6. Applications using TCP Sockets like
a. File transfer
b. Remote command execution
c. Chat
d. Concurrent server

A) APPLICATIONS USING TCP SOCKETS (FILE TRANSFER)

Objective:

To write a java program for file transfer using TCP Sockets.

Learning outcome:

After the completion of this experiment, student will be able to

- Develop a file transfer application by using TCP Socket
- Understand the reliability of TCP protocol.
- Understand how to transfer of computer files between a client and server on a computer network.
- Develop a client-server application by using TCP

Problem Statement:

To transfer computer files between a client and server on a computer network using TCP Sockets.

Concept:

Our aim is to send a file from the client machine to the server machine using TCP Communication.

System and Software tools required:

Package Required : Java Compiler
Operating System : UBUNTU
Minimum Requirement : Pentium III or Pentium IV with 2GB RAM 40 GB hard disk

Algorithm

Server

Step1: Import java packages and create class file server.
Step2: Create a new server socket and bind it to the port.
Step3: Accept the client connection
Step4: Get the file name and stored into the BufferedReader.
Step5: Create a new object class file and realine.
Step6: If file is exists then FileReader read the content until EOF is reached.
Step7: Stop the program.

Client

Step1: Import java packages and create class file server.
Step2: Create a new server socket and bind it to the port.
Step3: Now connection is established.
Step4: The object of a BufferReader class is used for storing data content which has been retrieved from socket object.
Step5: The content of file is displayed in the client window and the connection is closed.
Step6: Stop the program.

Execution of program:

Compiling the program: javac file name.java

Executing the program : java file name
Sample Coding:

Se.java

```
{/*… Register service on port 15123…*/
ServerSocketserverSocket = new ServerSocket(15123);
/*… Wait and accept a connection…*/
Socket socket = serverSocket.accept();
System.out.println("Accepted connection : " + socket);
/*… enter the source file name which is to be transferred to client…*/
File transferFile = new File ("sunithanandhinifiletransfer.doc");
byte [] bytearray = new byte [(int)transferFile.length()];
/*…FileInputStream is meant for reading streams of raw bytes such as image data…*/
FileInputStream fin = new FileInputStream(transferFile);
BufferedInputStream bin = new
BufferedInputStream(fin);bin.read(bytearray,0,bytearray.length);
/*… Get a communication stream associated with the socket…*/
OutputStreamos = socket.getOutputStream();
System.out.println("Sending Files...");
os.write(bytearray,0,bytearray.length);
os.flush();socket.close();
```

Cl.java

```
/*… Open your connection to a server, at port 15123…*/
Socket socket = new Socket("127.0.0.1",15123);
byte [] bytearray = new byte [filesize];
InputStream is = socket.getInputStream();
/*… enter the destinationfile name which is to be transferred …*/
FileOutputStream fos = new FileOutputStream("kalpanasonikafiletransfer.doc");
BufferedOutputStream bos = new BufferedOutputStream(fos);
bytesRead = is.read(bytearray,0,bytearray.length);
currentTot = bytesRead;
do { bytesRead = is.read(bytearray, currentTot, (bytearray.length-currentTot));
if(bytesRead>= 0) currentTot += bytesRead;}
while(bytesRead> -1);
bos.write(bytearray, 0 , currentTot);
bos.flush(); bos.close(); socket.close();
 }
}
```

B) APPLICATIONS USING TCP SOCKETS (REMOTE COMMAND EXECUTION)

Objective:
To create a program for executing the remote command using TCP Socket.
Learing outcomes:

After the completion of this experiment, student will be able to

 Develop a program to execute remote the application by using TCP Socket.

 Understand the reliability of TCP protocol.

 Understand how to execute remote commands in computer files between a client and server on a
computer network.

Problem Statement:

To create a program for executing a command in the system from a remote point.

System and Software tools required:

Package Required : Java Compiler
Operating System : UBUNTU
Minimum Requirement : Pentium III or Pentium IV with 2GB RAM 40 GB hard disk

Algorithm:

Step1: Start the program.
Step2: Create a server socket at the server side.
Step3: Create a socket at the client side and the connection is set to accept by the server socket using the
accept () method.
Step4: In the client side the remote command to be executed is given as input.
Step5: The command is obtained using the readLine () method of Buffer Reader.
Step6: Get the runtime object of the runtime class using getruntime ().
Step7: Execute the command using the exec () method of runtime.

Execution of program:

Compiling the program: javac file name.java
Executing the program : java file name

Sample Coding:

RemoteServer.java

```
/*…getInputStream()-This method take the permission to write the data from client program to
server program and server program to client program…*/
BufferedReader Buf =new BufferedReader(new InputStreamReader(System.in));
System.out.print(" Enter the Port Address : " );
Port=Integer.parseInt(Buf.readLine());
/* …Socket class is having a constructor through this Client program can request to server to get
connection…*/
ServerSocket ss=new ServerSocket(Port);
System.out.println(" Server is Ready To Receive a Command. ");
System.out.println(" Waiting ..... ");
/*… Wait and accept a connection…*/
Socket s=ss.accept();
if(s.isConnected()==true)
System.out.println(" Client Socket is Connected Succecfully. ");
InputStream in=s.getInputStream();
/* …getOutputStream()-This method is used to take the permission to read data from client
system by the server or from the server system by the client…*/
```

```
OutputStream ou=s.getOutputStream();
BufferedReader buf=new BufferedReader(new InputStreamReader(in));
String cmd=buf.readLine();
PrintWriter pr=new PrintWriter(ou);
pr.println(cmd);
/*... Runtime.getRuntime() method returns the runtime object associated with the
current Java
application ...*/
Runtime H=Runtime.getRuntime();
Process P=H.exec(cmd);
```

RemoteClient.java

```
BufferedReader Buf =new BufferedReader(new InputStreamReader(System.in));
System.out.print(" Enter the Port Address : " );
/*...Integer.parseInt() java method is used primarily in parsing a String method
argument into
an Integer object. The Integer object is a wrapper class for the int primitive data type
...*/
Port=Integer.parseInt(Buf.readLine());
Socket s=new Socket("localhost",Port);
if(s.isConnected()==true)
System.out.println(" Server Socket is Connected Succecfully. ");
InputStream in=s.getInputStream();
OutputStreamou=s.getOutputStream();
BufferedReader buf=new BufferedReader(new InputStreamReader(System.in));
BufferedReader buf1=new BufferedReader(new InputStreamReader(in));
PrintWriterpr=new PrintWriter(ou);
System.out.print(" Enter the Command to be Executed : " );
/*... readline() method read a line of text ...*/
pr.println(buf.readLine());
pr.flush();
String str=buf1.readLine();
```

c) **APPLICATIONS USING TCP SOCKETS (CHAT)**

Objective:

To implement a CHAT application, where the Client establishes a connection with the Server.

The Client and Server can send as well as receive messages at the same time. Both the Client and Server
exchange messages.

Learning Outcomes:

After the completion of this experiment, student will be able to

- Develop a chat application by using TCP Socket
- Understand the reliability of TCP protocol.
- Understand how the reliable in order delivery is obtained using this protocol
- Develop a client-server application by using TCP

Problem Statement:

 A server program to establish the socket connection with the client for performing chat.
 A client program which on establishing a connection with the server for performing chat.

Concept:

1. It uses TCP socket communication .We have a server as well as a client.
2. Both can be run in the same machine or different machines. If both are running in the machine, the
address to be given at the client side is local host address.
3. If both are running in different machines, then in the client side we need to specify the ip address of
machine in which server application is running.

System and Software tools required:

Package Required : Java Compiler
Operating System : UBUNTU
Minimum Requirement : Pentium III or Pentium IV with 2GB RAM 40 GB hard disk

Algorithm:

Server

Step1: Start the program and create server and client sockets.
Step2: Use input streams to get the message from user.
Step3: Use output streams to send message to the client.
Step4: Wait for client to display this message and write a new one to be displayed by the server.
Step5: Display message given at client using input streams read from socket.
Step6: Stop the program.

Client

Step1: Start the program and create a client socket that connects to the required host and port.
Step2: Use input streams read message given by server and print it.
Step3: Use input streams; get the message from user to be given to the server.
Step4: Use output streams to write message to the server.
Step5: Stop the program.

Execution of program:

Compiling the program: javac file name.java
Executing the program : java file name

Sample Coding:

GossipServer.java

```
ServerSocketsersock = new ServerSocket(3000);
System.out.println("Server ready for chatting");
Socket sock = sersock.accept( );
/*…reading from keyboard (keyRead object)…*/
BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));
/*…sending to client (pwrite object)…*/
OutputStreamostream = sock.getOutputStream();
PrintWriterpwrite = new PrintWriter(ostream, true);
/*… receiving from server ( receiveRead object)…*/
InputStreamistream = sock.getInputStream();
BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));
String receiveMessage, sendMessage;
while(true)
```

{if((receiveMessage = receiveRead.readLine()) != null)
{System.out.println(receiveMessage); }

GossipClient.java

Socket sock = new Socket("127.0.0.1", 3000);

/*…reading from keyboard (keyRead object)…*/

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

/*…sending to client (pwrite object)…*/

OutputStreamostream = sock.getOutputStream();

PrintWriterpwrite = new PrintWriter(ostream, true);

/*… receiving from server (receiveRead object)…*/

InputStreamistream = sock.getInputStream();

BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));

System.out.println("Start the chitchat, type and press Enter key");

String receiveMessage, sendMessage;

while(true)

{sendMessage = keyRead.readLine(); **/*... keyboard reading ...*/**

pwrite.println(sendMessage); **/*... sending to server...*/**

pwrite.flush(); **/*... flush the data...*/**

if((receiveMessage = receiveRead.readLine()) != null) **/*...receive from server...*/**

{System.out.println(receiveMessage);} **/*... displaying at DOS prompt**...*/

D) **APPLICATIONS USING TCP SOCKETS (CONCURRENT SERVER)**

Objective:

To write a program for concurrent communication of client to the server using TCP Socket.

Learning Outcomes:

After the completion of this experiment, student will be able to

- Understand the reliability of TCP protocol.
- Understand how a a server can handle multiple clients at the same time in parallel
- Develop a client-server application by using TCP

Problem Statement:

To have a reliable concurrent communication between client and the server.

Concept:

Concurrent Server: The server can be iterative, i.e. it iterates through each client and serves one request

at a time. Alternatively, a server can handle multiple clients at the same time in parallel, and this type of

a server is called a concurrent server.

System and Software tools required:

Package Required : Java Compiler

Operating System : UBUNTU

Minimum Requirement : Pentium III or Pentium IV with 2GB RAM 40 GB hard disk

Algorithm:

Server

Step 1: Create a socket and bind to the address. Leave socket unconnected.

Step 2 : Leave socket in passive mode, making it ready for use by a server.

Step 3: Repeatedly call accept to receive the next request from a client to handle the response with the
through socket.

Client

Step 1: Begin with a connection passed from the server (i.e., a socket for the connection).
Step 2: Use input streams; get the message from user to be given to the server.
Step 3: Use input streams read message given by server and print it.
Step 4: Use output streams to write message to the server.
Step 5: Close the connection and exit, i.e., slave terminates after handling all requests from one client.

Execution of program:

Compiling the program: javac file name.java
Executing the program : java file name
Sample Coding:

ConServer.java

```
/*… Register service on port 8020…*/
ServerSocketss=new ServerSocket(8500);
System.out.println("Waiting for client...");
while(true)
/*... ServerSocket in order to listen for and accept connections from clients...*/
{Socket s=ss.accept();
/*…getInputStream()-This method take the permission to write the data from client program to
server program and server program to client program…*/
BufferedReader br=new BufferedReader(new InputStreamReader(s.getInputStream()));
cli_name=br.readLine();
System.out.println("\nCLIENT NAME: "+cli_name);
no=Integer.parseInt(br.readLine());
sq=no*no;
PrintWriter pw=new PrintWriter(s.getOutputStream(),true);
pw.println(sq);
System.out.println("OUTPUT - The square of "+no+" is "+sq);}}}
```

ConClient1.java

```
{Socket s=new Socket("localhost",8500);
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
/*...Integer.parseInt() java method is used primarily in parsing a String method argument into
an Integer object. The Integer object is a wrapper class for the int primitive data type ...*/
int num=Integer.parseInt(br.readLine());
/* …getOutputStream()-This method is used to take the permission to read data from client
system by the server or from the server system by the client…*/
PrintWriterpw=new PrintWriter(s.getOutputStream(),true);
pw.println("Client 1");
pw.println(num);
```

```
BufferedReader br1=new BufferedReader(new InputStreamReader(s.getInputStream()));
intsqu=Integer.parseInt(br1.readLine());
System.out.println("Square of "+num+" is "+squ+"\n");
```

ConClient2.java

```
Socket s=new Socket("localhost",8500);
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("\nCLIENT 2:\nEnter the number to find square: ");
intnum=Integer.parseInt(br.readLine());
PrintWriter pw=new PrintWriter(s.getOutputStream(),true);
pw.println("Client 2");
pw.println(num);
BufferedReader br1=new BufferedReader(new InputStreamReader(s.getInputStream()));
intsqu=Integer.parseInt(br1.readLine());
System.out.println("Square of "+num+" is "+squ+"\n");
s.close();
```