Atharva Yadav
Roll No.127
Batch: s23

NETWORK LAB ASSG NO. 9

AIM: SOCKET PROGRAMMING USING UDP PROTOCOL

Theory: Datagram Sockets are Java's mechanism for network communication via UDP instead of TCP. Java provides
DatagramSocket to communicate over UDP instead of TCP. It is also built on top of IP. DatagramSockets can be used to both send and receive packets over the Internet. One of the examples where UDP is preferred over TCP is the live coverage of TV channels. In this aspect, we want to transmit as many frames to live audience as possible not worrying about the loss of one or two frames. TCP being a reliable protocol add its own overhead while transmission. Another example where UDP is preferred is online multiplayer gaming. In games like counter-strike or call of duty, it is not necessary to relay all the information but the most important ones. It should also be noted that most of the applications in real life uses careful blend of both UDP and TCP; transmitting the critical data over TCP and rest of the data via UDP.

Java Program for Client application

CODE:

```
// UDP Client import java.io.*; import java.net.*; public
class UDPClient {  public static void main(String[] args) {
DatagramSocket clientSocket = null;  try {
 // Create a DatagramSocket  clientSocket = new
DatagramSocket();
 InetAddress IPAddress = InetAddress.getByName("localhost");  byte[] sendData =
new byte[1024];  byte[] receiveData = new byte[1024];  while (true) {
 // Read input from user
 BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
 System.out.print("Enter your message: ");  String message =
inFromUser.readLine();  sendData = message.getBytes();
 // Send packet to server
 DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 9876);
clientSocket.send(sendPacket);  // Receive response from server
 DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
 clientSocket.receive(receivePacket);
 String serverResponse = new String(receivePacket.getData(), 0, receivePacket.getLength());
 System.out.println("Received from server: " + serverResponse);  }
 } catch (IOException e) {
 e.printStackTrace();  } finally {  if
(clientSocket != null) {
clientSocket.close();
 }
 }
 }
}
```

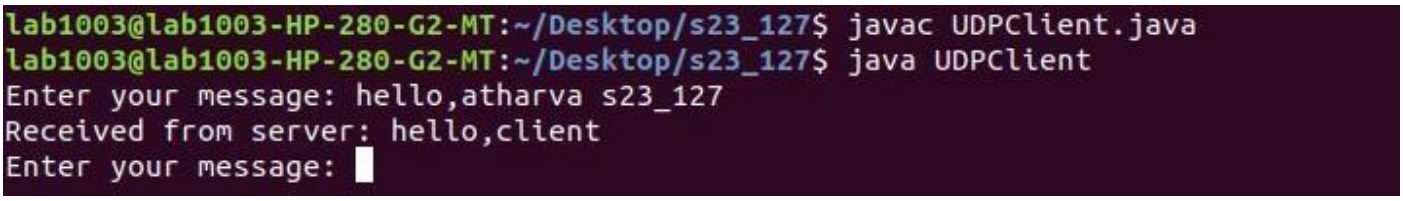
Java Program for Server application

CODE:

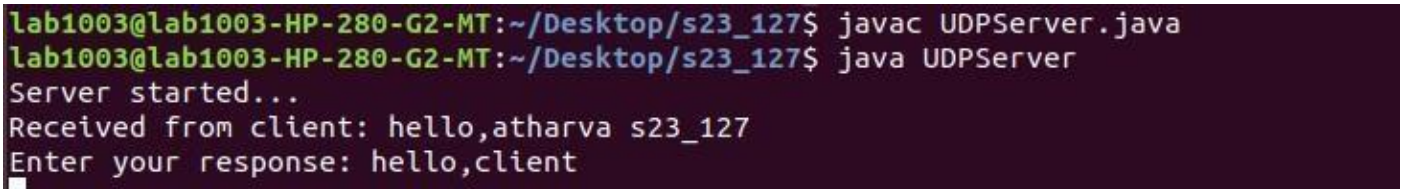
```
// UDP Server import java.io.*; import java.net.*; public

class UDPServer {  public static void main(String[] args) {

DatagramSocket serverSocket = null;  try {

 // Create a DatagramSocket, bind it to port 9876  serverSocket = new

DatagramSocket(9876);  byte[] receiveData = new byte[1024];  byte[]

sendData = new byte[1024];  System.out.println("Server started...");

while (true) {

 // Receive packet from client

 DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
serverSocket.receive(receivePacket);

 String clientMessage = new String(receivePacket.getData(), 0, receivePacket.getLength());

 System.out.println("Received from client: " + clientMessage);

 // Get client's address and port

 InetAddress clientAddress = receivePacket.getAddress();  int clientPort =

receivePacket.getPort();


 // Send response back to the client

 BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));

 System.out.print("Enter your response: ");  String serverResponse =

inFromUser.readLine();  sendData = serverResponse.getBytes();

 DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, clientAddress, clientPort);

serverSocket.send(sendPacket);

 }

 } catch (IOException e) {
```

```
 e.printStackTrace();  } finally {  if

(serverSocket != null) {

serverSocket.close();

}

 }

 }

}
```

OUTPUT: Client:



Server:



CONCLUSION : ideal for real-time applications where immediate response is crucial, even if sending data packets without error checking or guaranteed delivery, making it into the UDP (User Datagram Protocol) protocol. UDP prioritizes speed by likely explored creating sockets, the endpoints for communication, and delved skills to build applications that prioritize speed over guaranteed delivery. We The network assignment on socket programming using UDP equips us with the some data might be lost

Based On LO 5 : To observe and study the traffic flow and the contents of protocol frames.