ways of doing the same task.
  - Interface agents learn to better assist its users in four ways:
    - By observing and imitating the user
    - Through receiving positive and negative feedback from the user
    - By receiving explicit instructions from the user
    - By asking other agents for advice
- **Collaborative Agents:**
  - The goal of collaborative agents is to interconnect separately developed collaborative agents, thus enabling the ensemble to function beyond the capabilities of any of its members.
  - collaborative agents are to provide solutions to inherently distributed problems, such as distributed sensor network or air traffic control.
- **Information Agents:**
  - information agents seem a bit similar to interface agents
  - One distinction between interface and information agents, however, is that information agents are defined by what they do, in contrast to interface agents which are defined by what they are.
  - Information agents are most useful on the Web where they can help us with mundane tasks.
  - For example, we carry out actions that may consume long time (e.g. searching the Web for information).

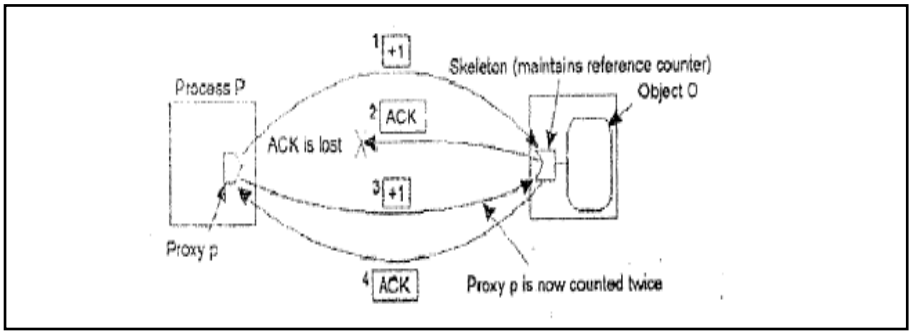
- **Reactive Agents:**
    - Reactive Agents act and respond in a stimulus-response manner to the present state of the environment in which they are embedded.
    - P. Maes highlights the following three key ideas which underpin reactive agents.
        - Emergent functionality: the dynamics of the interaction leads to the emergent complexity.
        - Task decomposition: a reactive agent is viewed as a collection of modules which operate autonomously and responsible for specific tasks (e.g. sensing, computation, etc.).
        - They tend to operate on representations that are close to raw sensor data.
- **Hybrid Agents:**
    - Hybrid Agents refer to those agents whose constitution is a combination of two or more agent philosophies within a singular agent.
    - The goal of having hybrid agents is the notion that the benefits accrued from having the combination of philosophies within a single agent is greater than the gains obtained from the same agent based on a singular philosophy
- **Mobile Agents:**
    - A software agent is a mobile software agent if it is able to migrate from host to host to work in a heterogeneous network environment.
    - This means we must also consider the software environment in which mobile agents exist.
    - This is called the mobile agent environment, which is a software system distributed over a network of heterogeneous computers and its primary task is to provide an environment in which mobile agents can run.

2. Software agents in Distributed system

Property	Common to all agents?	Description
Autonomous	Yes	Can act on its own
Reactive	Yes	Responds timely to changes in its environment
Proactive	Yes	Initiates actions that affects its environment
Communicative	Yes	Can exchange information with users and other agents
Continuous	No	Has a relatively long lifespan

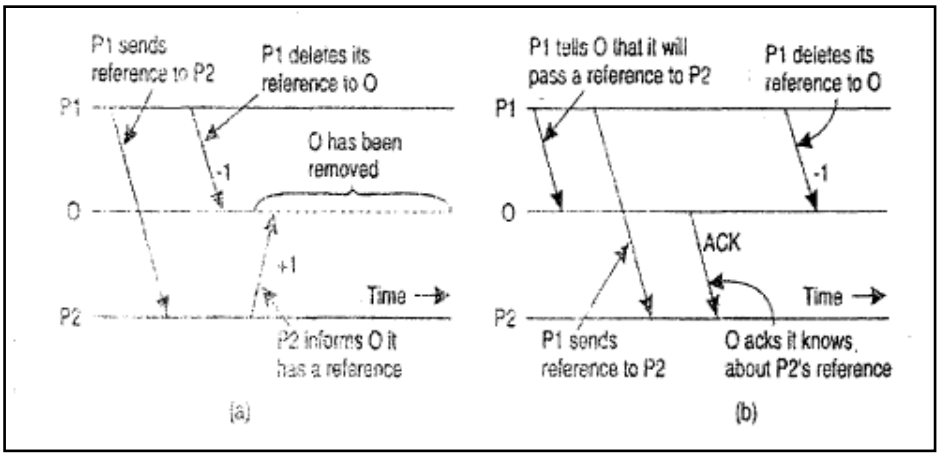
3.		**Attempt any TWO :**	**16 Marks**
	(a)	**Define distributed system. Describe the goals of distributed system.**	**8M**
	Ans:	A distributed system is a collection of independent computers that appears to its users as a single coherent system. **1.Making Resources Accessible** The main goal of a distributed system is to make it easy for the users (and applications) to access remote resources, and to share them in a controlled and efficient way. Resources can be just about anything, but typical examples include things like printers, computers, storage facilities, data, files, Web pages, and networks **2. Transparency** An important goal of a distributed system is to hide the fact that its processes. And resources are physically distributed across multiple computers. A distributed system that is able to present itself to users and applications as if it were only a single computer system is said to be transparent. **3.Openness** An open distributed system is a system that offers services according to standard rules that describe the syntax and semantics of those services. **4.Scalability** Scalability of a system can be measured along at least three different dimensions. First, a system can be scalable with respect to its size, meaning that we can easily add more users and resources to the system. Second, a geographically scalable system is one in which the	**(Definition :2 Marks, Any 3 Goals Each carries: 2 Marks)**

		users and resources may lie far apart. Third, a system can be administratively scalable, aiming that it can still be easy to manage even if it spans many independent administrative organizations.	
	(b)	**Explain servers general design issues of processes.**	**8M**
	Ans:	1. In the case of an iterative server, the server itself handles the request and, if necessary, returns a response to the requesting client. 2. A concurrent server does not handle the request itself, but passes it to a separate thread or another process, after which it immediately waits for the next incoming request. 3. A multithreaded server is an example of a concurrent server. An alternative implementation of a concurrent server is to fork a new process for each new incoming request. 4 Another issue that needs to be taken into account when designing a server is whether and how a server can be interrupted. For example, consider a user who has just decided to upload a huge file to an FTP server. Then, suddenly realizing that it is the wrong file, he wants to interrupt the server to cancel further data. 5. Whether or not the server is stateless. A stateless server does not keep information on the state of its clients, and can change its own state without having to inform any client	**(Any 4 issues Each issue carries: 2 Marks)**
	(c)	**What are the problem of unreferenced objects and how to remove it?**	**8M**
	Ans:	**The Problem of Unreferenced Objects** To explain how garbage collection works, we concentrate on garbage collecting distributed objects, in particular, remote objects. Recall that a remote object is implemented by having its entire state located at an object server, whereas clients have only a proxy. As we explained, a reference to a remote object is generally implemented as what we can now refer to as a (Proxy, skeleton) pair. The client-side proxy contains all the information to contact the object by means of its associated skeleton as implemented by the server. **Reference counting** Popular in uniprocessor system. It simply count the reference to that objects. Each time a reference to an object is created, a reference counter will be incremented and when reference is removed reference counter is decremented. As soon as reference counter reaches to zero object can be removed. 1. Simple reference counting Has many problem due to unreliable communication system. Counter may be decremented or incremented more than once for same proxy due to communication loss. Which will either remove object before handed or keeps it even when it is not needed.	**(Problem :2 Marks, Technique for remove problem any 3: 2 Marks Each)**

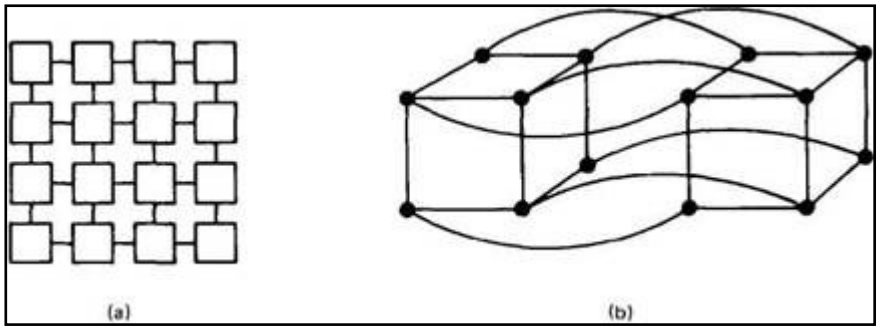


The problem of maintaining a proper reference count in the presence of unreliable communication

Another problem that needs to be resolved occurs when copying a remote reference to another process. If process P1 passes a reference to process P2, the object. or more precisely, its skeleton, will be unaware that a new reference, has been created. Consequently, if process P1 decides to remove its own reference, the reference counter may drop to zero and 0 may be deleted before P2 ever contacts it. This problem is illustrated in fig below (a).

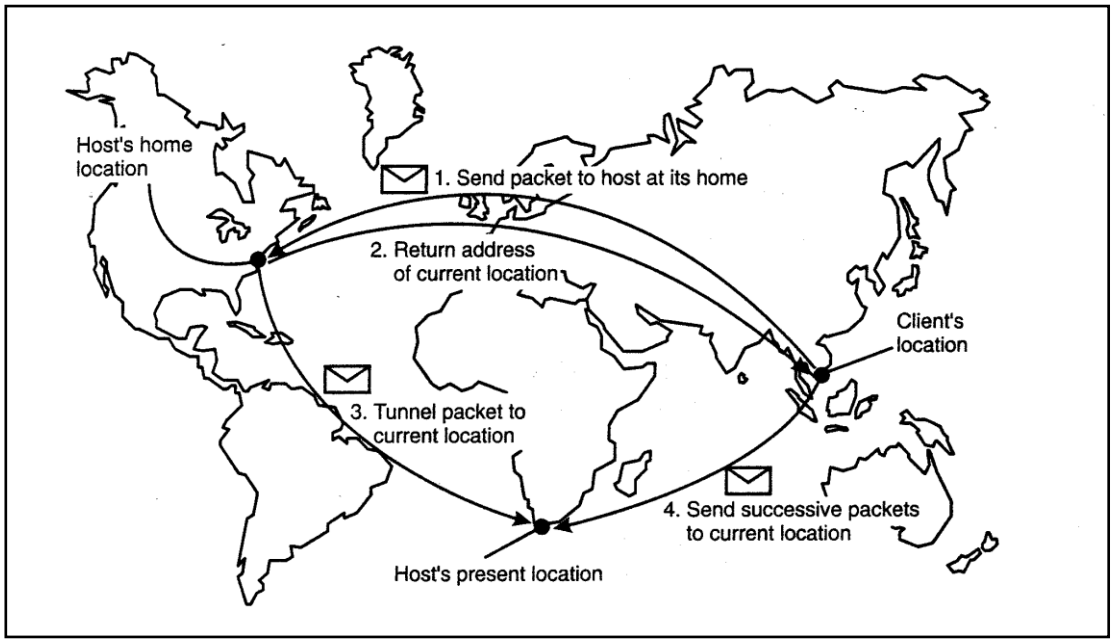


A solution is to let P1 inform the object's skeleton that it is going to pass a reference to process P2. In addition, a process is never allowed to remove a reference before the skeleton has acknowledged that it knows about the existence of that reference. This solution is shown in fig above (b). The acknowledgement sent by O to P2 confirming to P2 that O has registered the reference, will later permit P2 to delete its reference to O. As long as P2 is not sure that O knows about its reference, P2 is not allowed to request O to decrement the reference counter.

		1. **Reference Listing** A different approach to managing references, is to let a skeleton keep track of the proxies that have a reference to it. In other words, instead of counting references, a skeleton maintains an explicit first of all proxies that point to it. Such a reference list has the following important property. Removing a proxy from a list in which it did not occur. Also has no effect. Adding or removing proxies are thus idempotent operations. 2. **Identifying Unreachable entities** The collection of entities in distributed system may consist of entities that store reference to each other, but none of these entities can be reached from an entity in the root set, and as such, they should also be removed. What is needed is method by which all entities in a distributed system are traced. In general, this is done by checking which entities can be reached from the root set and subsequently removing all others. Such methods are generally called tracing based garbage collection. In contrast to distributed referencing discussed so far, tracing based garbage collection has inherent scalability problems, as it needs to trace all entities in a distributed system.	
4.		**Attempt any TWO :**	**16 Marks**
	(a)	**Write a short note on :** **(i) Homogeneous Multicomputer System.** **(ii) Heterogeneous Multicomputer System.**	**8M**
	Ans:	**(i) Homogeneous Multicomputer System.**  • **Grids** are easy to understand and lay out on printed circuit boards • They are best suited to problems that have an inherent two-dimensional nature, such as graph theory or vision (e.g., robot eyes or analyzing photographs). • **A hypercube** is an n–dimensional cube.	**(Diagram: 2 Marks, Explanation: 2 Marks)**

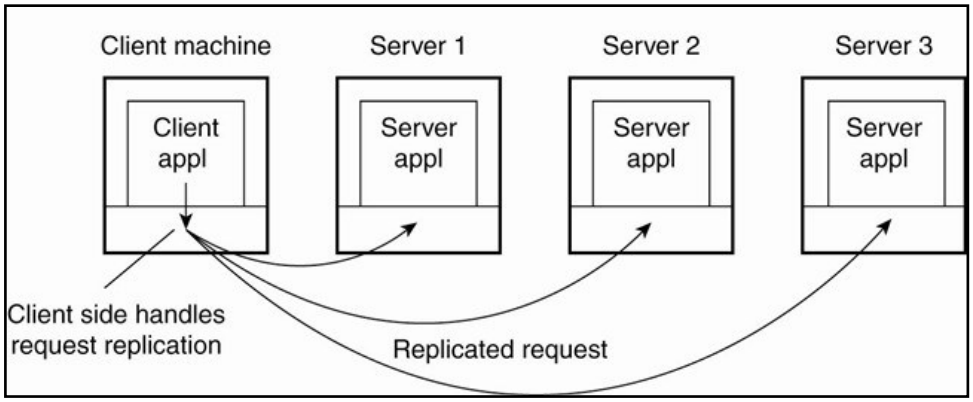
		• Each vertex is a CPU. • Each edge is a connection between two CPUs. **(ii) Heterogeneous Multicomputer System.** **1. Distributed Operating system:** • DOS is a tightly-coupled operating system for multiprocessors and homogenous multicomputer. • Its main goal is to hide and manage hardware resources. • This is a single timesharing system. • This property can be referred as single system image. • Distributed system is one that runs on collection of network machines but acts like virtual uniprocessor. **2. Network Operating System:** • NOS is loosely coupled operating system for heterogeneous multicomputer (LAN and WAN). • Its main goal is to offer local services to remote clients. • Typical example is network of workstation connected by LAN. • In this model each user have workstation for his exclusive use. • It provides shared, global file system accessible from all workstations. • File system is supported by 1 or more machines called file server. • As shown in following fig. file server accept request from user program running on other machine. • These machines are called clients. • Clients may read or write file.	**(Explanation of Heterogeneous Multicomputer System:4 Marks)**
	(b)	**Describe how to locate mobile entities naming and also give home based approaches for the same.**	**8M**
	Ans:	A popular approach to supporting mobile entities in large-scale networks is to Introduce a home location, which keeps track of the current location of an entity. Special techniques may be applied to safeguard against network or process failures. In practice, the home location is often chosen to be the place where an entity was created Each mobile host uses a fixed IP address. All communication to that IP address is initially directed to the mobile host's home agent. This home agent is located on the local-area network corresponding to the network address contained in the mobile host's IP address. In the case of IPy6, it is realized as a network-layer component. Whenever the mobile host moves to another network, it requests a temporary address that it can use for communication. This care-of address is registered at the home agent.	**(Diagram :2 Marks, Explanation: 6 Marks)**

When the home agent receives a packet for the mobile host, it looks up the host's current location. If the host is on the current local network, the packet is simply forwarded. Otherwise, it is tunneled to the host's current location, that is, wrapped as data in an IP packet and sent to the care-of address. At the same time, the sender of the packet is informed of the host's current location. This principle is shown in Figure Note that the IP address is effectively used as an identifier for the mobile host.



	(c)	**Describe different cloud deployment model.**	**8M**
	Ans:	**1. <u>Public Cloud Model:</u>** In this model all (general public) can access applications and services. Many providers exist, in which some of them are Google, Amazon, Microsoft etc. **Advantages:** • It is cost effective • It is flexible • It is reliable • It is location Independent • It is highly scalable **Disadvantages:** • It has low security as it is available in public and may be attacked by virus or DDoS. • It is less customizable than other clouds. **2. <u>Private Cloud Model:</u>** In this model only people of particular organization can access the applications and services. It can also be managed internally or by a third party.	**(2 marks for Each model)**

		Advantages: • As it is accessible within an organisation it has high security and privacy. • You can have more control over the cloud **Disadvantages:** • Restricted area • High pricing • Limited scalability **3. Hybrid Cloud Model:** This model is the combination of public and private Cloud models. In this you can perform non critical activities using public cloud and critical activities using private cloud. **Advantages:** • Scalability • Flexibility • Cost efficient **Disadvantages:** • There may be security issues. • Infrastructural Dependency **4. Community Cloud Model :** In this model a group of organizations can use the same cloud ie., here the infrastructure is shared by a group of organizations and can be maintained internally or with the help of third party. **Advantages:** • Cost effective **Disadvantages:** • Security	
5.		**Attempt any TWO :**	**16 Marks**
	(a)	**Explain about RPC and RMI.**	**8M**
	Ans:	**RPC :** When a process on machine A calls' a procedure on machine B, the calling process on A is suspended, and execution of the called procedure takes place on B. Information can be transported from the caller to the callee in the parameters and can come back in the procedure result. No message passing at all is visible to the programmer. This method is known as Remote Procedure Call, or often just RPC. **RMI :** RMI (Remote Method Invocation) is a way that a programmer, using the Java programming language and development environment, can write object-oriented programming in which objects on different computers can interact in a distributed network. RMI is the Java version of what is generally known as a remote procedure call (RPC), but with the ability to pass one or more objects along with the request. The object can include information that will change the service that is performed in the remote computer.	**(Explain about RPC :4 Marks)** **(Explain about RMI: 4 Marks)**

	(b)	Explain about client-side software for distribution transparency.	8M
	Ans:	 • Client software includes more than just user interfaces. • Client should not be aware that it is communicating with remote process. • Sometimes, parts of the processing and data level in a client-server application are executed on the client side. • **Distribution transparency**- o client software includes components for achieving distribution transparency. o distribution is less transparent to servers for reasons of performance and correctness. • **Access transparency** – o This is handled through the generation of a client stub from an interface definition of what the server has to offer. o The stub provides the same interface as available at the server, but hides the possible differences in machine architectures, as well as the actual communication. • There are different ways to handle **location, migration, and relocation transparency.** o For **example**, when a client is already bound to a server, the client can be directly informed when the server changes location. o In this case, the client's middleware can hide the server's current geographical location from the user. o Client also transparently rebind to the server if necessary. o At worst, the client's application may notice a temporary loss of performance. • **Replication transparency** o In a similar way, many distributed systems implement replication transparency by means of client-side solutions. o For example, imagine a distributed system with replicated servers, Such replication can be achieved by forwarding a request to each replica, as shown in Figure (d).	**(Diagram :4 Marks)** **(Any 4 transparency , Each transparency : 1 Mark)**

- Client-side software can transparently collect all responses and pass a single response to the client application.
- **Failure transparency**-
  - Masking communication failures with a server is typically done through client middleware.
  - For example, client middleware can be configured to repeatedly attempt to connect to a server, or perhaps try another server after several attempts.
- **Concurrency transparency**:
  - This can be handled through special intermediate servers, notably transaction monitors.
  - This requires less support from client software.
- **Persistence transparency** is often completely handled at the server.

(c) Discuss the impact of cloud computing on users. **8M**

Ans:

i. Security and Privacy

1. The main challenge to cloud computing is how it addresses the security and privacy concerns of businesses thinking of adopting it.
2. The fact that the valuable enterprise data will reside outside the corporate firewall raises serious concerns.
3. Hacking and various attacks to cloud infrastructure would affect multiple clients even if only one site is attacked.
4. These risks can be mitigated by using security applications, encrypted file systems, data loss software, and buying security hardware to track unusual behavior across servers.

ii. Service Delivery and Billing

1. It is difficult to assess the costs involved due to the on-demand nature of the services.
2. Budgeting and assessment of the cost will be very difficult unless the provider has some good and comparable benchmarks to offer.
3. The service-level agreements (SLAs) of the provider are not adequate to guarantee the availability and scalability.
4. Businesses will be reluctant to switch to cloud without a strong service quality guarantee.

iii. Interoperability and Portability

Businesses should have the leverage of migrating in and out of the cloud and switching providers whenever they want, and there should be no lock-in period.
Cloud computing services should have the capability to integrate smoothly with the on-premise IT
It is important to monitor the service being provided using internal or third-party tools.

(Any 2 impacts each carries: 4 Marks)

		iv. Performance and Bandwidth Cost Businesses can save money on hardware but they have to spend more for the bandwidth. 1. This can be a low cost for smaller applications but can be significantly high for the data-intensive appli 2. Delivering intensive and complex data over the network requires sufficient bandwidth. 3. Because of this, many businesses are waiting for a reduced cost before switching to the cloud.	
6.		**Attempt any TWO :**	**16 Marks**
	(a)	**Describe about message-oriented and stream-oriented communication.**	**8M**
	Ans:	Different forms of Message-oriented communication **Message-oriented transient communication** Message is stored only so long as sending/receiving application are executing Discard message if it can't be delivered to next server/receiver Example: transport-level communication services offer transient communication Example: Typical network router – discard message if it can't be delivered next router or destination Berkeley socket MPI **Stream-Oriented Communication** • With RPC, RMI and MOM, the effect that time has on correctness is of little consequence. • However, audio and video are time-dependent data streams – if the timing is off, the resulting "output" from the system will be incorrect. • Time-dependent information – known as "continuous media" communications. • Example: voice: PCM: 1/44100 sec intervals on playback. • Example: video: 30 frames per second (30-40 msec per image). • KEY MESSAGE: Timing is crucial! Stream Definition: A (continuous) data stream is a connection-oriented communication facility that supports isochronous data transmission Some common stream characteristics: o Streams are unidirectional o There is generally a single source , and one or more sinks o Often, either the sink and/or source is a wrapper around hardware (e.g., camera, CD device, TV monitor, dedicated storage)	**(Description of message-oriented communication: 4 Marks)** **(Description of stream-oriented communication:4 Marks)**
	(b)	**Describe how processes migrate in heterogeneous system.**	**8M**
	Ans:	The migrated code can be easily executed at the target machine. This assumption is in order when dealing with homogeneous systems. In general, however, distributed systems are constructed on a heterogeneous collection of platforms, each having their own operating system and machine architecture. Migration in such systems requires that each	**(Explanation: 8 Marks)**

		platform is supported, that is, that the code segment can be executed on each platform. There are, in principle, three ways to handle migration: 1. Pushing memory pages to the new machine and resending the ones that are later modified during the migration process. 2. Stopping the current virtual machine; migrate memory, and start the new virtual machine. 3. Letting the new virtual machine pull in new pages as needed, that is, let processes start on the new virtual machine immediately and copy memory pages on demand.	
	(c)	**Write a short note on SAas and PAas**	**8M**
	Ans:	**Software as a Service (SaaS) Model:** • In this model a software is deployed on hosted service. • It is accessible through internet. • It allows providing software application to the users. • Billing and Invoicing System, customer relationship management (CRM) applications, Help Desk Applications are some of SaaS applications. • The software's license is available based on usage or subscription. • They are cost effective and requires less maintenance. • In this multiple users can share an instance and is not required to code functionality of each user. • Scalability, efficiency, performance are the benefits of SaaS. • The issues with this model are Lack of portability between SaaS clouds and browser based risks. **Platform as a Service (PaaS) Model:** • This model acts as a run time environment. • It allow to develop and deploy tools required for the applications. • It has a special feature that helps non developers to create web applications. • This also offers API and development tools required to develop an application. • The benefits of this model are low cost of ownership and scalable solutions. • But the disadvantage is, in PaaS the consumer's browser has to maintain reliable and secure connections to the provider systems. There is also a lack of probability between PaaS clouds.	**(SaaS Model :4 Marks)** **(PaaS Model: 4 Marks)**