1.9 Models

The system models of the distributed systems are classified in three parts.

1. Architectural models
2. Fundamental models
3. Client-server models

The explanation of each of the models is given below.

1.9.1 Architecture Models

In a distributed system, the architectural model is concerned with the placement different parts of the system and the relationships among them. Since there a various components of the distributed system, they must be structured in such a w that the present and future demands of the system are met. So a architectural mod will first simplify and abstract the individual components of the distributed syste

Then it will consider the placement of the components in the network and finally it will also consider the interrelationship between the components.

There are basically three architectural models for the distributed systems.

A) Workstations/servers model ;

B) Processor pool (thin client) model ;

C) Integrated model.

We will elaborate in brief on all the above models.

A) Workstation Model

The main components of the workstation models are

a) Workstations

b) Application programs

c) Various servers

d) Interface

e) Unique Use ID

f) Filestore.

In a workstation model, each user has a workstation. There can be a network of workstations or a cluster of workstations.

The application programs run on the workstation. Various services are to be provided by the distributed system. Services may be like managing the directory, authentication, news, printing, gateway, mail, etc. There are specialized servers who are responsible for giving the designated services in the workstation model.

The workstations may share a common set of resources or a common interface. They may be in the integrated fashion. There is a network of workstations but a user ID is unique across the entire network of workstations. Any user is allowed to use any workstation.

The **filestore** in the distributed systems must be system wide and this is a mandatory aspect of the distributed system. Some of the workstations may have their own file stores but due to transparency feature of the distributed systems, the files must be accessible to other users of the system.

A user can run the application programs remotely on other workstations. If a cluster management system is used then the user can submit the jobs transparently to the network of workstations.

Since there are many jobs running in the distributed environment the cluster management system handles the resource allocation, scheduling and queueing aspects of the jobs.

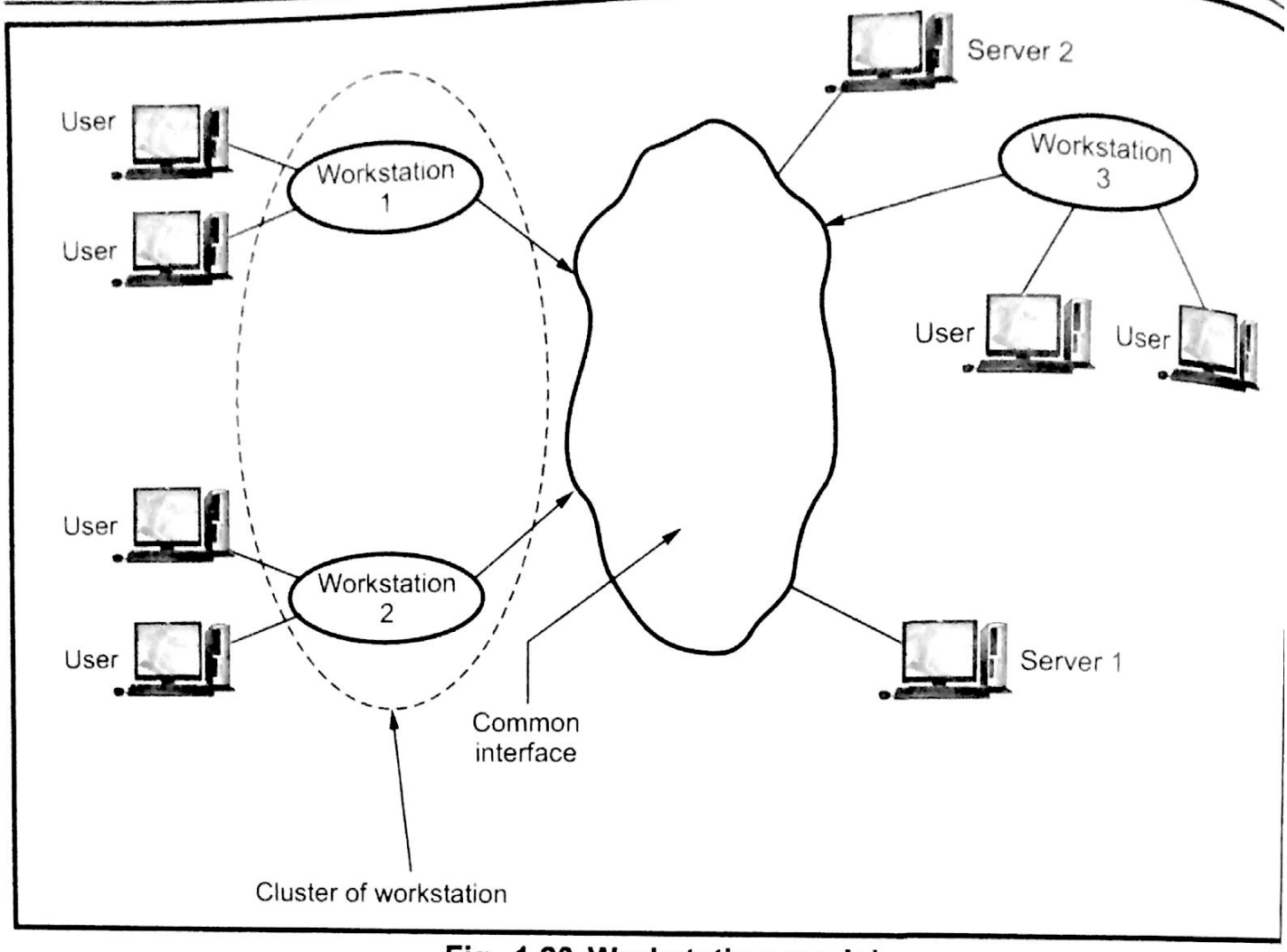

Fig. 1.23 Workstation model

Important features of workstation model :

1) Users with unique id

They can remotely login to any workstation.

2) Common interface

Used by all workstation and servers.

3) Specialized servers

Specialized services are provided by the servers in the workstation model.

B) Processor Pool (thin client) Model

In a workstation model, there may be a case that there are many idle workstations or many idle computing resources. They can be completely ignored, but it is not an intelligent way of working. The options possible are

- In a distributed environment, the idle processors can be utilised by the other O.S. on the system. Some processes can run on the idle processor and time can be consumed.
- This can be done manually also. The user can move the processes on the idle machine.

In the processor pool model, there is a collection of processes and the CPU is namically assigned to processes on demand.

The processor pool model of the distributed system is a collection of terminals for cess to the system. It consists of a

a) Pool of a processors

The pool of processors can be distributed memory clusters or shared memory servers.

b) Servers

There can be other servers like the fileservers.

The processor pool model is becoming very common now days. This is because mponents like X-Terminals, sunrays, this clients; network interface computers (NICs) c. are being added to the system.

Advantages of Processor Pool Models

- It is having centralized hardware and software and hence easier to manage.
- Addition and removal of pool processors is easy. This is possible without affecting other users.
- The old pool processors never die. They gradually outlive their usefulness.
- Since the softwares are centralized, the licensing cost is reduced.
- The processor utilisation is better as compared to workstation model.
- Migration to new hardware platforms is easier.

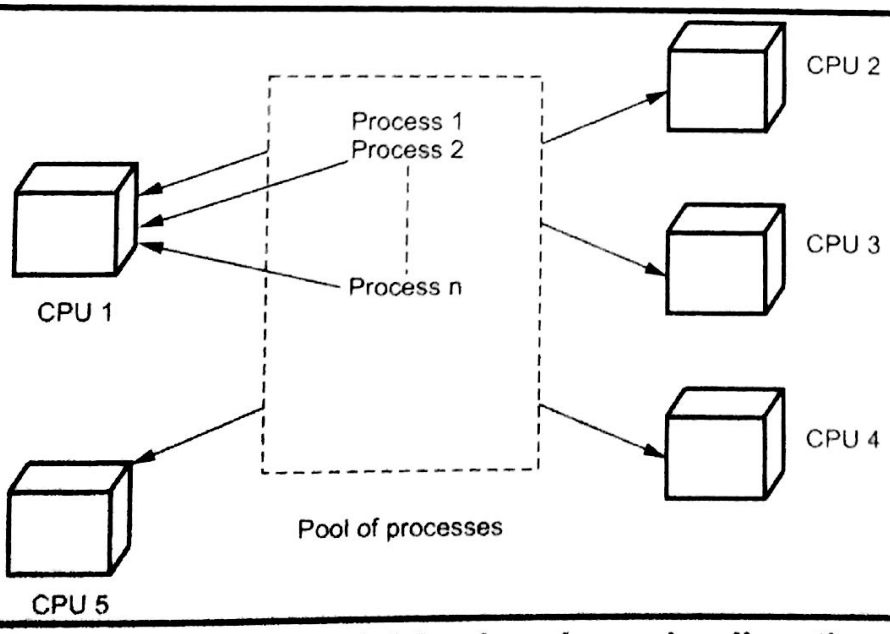

Fig. 1.24 Processor pool model having dynamic allocation of CPU's

C) The Integrated Model

In the totally integrated model all platforms are on single network-wide stributed O.S.

The main features of the integrated model are

- Completely seamless
- Completely transparent
- Completely mythical

The various platforms across the distributed system are completely separate or seamless with each other. The transparency feature is also embedded in the completely mythical or traditional model of a distributed system.

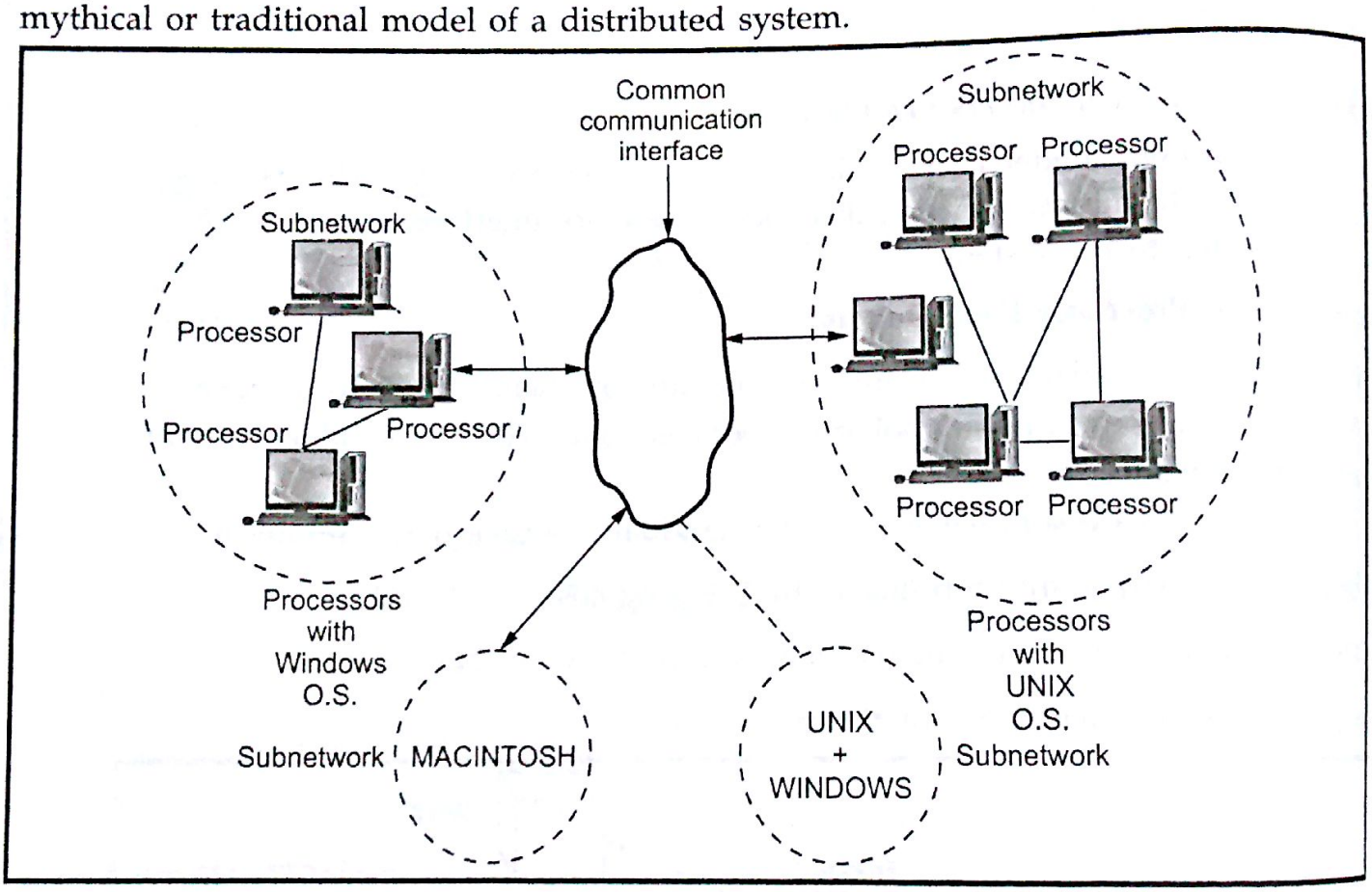

Fig. 1.25 Totally Integrated model of distributed system

1.9.2 Fundamental Models

Some of the properties are common in all architectural models. The fundamental models are concerned with the formal description of such properties. Depending on the properties, the fundamental models are classified as

A) The Interaction Model

B) The Failure Model

C) The Security Model.

Every model which is designed should take care of the following points.

- It must necessarily specify all the entities in the system.
- It must also specify the necessary means and ways of communication among various entities.

- It should also mention the characteristics which will affect either the individual or the collective behaviour of the entities.
- The model should explicitly mention all the assumption made about the system being considered.
- The model must provide means to make all generalizations based on basic assumptions.

We will elaborate on the various models below.

A) The Interaction Model

The interaction model deals with the interaction feature of the distributed system. In the system, there are various processes. These processes communicate with each other by various communication mechanisms. They send and receive messages among each other. The interaction model must be designed in such a manner that the communication between various processes is not delayed.

In a distributed system there are various ways in which communication can take place.

- There may be various servers providing specific services. For example,
  a) Various file servers
  b) DOMAIN name server
  c) Network information service provider.
- Many processes may communicate with each other for specific purpose.

  For example, video conferencing.

The processes in this system communicate with each other in the real time nvironment. Various distributed algorithms are designed to enhance the ommunication between various processors in a better sophisticated manner.

Since communication is the basic feature in the interactive model, its performance valuation is a matter of concern.

Various issues related to performance of communication channels are :

a) Network traffic

The network traffic leads to delay in transmission and reception.

b) Bandwidth

Bandwidth deals with the total amount of information that can be transmitted at a iven time. If there are various processes sharing the same bandwidth, then it may ead to delay of transmission and reception.

c) Jitter

When a series of messages are to be transmitted, there may be variation in the time taken to send each part of the message. This is called as delay. For example, if a sound file is to be transmitted and there is variation is transmission of the samples, it results in noise.

So, the performance of a communication channel is an important issue to be handled in interactive models.

There are two types of interactive models :

i) Synchronous models and

ii) Asynchronous models.

i) Synchronous Model

In a synchronous model, strong assumption about time is made.

- It is assumed that every processes execution time is known perfectly with lower bounds and upper bounds.
- It is also assumed that any message transmitted over a channel takes a specific amount of time.

A synchronous model of a distributed system can be built if the processors are guaranteed with sufficient processor cycles and network capacity.

ii) Asynchronous Model

It is very difficult to design an synchronous distributed system with strong time assumptions.

An alternative model in the asynchronous model. In such type of models there is no bound on

- Process speed.
- Message transmission time.

B) Failure Model

The failure model of a distributed system must be capable of handling all the failures.

- Failure handling

In a distributed system, there are various processes and many communication channels. Hence there may be failures in processes or communication channels or both.

Three common types of failures occurring in a distributed system are :

- Omission failures
- Arbitrary failures and
- Timing failures

We will discuss in short about the above failures.

- **Omission failures**

  In the processes or the communication systems fails to perform actions that it is supposed to do, then this type of failure is classified as omission failure.

There can be

i) Process omission failures and

ii) Communication omission failures.

i) Process omission failures

The process omission failure is said to occur when a process crashes. A crash is said to occur, when a process halts and does not respond for any request. Other processes running in the system may be able to detect the failure. It may also be possible, that the process sending request to failure process may wait assuming that the process is slow and be in the waiting state for quite a long time. By this way, it is difficult to detect a process crash.

If the processes can certainly detect that a process has crashed, then it is called a **fail-stop crash.** This method requires a time out to come up to a certain decision. For example,

The Fig. 1.26 shows the description of a ail-stop crash. If a process P is requesting rocess Q for a certain service, process P oes in the waiting state. If, after a certain me equal to the maximum allowable time he timeout), there is no response from Q, nen P can conclude that a fail-stop crash has ccurred for process Q.

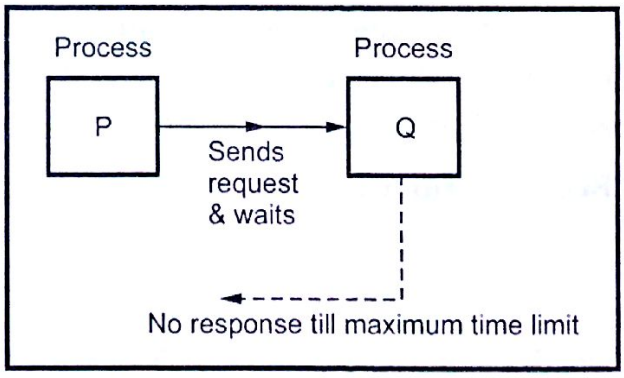

Fig. 1.26 Fail-stop crash

ii) Communication omission failures

A communication omission failure is said to occur if there is loss in the message transmission process.

For example if a process A wants to send message to process B, the transfer is rough a buffer. Process A sends the data to buffer A from where it is transmitted to uffer B of process B. There may be a loss of data in the communication channel.

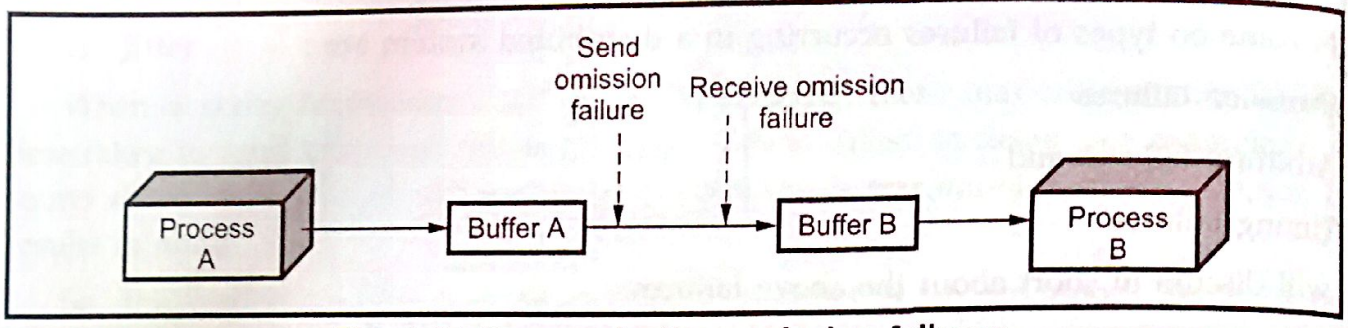

Fig. 1.27 Communication omission failures

If the data is lost from buffer A, 'Send omission failure' occurs.

Similarly, if the data is not received in buffer B, 'Receive omission failure' is said to occur.

Handling of failures is very difficult in a distributed system. The following are the features related to failures.

- **Tolerating failures**

  If a system is big enough for example, an internet, its clients are designed in such a fashion that they can tolerate failure. In a bigger environment, it is not possible to handle the failure of any component immediately. For example A server is called on the internet and it is not available immediately, the client is responded with the corresponding message. The client is not in a waiting state but completely free to do some other task.

- **Detecting failures**

It is possible to detect some failures on the system. In a proper coding, some error messages may be embedded which are executed during the respective conditions.

But in a distributed environment it is a challenge to detect the failures. For example if a remote server crashes a failure occurs, then it is difficult to detect the circumstances at that moment of time.

- **Recovery from failure**

In some cases, it is possible to recover the data. Recovery means getting the system or the process to its original state after a crash or failure.

It is a challenge to recover failures in the distributed system. The software must be designed in such a way. That after a crash, the system automatically comes back to its original state.

- **Masking failures**

Some failures after detection can be made hidden. For example, if a data is sent from one process to another and it is lost during reception, it can be retransmitted again.

- **Redundancy**

The failures could be tolerated and the services could be made available to the users by adding some redundant components in the system.

For example,

- There could always be one backup server.
- Data can be replicated on various servers. If there is fault on one server, immediately a replacement could be done.

In a distributed environment, if one of the components fail, only the work that is involving a failed component is affect. Rest remains unaffected and the working continues.

C) Security Model

We have discussed the security aspect in section 1.7.1.

The security of a distributed system can be achieved by

a) Securing processes of the distributed systems.

b) Securing channels of the distributed systems.

c) Protecting the objects against unauthorized access.

a) Security of Processes

The processes in the distributed systems communicate with each other by passing messages. There may be threats to the processes and to messages.

Threats to processes may be of the following types.

i) A message in being received by the receiver, but the sender remains unknown.

ii) A message received may not be from the intended server.

b) Security of Communication Channels

To maintain the security of communication channels, threats to communication channels must be defeated.

The threats to communication channels can be :

- Since messages travel across the network, the enemy can include or intrude the data and add some falsely information which is transmitted across the channel.
- The intruders may collect some confidential information across the channel and misuse it. For example, anybody's credit card number can be hacked through the communication channel.

c) Security of Objects Against Unauthorized Access

Many clients store their objects on the servers. These objects may be of a particular application. The clients access the objects. Accessing rights are given to each client about the objects. It is not necessary that every client can access every object. But intruders may get into the system and access the unauthorized objects by throating.

The threats to processes and communication channels can be defeated by security techniques like cryptography.

The security model to be designed must take all the above aspects into consideration.

1.9.3 Client Server Model

The architecture of client server model of a distributed system basically consists of 2 main entities.

- **Clients**

They are sending messages to the servers.

- **Servers**

They are receiving messages from clients and providing them with necessary services.

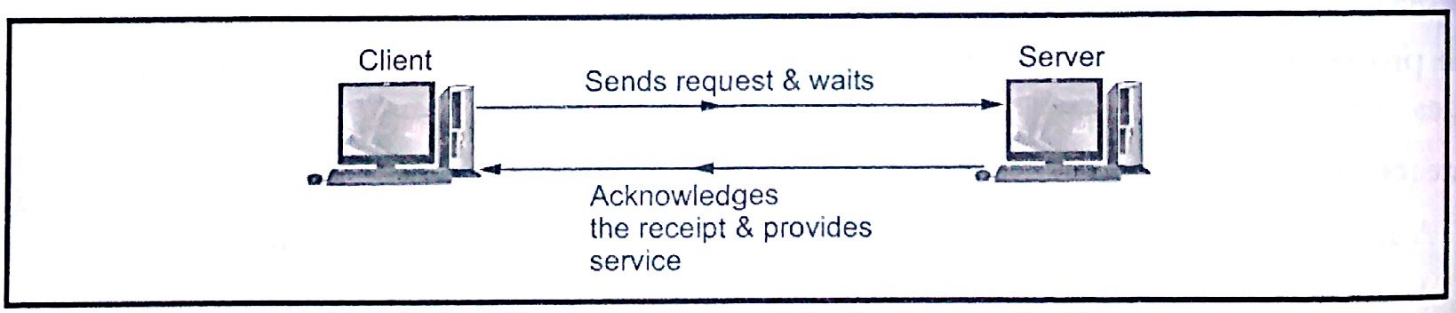

Fig. 1.28 Client server model of communication

In a distributed environment, it is sometimes very difficult to distinguish between a client and server. The server may many times act as a client and forward the messages to different file servers.

Researchers have said that there may be distinction in the various applications in the client server model. They can be done on following basis

1) The user interface level.
2) The processing level.
3) The user data level.

1) The user interface level

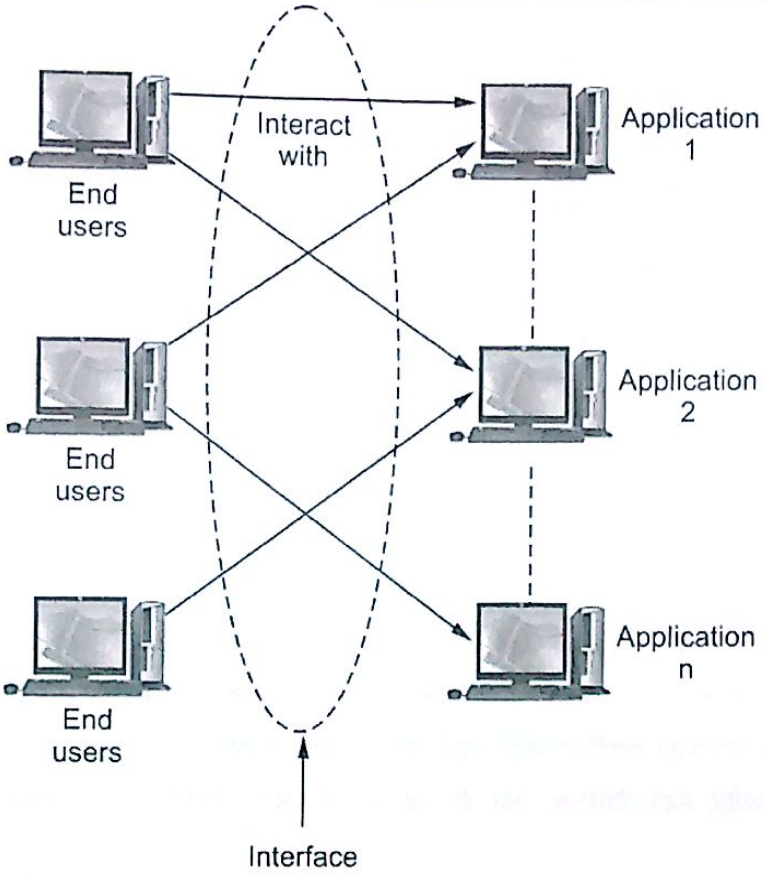

Fig. 1.29 Interface level

The clients basically have the user interface level. Here the clients interact with various applications via an interface.

The interface can be

- A simple character based screen.
- A graphical display with certain pop-up menus. e.g. Xwindows.
- A graphical display with drag and drop operations.

2) The processing level

The client server applications are basically separated at three levels.

- Interaction level

The part which works with the interaction with a user.

- Operative level

The part that mainly has the code or the operational and functional details.

- Database level

The part that deals with the database connectivity issues and handles it.

The middle part i.e. the operative level is placed in the processing level.

3) The user data level

In the client server model, the data level is the one that contains the programs which actually interacts with the database servers.

Persistence is an important property of this level. This property saves the data for further use even if no application is running.

The responsibility of the data level is to maintain consistent data.

Question Bank 1

1. *Name five reasons why to build distributed systems.*
2. *What is the difference between a client/server and a distributed system ?*
3. *Is a three-tier architecture a distributed systems ?*
4. *Why do we not build every system as a distributed system ?*
5. *What is the relationship between requirements engineering and distributed systems ?*
6. *What are the eight dimensions of transparency in distributed systems ? Where they are used ?*
7. *What are the differences between performance and scaling transparency ?*
8. *Explain eight reasons why distribution has to be considered during the design of distributed objects.*
9. *On which layer does object middleware build ?*
10. *Which layers does the middleware implement ?*
11. *Identify the three ways to resolve data heterogeneity.*
12. *What is location transparency ?*
13. *What are the differences between location transparency and access transparency ?*