**AIM:** To build client server application

**PROBLEM STATEMENT:** Design and implement client server application using RPC/ RMI mechanism (Java)

#### PREREQUISITES: Basic knowledge of java programming

**COURSE OBJECTIVE:** Enhance programming skills

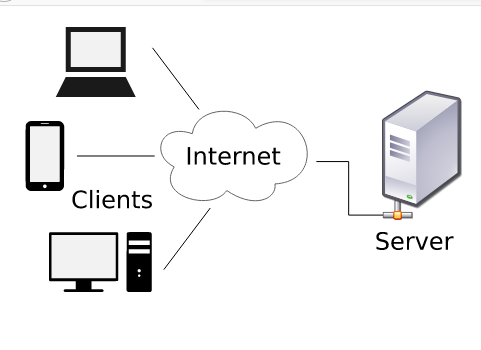
**COURSE OUTCOME:** Able to analyze the mechanism of peer to peer systems and Distributed File Systems

**THEORY:**

The client-server model describes how a server provides resources and services to one or more clients. Examples of servers include web servers, mail servers, and file servers. Each of these servers provide resources to client devices, such as desktop computers, laptops, tablets, and smartphones.

The client and the server must of course use the same communication protocol. A server is generally capable of serving multiple clients simultaneously.  
  
Characteristics of a client:

1. It is the first active (or master);
2. Sends requests to the server;
3. It expects and receives responses from the server.



Remote Method Invocation (RMI) is an API which allows an object to invoke a method on an object that exists in another address space, which could be on the same machine or on a remote machine. Through RMI, object running in a JVM present on a computer (Client side) can invoke methods on an object present in another JVM (Server side). RMI creates a public remote server object that enables client and server side communications through simple method calls on the server object.

**Working of RMI** :

The communication between client and server is handled by using two intermediate objects: Stub object (on client side) and Skeleton object (on server side).

**Stub Object**

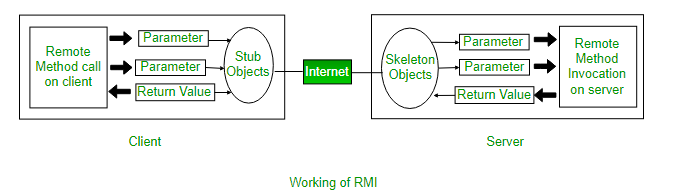
The stub object on the client machine builds an information block and sends this information to the server. The block consists of

* An identifier of the remote object to be used
* Method name which is to be invoked
* Parameters to the remote JVM

**Skeleton Object**

The skeleton object passes the request from the stub object to the remote object. It performs following tasks

* 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 implement Interface**

1. Defining a remote interface
2. Implementing the remote interface
3. Creating Stub and Skeleton objects from the implementation class using rmic (rmi complier)
4. Start the rmi registry
5. Create and execute the server application program
6. Create and execute the client application program.

**Step1: Defining the remote interface**  
The first thing to do is to create an interface which will provide the description of the methods that can be invoked by remote clients. This interface should extend the Remote interface and the method prototype within the interface should throw the RemoteException.

**Step 2: Implementing the remote interface**

The next step is to implement the remote interface. To implement the remote interface, the class should extend to UnicastRemoteObject class of java.rmi package. Also, a default constructor needs to be created to throw the java.rmi.RemoteException from its parent constructor in class.

**Step 3:  Creating Stub and Skeleton objects from the implementation class using rmic**  
The rmic tool is used to invoke the rmi compiler that creates the Stub and Skeleton objects. Its prototype srmic classname. For above program the following command need to be executed at th command prompt rmic Search Query

**STEP4:Start the rmiregistry**  
Start the registry service by issuing the following command at the command prompt start rmiregistry

**STEP 5: Create and execute the server application program**  
The next step is to create the server application program and execute it on a separate command prompt.

* The server program uses createRegistry method of LocateRegistry class to create rmiregistry within the server JVM with the port number passed as argument.
* The rebind method of Naming class is used to bind the remote object to the new name.

**Step 6: Create and execute the client application program**  
The last step is to create the client application program and execute it on a separate command prompt .The lookup method of Naming class is used to get the reference of the Stub object.

**RPC:**

(Remote Procedure Call) is a technique used for for constructing distributed, client-server based applications. It provides a communication mechanism by which a procedure stored on remote machine can be called by the client. The called procedure need not to be exist in the same address space as the calling procedure. The two processes may be on the same system, or they may be on different systems with a network connecting them. By using RPC, programmers of distributed applications avoid the details of the interface with the network. The transport independence of RPC isolates the application from the physical and logical elements of the data communications mechanism and allows the application to use a variety of transports.

• RPC makes the client/server model of computing more powerful and easier to program.

• The main components of RPC architecture are Clent, Client Stub, Server, Server Stub and RPC Runtime.

**Working:**

The execution phases of an RPC thread, as shown in Execution Phases of an RPC Thread include the following:

1. The RPC operation begins in the client machine by client program, as a client application makes an RPC to its stub.
2. The Client stub packs the parameters to be sent to server process in the form of a message.
3. The RPC Runtime at the client forwards the message to server through the communication network.
4. RPC Runtime component at server side forwards the message to server stub.
5. Server stub unpacks the parameters and hands over the parameters to the remote procedure.
6. The remote procedure gets executed, and results are sent back to the client in the reverse direction

**Applications :**

A client/server application is a piece of software that runs on a client computer and makes requests to a remote server. Many such applications are written in high-level visual programming languages where UI, forms, and most business logic reside in the client application.

**CONCLUSION:**

Thus, students understood Client Server Architecture

**FAQ’s:**

What is Client Server Architecture ?

What is meant by RPC?

What is Stub and Skeleton/?