NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCE

Computer Network Lab (CL-307)

Lab Session 03

Awais Ahmed || Faizan Yousuf || Munim Ali
awais.ahmed@nu.edu.pk || faizan.yousuf@nu.edu.pk || munim.ali@nu.edu.pk

OBJECTIVE

1. Study of Socket Programming
2. Finding the IP Address
3. TCP-One Way Communication
4. TCP-Two Way Communication
5. UDP-One Way Communication
6. UDP-Two Way Communication
7. Ping Command
8. File transfer using TCP
9. Broadcasting

FINDING THE IP ADDRESS

OBJECTIVE: To write a java program to find the IP address of the system.

ALGORITHM:

1. Start
2. Declare a variable 'ip' as a static InetAddress.
3. Using the function getLocalHost() to find the address of the system.
4. Get the name of the system by using the getHostName() function.
5. By specifying the system name,find out the IP address of the system using the function getByName().
6. Stop.

FINDING IP ADDRESS

SOURCE CODE:

```
import java.io.*;
import java.net.*;
class address
{
public InetAddressip;
public static void main(String args[])throws UnknownHostException
{
InetAddress ip=InetAddress.getLocalHost();
System.out.println("\n IP address is :"+ip);
String s1=ip.getHostName();
```

```
System.out.println("system number is:"+s1);
InetAddress ip1=InetAddress.getByName("system 10");
System.out.println("\n name of other system is :"+ip1);
}
}
```

TCP-ONE WAY COMMUNICATION

OBJECTIVE: To write a java program to implement one way communication using TCP(Transmission Control Protocol).

ALGORITHM:

SERVER:

1. Start the program.
2. Import .net package and other packages.
3. Declare objects for ServerSocket, Socket and PrintStream to transfer data.
4. Using PrintStream transfer the data in OutputStream via a port.
5. Run an loop to send data from server until an “end or exit” string is transferred.
6. If “end or exit” is encountered, close the socket and exit the program.

CLIENT:

1. Start the program.
2. Import .nrt package and other packages.
3. Declare objecta for Socket and DataInputStream to receive server data.
4. Run an loop to receive data from server and store it in a string using DataInputStream.
5. Prunt the string to display the server data and exit when an “end or exit” Message is encountered.

SOURCE CODE:

CLIENT:

```
import java.io.*;
import java.net.*;
class client
{
public static void main(String args[])throws IOException
{
Socket s=new Socket("localHost",8000);
DataInputStream in=new DataInputStream(s.getInputStream());
while(true)
{
String str=in.readLine();
System.out.println("Message Received:"+str);
if(str.equals("end"))
```

```
{
s.close();
break;
}
}
}
}
```

SOURCE CODE:

SERVER:

```
import java.io.*;
import java.net.*;
class server
{
public static void main(String a[])throws IOException
{
ServerSocket ss=new ServerSocket(8000);
Socket s=ss.accept();
DataInputStream in=new DataInputStream(System.in);
PrintStream dos=new PrintStream(s.getOutputStream());
while(true)
{
System.out.println("enter message to send:");
String str=in.readLine();
dos.println(str);
if(str.equals("end"))
{
ss.close();
break;
}
}
}
}
```

TCP-TWO WAY COMMUNICATION

OBJECTIVE: To write a java program to implement two way communication using TCP(Transmission Control Protocol).

ALGORITHM:

SERVER:

1. Start the Program.
2. Import .net package and other packages.
3. declare objects for Server4Scoket,SAcoket and PrintStream to transfer server data.
4. Declare objects for Socket and DataInputStream to receive client data.
5. Using PrintStream transfer the data in OutputStream via a port.
6. Run an loop to send data from server until an "end or exit" String is transferred.
7. Using the same loop receive data from server and store it in a String Using DataInputStream.
8. print the String to display the server data and exit when an "end or exit" Message is encountered.

CLIENT:

1.Start the program.
2.Import .net package and other packages.
3.Declare objects for Socket, Socket and PrintStream to transfer54 the client data.
4.Declare objects for Socket and DataInputStream to receive server the data.
5.Using PrintStream transfer the data in OutputStream via a port.
6.Run an loop to send data from server until an "end or exit" string is transferred.
7.Using the same loop receive data from server and store it in a String using DataInputStream.
8.Print the string to display the server data and exit when an "end or exit" Message is encountered.

UDP-ONE WAY COMMUNICATION

OBJECTIVE: To write a program in java to perform one way message transfer using the User Datagram Protocol (UDP).

ALGORITHM:

SERVER:

1. Import .net and other necessary packages.
2. Declare objects for DatagramSocket and DatagramPacket to receive packet data from server.
3. Declare an object for InetAddress of the LocalHost.
4. Get user input data and convert it into bytes and send the bytes using DatagramPacket and DatagramSocket.
5. Get user input in an loop and send data until the user input points end.
6. If end is encountered, exit sending data and program.

CLIENT:

1. Import .net and other necessary Packages.
2. Declare objects for DatagramSocket and Datagrampacket to send packet data from server.
3. Declare an object for InetAddress of the LocalHost.
4. Receive the server data using receive() method and store it to a string.
5. Run an loop and store the data in the string until the received message points end.
6. Print the string unless it encounters end.

SOURCE CODE:

SENDER:

```
import java.io.*;
import java.net.*;
class sender
{
DatagramSocket ds;
DatagramPacketdp;
byte buff[]=new byte[1024];
String str,str1;
booleani=true;
public void send() throws IOException
{
while(i)
{
ds=new DatagramSocket();
DataInputStream in=new DataInputStream(System.in);
System.out.println("Enter the msg:");
str=in.readLine();
buff=str.getBytes();
dp=new DatagramPacket(buff,buff.length,InetAddress.getLocalHost(),8000);
ds.send(dp);
System.out.println("do u want to continue:yes or no");
str1=in.readLine();
if(str1.equals("yes"))
{
i=true;
}
else
{
i=false;
}
}
}
public static void main(String args[])throws IOException
{
sender se=new sender();
se.send();
}
}
```

RECEIVER:

```
import java.io.*;
import java.net.*;
class receiver
{
DatagramSocket ds;
DatagramPacketdp;
byte buff[]=new byte[1024];
String str;
public void receive() throws IOException
{
ds=new DatagramSocket(8000);
while(true)
{
dp=new DatagramPacket(buff,buff.length);
ds.receive(dp);
str=new String (dp.getData(),0,0,dp.getLength());
System.out.println(str);
System.out.println("InetAddress:"+dp.getAddress());
}
}
public static void main(String args[])throws Exception
{
receive re=new receive();
re.receive();
}
}
```

UDP-TWO WAY COMMUNICATION

OBJECTIVE: To write a java program to perform two way message transfer using the user datagram protocol(UDP).

ALGORITHM:

SERVER:

1. Start the program.
2. Import.net and other necessary packages.
3. Declare objects for datagramSocket and DatagramPacket to receive packet data from server.
4. Receive an object for InetAddress of the LocalHost.
5. Receive the client data using receive() method and store it to a string.
6. Run a loop and store the data in the string until the received message points end.
7. Print the string unless it encounters end.
8. Get user input in the same loop and send data until the user input points end.
9. Convert the user input into bytes and send the byte using DatagramPacket and DatagramSocket.
10. If end is encountered, exit sending data and program.

CLIENT:

1. Start the program.
2. Import.net and other necessary packages.
3. Declare objects for datagramSocket and DatagramPacket to receive packet data from server.
4. Declare an object for InetAddress of the LocalHost.
5. Receive the Server data using receive() method and store it to a string.
6. Run a loop and store the data in the string until the received message points end.
7. Print the string unless it encounters end.
8. Get user input in the same loop and send data until the user input points end.
9. Convert the user input into bytes and send the byte using DatagramPacket and DatagramSocket.
10. If end is encountered, exit sending data and program.

PING COMMAND

OBJECTIVE: To write a program in java to demonstrate the usage of PING command.

ALGORITHM:

1. Start the program.
2. Import.net and other necessary packages.
3. Initialize the ping server with both sockets as null value.
4. Start the ServerSocket.
5. At the client end, give the IP address of the server.
6. The client program is then started by starting socket.
7. At the receiver end, the server is pinged.

SOURCE CODE:

PING SERVER:

```
import java.io.*;
import java.net.*;
public class pingserver
{
public static void main(String a[])
{
String line1,line2;
inti;
ServerSocket es;
DataInputStream di;
PrintStream ps;
Socket csoc;
es=null;
csoc=null;
try
{
```

```
es=new ServerSocket(9999);
}
catch(Exception e)
{
System.out.println(e);
}
System.out.println("ping server");
try
{
csoc=es.accept();
di=new DataInputStream(csoc.getInputStream());
ps=new PrintStream(csoc.getOutputStream());
for(i=0;i<4;i++)
{
line1=di.readLine();
System.out.println("pinged by client");
ps.println(line1+"reply from host:bytes=3<time<1ms TT<=128");
}
di.close();
ps.close();         }
catch(Exception e)
{
System.out.println(e);
}
}
}
```

PING CLIENT:

```
import java.io.*;
import java.net.*;
public class pingclient
{
public static void main(String args[])
{
PrintWriter out=null;
inti,j,k;
BufferedReadernetworkIn=null;
try
{
System.out.println("enter the IP address:"); DataInputStream in
= new DataInputStream(System.in); String ip = in.readLine();
Socket thesocket = new Socket(ip, 9999);
networkIn = new BufferedReader(new InputStreamReader(System.in)); out =
new PrintWriter(thesocket.getOutputStream()); System.out.println("\npinging"
+ ip + "with 32 bytes of data\n"); for (i = 0; i< 4; i++)
{
out.println(ip);
out.flush();
```

```
String inp = networkIn.readLine();
if (inp != null)
{
for (j = 0; j < 10000; j++)
{
for (k = 0; k < 50000; k++)
{
}
}
System.out.println("reply from" + inp);
}
else
{
for (i = 0; i< 4; i++)
{
for (j = 0; j < 10000; j++)
{
for (k = 0; k < 50000; k++)
{
}
System.out.println("\nrequest time out");}}}}
catch (IOException e)
{
for (i = 0; i< 4; i++)
{
for (j = 0; j < 1000; j++)
{
for (k = 0; k < 5000; k++)
{}}
System.out.println("\nrequest timed out");}}
try
{
if(networkIn!=null)
networkIn.close();
if(out!=null)
out.close();
}
catch(Exception e){
System.out.println("\nrequested time out");
}
}
}
```

IMPLEMENTATION OF FTP

OBJECTIVE: To write a java program to perform **"File Transfer Protocol"**.

ALGORITHM:

SERVER SIDE:

1. Import the java packages and create class fileserver.
2. String of argument is passed to the args[].
3. Create a new server socket and bind it to the port.
4. Accept the client connection at the requested port.
5. Get the filename and stored into the BufferedReader.
6. Create a new object class file and readline.
7. If File is exists then FileReader read the content until EOF is reached.
8. Else Print FileName doesn't exist.
9. End of main.
10. End of FileServer class.

CLIENT SIDE

1. Import the java packages and create class fileClient.
2. String of argument is passed to the args[].
3. The connection between the client and server is successfully established.
4. The object of a BufferReader class is used for storing data content which have been retrieved from socket object s.
5. The content are read and stored in inp until the EOF is reached.
6. The content of file are displayed in displayed in client window and the connection is closed.
7. End of main.
8. End of Fileclient class.

BROADCASTING

OBJECTIVE: To write a java program to send a single message to multiple clients.

ALGORITHM:

SENDER:

1. Start the program.
2. Import .net and other necessary packages.
3. The DatagramSocket and the DataInputStream are initialized.
4. The buffer is declare for message.
5. The message is obtained from the user.
6. The message is delivered to the client one by one with the help of the Datagram Packet.
7. The client details is obtained.
8. If message is 'bye' then the server quits.

CLIENT:

1. Start the program.
2. Import .net and other necessary packages.
3. The DatagramSocket and DatagramPacket are initialized.
4. The message from sender is taken as the packet from buffer.
5. The message is displayed along with the IP address of the server.
6. If the message is 'bye' then the client quits from the transmission and the connection is terminated.

TASK 01:

Write a java program to perform two way message transfer using the Transmission Control Protocol (TCP).

TASK 02:

Write a java program to perform two way message transfer using the User Datagram Protocol (UDP).

TASK 03:

Write a java program
(i) To transfer a computer file from client to server over a network.
(ii) To send a broadcast message to multiple clients.