**G. H. RAISONI COLLEGE OF ENGG., NAGPUR**

**(An Autonomous Institute)**

**Department of Computer Science & Engg.**



**Date: 14-09-2021**

**Practical Subject: DCN Pr A2**

**Session: 2021-22**

**Student Details:**

| **Roll Number** | 44 |
| --- | --- |
| **Name** | Anand Suralkar |
| **Semester** | 9th |
| **Section** | A |
| **Batch** | CSE |

**Practical Details: Practical Number-5;**

| Practical Aim | Implement client-server communication using TCP and UDP protocol |
| --- | --- |
| Theory & Syntax | The client-server architecture is a way of building computer systems. It consists in splitting the system into  two categories of programs: clients and severs.  A client is a program or a process which connects to another program (the server) and lets it carry out a  specific task. In particular it might require supplying some data from the server.  A server is a program or a process which provides services for clients. For example, it might supply some  data or a result of processing data to clients.  A communication (between a server and a client) requires establishing specific rules (a language in which  programs talk to each other). These rules are defined by the protocols.  A protocol is a language of communication among programs; in particular between a client and a server.  A fundamental notion for exchanging information through the network is a socket.  A socket is one end-point of a two-way communication link between two programs running on the network  Usually, communication using sockets is based on the TCP or the UDP protocol.  TCP (Transmission Control Protocol) is a connection-based protocol that provides a reliable flow of data  between two computers. It guarantees that the sent data are not lost and arrive in the proper order to the  receiver.  UDP (User Datagram Protocol) is a protocol that sends independent packets of data, called datagrams,  from one computer to another with no guarantees about arrival. UDP is not connection-based like TCP.  Datagrams may arrive to the receiver in an arbitrary order, some of them might be lost.  Identification of hosts in the network is made using IP (Internet Protocol) addresses. An IP address is a 32-  bit number (or a 128-bit for IPv6) typically represented using the dot notation as a sequence of four (or  eight) numbers separated with dots (e.g. 192.33.71.12). A human-friendly identification of hosts involves  DNS (Domain Name Service) which gives them names like: &quot;google.com&quot;.  Such an identification is not enough for inter-process communication, as there may be several processes  running on a host. TCP and UDP protocols use ports for delivering data to a receiver. A port is identified by  a 16-bit number.  In connection-based communication such as TCP, a server application binds a socket to a specific port  number. This has the effect of registering the server with the operating system to receive all data destined  for that port. A client can then talk with the server at the server&#39;s port.  In datagram-based communication such as UDP, the datagram packet contains the port number of its  destination. Further, the packets are routed to the appropriate application.  In TCP (connection-based) socket: A typical interaction between a client and a server is based on the  following model.  1. The server binds a socket to a specific port number and starts waiting for clients.  2. A client initializes a connection with the service specified by its host name and port number.  3. The server accepts the connection made by the client and creates a new socket for communicating  with it.  4. From the point of view of the client it is (usually) the socket which was used to initialize the  connection.  So we use two types of sockets in Java:  1. server sockets (the class ServerSocket) - used by servers,  2. client sockets (the class Socket) - used by clients; servers use it too for communication with clients.  Programming a client communicating with a server is quite simple in Java. It involves the following steps:  1. creating a socket,  2. obtaining streams associated with the socket,  3. sending requests to the server by means of the output stream,  4. receiving server responses from the input stream,  5. closing streams,  6. Closing the socket.  The fundamental issue related to the client-server communication is the protocol. Each server may use its  own protocol. However, there exist Internet services which use the well-known, standard protocols. Here  are some of them:  HTTP (Hypertext Transfer Protocol) - used for accessing web pages  FTP (File Transfer Protocol) - used for transferring files  SMTP (Simple Mail Transfer Protocol) - used for sending mail  POP3 (Post Office Protocol) - getting mail from a server  IMAP (Internet Message Access Protocol) - getting mail from a server  TELNET - access to remote hosts  TIME - get the current time from a host  DAYTIME - get data and time  NTP (Network Time Protocol) - for getting the exact time  NNTP (Network News Transfer Protocol) - newsgroups support  IRC - Internet Relay Chat - internet chat  ECHO - the echo server  DICT - the dictionary service |
| Program | import java.net.\*;  import java.io.\*;  class DateTime {  public static void main(String[] args) {  String host = null;  int port = 13;  try {  host = args[0]; // hostname  // create the socket  Socket socket = new Socket(host, port);  // Get the socket&#39;s input stream and  // decorate it with a decoder and a buffer  BufferedReader br = new BufferedReader(  new InputStreamReader(  socket.getInputStream()  )  );  // Get the server&#39;s response (date and time)  String line;  while ((line = br.readLine()) != null) {  System.out.println(line);  }  // Close streams and the socket  br.close();  socket.close();  } catch (UnknownHostException exc) {  System.out.println(&quot;Unknown host: &quot; + host);  } catch (Exception exc) {  exc.printStackTrace();  }  }  } |
| Output |  |
| Conclusion | Implemented client-server communication using TCP and UDP protocol in java in linux |