Practical 12 : Understanding of Client and Server Architecture. Implementation of client and server mechanism using socket programming.

Software & Hardwere Requirements:
NetBeans , JRE

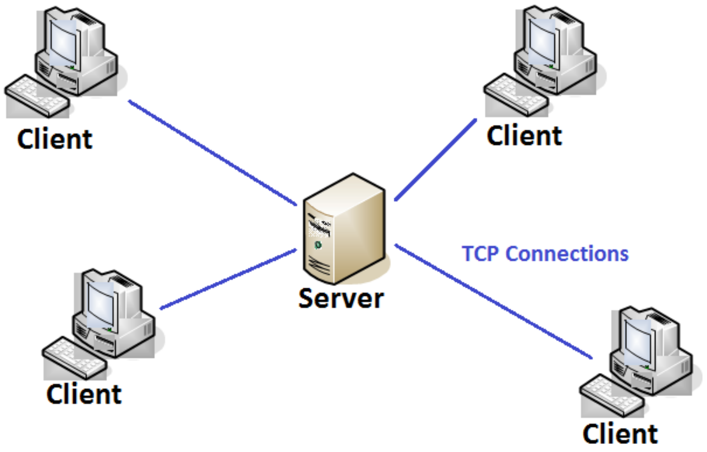
Knowledge requirements: basic knowledge of Programming Language.

Theory:

A network is composed of computers which is either a client or a server. A server is a program that is offering some service whereas a client is a program that is requesting some service. Servers are powerful computers or processes dedicated to managing disk drives (file servers), printers (print servers), or network traffic (network services) whereas clients are PCs or workstations on which users run applications. Clients rely on servers for resources, such as files, devices and even processing power.

When these programs are executed, as a result, a client and a server process are created simultaneously and these two processes communicate with each other by reading from and writing to sockets as shown in figure Client Server Limi Kalita / (IJCSIT) International Journal of Computer Science and Information Technologies, These sockets are the programming interfaces provided by the TCP and UDP protocols for stream and datagram communication respectively of the transport layer which is a part of the TCP/IP stack.

When creating a network application, the developer's main task is to write the code for both the client and server programs. The client/server application that is covered here is a proprietary client/server application. A single developer (or development team) creates both the client and server programs, and the developer has complete control over what goes in the code. But because the code does not implement a public-domain protocol, other independent developers will not be able to develop code that interoperates with the application. When developing a proprietary application, the developer must be careful not to use one of the well-known port numbers defined in the RFCs.



Server side Code

```
import java.net.*;
import java.io.*;
public class  chatserver
{
        public static void main(String args[]) throws Exception
        {
                ServerSocket ss=new ServerSocket(2000);
                Socket sk=ss.accept();
                BufferedReader
             cin=newBufferedReader(newInputStreamReader(sk.getInputSm()));
                PrintStream cout=new PrintStream(sk.getOutputStream());
                BufferedReaderstdin=newBufferedReader(InputStreamReader(System.in));
                String s;
                while (  true )
                {
                        s=cin.readLine();
                        if (s.equalsIgnoreCase("END"))
                        {
                                cout.println("BYE");
                                break;
                         }
                        System. out.print("Client : "+s+"\n");
                        System.out.print("Server : ");
                        s=stdin.readLine();
                        cout.println(s);
                }
                ss.close();
                sk.close();
                cin.close();
                cout.close();
                stdin.close();
        }
}
```

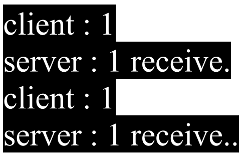
Client side code:-

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.Socket;

public class  chatclient
{
        public static void main(String args[]) throws Exception
        {
                Socket a=new Socket("localhost",2000);
                BufferedReader
sin=new BufferedReader(new InputStreamReader(a.getInputStream()));
                PrintStream sout=new PrintStream(a.getOutputStream());
                BufferedReader
 stdin=new BufferedReader(new InputStreamReader(System.in));
                String s;
                while (  true )
                {
                        System.out.print("Client : ");
                        s=stdin.readLine();
                        sout.println(s);
                        s=sin.readLine();
                        System.out.print("Server : "+s+"\n");
                        if ( s.equalsIgnoreCase("BYE") )
                          break;
                }
                 sk.close();
                 sin.close();
                 sout.close();
                stdin.close();
        }
}
```

Output

Server :



```
client : 1
server : 1 receive.
client : 1
server : 1 receive..
```

Client :

```
client : hi
server : hi,good morning.
client : bye..
server :bye,nice to meet you.
```

CONCLUSION:

In this practical we study about client and server architecture.

And we write program for server and client socket programming…