Assignment No:-2

Q.1) client / server Architecture.

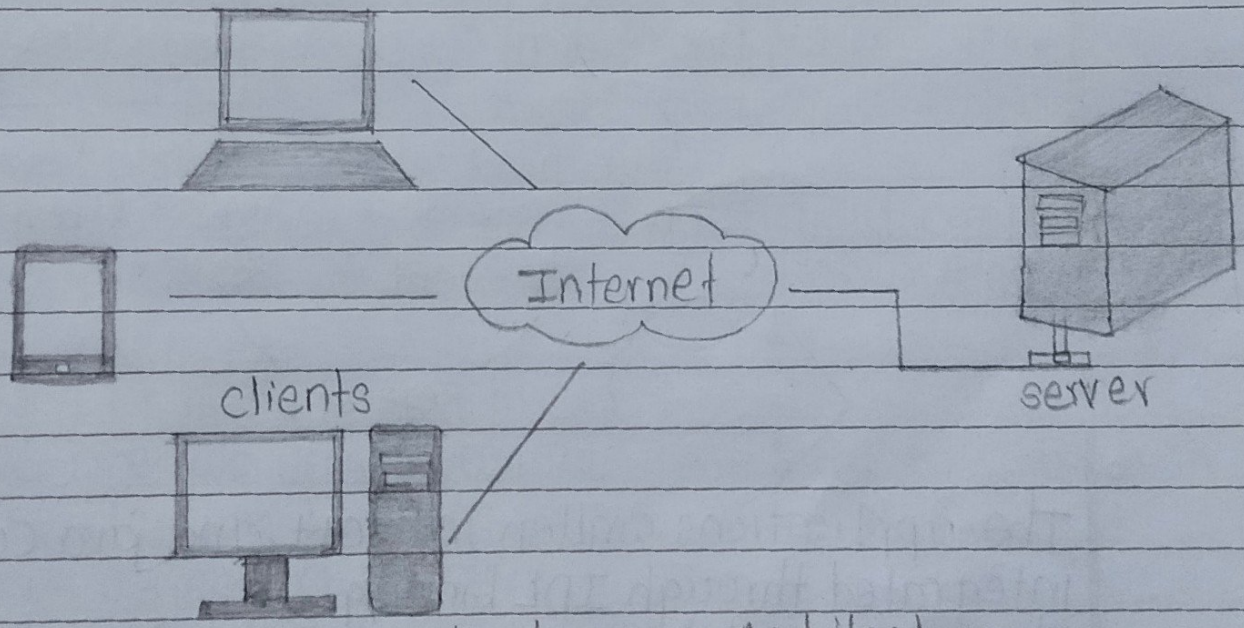
→ This is the early age technology which seperates the roles of computers as client and server. This technology is still powerful and popular amongst the network technologies to establish communication between two or more machines.

The 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 mainely responsible for the organization and retrieval of data. the application client handles the user interaction through variety of graphical user interface of the application.

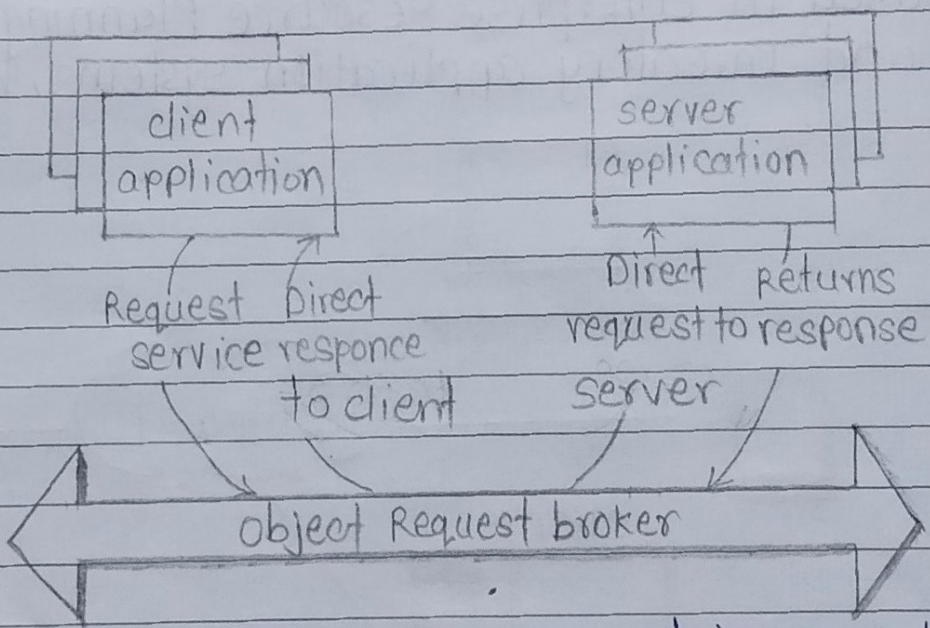
For example: the client-server model has been widely used in Enterprise Resource Planning (ERP), billing, and Inventory application systems, banking etc.



client-server Architecture.

Q.2) 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 application 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 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 binding.
The CORBA architecture is composed of the following components:

i) IDL: CORBA uses IDL contracts to specify the application boundaries and to establish interfaces with its clients. The IDL provides a machanism by which the distributed application components interfaces, inherited classes, events, attributes, and exceptions can be specified.

ii) ORB: It acts as the object bus or the bridge, providing the communication infrastructure to send and receive request/responces from the client and server. It establishes the foundation for the distributed application objects, achieving interoperability in a heterogeneous environment.

Advantages of CORBA:

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

Q.3) Java 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.
Throught 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.
The block consist 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:
a. It calls the desired method on the real object present on the server.
b. It forwards the parameters received from the stub object to the method.

steps to implement Interface:
i) Defining a remote interface.
ii) Implementing the remote interface.
iii) creating stub and skeleton objects from the implementation class using rmic (rmi compiler).
iv) start the rmiregistry.
v) create and execute the server application program.
vi) create and execute the client application program.

- Java RMI is a mechanism that allows one to invoke a method on an object that exists in another address space.
- The other address space could be on the same machine or a different one.
- The RMI machanism is basically an object-oriented RPC machanism.

Remote classes and Interfaces:

The remote interface must have the following properties
i) The interface must be public.
ii) The interface must be public.
iii) Every method in the interface must declare that it throws java.rmi.RemoteException.
other exceptions may also be thrown.

The Remote class itself has the following properties.
i) it must implement a Remote interface
ii) It should extend the java.rmi.server.UnicastRemote object class. Objects of such a class exist in the address space of the server and can be invoked remotely.
iii) It can have methods that are not in its Remote interface These can only be invoked locally.

The skeleton Interface:

- The interface skeleton is used solely by the implementation of skeletons generated by the rmic compiler A skeleton for a remote object is a server-side entity that dispatches calls to the actual remote object implementation.
- The skeleton is responsible for dispatching the call to the actual object implementation. When a skeleton receives an incoming method invocation it does no following:

a. unmarshals (reads) the parameter for the remote method.
b. invokes the method on the actual remote object implementation.
c. marshals (writes and transmits) the result (return va

or exception) to the caller.

Stub object :-

i) are surrogates that support exactly the same set of remote interfaces defined by the actual implementation of a remote object.
ii) A stub for a remote object acts as a client's local representative or proxy for the remote object.
iii) In RMI, a stub for a remote object implements the same set of remote interfaces that a remote object implements.

The Registry Interface:

- The java.rmi.registry remote interface provides methods for lookup, binding, rebinding, unbinding and listing the contents of a registry. The java.rmi.Naming class uses the registry remote interface to provide URL-based naming.
- The REGISTRY_PORT is the default port of the registry.
- The lookup method returns the remote object bound to the specified name.
- The bind method associates the name with the remote object, obj.

Steps to execute RMI Application :-

i) Compile the interface program with rmic to create stub and skeleton.
ii) Start the RMI Registry.
   a. on Linux : rmiregistry &.
   b. on Windows : start rmiregistry.
iii) Run the Remoteserver server object, i.e. server-side java program.

iv) Run the RMI client 1 program, i.e., client-side java program.

Example:
i) Interface program.
ii) Implementation program of interface.
iii) server side program.
iv) Client side program
v) To run the application.

Q.4) Microsoft DCOM(Distributed component object Model)
→ It is a remote protocol designed by microsoft to invoke RPCs. It consists of a set of extensions layered on the microsoft Remote Procedure Call Extensions.

High-level application/protocol
DCOM
RPC

(DCOM Protocol stack): Higher-level application use the DCOM client to obtain object references ormake 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 a binary standard.
DCOM provides the ability to use and reuse components dynamically, without recompiling on 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) DCOMLaunch.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 of the remote computer.