Name - Kale Sayali Kantilal
Sub - Web Services
Assignment - III

Assignment - II

PAGE NO.
DATE. / /

① Explain Core Distributed Computing Technology.

→

❀ Distributed Computing :-

Distributed computing is a branch of computing in a network environment multiple computers work together on a single project.

Importance of Distributed Computing :-

① Distributed Computing helps to improve performance of large scale project combining the power of multiple machines.

② Distributed components allow different users or computers to share information.

The following are the core Distributed Computing Technology :-

[1] Client / Server :-

This is the early age technology which separate the roles of computers as client and server.

This technology is still powerful and popular amongst the network technolgies to establish communication between two or more machines.

The large and early stage

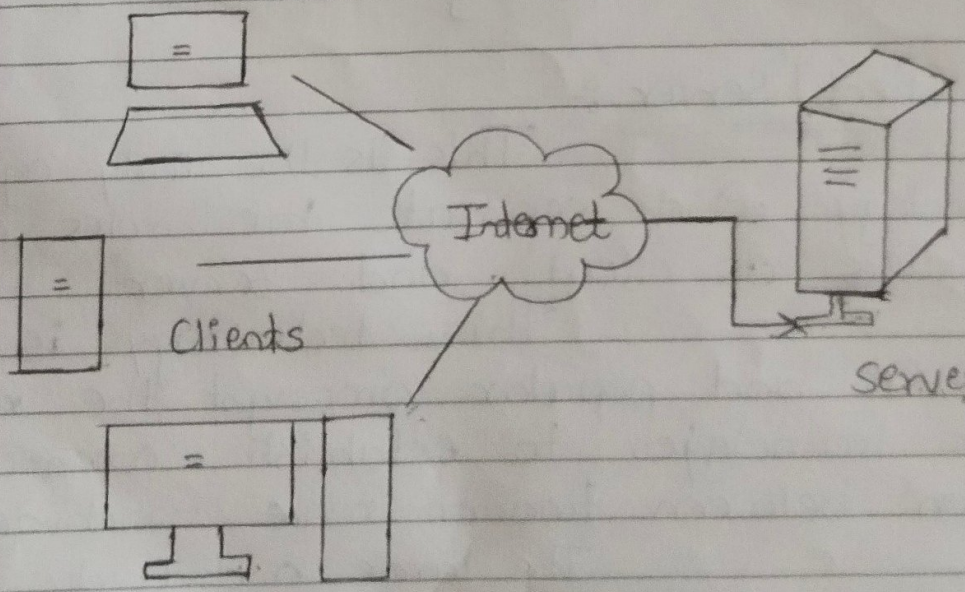
of this technology used two-tier business applications.

In this model, the first (upper) tier handles the presentation and business logic of the user application (client) and the second (lower) tier handles the application organization and its data storage (server).

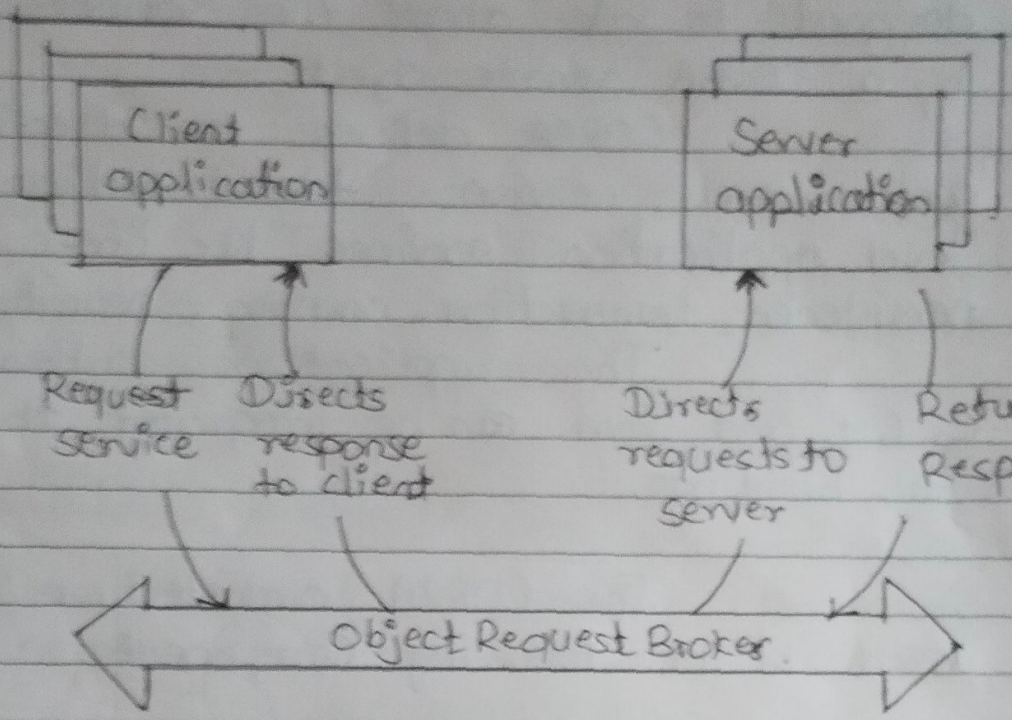
In general, the server is a database server that is mainly responsible for the organization and retrieval of data. The application client handles the user interaction through variety of graphical user interface of the application.

For Example:-

The client-server model has been widely used in Enterprise Resources Planning (ERP), billing, and Inventory application systems, banking, etc.



① CORBA (Common Object Request Broker Architecture)



⇒ It is an industry wide, open standard initiative.

⇒ It is developed by the Object Management Group (OMG).

⇒ It is developed to enable distributed computing that supports a wide range of application environments.

⇒ It provides an object-oriented solution that does not enforce any proprietary protocols or any particular programming language, Operating system, or hardware platform.

⇒ By adopting CORBA, the applications can reside and run on any hardware

platform located anywhere on the network, and can be written in any language.

Interface Definition language (IDL) is a specific interface language designed to talk about the services provided by a CORBA remote object.

CORBA defines a collection of system-level services for handling low-level application services like life-cycle, persistence, transaction, naming, security.

The applications written in C, C++ and Java can be integrated through IDL bindings.

The CORBA architecture is composed of the following components.

(i) IDL
(ii) ORB

* Advantages of CORBA :-

(i) OS and programming-language independance.
(ii) Legacy and custom application integration.
(iii) Rich distributed object infrastructure
(iv) Location transparency

③ RMI (Remote Method Invocation)

As the name specifies Java invented RMI APIs for communicating methods on any machine remotely.

This is pure Java solution for handling distributed communication.

Through RMI, object running on a client computer can invoke methods on an object present on server.

* Working of RMI :-

There are two special objects designed to establish communication between client and server.

(i) Stub object (client side)
(ii) skeleton object (server side)

(i) Stub Object :-

The stub object on the client machine builds an information block and sends this information to the server.

This block consists of

(a) An identifier of the remote object to be used.
(b) Method name which is to be invoked.
(c) Parameters to the remote JVM.

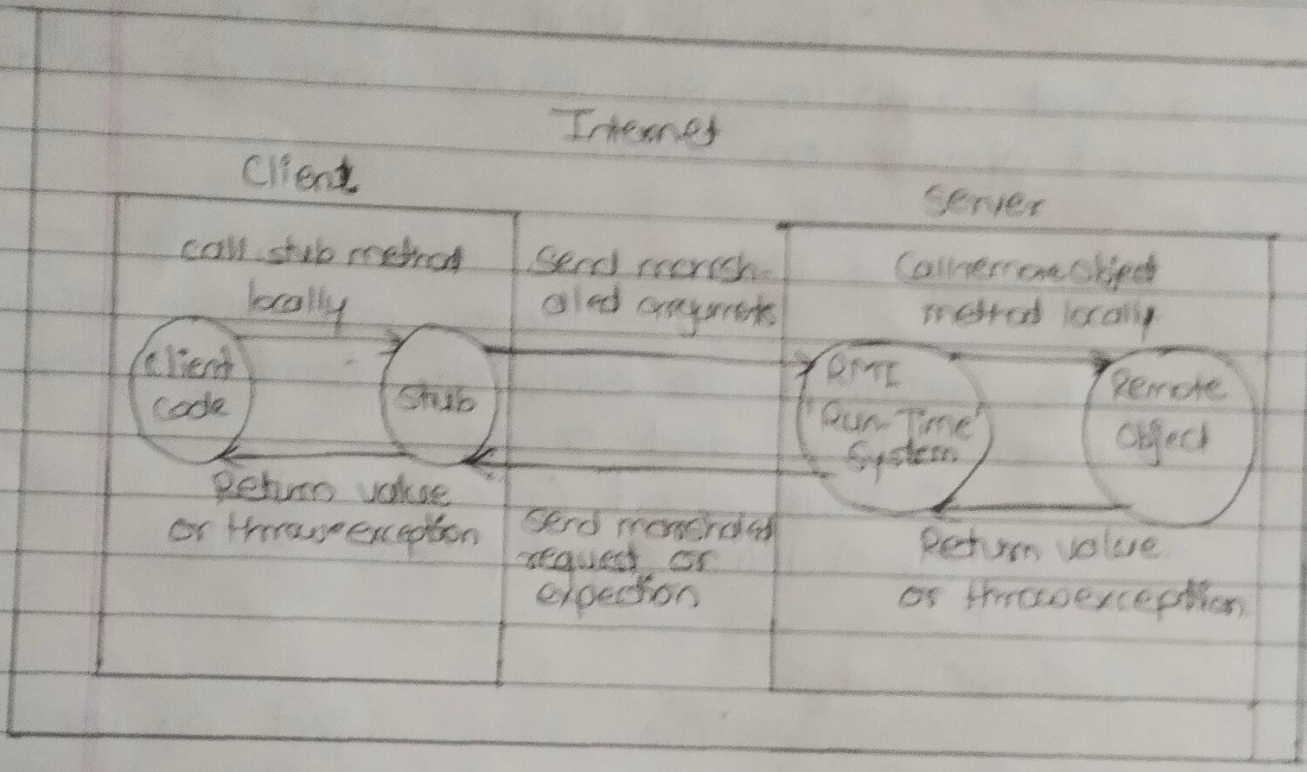
(ii) Skeleton Object :-

The skeleton object passes the request from the stub object to the

remote object. It works as;

ⓐ It calls the desired method on the real object present on the server.

ⓑ It forwards the parameters received from the stub object to the method.



* Steps to execute RMI Application in program.

① Interface Program :-

```
import java.rmi.*;
public interface addServerIntf extends Remote
{
  double add(double d1, double d2) throws
                                RemoteException;
}
```

④ Microsoft DCOM (Distributed Component Object Model)

It is a remote protocol designed by Microsoft to invoke RPCs. It contains of a set of extensions layered on the Microsoft Remote Procedure Call Extensions.

Higher-level application / Protocol
DCOM
RPC

*DCOM Protocol stack:-

Higher level applications use the DCOM client to obtain object references or make ORPC calls on the object. The DCOM client uses the Remote Procedure Call Protocol Extensions, to communicate with the object server.

The object server constitutes an object resolver service and one or more object exporters. Objects are contained in object exporters.

DCOM is language and platform independent.

DCOM is binary standard.

DCOM provides the ability to use and reuse components dynamically, without recompiling, on any platform and language neutral principle.

However, DCOM do not have any absolute way of addressing an object instance. everything is done through object interfaces.

* Marshalling:-

Marshalling helps to pass data from one COM object instance to another on a different computer.

* The steps in DCOM Communication.

(i) The client computer requests the remote computer to create an object by its CLSID or PROGID. If the client passes the APPID, the remote computer looks up the CLSID using the PROGID.

(ii) The remote machine checks the APPID and verifies the client has permissions to create the object.

(iii) DCOM Launch.exe (if an exe) or DLLHOST.exe (if a dll) will create an instance of the class the client computer requested.

(iv) The communication gets established.

(v) The Client can now access all functions in the class on the remote computer.