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 judgment 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.

Marks

1) Attempt any FIVE of the following: 20

a) Write down the goals of Distributed operating system.
(Any two goals with description; each goal consist - 2 marks)

Ans:

1. **Should make resources easily accessible**
   - **a.** make it easy for the users (and applications) to access remote resources
   - **b.** To share them in a controlled and efficient way.

2. **Distributed Transparency:**
   - **a.** should reasonably hide the fact that resources are distributed across a network
   - **b.** hide the fact that its processes and resources are physically distributed across multiple computers – systems should be transparent

3. **Openness:**
   - **a.** should be open
   - **b.** Offer services according to standard rules that describe the syntax and semantics of those services.

4. **Scalability:**
   - **a.** Should be scalable.
   - **b.** Scalability of a system is measured with respect

- Size - can easily add more users and resources to the system.
- Geographic extent - a geographically scalable system is one in which the users and resources may lie far apart.
- Administrative scalability - can be easy to manage even if it spans many independent administrative organizations.

b) Describe basic RPC operations.

(Any four operations each consist - 1 mark)

Ans:

A remote procedure call occurs in the following steps:

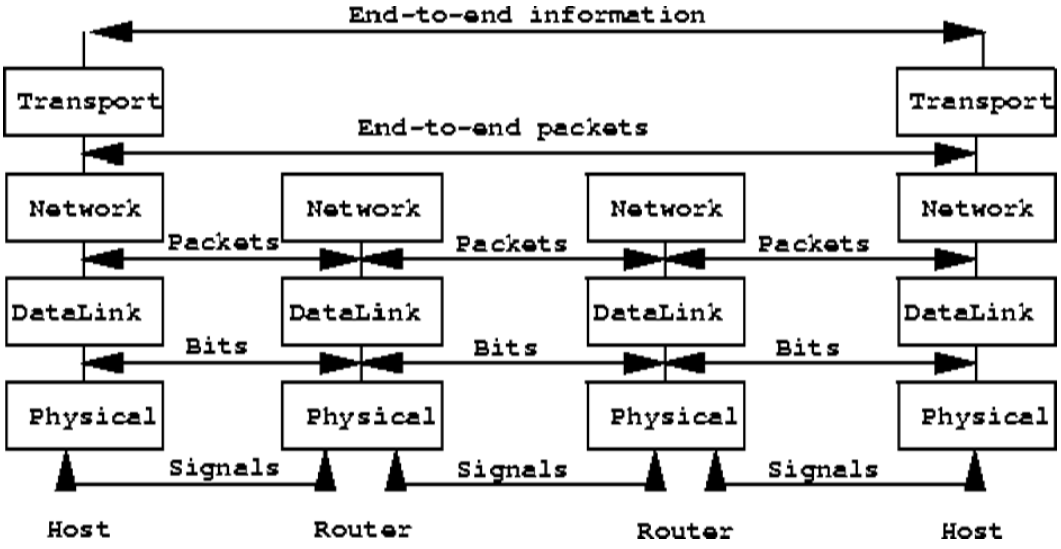
1. The client procedure calls the client stub in the normal way.
2. The client stub builds a message and traps to the kernel.
3. The kernel sends the message to the remote kernel.
4. The remote kernel gives the message to the server stub.
5. The server stub unpacks the parameters and calls the server.
6. The server does the work and returns the result to the stub.
7. The server stub packs it in a message and traps to the kernel.
8. The remote kernel sends the message to the client's kernel.
9. The client's kernel gives the message to the client stub.
10. The stub unpacks the result and returns to the client.

c) Explain about low level protocol.

(Figure - 2 marks; Protocol - 2 marks)

Ans:

Figure 1: Protocol Layers



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)

d) Describe the concept of threads in distributed system.

(Definition - 1 mark; multithreaded client with suitable theory - 1.5 marks; multithreaded client with suitable theory - 1.5 marks)

Ans:

Thread: A thread is the smallest unit of processing that can be performed in an OS. In most modern operating systems, a thread exists within a process - that is, a single process may contain multiple threads.

Multithreaded Client:

- 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.

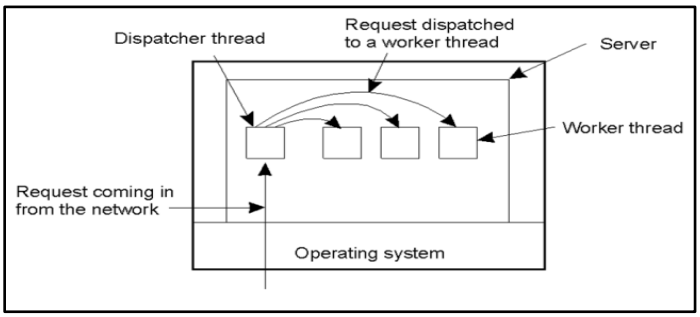
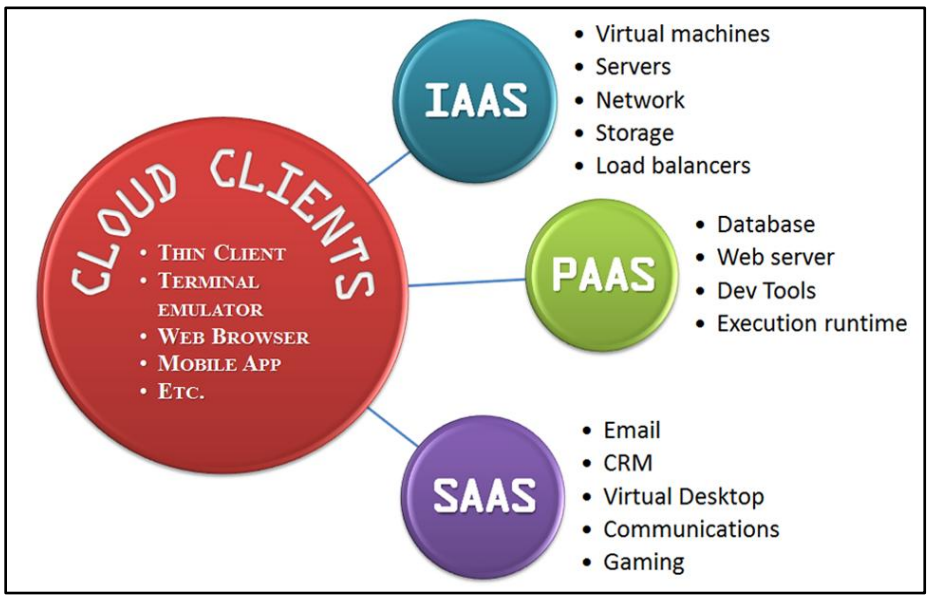


Fig: Multithreaded server

- 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.

e) Enlist SPI model and describe any one in detail.
(*Diagram - 1 mark; each type - 1 mark)*

Ans:



Following three models are also called as service models.
Infrastructure as a Service (IaaS) Model:

- This model offers access to physical resources like physical and virtual machines, data storage. Virtual Machines
- They are preinstalled as an operating system.
- They allow to store a specific data in different locations and also computing resources can be easily scaled up and down.
- It offers disk storage, local area network, load balancers, IP addresses, software bundles etc.
- This model also allows the cloud provider to locate infrastructure over the internet.

- IaaS helps in controlling the computing resources via administrative access to virtual machines.
- There are also some issues along with the benefits and the issues are compatibility with legacy security vulnerabilities, virtual machine sprawl etc.

Platform as a Service (PaaS) Model:

- This model acts as a run time environment.
- It allows developing and deploying 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.

Software as a Service (SaaS) Model:

- In this model 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.

f) What is recursive name resolution?

(Diagram - 2 marks; Related Theory - 2 marks)

Ans:

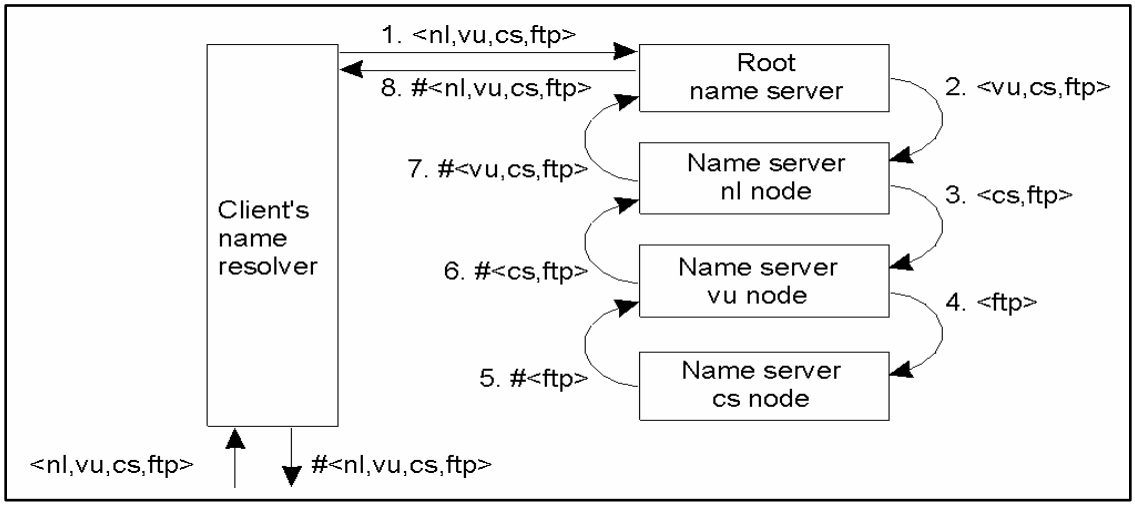


Fig. Recursive Name resolution

Name resolution is the process of mapping an object's name to the object's properties, such as its location.

Instead of returning each intermediate result back to the client's name resolver, each name server passes the result to the next name server it finds

- Puts a higher performance demand on each name server (higher level nodes: iterative)
- Caching results is more effective
- Communication costs may be reduced

g) Differentiate between process and thread.
(Any four Differences; Each for 1 mark)

Ans:

Sr. No.	Process	Thread
01	Processes are the abstraction of running programs	Threads are the unit of execution in a process
02	Process is heavy weight or resource intensive	Thread is light weight taking lesser resource than a process.
03	Process switching needs interaction with operating system.	Thread switching does not neet to interact with operating system
04	In multiple processing environments each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
05	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
06	In multiple processes each process operates independently of the others	One thread can read, write or change another threads data.
07	If one process is blocked then no other process can execute until the first process is unblocked	While one thread is blocked and waiting second thread in the same task can run.

2) Attempt any Four of the following: 16

a) Define and illustrate about distributed system
(Definition - 2 marks; any two Goals - 2 marks)

Ans:

A distributed system is a collection of independent computers that appear to users of the system as a single computer.

This definition has two aspects:

1. **Deals with hardware:** Machines are autonomous.
2. **Deals with software:** The users think of system as a single computer.

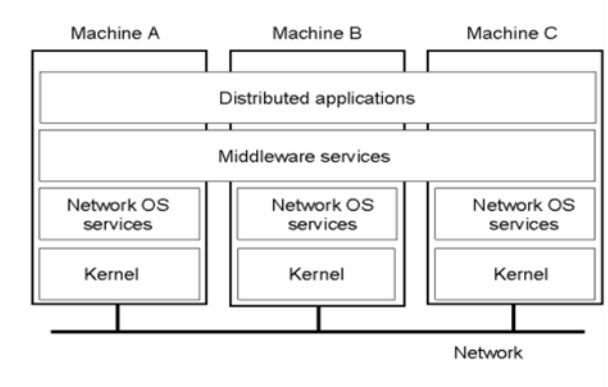
Example: An example of large bank with thousands of branch offices all over the world. Each office has a master computer. Master computer stores local accounts and handle local transactions.

Goals of distributed system:

1. Connecting users and resources:
2. Goals of distributed system:
3. Openness
4. Scalability
5. Concurrency

b) Describe about middleware concept.
(Diagram - 2 marks; related theory - 2 marks)

Ans:



General structure of a distributed system as middleware.

- Middleware is a additional layer atop of NOS implementing general-purpose services.
- Its main goal is to be providing the distribution transparency.
- **Middleware can provide following services:**
  (a) Communication
  (b) Naming
  (c) Persistence storage
  (d) Security
  (e) Distributed transaction

c) Explain different forms of communication in message oriented communication
(Any two forms - 4 marks; each consist - 2 marks with suitable points)

Ans:

Different forms of Message-oriented communication

Persistent communication:

- Messages are stored until (next) receiver is ready
- Examples: email, pony express

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

Message-oriented synchronicity communication

- Asynchronous communication
  - Sender continues immediately after it has submitted the message
  - Need a local buffer at the sending host
- Synchronous communication
  - Sender blocks until message is stored in a local buffer at the receiving host or actually delivered to sending
  - Variant: block until receiver processes the message

d) Write any three ways to handle code migration
(Diagram - 2 marks; any two segments - 2 marks)

Ans:

A process consists of three segments:

- Code segment
- Resource segment
- Execution segment

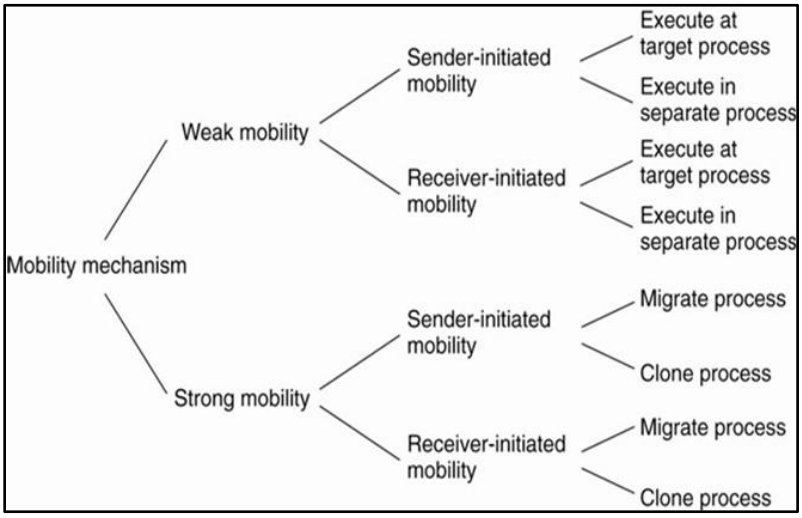
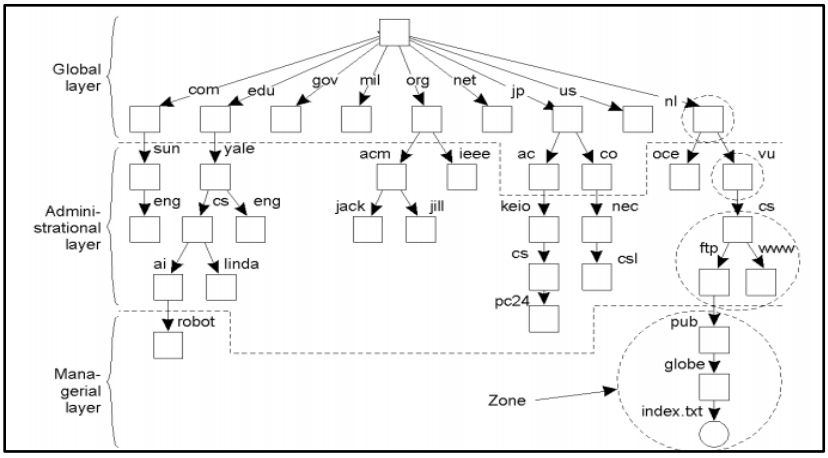


Figure (l) Alternatives for code migration.

e) Explain how implementation of name space is done
(Diagram with related theory - 4 marks)

Ans:



(Figure partitioning of the DNS name space)

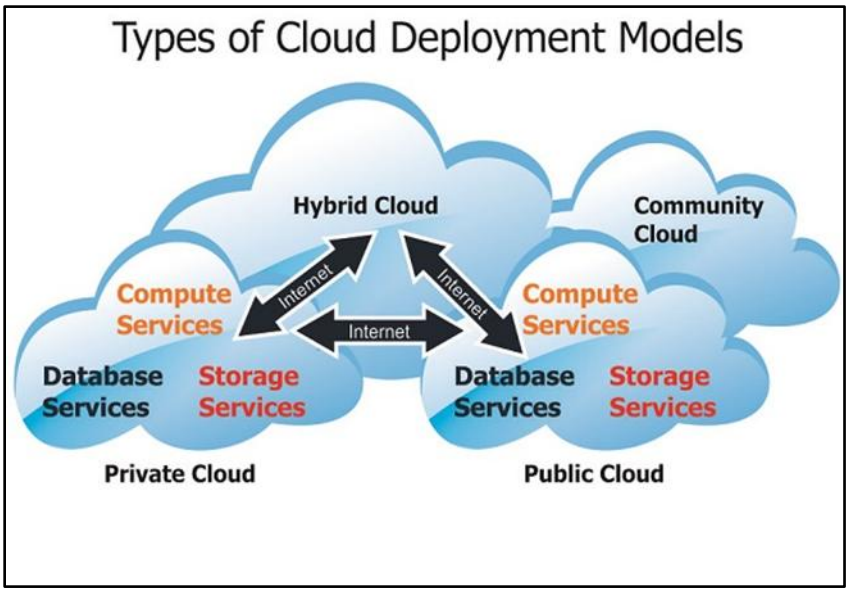
- Figure shows an example of the partitioning of part of the DNS name space, including the names of files within an organization that can be accessed through the Internet, for example, Web pages and transferable files.
- The name space is divided into non overlapping parts, called zones in DNS.
- A zone is a part of the name space that is implemented by a separate name server.

- Naming service: a service that allows users and processes to add, remove, and lookup names
- Name spaces for large-scale widely distributed systems are typically organized hierarchically
- Three layers used to implement such distributed name spaces
- Global layer: root node and its children
- Administrational layer: directory nodes within a single organization
- Managerial layer

f) State about different cloud deployment models.
(Diagram - 2 marks; any two models - 2 marks)

Ans:

Cloud Deployment Model



Public Cloud Model:
In this model all (general public) can access applications and services. Many providers exist, in which some of them are Google, Amazon, and 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.

Private Cloud Model:

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.

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

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

Community Cloud Model: In this model a group of organizations can use the same cloud i.e., 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

3) Attempt any FOUR of the following: 16

a) Write down the advantages of distributed operating system over centralized operating system.

(Each suitable advantage carries - 1 mark)

Ans:

Speed:-When used to implement parallel processing where only goal is to achieve maximum speed on a single problem, distributed systems can achieve very high speed as compared to the centralized ones.

Inherent Distribution:-Another reason for building a distributed system is that some applications are inherently distributed. Banking, Airline reservation etc. are examples of the applications that are inherently distribute. When all the branches of a bank are connected, we have a commercial distributed system.

Reliability: - Another potential advantage of a distributed system over a centralized one is higher reliability. By distributing the workload over many machines, a single chip failure will bring down at most one machine, leaving the rest intact. Ideally, if 5 percent of the machines are down at any moment, the system should be able to work with a 5 percent loss in performance. For critical applications, such as control of nuclear reactors or aircraft, using a distributed system to achieve high reliability may be a dominant consideration

Incremental Growth:- Finally, incremental growth is also potentially a big plus. Often a company will buy a mainframe with the intention of doing all its work on it. If the company prospers and the workload grows, at a certain point the mainframe will no longer be adequate.

b) Explain the process of static remote invocation

(Explanation - 2 marks; static remote invocation - 2 marks)

Ans:

The **RMI** (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.

The RMI provides remote communication between the applications using two objects *stub* and *skeleton*.

Steps to write the RMI program

There is given the 6 steps to write the RMI program.

- Create the remote interface
- Provide the implementation of the remote interface
- Compile the implementation class and create the stub and skeleton objects using the rmic tool
- Start the registry service by rmiregistry tool
- Create and start the remote application
- Create and start the client application

Static Remote Invocation

- Addition of new code without updating client code is not possible
- Programming is made easy
- Type checking is robust
- Code is self-explanatory
- Good performance in terms if flow of control
- Tools cant check services that will be provided at runtime

c) Explain How Quality of Service can be achieved in stream oriented communication.
(Explanation - 2 marks; QOS - 2 marks)

Ans:

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!

Definition:

- A (continuous) data stream is a connection-oriented communication facility that supports isochronous data transmission

Some common stream characteristics:

- Streams are unidirectional
- There is generally a single source , and one or more sinks
- Often, either the sink and/or source is a wrapper around hardware (e.g., camera, CD device, TV monitor, dedicated storage)

Quality of Service

- Definition: "ensuring that the temporal relationships in the stream can be preserved".
- QoS is all about three things:
  - Timeliness,
  - Volume and
  - Reliability.
- Most current operating systems and networks do not include the QoS management facilities
- Bleeding edge of the discipline

Must specifying the following:

- The required bit rate at which data should be transported.
- The maximum delay until a session has been set up (i.e., when an application can start sending data).

- The maximum end-to-end delay (i.e., how long it will take until a data unit makes it to a recipient).
- The maximum delay variance or jitter.

d) What are the various alternatives of code migration
(Explanation - 2 marks; Diagram 2 marks)

Ans:

The various alternatives for code migration are summarized in Fig. 3-8.

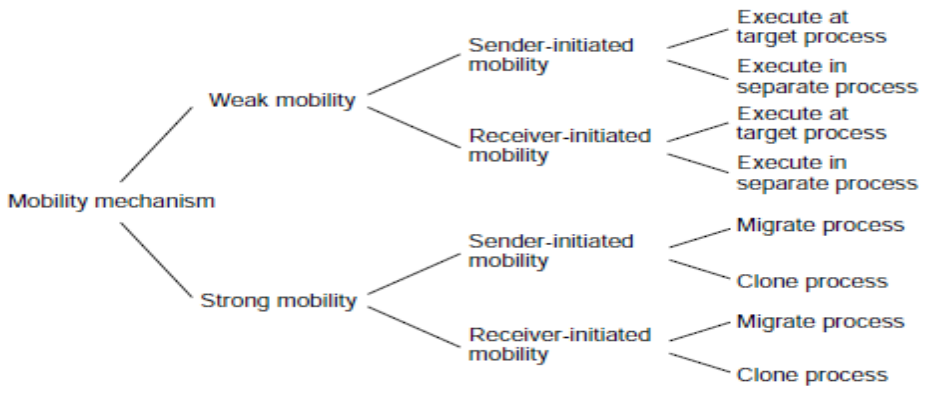


Figure 3-8. Alternatives for code migration.

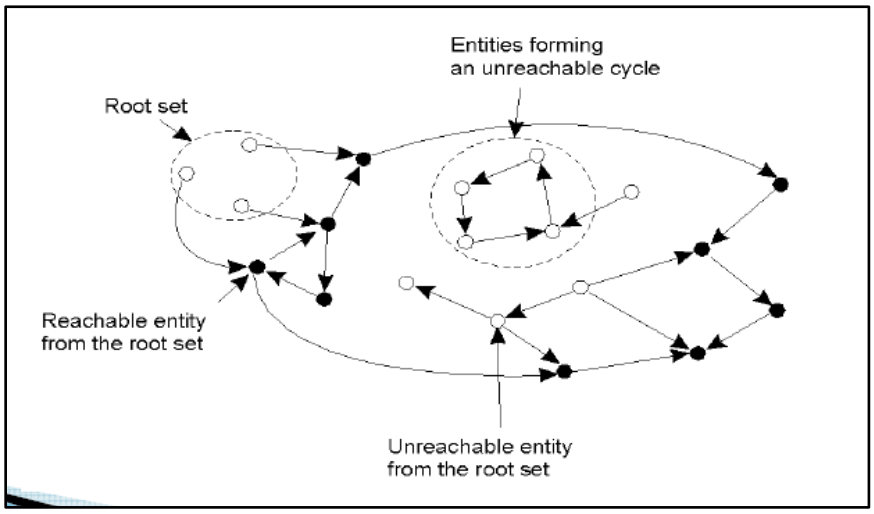
In the case of weak mobility, it also makes a difference if the migrated code is executed by the target process, or whether a separate process is started. For example, Java applets are simply downloaded by a Web browser and are executed in the browser's address space. The benefit of this approach is 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. Note that this solution does not solve the resource-access problems just mentioned. Instead of moving a running process, also referred to as process migration, strong mobility can also be supported by remote cloning. In contrast to process migration, cloning yields an exact copy of the original process, but now running on a different machine. The cloned process is executed in parallel to the original process. In UNIX systems, remote cloning takes place by forking off a child process and letting that child continue on a remote machine. The benefit of cloning is that the model closely resembles the one that is already used in many applications. The only difference is that the cloned process is executed on a different machine. In this sense, migration by cloning is a simple way to improve distribution transparency.

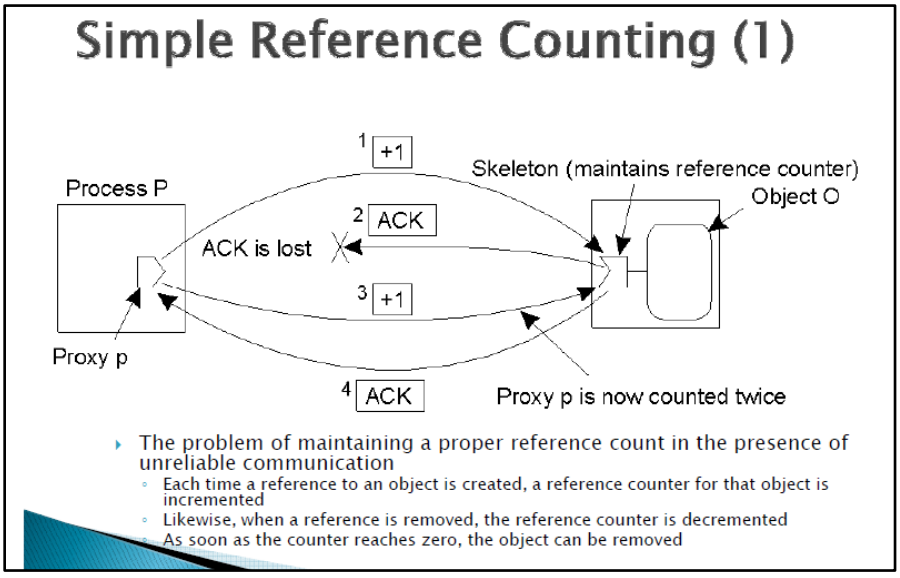
e) What is reference listing and reference counting

(Reference listing - 2 marks, static reference counting 2 - marks)

Ans:

As soon as an entity can no longer be accessed, it should be removed An example of a graph representing objects containing references to each other





Reference listing:-

Instead of counting references, a skeleton maintains an explicit list of all proxies that point to it Adding or removing proxies are idempotent operations – they can be repeated without affecting

the end result Consequently, reference listing does not require communication to be reliable Main drawback: may scale badly if the skeleton needs to keep track of many references

f) Explain Elements of grid computing system
(*Each suitable element carries - 1 mark*)

Ans:

Elements of Grid Computing

Grid computing combines elements such as distributed computing, high-performance computing and disposable computing depending on the application of the technology and the scale of operation. Grids can create a virtual supercomputer out of the existing servers, workstations and personal computers.

Present-day grids encompass the following types:

- Computational grids, in which machines will set aside resources to "number Crunch" data or provide coverage for other intensive workloads
- Scavenging grids, commonly used to find and harvest machine cycles from idle servers and desktop computers for use in resource-intensive tasks (scavenging is usually implemented in a way that is unobtrusive to the owner/user of the processor)
- Data grids, which provide a unified interface for all data repositories in an organization, and through which data can be queried, managed and secured.
- Market-oriented grids which deal with price setting and negotiation, grid economy management and utility driven scheduling and resource allocation.

The key components of grid computing include the following.

- Resource management: a grid must be aware of what resources are available for different tasks
- Security management: the grid needs to take care that only authorized userscan access and use the available resources
- Data management: data must be transported, cleansed, parceled and processed
- Services management: users and applications must be able to query the grid in an effective and efficient manner

4). Attempt any FOUR of the following: 16

a) What are the issues concerned with parameter passing in RPC system
(*Any four issues each issue carries - 1 mark*)

Ans:

RPC Issues

Marshalling:

Parameters must be *marshalled* into a standard representation. Parameters consist of simple types (e.g., integers) and compound types (e.g., C structures or Pascal records). Moreover, because each type has its own representation, the types of the various parameters must be known to the modules that actually do the conversion. For example, 4 bytes of characters would be uninterrupted, while a 4-byte integer may need to the order of its bytes reversed.

Semantics:

Call-by-reference not possible: the client and server don't share an address space. That is, addresses referenced by the server correspond to data residing in the client's address space. One approach is to simulate call-by-reference using *copy-restore*. In copy-restore, call-by-reference parameters are handled by sending a copy of the referenced data structure to the server, and on return replacing the client's copy with that modified by the server. However, copy-restore doesn't work in all cases. For instance, if the same argument is passed twice, two copies will be made, and references through one parameter only changes one of the copies.

Binding:

How does the client know *who* to call, and *where* the service resides? The most flexible solution is to use *dynamic binding* and find the server at run time when the RPC is first made. The first time the client stub is invoked, it contacts a name server to determine the transport address at which the server resides.

Transport protocol:

What transport protocol should be used?

Exception handling:

How are errors handled?

Binding We'll examine one solution to the above issues by considering the approach taken by Birrell and Nelson [Binding consists of two parts:

Naming refers to what service the client wants to use. In B&N, remote procedures are named through *interfaces*. An interface uniquely identifies a particular service, describing the types and numbers of its arguments. It is similar in purpose to a type definition in programming languauges. For example, a ``phone'' service interface might specify a single string argument that returns a character string phone number.

Locating refers to finding the transport address at which the server actually resides. Once we have the transport address of the service, we can send messages directly to the server.

b) Describe client side software for distribution transparency.
(Explanation - 2 mark; Diagram - 2 mark)

Ans:

1. Client-Side Software for Distribution Transparency

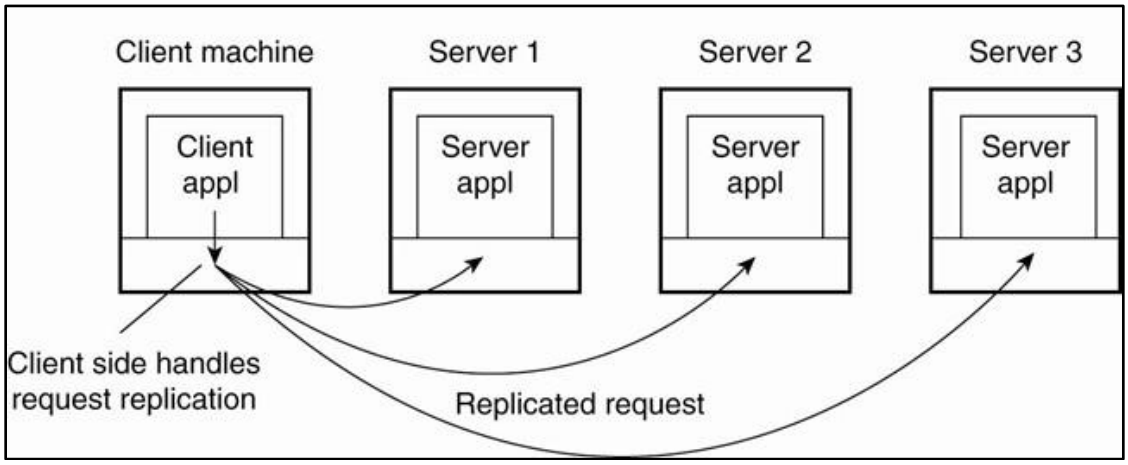


Figure d Transparent replication of a server using a client-side solution.

- 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**-
  - client software includes components for achieving distribution transparency.
  - distribution is less transparent to servers for reasons of performance and correctness.
- **Access transparency** –
  - This is handled through the generation of a client stub from an interface definition of what the server has to offer.
  - 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.**
  - For **example**, when a client is already bound to a server, the client can be directly informed when the server changes location.
  - In this case, the client's middleware can hide the server's current geographical location from the user.
  - Client also transparently rebind to the server if necessary.
  - At worst, the client's application may notice a temporary loss of performance.

- **Replication transparency**
  - In a similar way, many distributed systems implement replication transparency by means of client-side solutions.
  - 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).
  - 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) Explain client to server binding.

(Explanation - 2 marks; Diagram - 2 marks)

Ans:

Second Design Issue: where clients contact a server?

- In all cases, clients send requests to an end point, also called a port, at the machine where the server is running.
- Each server listens to a specific end point.
- How do clients know the end point of a service? Solution is to globally assign end points for well-known services.
- There are many services that do not require a preassigned end point.
- In that case, a client will first have to look up the end point.
- One solution is to have a special daemon running on each machine that runs servers.
- The daemon keeps track of the current end point of each service implemented by a co-located server.
- The daemon itself listens to a well-known endpoint.
- A client will first contact the daemon, request the end point, and then contact the specific server, as shown in Figure (g).
- It is common to associate an end point with a specific service.
- However, actually implementing each service by means of a separate server may be a waste of resources.

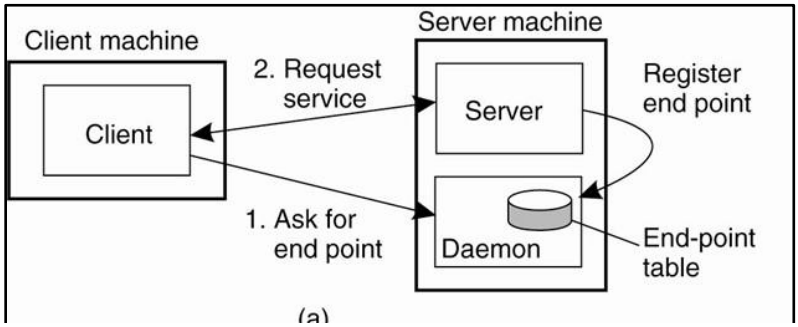


Figure (g). Client-to-server binding using a daemon

Third Design issue: 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 transmission.
- There are several ways to do this. One approach that works only too well in the current Internet is for the user to abruptly exit the client application which will automatically break the connection to the server, immediately restart it, and pretend nothing happened. The server will eventually tear down the old connection, thinking the client has probably crashed. A much better approach for handling communication interrupts is to develop the client and server such that it is possible to send out-of-band data, which is data that is· to be processed by the server before any other data from that client.
  One solution is to let the server listen to a separate control end point to which the client sends out-of-band data, while at the same time listening (with a lower priority) to the end point through which the normal data passes

d) Describe simple solution for locating entities.
(Explanation - 2 marks; Solution - 2 marks)

Ans:

Simple solutions:

Simple Solutions that work in a LAN environment

- Broadcasting & Multicasting
- Message containing identifier of the entity is broadcast; machine with an access point for the entity replies with the address of the access point – ARP protocol for finding the data-link address of a machine given the IP address
- Forwarding pointers

- When an entity moves from A to B, it leaves behind a reference to its new location at B

Broadcasting

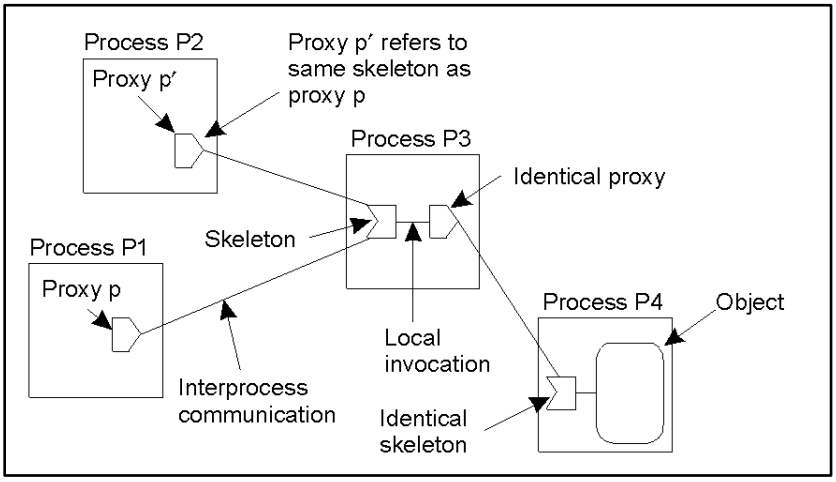
- Simply broadcast the id to each machine, each machine is requested to check whether it has that entity, and if so, send a reply message containing the address of the access point.
  - Can never scale beyond local-area networks (think of ARP/RARP).
  - Requires all processes to listen to incoming location requests

Multicasting:

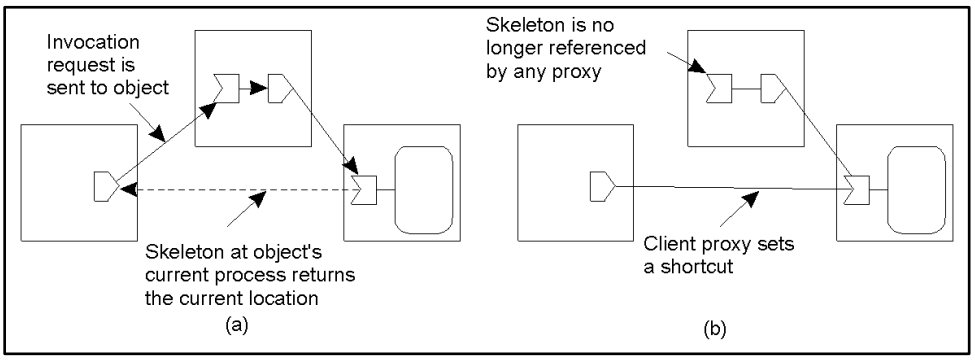
- It can be used to locate entities in point to point network.
- Example: Internet supports network level multicasting by allowing hosts to join a specific multicast group. Such groups are identified by a multicast address.
- When a host sends a message to multicast address, the network layer provides a best effort service to deliver that message to all group members.
- A multicast address can be used as a general location service for multiple entities.
- **Example:** Consider an organization where each employee has his or her own mobile computer. When such computer connects to locally available network. It is dynamically assigned an IP address. It joins a specific multicast group. When a process wants to locate computer A, it sends a where is request to multicast group. If A is connected , it responds with its current IP address.

Forwarding pointer:

- Each time an entity moves, it leaves behind a pointer telling where it has gone to. Dereferencing can be made entirely transparent to clients by simply following the chain of pointers
- Update a client's reference as soon as present location has been found
- **Geographical scalability problems:**
  - Long chains are not fault tolerant
  - Increased network latency at dereferencing
  - Essential to have separate chain reduction mechanisms
  - Forwarding pointers for distributed objects Each forwarding pointer is implemented as a (proxy, skeleton) pair A skeleton (i.e., server-side stub) contains a local reference to the actual object or a local reference to a proxy (i.e., client-side stub) for the object Skeleton (entry items for remote references) Proxies (exit items) When an object moves from A to B, it leaves behind a proxy in its place in A and installs a skeleton that refers to it in B. Transparent to the client



- The principle of forwarding pointers using (proxy, skeleton) pairs.
- skeletons acts as entry items for remote references, proxies as exit items,
- Whenever move from A to B, proxy installed on A and referring skeleton on B.

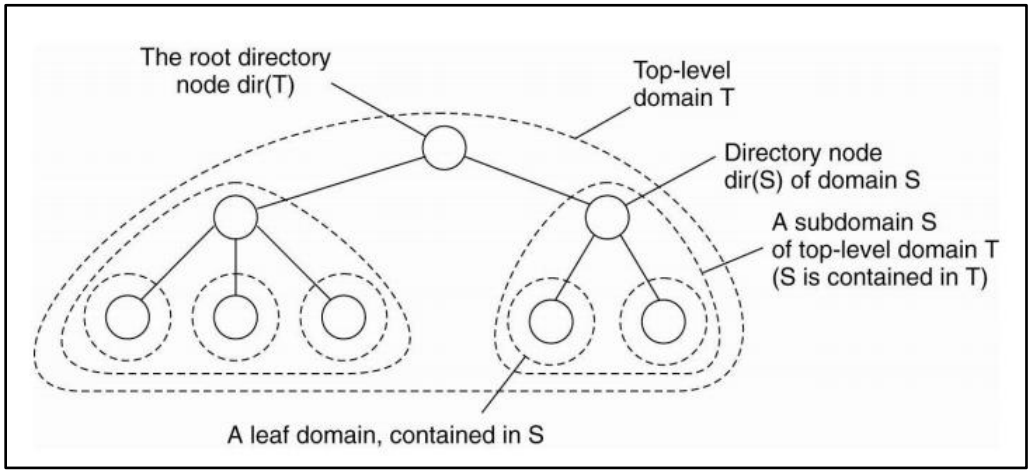


- Redirecting a forwarding pointer, by storing a shortcut in a proxy
- The response can be sent directly, or along the whole reverse path, thus updating all intermediate proxies
- Pointer forwarding is fully transparent,

e) Explain hierarchical location services for locating mobile entities.
(Explanation - 2 marks; Diagram - 2 marks)

Ans:

Hierarchical Approach



Hierarchical organization of a location service into domains, each having an associated directory node. Basic idea: Build a large-scale search tree for which the underlying network is divided into hierarchical domains. Each domain is represented by a separate directory node dir(d) Leaf domains typically correspond to a local-area network or a cell The root (directory) node knows all the entities Each entity currently in a domain D is represented by a location record in the directory node dir(D) which is the entity's current address or a pointer

f) Describe grid architecture with neat diagram
(Explanation - 2 marks; Diagram - 2 marks)

Ans:

Grid computing architecture:

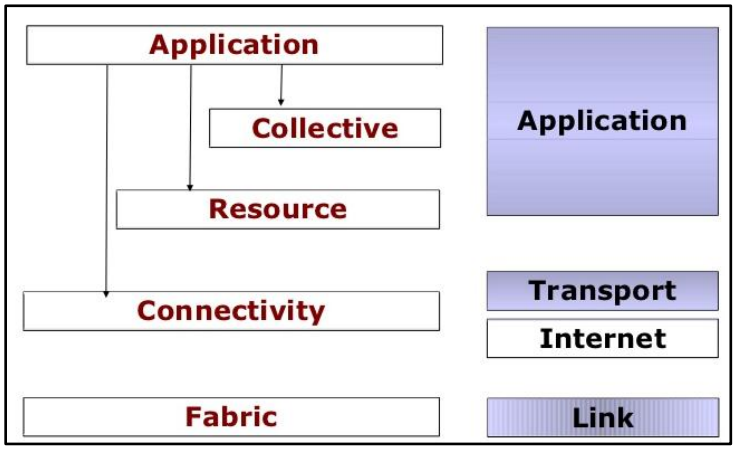


Fig.2. Grid computing layered architecture

- The architecture consists of four layers.
- **The lowest fabric layer**
  - This provides interfaces to local resources at a specific site.
  - These interfaces are tailored to allow sharing of resources within a virtual organization.
- **The connectivity layer**
  - This consists of communication protocols for supporting grid transactions that span the usage of multiple resources.
  - For example, protocols are needed to transfer data between resources, or to simply access a resource from a remote location.
  - The connectivity layer will contain security protocols to authenticate users and resources.
- The **resource layer**
  - This is responsible for managing a single resource.
  - It uses the functions provided by the connectivity layer and calls directly the interfaces made available by the fabric layer.
  - For example, this layer will offer functions for obtaining configuration information on a specific resource.

- **The collective layer**.
  - It deals with handling access to multiple resources and typically consists of services for resource discovery, allocation and scheduling of tasks onto multiple resources, data replication, and so on.
  - Unlike the connectivity and resource layer, which consist of a relatively small, standard collection of protocols, the collective layer may consist of many different protocols for many different purposes, reflecting the broad spectrum of services it may offer to a virtual organization.

- **The application layer**
  - It consists of the applications that operate within a virtual organization and which make use of the grid computing environment.

5). Attempt any FOUR of the following **16**

(a) Describe the concept of Homogeneous multicomputer system

(Diagram - 2 marks; explanation - 2 marks)

Ans:

Homogeneous Multicomputer System:

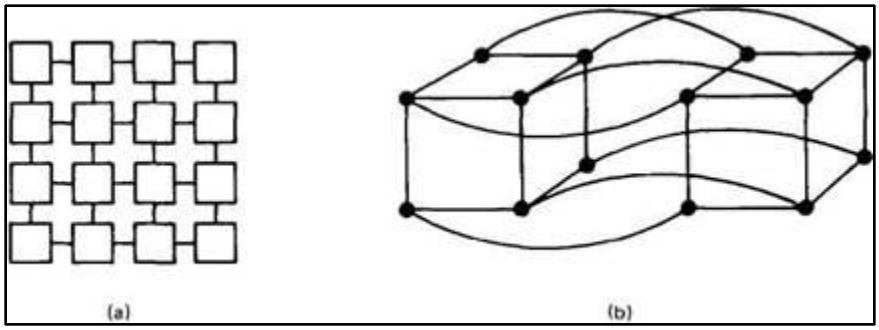


Fig. (a) Grid. (b) Hypercube.

- **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.
- Each vertex is a CPU.
- Each edge is a connection between two CPUs.

Software Concepts:

- **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.

➢ **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.

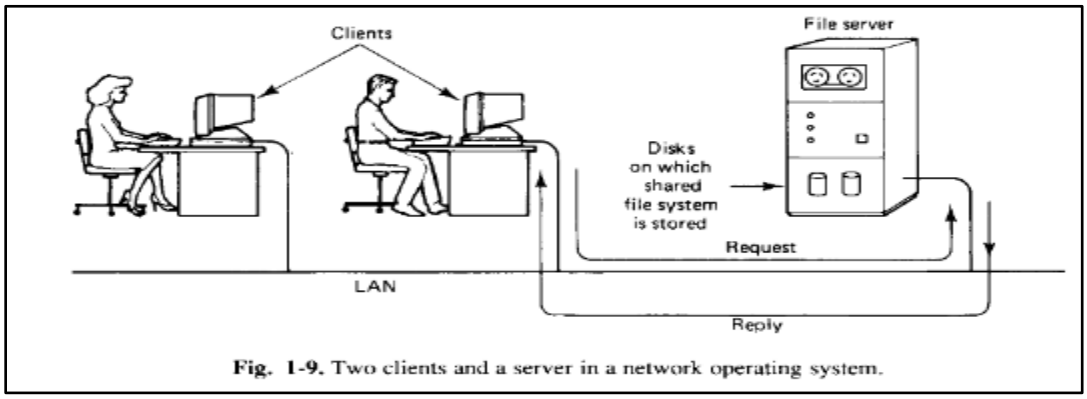
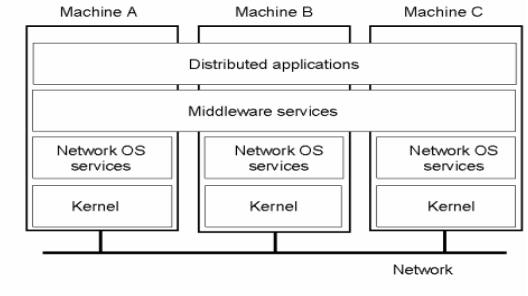


Fig. 1-9. Two clients and a server in a network operating system.

➢ **Middleware:**

- Middleware is a additional layer atop of NOS implementing general-purpose services.
- Its main goal is to be providing the distribution transparency.
- In an open middleware based distributed system the protocol used by each middleware layer should be the same as well as the interface they offer to application.



General structure of a distributed system as middleware.

(b) Write down the design issues for RMI

(*Any two issues each consist of 2 marks*)

Ans:

- **Handling errors**
  - **retry request?**
  - **duplicate filtering?**
  - **retransmission of results?**
- **Semantics**
  - **Maybe**
  - **At least once**
  - **At most once**
- **like a local call**
  - **marshalling/unmarshalling**
  - **locating remote objects**
  - **accessing/syntax**
- **latency more likely to fail**
- **Errors/exceptions: failure of the network? server? hard to tell consistency on the remote machine. Argus: incomplete transactions, abort, restore states [as if the call was never made]**

(c) (Write a note on agent technology
(Explanation - 2 marks; types of agents - 2 marks)

Ans:

- "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)

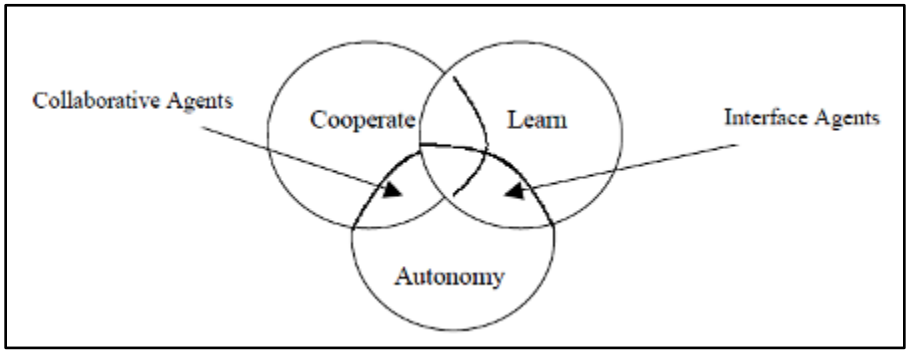


Figure (n) A partial view of agent classification

- **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.

(d) Describe the problem of unreturned objects.
(Related theory - 4 marks)

Ans:

Distributed operating system:

Integration of system services presenting a transparent view of a multiple computer system with distributed resources and control.

Consisting of concurrent processes accessing distributed shared or replicated resources through message passing in a network environment.Problems Unique to Distributed Systems

- **Distributed Operating Systems:**
  - **Characteristics:** Global view of file system, name space, time, security, computational power.
  - **Goal:** Single computer view of multiple computer system (transparency)
- **Efficiency problem:**
  - Communication delays
  - Data propagation
  - Overhead of communication protocols
  - Load distribution
- **Consistency Problem:**
  - User's perspective:
  - Uniformity in using the system

- Predictability of the system's behavior

- **System's perspective:**
  - Integrity maintenance
  - Concurrency control
  - Failure handling
  - Recovery procedures
- **Robustness Problems:**
  - Fault tolerance
  - What to do when a message is lost?
  - Handling of exceptional situations and errors
  - Changes in the system topology
  - Long message delays
  - Inability to locate a server
  - Security for the users and the system
- **Design Issue Example:**

  The resources in a distributed system are spread across different computers and a naming scheme has to be devised so that users can discover and refer to the resources that they need.

  An example of such a naming scheme is the URL (Uniform Resource Locator) that is used to identify WWW pages. If a meaningful and universally understood identification scheme is not used then many of these resources will be inaccessible to system users.

(e) Describe the impact of cloud computing on users
(Four Related impact each consist of 01 mark)

Ans:

- Cloud computing challenges have always been there.
- Companies are increasingly aware of the business value that cloud computing brings and are taking steps towards transition to the cloud.
- A smooth transition entails a thorough understanding of the benefits as well as challenges involved.
- Like any new technology, the adoption of cloud computing is not free from issues. Some of the most important challenges are as follows.
  - **Security and Privacy**
    - The main challenge to cloud computing is how it addresses the security and privacy concerns of businesses thinking of adopting it.
    - The fact that the valuable enterprise data will reside outside the corporate firewall raises serious concerns.
    - Hacking and various attacks to cloud infrastructure would affect multiple clients even if only one site is attacked.
    - 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.

- **Service Delivery and Billing**
  - It is difficult to assess the costs involved due to the on-demand nature of the services.
  - Budgeting and assessment of the cost will be very difficult unless the provider has some good and comparable benchmarks to offer.
  - The service-level agreements (SLAs) of the provider are not adequate to guarantee the availability and scalability.
  - Businesses will be reluctant to switch to cloud without a strong service quality guarantee.
- **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.
- **Reliability and Availability**
  - Cloud providers still lack round-the-clock service; this results in frequent outages.
  - It is important to monitor the service being provided using internal or third-party tools.
  - It is vital to have plans to supervise usage, SLAs, performance, robustness, and business dependency of these services.
- **Performance and Bandwidth Cost**
  - Businesses can save money on hardware but they have to spend more for the bandwidth.
  - This can be a low cost for smaller applications but can be significantly high for the data-intensive applications.
  - Delivering intensive and complex data over the network requires sufficient bandwidth.
  - Because of this, many businesses are waiting for a reduced cost before switching to the cloud.

All these challenges should not be considered as road blocks in the pursuit of cloud computing. It is rather important to give serious consideration to these issues and the possible ways out before adopting the technology.

(f) Mention difference between cloud computing and grid computing
(For every point - 1/2 marks)

Ans:

Feature	Grid	Cloud
Resource Sharing	Collaboration (VOs, fair share).	Assigned resources are not shared.
Resource Heterogeneity	Aggregation of heterogeneous resources.	Aggregation of heterogeneous resources.
Virtualization	Virtualization of data and computing resources.	Virtualization of hardware and software platforms.
Security	Security through credential delegations.	Security through isolation.
High Level Services	Plenty of high level services.	No high level services defined yet.
Architecture	Service orientated.	User chosen architecture.
Software Dependencies	Application domain-dependent software.	Application domain independent software.
Platform Awareness	The client software must be Grid-enabled.	The SP software works on a customized environment.
Software Workflow	Applications require a predefined workflow of services.	Workflow is not essential for most applications.
Scalability	Nodes and sites scalability.	Nodes, sites, and hardware scalability.
Self-management	Reconfigurability.	Reconfigurability, self-healing.
Centralization Degree	Decentralized control.	Centralized control (until now).
Usability	Hard to manage.	User friendliness.
Standardization	Standardization and interoperability.	Lack of standards for Clouds interoperability.
User Access	Access transparency for the end user.	Access transparency for the end user.
Payment Model	Rigid.	Flexible.
QoS Guarantees	Limited support, often best-effort only.	Limited support, focused on availability and uptime.

6). Attempt any FOUR of the following. 16

(a) Write down difference between distributed operating system and network operating system
(For every point - 1/2 marks)

Ans:

DOS		NOS	
1	Tightly coupled operating systems for multiprocessors and homogenous multi-computers	**1**	Loosely coupled operating system for heterogeneous multi computers
2	Follow n tire architecture	**2**	Follow 2 tire architecture
3	Degree of transparency is very high	**3**	Degree of transparency is very low
4	For communication shared memory is used	**4**	For communication files are used
5	Resource management is global	**5**	Resource management is per node
6	Low scalability	**6**	High scalability
7	Openness is closed	**7**	Openness is open
8	Goal of DOS is to hide and manage hardware resources	**8**	Goal of NOS is to offers local services to remote clients

(b) Describe distributed objects with working of client side
(Related Theory - 4 marks)

Ans:

1. The idea of distributed objects is an extension of the concept of remote procedure calls.
2. In a system for distributed objects, the unit of distribution is the object. That is, a client imports a "something" "full blown" object-based distributed systems include Corba and DCOM. Binding a client to object, Key feature of an object: it encapsulates data, the state, and the operations on those data, the methods.
3. Methods are made available through an interface.
4. The separation between interfaces and the objects implementing these interfaces is crucial for distributed systems.

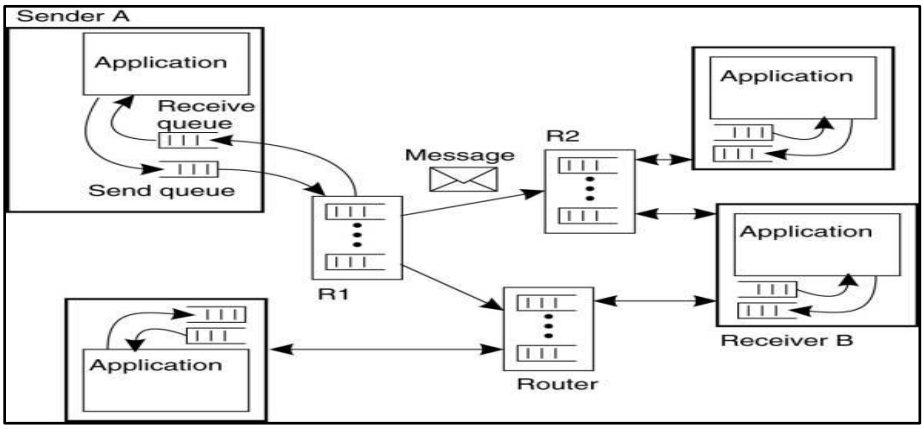
(c) Describe general architecture of message queuing system for persistence communication
(Explanation - 2 marks; diagram - 2 marks)

Ans:

General Architecture of a Message-Queuing System

- Message-Queuing System Architecture
- Messages are "put into" a source queue.

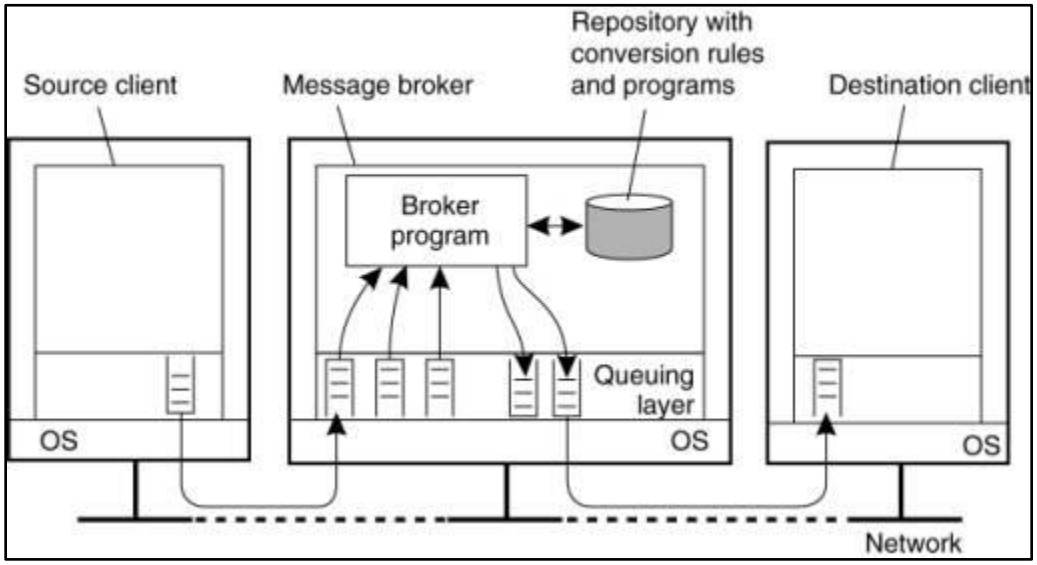
- They are then "taken from" a destination queue.
- Queues are managed by queue managers
- They move a message from a source queue to a destination queue.
- Special queue managers operate as routers or relays: they forward incoming messages to other queue managers.
- The general organization of a message-queuing system with routers.



The Role of Message Brokers

- Often, there's a need to integrate new/existing apps into a "single, coherent Distributed Information System (DIS)".
- Problem: different message formats exist in legacy systems (cooperation and adherence to open standards was not how things were done in the past).
- It may not be convenient to "force" legacy systems to adhere to a single, global message format (cost!?).
- It is often necessary to live with diversity (there's no choice).
- Solution: the "Message Broker".

- Message Brokers (aka "Interface engine")
- In message-queuing systems, conversions are handled by special nodes in a queuing network, known as message brokers.
- A message broker acts as an application-level gateway in a message-queuing system.
- Purpose - convert incoming messages so that they can be understood by the destination application.
- Note: a message broker is just another application - not considered to be an integral part of the queuing system.
- Message brokers can be simple (reformat messages) or complex (find associated applications, convert data)
- The general organization of a message broker in a message-queuing system.



Message-Queuing (MQ) Applications

General-purpose MQ systems support a wide range of applications, including:

- Electronic mail.
- Workflow.
- Groupware.
- Batch Processing.

Most important MQ application area:

- Integration of a widely dispersed collection of database applications (which is all but impossible to do with traditional RPC/RMI techniques).

(d) Describe migration in heterogeneous system

(Any four points each for 01 mark)

Ans:

Migration in heterogeneous system:

- Heterogeneous systems are platforms with different operating systems and/or different machine architectures
- Problems with heterogeneous systems is similar to those of portability Solutions
  - Highly portable languages
- Scripting languages, interpreted languages (Java)
  - Migrating computing environments
  - Migrating virtual machines
- Distributed systems are constructed on a heterogeneous collection of platforms.
- Distributed systems each having their own operating system and machine architecture.
- Migration in such systems requires that each platform is supported, That the code segment can be executed on each platform.

- Also, we need to ensure that the execution segment can be properly represented at each platform.three ways to handle migration (which can be combined)

(e) Explain software agents in distributed system
(Explanation - 2 marks; types - 2 marks)

Ans:

Software agents

- "An agent is an entity thatacts 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)

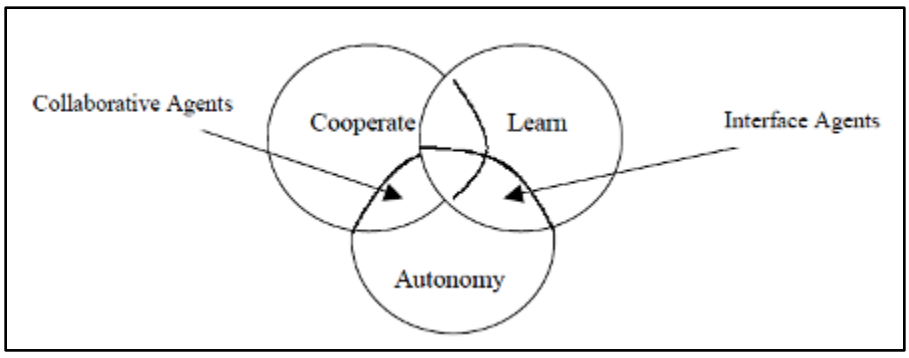


Figure (n) A partial view of agent classification

- **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, and then it will try to suggest better ways of doing the same task.

- **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.

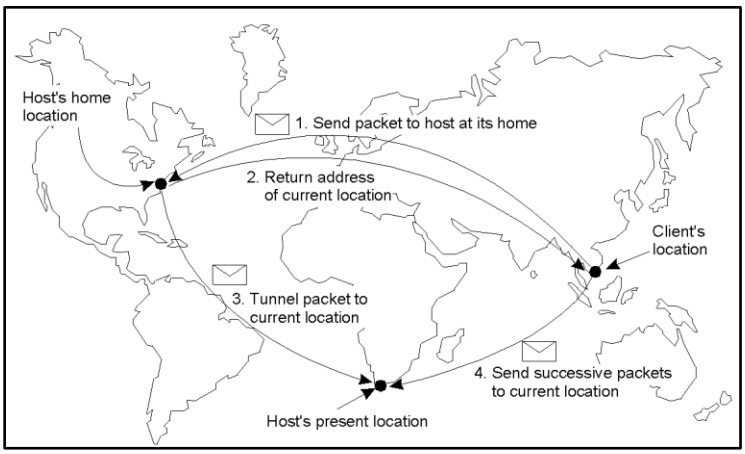
This is called the mobile agent environment, which is a software system distributed over a network of heterogeneous computers and its primary task is to provide an environment in which mobile agents can run.

(f) Describe home based approaches for locating Mobile Entities
(Two types each - 2 marks)

Ans:

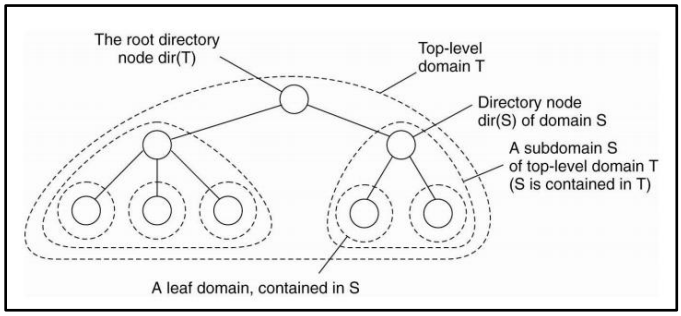
1. **Locating Mobile Entities**
   - **Home-based approaches**
   - **Hierarchical approaches**

- **Home Based approach:**



The principle of Mobile IP. home location, home agent in home LAN, fixed IP address, whenever the mobile host in another network requests a temporary care of-address, registered afterwards at the home agent.

- **Hierarchical Approach**



Hierarchical organization of a location service into domains, each having an associated directory node.Basic idea: Build a large-scale search tree for which the underlying network is divided into hierarchical domains. Each domain is represented by a separate directory node dir(d) Leaf domains typically correspond to a local-area network or a cell The root (directory) node knows all

the entities Each entity currently in a domain D is represented by a location record in the directory node dir(D) which is the entity's current address or a pointer