Name: Soham Desai
XIE ID: 202003021
Roll no:11
Batch: B

Experiment No: 6

Aim: Study and Implement Socket Programming using TCP

LO 4: Implement the socket programming for client server architecture.

Theory:

A socket programming interface provides the routines required for interprocess communication between applications, either on the local system or spread in a distributed, TCP/IP based network environment. Once a peer-to-peer connection is established, a socket descriptor is used to uniquely identify the connection. The socket descriptor itself is a task specific numerical value.

One end of a peer-to-peer connection of a TCP/IP based distributed network application described by a socket is uniquely defined by

- Internet address
  for example 127.0.0.1 (in an IPv4 network) or FF01::101 (in an IPv6 network).

- Communication protocol
  - User Datagram Protocol (UDP)
  - Transmission Control Protocol (TCP)
- Port
  A numerical value, identifying an application. We distinguish between
  - "well known" ports, for example port 23 for Telnet
  - user defined ports

Socket applications were usually C or C++ applications using a variation of the socket API originally defined by the Berkeley Software Distribution (BSD). The JAVA language also provides a socket API. JAVA based Client/Server applications exploit those socket services.

Socket programming interfaces have been standardized for ease of portability by The Open Group for example.

Besides TCP/IP based sockets, UNIX systems provide socket interfaces for interprocess communication (IPC) within the local UNIX host itself. Those UNIX sockets use the local file system for interprocess communication.

z/VSE provides TCP/IP based socket services. They can be used for IPC too, although they are primarily aimed for network communication only.

Name: Soham Desai
XIE ID: 202003021
Roll no:11
Batch: B

Code :

1. **Client :**

```
import java.net.*;
import java.io.*;
public class Client {
   private Socket socket = null;
   private DataInputStream input = null;
   private DataOutputStream out = null;
   public Client(String address, int port) {
      try {
         socket = new Socket(address, port);
         System.out.println("Connected");
         input = new DataInputStream(System.in);
         out = new DataOutputStream(socket.getOutputStream());
      } catch (IOException u) {
         System.out.println(u);
      }
      String line = "";
      while (!line.equals("Over")) {
         try {
            line = input.readLine();
            out.writeUTF(line);
         } catch (IOException i) {
            System.out.println(i);
         }
      }
      try {
         input.close();
         out.close();
         socket.close();
      } catch (IOException i) {
         System.out.println(i);
      }
   }
   public static void main(String[] args) {
      Client client = new Client("127.0.0.1", 5000);
   }
    }
```

Name: Soham Desai
XIE ID: 202003021
Roll no:11
Batch: B

2. Server:

```
import java.net.*;
import java.io.*;
public class Server {
   private Socket socket = null;
   private ServerSocket server = null;
   private DataInputStream in = null;
   public Server(int port) {
      try {
         server = new ServerSocket(port);
         System.out.println("Server started");
         System.out.println("Waiting for a client ...");
         socket = server.accept();
         System.out.println("Client accepted");
         in = new DataInputStream(
               new BufferedInputStream(socket.getInputStream()));
         String line = "";
         while (!line.equals("Over")) {
            try {
               line = in.readUTF();
               System.out.println(line);
            } catch (IOException i) {
               System.out.println(i);
            }
         }
         System.out.println("Closing connection");
         socket.close();
         in.close();
      } catch (IOException i) {
         System.out.println(i);
      }
   }
   public static void main(String[] args) {
      Server server = new Server(5000);
   }
}
```

Name: Soham Desai
XIE ID: 202003021
Roll no:11
Batch: B

Output:

1. **Client :**

```
Connected
hello world
This is socket programming
thank you for using!!
Over
```

2. **Server:**

```
Server started
Waiting for a client ...
Client accepted
hello world
This is socket programming
thank you for using!!
Over
Closing connection
```

Conclusion: From this experiment we have learned to do socket programming using TCP in Java and also how it actually works.