NETWORK AND COMMUNICATION LABORATORY

(V semester of B.Tech)

As per the curriculam and syllabus

Of

Bharath Institute of Higher Education & Research

PREPARED BY
DR.YOGESH RAJ KUMAR

NEW EDITION

Department of information technology

Bharath
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Declared as Deemed - to - be - University under section 3 of UGC Act 1956)
ACCREDITED WITH 'A' GRADE BY NAAC

NETWORK AND COMMUNICATION

(V semester of B.Tech)

As per the curricullam and syllabus

Of

Bharath Institute of Higher Education & Research

PREPARED BY
DR.YOGESH RAJ KUMAR

Department of information technology

Bharath
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Declared as Deemed - to - be - University under section 3 of UGC Act 1956)
ACCREDITED WITH 'A' GRADE BY NAAC

SCHOOL OF COMPUTING

DEPARTMENT OF INFORMATION TECHNOLOGY

LAB MANUAL

SUBJECT NAME: NETWORK AND COMMUNICATION LABORATORY

SUBJECT CODE: U20ITCJO3

Regulation - 2020

U20ITCJ03	NETWORK AND COMMUNICATION LABORATORY	L	T	P	C
	Total Contact Hours - 75	**3**	**0**	**2**	**4**
	Prerequisite :network and communication				
	Lab Manual Designed by – Dept. of information technology				
OBJECTIVES: This laboratory course is intended to make the students know about Networking Concepts and Protocols.					

COURSE OUTCOMES (COs)	
CO1	Summarize the models in computer networks
CO2	
CO3	
CO4	

MAPPING BETWEEN COURSE OUTCOMES & PROGRAM OUTCOMES (3/2/1 INDICATES STRENGTH OF CORRELATION) 3- High, 2- Medium, 1-Low															
COs	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**
CO1															
CO2															
CO3															
CO4															
CO5															
CO6															
(Tick mark or level of correlation: 3-High, 2-Medium, 1-Low)															

VISION AND MISSION OF THE INSTITUTE

VISION

“Bharath Institute of Higher Education & Research (BIHER) envisions and constantly strives to provide an excellent academic and research ambience for students and members of the faculties to inherit professional competence along with human dignity and transformation of community to keep pace with the global challenges so as to achieve holistic development.”

MISSION

- To develop as a Premier University for Teaching, Learning, Research and Innovation on par with leading global universities.
- To impart education and training to students for creating a better society with ethics and morals.
- To foster an interdisciplinary approach in education, research and innovation by supporting lifelong professional development, enriching knowledge banks through scientific research, promoting best practices and innovation, industry driven and institute-oriented cooperation, globalization and international initiatives.
- To develop as a multi-dimensional institution contributing immensely to the cause of societal advancement through spread of literacy, an ambience that provides the best of international exposures, provide health care, enrich rural development and most importantly impart value-based education.
- To establish benchmark standards in professional practice in the fields of innovative and emerging areas in engineering, management, medicine, dentistry, nursing, physiotherapy and allied sciences.
- To imbibe human dignity and values through personality development and social service activities.

VISION AND MISSION OF THE DEPARTMENT

VISION

To be an excellence in education and research in Information Technology
producing global scholars for improvement of the society

MISSION

- To provide sound fundamentals, and advances in Information Technology, Software Engineering, data Communications and Computer Applications by offering world class curriculum.
- To create ethically strong leaders and expert for next generation IT.
- To nurture the desire among faculty and students from across the globe to perform outstanding and impactful research for the benefit of humanity and, to achieve meritorious and significant growth.

PROGRAM EDUCATIONAL OBJECTIVES (PEO)

The Program Educational Objectives (PEOs) of Information technology are listed below: The graduate after 3-5 years of programme completion will

PEO1: PREPARATION
To provide students with sound fundamental in Mathematical, Scientific and Engineering fundamentals necessary to formulate, analyse, and comprehend the fundamental concepts essential to articulate, solve and assess engineering problems and to prepare them for research & development and higher learning.

PEO2: CORE COMPETENCE
To apply critical reasoning, quantitative, qualitative, designing and programming skills, to identify, solve problems and to analyze the experimental evaluations, and finally making appropriate decisions along with knowledge of computing principles and applications and be able to integrate this knowledge in a variety of industry and inter-disciplinary setting.

PEO3: PROFESSIONALISM
To broaden knowledge to establish themselves as creative practicing professionals, locally and globally, in fields such as design, development, problem solving to production support in software industries and R&D sectors.

PEO4: SKILL
To provide better opportunity to become a future researchers / scientist with good communication skills so that they may be both good team-members and leaders with innovative ideas for a sustainable development.

PEO5: ETHICS
To be ethically and socially responsible solution providers and entrepreneurs in Computer Science and other engineering discipline.

PROGRAMME OUTCOMES

PO 1	**Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem Analysis:** Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
PO 3	**Design/Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO 4	**Conduct Investigations of Complex Problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions for complex problems.
PO 5	**Modern Tool Usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
PO 6	**The Engineer and Society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO 7	**Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO 8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO 9	**Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO 10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO 11	**Project Management and Finance:** Demonstrate knowledge and understanding of the

	engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO 12	**Life-long Learning:** Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

PROGRAMME SPECIFIC OUTCOME

PSO 1	**Programming Design :** Design and develop algorithm for real life problems using latest technologies and solve it by using computer programming languages and database technologies .
PSO 2	**IT Business Scalable Design :** Analyze and recommend computing infrastructures and operations requirements and Simulate and implement information networks using configurations, algorithms, suitable protocol and security for valid and optimal connectivity.
PSO 3	**Intelligent Agents Design :** Design and execute projects for the development of data modeling, data analytics and knowledge representation in various domain.

PART - B CONTENT OF THE COURSE

COURSE CONTENTS

UNIT – I : INTRODUCTION TO DATA COMMUNICATION.
Data Communications – Networks - Network Types - Internet History - Standards and Administration. Networks Models: Protocol Layering - TCP/IP Protocol suite - The OSI model.
Introduction to Physical Layer - 1 : Data and Signals - Digital Signals - Transmission Impairment - Data Rate limits - Performance.

UNIT – II : DIGITAL TRANSMISSION.
Digital Transmission: Digital to digital conversion (Only Line coding: Polar, Bipolar and Manchester coding). Physical Layer-2: Analog to digital conversion (only PCM), Transmission Modes) Analog Transmission: Digital to Analog conversion.

UNIT – III : BANDWIDTH UTILIZATION.
Bandwidth Utilization: Multiplexing and Spread Spectrum – Switching : Introduction - Circuit Switched Networks - Packet switching - Error Detection and Correction: Introduction - Block coding - Cyclic codes, Checksum.

UNIT – IV : DATA LINK CONTROL.
Data link control: DLC services - Data link layer protocols - Point to Point protocol (Framing, Transition phases). Media Access control: Random Access - Controlled Access - Channelization. Introduction to Data-Link Layer: Introduction - Link-Layer Addressing - ARP. IPv4 Addressing and subnetting: Classful – DHCP – NAT.

UNIT – V : WIRED LANS ETHERNET.
Wired LANs Ethernet: Ethernet Protocol - Standard Ethernet - Fast Ethernet - Gigabit Ethernet and 10 Gigabit Ethernet. Wireless LANs: Introduction, IEEE 802.11 Project and Bluetooth.
Other wireless Networks: Cellular Telephony.

LIST OF EXPERIMENTS:

1. Create a socket (TCP) between two computers and enable file transfer between them.
2. Write a program to develop a simple Chat TCP application.
3. Write a program to develop a simple Chat UDP application.
4. Write a socket Program for Echo/Ping/Talk commands.
5. Implementation of Stop and Wait Protocol and Sliding Window Protocol.
6. Implementation of DNS, SNMP and File Transfer application using TCP and UDP Sockets.
7. Create a socket for HTTP for web page upload and download.
8. Write a program to display the client’s address at the server end.
9. Study of Network simulator (NS).and Simulation of Congestion Control Algorithms using NS
10. Perform a case study about the different routing algorithms to select the network path with its optimum and economical during data transfer.
    i. Link State routing
    ii. Flooding
    iii. Distance vector

CONTENT

1. Create a socket (TCP) between two computers and enable file transfer between them.

AIM
To Perform File Transfer in Client & Server Using TCP/IP.

ALGORITHM

CLIENT SIDE
1. Start.
2. Establish a connection between the Client and Server.
3. Socket ss=new Socket(InetAddress.getLocalHost(),1100);
4. Implement a client that can send two requests. i) To get a file from the server.
ii) To put or send a file to the server.
5. After getting approval from the server ,the client either get file from the server or send
6. file to the server.

SERVER SIDE
1. Start.
2. Implement a server socket that listens to a particular port number.
3. Server reads the filename and sends the data stored in the file for the'get' request.
4. It reads the data from the input stream and writes it to a file in theserver for the 'put' instruction.
5. Exit upon client's request.
6. Stop.

PROGRAM
CLIENT SIDE

```
import java.net.*;
import java.io.*;
public class FileClient{
public static void main (String [] args ) throws IOException {
int filesize=6022386; // filesize temporary hardcoded
long start = System.currentTimeMillis();
int bytesRead;
int current = 0;
// localhost for testing
Socket sock = new Socket("127.0.0.1",13267);
System.out.println("Connecting...");
// receive file
byte [] mybytearray = new byte [filesize];
        InputStream is = sock.getInputStream();
FileOutputStream fos = new FileOutputStream("source-copy.pdf");
BufferedOutputStream bos = new BufferedOutputStream(fos);
bytesRead = is.read(mybytearray,0,mybytearray.length);

current = bytesRead;

// thanks to A. Cádiz for the bug fix

do {
bytesRead =
    is.read(mybytearray, current, (mybytearray.length-current));
if(bytesRead >= 0) current += bytesRead;
} while(bytesRead > -1);
bos.write(mybytearray, 0 , current);
   bos.flush();
   long end = System.currentTimeMillis();
   System.out.println(end-start);
   bos.close();
         sock.close();
         }}
```

SERVER SIDE

```
import java.net.*;
   import java.io.*;
   public class FileServer
   {
         public static void main (String [] args ) throws IOException {
   ServerSocket servsock = new ServerSocket(13267);
   while (true) {
System.out.println("Waiting...");
Socket sock = servsock.accept();
System.out.println("Accepted connection : " + sock);
File myFile = new File ("source.pdf");
byte [] mybytearray = new byte [(int)myFile.length()];
FileInputStream fis = new FileInputStream(myFile);
BufferedInputStream bis = new BufferedInputStream(fis);
bis.read(mybytearray,0,mybytearray.length);
OutputStream os = sock.getOutputStream();
System.out.println("Sending...");
os.write(mybytearray,0,mybytearray.length);
os.flush();
sock.close();
}}}
```

OUTPUT

SERVEROUTPUT

```
C:\Program Files\Java\jdk1.6.0\bin>javac FServer.java
C:\Program Files\Java\jdk1.6.0\bin>java FServer
Waiting for clients...
Connection Established
Client wants file:network.txt
```

CLIENTOUTPUT

```
C:\Program Files\Java\jdk1.6.0\bin>javac FClient.java
C:\Program Files\Java\jdk1.6.0\bin>java FClient
Connection request.....Connected
Enter the filename: network.txt
Computer networks: A computer network, often simply referred to as a network, is a collection of computers and
devices connected by communications channels that facilitates communications among users and allows users to
share resources with other user .
```

RESULT

Thus the File transfer Operation is done & executed successfully.

2. Write a program to develop a simple Chat TCP application.

AIM
To write a client-server application for chat using TCP .

ALGORITHM

CLIENT
1. Start the program
2. Include necessary package in java
3. To create a socket in client to server.
4. The client establishes a connection to the server.
5. The client accept the connection and to send the data from client to server.
6. The client communicates the server to send the end of the message
7. Stop the program.

SERVER

1. Start the program
2. Include necessary package in java
3. To create a socket in server to client
4. The server establishes a connection to the client.
5. The server accept the connection and to send the data from server to client and
6. vice versa
7. The server communicate the client to send the end of the message.
8. Stop the program.

PROGRAM

TCPserver1.java
```
import java.net.*;
import java.io.*;
public class TCPserver1
{
public static void main(String arg[])
{
ServerSocket s=null;
String line;
DataInputStream is=null,is1=null;
PrintStream os=null;
Socket c=null;
        try
        {
        s=new ServerSocket(9999);
}

catch(IOException e)
                                        {
System.out.println(e);
}
   try
{
c=s.accept();
is=new DataInputStream(c.getInputStream());
is1=new DataInputStream(System.in);
  os=new PrintStream(c.getOutputStream());
  do
  {

  line=is.readLine();
  System.out.println("Client:"+line);
  System.out.println("Server:");
  line=is1.readLine();
  os.println(line);
}
  while(line.equalsIgnoreCase("quit")==false);
  is.close();
  os.close();
  }
  catch(IOException e)
  {
  System.out.println(e);
  }
}
}
```

TCPclient1.java

```
import java.net.*;
import java.io.*;
public class TCPclient1
{
public static void main(String arg[])
{
Socket c=null;
String line;
DataInputStream is,is1;
PrintStream os;
try
{
c=new Socket("10.0.200.36",9999);
}
catch(IOException e)
{
System.out.println(e);
}
try
{
os=new PrintStream(c.getOutputStream());
is=new DataInputStream(System.in);
is1=new DataInputStream(c.getInputStream());
do
{
System.out.println("Client:");
line=is.readLine();
os.println(line);
System.out.println("Server:" + is1.readLine());
}
while(line.equalsIgnoreCase("quit")==false);
is1.close();
os.close();
}
catch(IOException e)
{
System.out.println("Socket Closed!Message Passing is over");
}}
```

OUTPUT:

SERVER

```
C:\Program Files\Java\jdk1.5.0\bin>javac TCPserver1.java
Note: TCPserver1.java uses or overrides a deprecated API.
Note: Recompile with -deprecation for details.
C:\Program Files\Java\jdk1.5.0\bin>java TCPserver1
Client: Hai Server
Server:Hai Client
Client: How are you
Server:Fine
Client: quit
Server:quit
```

RESULT

Thus the above program a client-server application for chat using TCP / IP was and successfully executed.

3. Write a program to develop a simple Chat UDP application

AIM
To write a program to implement simple client-server application using UDP.

ALGORITHM
CLIENT SIDE

1. Create a datagram socket with server’s IP address.
2. Create datagram packets with data, data length and the port address.
3. Send the datagram packets to server through datagram sockets
4. Receive the datagram packets from server through datagram sockets
5. Close the socket.

SERVER SIDE

1. Create a datagram socket with port address.
2. Create datagram packets with data, data length and the port address.
3. Send the datagram packets to client through datagram sockets
4. Receive the datagram packets from client through datagram sockets
5. Close the socket.

UDPserver.java
```
import java.io.*;
import java.net.*;
class UDPserver
{
public static DatagramSocket ds;
public static byte buffer[]=new byte[1024];
public static int clientport=789,serverport=790;
public static void main(String args[])throws Exception
{
ds=new DatagramSocket(clientport);
System.out.println("press ctrl+c to quit the program");
BufferedReader dis=new BufferedReader(new InputStreamReader(System.in));
InetAddress ia=InetAddress.getByName("localhost");
while(true)
{
DatagramPacket p=new DatagramPacket(buffer,buffer.length);
ds.receive(p);
String psx=new String(p.getData(),0,p.getLength());
System.out.println("Client:" + psx);
        System.out.println("Server:");String str=dis.readLine();
if(str.equals("end"))
break;

buffer=str.getBytes();

ds.send(new DatagramPacket(buffer,str.length(),ia,serverport));

}
}
  }
```

UDP CLIENT.JAVA

```
import java .io.*;
import java.net.*;
  class UDPclient
  {
  public static DatagramSocket ds;
  public static int clientport=789,serverport=790;
  public static void main(String args[])throws Exception
                {
byte buffer[]=new byte[1024];
ds=new DatagramSocket(serverport);
  BufferedReader dis=new BufferedReader(new InputStreamReader(System.in));
  System.out.println("server waiting");InetAddress ia=InetAddress.getByName("10.0.200.36");
  while(true)
  {
  System.out.println("Client:");
                String str=dis.readLine();
                if(str.equals("end")) break;
buffer=str.getBytes();
 ds.send(new DatagramPacket(buffer,str.length(),ia,clientport));
 DatagramPacket p=new DatagramPacket(buffer,buffer.length);
 ds.receive(p);String psx=new String(p.getData(),0,p.getLength());
 System.out.println("Server:" + psx);
 }
 }
 }
```

OUTPUT

Server

```
C:\Program Files\Java\jdk1.5.0\bin>javac UDPserver.java
C:\Program Files\Java\jdk1.5.0\bin>java UDPserver
press ctrl+c to quit the program
Client:Hai Server
Server:Hello Client
Client:How are You
Server:I am Fine what about you
```

CLIENT

```
C:\Program Files\Java\jdk1.5.0\bin>javac UDPclient.java
C:\Program Files\Java\jdk1.5.0\bin>java UDPclientserver
Waiting
Client:Hai Server
Server:Hello Clie
Client:How are YouServer:I am Fine
Client:end
```

RESULT

Thus the above program a client-server application for chat using UDP was executed and successfully

4. Write a socket Program for Echo/Ping/Talk commands.

AIM

To write a socket program for between two computers and enable file transfer between them.

ALGORITHM

CLIENT SIDE

1. Start the program.
2. Create a socket which binds the Ip address of server and the port address to acquire service.
3. After establishing connection send a data to server.
4. Receive and print the same data from server.
5. Close the socket.
6. End the program.

SERVER SIDE

1. Start the program.
2. Create a server socket to activate the port address.
3. Create a socket for the server socket which accepts the connection.
4. After establishing connection receive the data from client.
5. Print and send the same data to client.
6. Close the socket.
7. End the program.

PROGRAM

ECHO CLIENT

```
import java.io.*;
import java.net.*;
public class eclient
{
public static void main(String args[])
{

Socket c=null;

String line;

DataInputStream is,is1;

PrintStream os;

try

{

c=new Socket("localhost",8080);

}

catch(IOException e)

{
```

```
System.out.println(e);

}



try
        {

os=new PrintStream(c.getOutputStream());
is=new DataInputStream(System.in);

is1=new DataInputStream(c.getInputStream());

do
{
    System.out.println("client");
line=is.readLine();
os.println(line);
if(!line.equals("exit"))
System.out.println("server:"+is1.readLine());
}while(!line.equals("exit"));
}
catch(IOException e)
{
System.out.println("socket closed");
}}}
```

Echo Server:

```
import java.io.*;
import java.net.*;
import java.lang.*;
public class eserver
{
public static void main(String args[])throws IOException
{
ServerSocket s=null;
String line;

DataInputStream is;

PrintStream ps;

Socket c=null;
try
{
s=new ServerSocket(8080);
}
catch(IOException e)
{
System.out.println(e);
}
try
{
c=s.accept();
is=new DataInputStream(c.getInputStream());
ps=new PrintStream(c.getOutputStream());
while(true)
{
line=is.readLine();
System.out.println("msg received and sent back to client");
ps.println(line);
}




}
catch(IOException e)
{
System.out.println(e);
}
}
}
```

OUTPUT

CLIEN

```
Enter the IP address 127.0.0.1
CONNECTION ESTABLISHED
Enter the data BIHER
Client received BIHER
```

SERVER

```
CONNECTION ACCEPTED
Server received BIHER
```

RESULT

Thus the program for simulation of echo server was written & executed.

5. Implementation of Stop and Wait Protocol and Sliding Window Protocol

Aim:

To provide a reliable data transfer between two nodes over an unreliable network using the Stop andWait Protocol.

Problem Statement:

1. Write a C program to implement Go-Back N ARQ using Stop and Wait protocol.
2. Write a C program to implement Selective Repeat ARQ using Stop and Wait protocol.
3. Write a C program to implement Go-Back N ARQ using Sliding Window protocol.
4. Write a C program to implement Selective Repeat ARQ using Sliding Window protocol.
5. Write a C program to implement A One-Bit Sliding Window Protocol.

Algorithm:

1. Start the program.
2. Get the frame size from the user
3. To create the frame based on the user request.
4. To send frames to server from the client side.
5. If your frames reach the server it will send ACK signal to client otherwise it will send NACK signal toclient.
6. Stop the program

Program:

```
#include<stdio.h>
#include<conio.h>
char str[20][20]; int
i,n;
void sender()
{
printf("\n\t********* SENDER ***********\n");
printf("\n\t\t%s",str[i]);
printf("\n\t\tdata %d sent",i);
printf("\n\t\tWaiting for ACK %d\n",i);
}
void receiver()
{
printf("\n\t********* RECEIVER ***********\n");
printf("\n\t\tdata %d received",i);
printf("\n\t\tPress any key to send ACK\n");
 getch();
printf("\n\t\tACK %d sent\n",i);
}
void main()
{
clrscr();
printf("\nStop and wait starts\nEnter the no. of data to send");
scanf("%d",&n);
printf("\nEnter the data\n");
for(i=1;i<=n;i++)
scanf("%s",str[i]); printf("\nStart
sending data"); for(i=1;i<=n;i++)
{
sender();
 receiver();
}
printf("\nAll data sent");
getch();
```

Output:

Observation:

Stop and Wait ARQ mainly implements Sliding Window Protocol concept with Window Size 1 Used in Connection-oriented communication. It offers error and flow control and It is used in Data Link and Transport Layers.

Result:

Thus the program was written & executed.

5A. SLIDING WINDOW PROTOCOL

Aim:

To write a C program to implement sliding window protocol.

Algorithm:

Sender:

1. Establish socket connection to the receiver.
2. After successful connection with the receiver, acknowledgement will be received
3. Get the number of frames need to be sent by the sender.
4. Each frame will have unique sequence number and sent in order
5. The sender will wait for the acknowledgement for every successful transmission of each frame from receiver side.
6. In case of unsuccessful transmission, the sender will not receive any acknowledgement and again the particular frame will be re-transmitted.

Receiver:

1. Establish the sender's socket connection.
2. After successful connection, send acknowledgement to the sender
3. With each frame received with their unique sequence number, acknowledgement message will be sent.
4. In case of faulty frame structure, the receiver will not send any response and it waits for the sender to retransmit the message again.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
int temp1,temp2,temp3,temp4;
int winsize=8;
int noframes,moreframes,i;
int receiver(int);
int simulate(int);
temp1=0; temp2=0;
temp3=0; temp4=0;
clrscr();
for(i=0;i<200;i++)
rand(); noframes=5;
printf("\nNo. of frames is %d",noframes);
getch();
moreframes=noframes;
while(moreframes>0)
{
temp1=simulate(winsize);
winsize=temp1; temp4+=temp1;
if(temp4>noframes)
temp4=noframes;
for(i=temp3+1;i<=temp4;i++)
printf("\nSending frame %d",i);
getch(); temp2=receiver(temp1);
temp3+=temp2;
if(temp3>noframes)
temp3=noframes;
printf("\nAcknowledgement for the frames upto %d",temp3);
getch();
moreframes-=temp2;
temp4=temp3;
if(winsize<=0)
winsize=8;
}
printf("\nEnd of SLIDING WINDOW PROTOCOL");
getch();
}
int receiver(int temp1)
{
int i;
for(i=0;i<100;i++)
rand();
i=rand()%temp1;
return 1;
}
int simulate(int winsize)
{
int temp1,i;
for(i=0;i<50;i++)
temp1=rand(); if(temp1==0)
temp1=simulate(winsize);
i=temp1%winsize; if(i==0)
return winsize;
else
return temp1%winsize;
}
```

Output:

Observation:

A sliding window protocol is a feature of packet-based data transmission protocols. Sliding window protocols are used where reliable in-order delivery of packets is required, such as in the Data Link Layer (OSI model) as well as in the Transmission Control Protocol (TCP).

Result:

Thus the program was written & executed.

6. Implementation of DNS, SNMP and File Transfer application using TCP and UDP Sockets.

a.DNS

Aim

To write a java program for Dns application program

Algorithm

1. Start the program.
2. Get the frame size from the user
3. To create the frame based on the user request.
4. To send frames to server from the client side.
5. If your frames reach the server it will send ACK signal to client otherwise it will send NACK signal to client.
6. Stop the program

Program

/ UDP DNS Server
Udpdnsserver

```
.java import java.io.*;
import java.net.*;
 public class udpdnsserver
{
private static int indexOf(String[] array, String str)
 {
str = str.trim();
for (int i=0; i < array.length; i++)
{
if (array[i].equals(str)) return i;
 }
return -1;
 }
public static void main(String arg[])throws IOException
 {
String[] hosts = {"yahoo.com", "gmail.com","cricinfo.com", "facebook.com"};
String[] ip = {"68.180.206.184", "209.85.148.19","80.168.92.140", "69.63.189.16"};
System.out.println("Press Ctrl + C to Quit");
 while (true)
{
DatagramSocket serversocket=new DatagramSocket(1362);
byte[] senddata = new byte[1021];
 byte[] receivedata = new byte[1021];
DatagramPacket recvpack = new DatagramPacket
(receivedata, receivedata.length);
serversocket.receive(recvpack);
String sen = new String(recvpack.getData());
 InetAddress ipaddress = recvpack.getAddress();
 int port = recvpack.getPort();
 String capsent;
 System.out.println("Request for host " + sen);

if(indexOf (hosts, sen) != -1)
capsent = ip[indexOf (hosts, sen)];
else capsent = "Host Not Found";
senddata = capsent.getBytes();
 DatagramPacket pack = new DatagramPacket
(senddata, senddata.length,ipaddress,port);
serversocket.send(pack);
serversocket.close();
}
}
}
```

//**UDP DNS Client –**

```
 Udpdnsclient
.java import java.io.*;
 import java.net.*;
 public class udpdnsclient
 {
 public static void main(String args[])throws IOException
 {

 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
DatagramSocket clientsocket = new DatagramSocket();
 InetAddress ipaddress;
 if (args.length == 0)
 ipaddress = InetAddress.getLocalHost();
else
 ipaddress = InetAddress.getByName(args[0]);
 byte[] senddata = new byte[1024];
byte[] receivedata = new byte[1024];
int portaddr = 1362;
System.out.print("Enter the hostname : ");
String sentence = br.readLine();
Senddata = sentence.getBytes();
DatagramPacket pack = new DatagramPacket(senddata,senddata.length, ipaddress,portaddr);
clientsocket.send(pack);
DatagramPacket recvpack =new DatagramPacket(receivedata,receivedata.length);
clientsocket.receive(recvpack);
String modified = new String(recvpack.getData());
System.out.println("IP Address: " + modified);
clientsocket.close();
}
}
```

OUTPUT

Server

```
$ javac udpdnsserver.java $ java udpdnsserver Press Ctrl + C to Quit Request for host
yahoo.com Request for host cricinfo.com Request for host youtube.com
```

Client

```
$ javac udpdnsclient.java $ java udpdnsclient Enter the hostname : yahoo.com IP Address:
68.180.206.184 $ java udpdnsclient Enter the hostname : cricinfo.com IP Address:
80.168.92.140 $ java udpdnsclient Enter the hostname : youtube.com IP Address: Host Not
Found
```

Result:

Thus the program was written & executed.

b. File Transfer

AIM

To write a java program for applaction using TCP and UDP Sockets Liks

Program

File Client

```
import java.io.*;
import java.net.*;
import java.util.*;
class Clientfile
{		public static void main(String args[])
{
Try
{
BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
Socket clsct=new Socket("127.0.0.1",139);
DataInputStream din=new DataInputStream(clsct.getInputStream());
DataOutputStream dout=new DataOutputStream(clsct.getOutputStream());
System.out.println("Enter the file name:");

String str=in.readLine();
dout.writeBytes(str+'\n');
System.out.println("Enter the new file name:");
String str2=in.readLine();
String str1,ss;
FileWriter f=new FileWriter(str2);
char buffer[];
while(true)
{		str1=din.readLine();
if(str1.equals("-1")) break;
System.out.println(str1);
buffer=new char[str1.length()];
str1.getChars(0,str1.length(),buffer,0);
f.write(buffer);
```

```
  }
f.close();
clsct.close();
}
catch (Exception e)
{
 System.out.println(e);
}
}
}
```

Server

```
import java.io.*;
import java.net.*;
import java.util.*;
class Serverfile
{          public static void main(String args[])
{
Try
{
 ServerSocket obj=new ServerSocket(139);
while(true)
{
 Socket obj1=obj.accept();
DataInputStream din=new DataInputStream(obj1.getInputStream());
DataOutputStream dout=new DataOutputStream(obj1.getOutputStream());
String str=din.readLine();
FileReader f=new FileReader(str);
BufferedReader b=new BufferedReader(f);
String s;
while((s=b.readLine())!=null)
{          System.out.println(s);
dout.writeBytes(s+'\n');
 }
f.close();
dout.writeBytes("-1\n");
 }          }
catch(Exception e)
{          System.out.println(e);}
 }
}
```

<u>OUT PUT</u>

```
Hardware: x86 Family 6 Model 23 Stepping 10 AT/AT COMPATIBLE – Software: Windows
2000 Version  5.1 (Build 2600 Multiprocessor Free)
```

RESULT

Thus the SNMP program was displayed.

7. Create a socket for HTTP for web page upload and download

Algorithm

1.Start the program.
2.Get the frame size from the user
3.To create the frame based on the user
request.4.To send frames to server from the
client side.
5.If your frames reach the server it will send ACK signal to client otherwise it
willsend NACK signal to client.
6.Stop the program

Program :

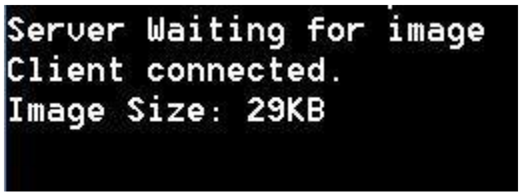
```
import javax.swing.*;
import  java.net.*;
import
java.awt.image.*;
import
javax.imageio.*;
import java.io.*;
import
java.awt.image.BufferedImage;
import
java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import
javax.imageio.ImageIO;

public class Client{
  public static void main(String args[]) throws
    Exception{Socket soc;
    BufferedImage img = null;
    soc=new
    Socket("localhost",4000);
    System.out.println("Client is running.
    ");try {
      System.out.println("Reading image from disk. ");
      img = ImageIO.read(new
      File("digital_image_processing.jpg"));
      ByteArrayOutputStream baos = new
      ByteArrayOutputStream();
```

```
                        ImageIO.write(img, "jpg", baos);
         baos.flush();
         byte[] bytes =
         baos.toByteArray();
         baos.close();


         System.out.println("Sending image to server. ");

                        OutputStream out =
         soc.getOutputStream();DataOutputStream dos = new
         DataOutputStream(out); dos.writeInt(bytes.length);
         dos.write(bytes, 0, bytes.length);
         System.out.println("Image sent to server.
         ");

         dos.close();
         out.close();
       }catch (Exception e) {
         System.out.println("Exception: " +
         e.getMessage());soc.close();
       }
         soc.close();
   }
}


import
java.net.*;
import
java.io.*;
import
java.awt.image.*;
import
javax.imageio.*;
import javax.swing.*;

class Server {
   public static void main(String args[]) throws Exception{
      ServerSocket server=null;
      Socket socket;
      server=new ServerSocket(4000);
      System.out.println("Server Waiting for
      image");

      socket=server.accept();
      System.out.println("Client connected.");
```

```
    InputStream in = socket.getInputStream();
    DataInputStream dis = new
    DataInputStream(in);

    int len = dis.readInt();
    System.out.println("Image Size: " + len/1024 +
    "KB");byte[] data = new byte[len];
    dis.readFully(dat
    a);dis.close();
            in.close();

            InputStream ian = new
            ByteArrayInputStream(data);BufferedImage
            bImage = ImageIO.read(ian);

            JFrame f = new JFrame("Server");
            ImageIcon icon = new
            ImageIcon(bImage);JLabel l = new
            JLabel();


            l.setIcon(ico
            n);f.add(l);
            f.pack();
            f.setVisible(tru
            e);
  }
}
```

Output

When you run the client code, following output screen would appear on client side.



```
Server Waiting for image
Client connected.
Image Size: 29KB
```

RESULT

Thus the program was implementing to socket for HTTP for web page upload anddownload.

8. Write a program to display the client's address at the server end

Aim:

To implement a java program for simulating ping command.

Algorithm:

1. Start the program.
2. Get the frame size from the user
3. To create the frame based on the user request.
4. To send frames to server from the client side.
5. If your frames reach the server it will send ACK signal to client otherwise it will send NACKsignal to client.

Program

```
//pingclient.j
ava import
java.io.*;
import
java.net.*;

import
 java.util.Cal
 endar;class
 pingclient
{
public static void main(String args[])throws Exception
{

String str; int c=0; longt1,t2;
Socket s=new Socket("127.0.0.1",5555);

DataInputStream dis=new
DataInputStream(s.getInputStream());PrintStream
out=new PrintStream(s.getOutputStream());
while(c<4)
{
```

```
t1=System.currentTimeMillis();
str="Welcome to network
programming world";
out.println(str);
System.out.println(dis.readLine());
t2=System.currentTimeMillis();
System.out.println(";TTL="+(t2-
t1)+"ms"); c++;
}


s.close();
}
}
```

//pingserver.java import

```
java.io.*;
import          java.net.*;
import          java.util.*;
import  java.text.*;class
pingserver
{
public static void main(String args[])throws Exception
{
ServerSocket ss=new
ServerSocket(5555);Socket
s=ss.accept();
int c=0; while(c<4)
{
DataInputStream dis=new
DataInputStream(s.getInputStream());PrintStream
out=new PrintStream(s.getOutputStream());
String str=dis.readLine();
out.println("Reply from"+InetAddress.getLocalHost()+";Length"+str.length()); c++;
}
s.close();
}
}
```

Output:

Observation:

Ping is a computer network administration software utility used to test the reachability ofa host on an Internet Protocol (IP) network. It measures the round-trip time for messages sent from the originating host to a destination computer that are echoed back to the source. The name comes from active sonar terminology that sends a pulse of sound and listens for the echo to detect objects under water, although it is sometimes interpreted as a bacronym to packet Internet groper.

Result:

_Thus the program was written & executed.

9. Study of Network simulator (NS).and Simulation of Congestion Control Algorithms using NS

Aim:

To Study of Network simulator (NS).and Simulation of Congestion Control Algorithms usingNS

NET WORK SIMULATOR (NS2)
Ns overview

- Ns programming: A Quick start
- Case study I: A simple Wireless network
- Case study II: Create a new agent in Ns

Ns overview

- Ns Status
- Periodical release (ns-2.26, Feb 2003)
- Platform support
- FreeBSD, Linux, Solaris, Windows and Mac

Ns unctionalities
Routing, Transportation, Traffic sources,Queuing
disciplines, QoS

Wireless

Ad hoc routing, mobile IP, sensor-MAC
Tracing, visualization and various utilitie
NS(Network Simulators)

Most of the commercial simulators are GUI driven, while some network simulators are CLI driven. The network model / configuration describes the state of the network (nodes,routers, switches, links) and the events (data transmissions, packet error etc.). An important output of simulations are the trace files. Trace files log every packet, every event that occurred in the simulation and are used for analysis. Network simulators can also provide other tools to facilitate visual analysis of trends and potential trouble spots.

Most network simulators use discrete event simulation, in which a list of pending "events" is stored, and those events are processed in order, with some events triggering future events—such as the event of the arrival of a packet at one node triggering the event of the arrivalof that packet at a downstream node.

Simulation of networks is a very complex task. For example, if congestion is high, then estimation of the average occupancy is challenging because of high variance. To estimate the likelihood of a buffer overflow in a network, the time required for an accurate answer can be extremely large. Specialized techniques such as "control variates" and "importance sampling" have been developed to speed simulation.

Examples of network simulators

There are many both free/open-source and proprietary network simulators. Examples of notable network simulation software are, ordered after how often they are mentioned in researchpapers:

1. ns (open source)
2. OPNET (proprietary software)
3. NetSim (proprietary software)

Uses of network simulators

Network simulators serve a variety of needs. Compared to the cost and time involved in setting up an entire test bed containing multiple networked computers, routers and data links, network simulators are relatively fast and inexpensive. They allow engineers, researchers to test scenarios that might be particularly difficult or expensive to emulate using real hardware - for instance, simulating a scenario with several nodes or experimenting with a new protocol in the network. Network simulators are particularly useful in allowing researchers to test new networking protocols or changes to existing protocols in a controlled and reproducible environment. A typical network simulator encompasses a wide range of networking technologies and can help the users to build complex networks from basic building blocks such as a variety of nodes and links. With the help of simulators, one can design hierarchical networks using various types of nodes like computers, hubs, bridges, routers, switches, links, mobile units etc.

Various types of Wide Area Network (WAN) technologies like TCP, ATM, IP etc. and Local Area Network (LAN) technologies like Ethernet, token rings etc., can all be simulated with a typical simulator and the user can test, analyze various standard results apart from devising some novel protocol or strategy for routing etc. Network simulators are also widely used to simulate battlefield networks in Network-centric warfare

There are a wide variety of network simulators, ranging from the very simple to the very complex. Minimally, a network simulator must enable a user to represent a network topology, specifying the nodes on the network, the links between those nodes and the traffic between the nodes. More complicated systems may allow the user to specify everything about the protocols used to handle traffic in a network. Graphical applications allow users to easily visualize the workings of their simulated environment. Text-based applications may provide a less intuitive interface, but may permit more advanced forms of customization.

Packet loss
occurs when one or morepacketsof data travelling across a computer networkfail to reachtheir destination. Packet loss is distinguished as one of the three main error types encountered in digital communications; the other two being bit errorand spurious packets caused due to noise. Packets can be lost in a network because they may be dropped when a queue in the network node overflows. The amount of packet loss during the steady state is another important property of a congestion control scheme. The larger the value of packet loss, the more difficult it is for transportlayer protocols to maintain high bandwidths, the sensitivity to loss of individual packets, as well as to frequency and patterns of loss among longer packet sequences is strongly dependent on the application itself.

Throughput

This is the main performance measure characteristic, and most widely used.
Incommunicationnetworks, such asEthernetorpacket radio, throughputor network throughputis the average rate of successfulmessage delivery over a communication channel. Thethroughput is usually measured inbitsper second (bit/s orbps), andsometimes indata packetspersecond or data packets pertime slotThis measure how soon the receiver is able to get a certainamount of data send by the sender. It is determined as the ratio of the total data received to theend to end delay. Throughput is an important factor which directly impacts the networkperformance

Delay
Delay is the time elapsed while a packet travels from one point e.g., source premise or network ingress to destination premise or network degrees. The larger the valueof delay, the moredifficult it is for transport layer protocols to maintain highbandwidths. We will calculate end to end delay

Queue Length
A queuing system in networks can be described as packets arriving for service, waiting for service if it is not immediate, and if having waited for service, leaving thesystem after being served. Thus queue length is very important characteristic to determine that how well the active queue management of the congestion control
algorithm has been working.

RESULT
Thus the study of Network simulator (NS2)was studied

10. Perform a case study about the different routing algorithms to select the network path with its optimum and economical during data transfer.

i. Link State routingAim:
To study the link state routing

Link State routing
Routing is the process of selecting best paths in a network. In the past, the term routing was also used to mean forwarding network traffic among networks. However this latter function is much better described as simply forwarding. Routing is performed for many kinds of networks, including the telephone network (circuit switching), electronic data networks (such as the Internet), and transportation networks. This article is concerned primarily with routing in electronic data networks using packet switching technology.

In packet switching networks, routing directs packet forwarding (the transit of logically addressed network packets from their source toward their ultimate destination) through intermediate nodes. Intermediate nodes are typically network hardware devices such as routers, bridges, gateways, firewalls, or switches. General-purpose computers can also forward packets and perform routing, though they are not specialized hardware and may suffer from limited performance. The routing process usually directs forwarding on the basis of routing tables which maintain a record of the routes to various network destinations. Thus, constructing routing tables, which are held in the router's memory, is very important for efficient routing. Most routing algorithms use only one network path at a time. Multipath routing techniques enable the use of multiple alternative paths.

In case of overlapping/equal routes, the following elements are considered in order to decide which routes get installed into the routing table (sorted by priority):

1. *Prefix-Length*: where longer subnet masks are preferred (independent of whether it iswithin a routing protocol or over different routing protocol)
2. *Metric*: where a lower metric/cost is preferred (only valid within one and the same routing protocol)
3. *Administrative distance*: where a lower distance is preferred (only valid between differentrouting protocols)

Routing, in a more narrow sense of the term, is often contrasted with bridging in its assumption that network addresses are structured and that similar addresses imply proximity within the network. Structured addresses allow a single routing table entry to represent the route to a group of devices. In large networks, structured addressing (routing, in the narrow sense) outperforms unstructured addressing (bridging). Routing has become the dominant form of addressing on the Internet. Bridging is still widely used within localized environments.

ii. Flooding

Flooding s a simple routing algorithm in which every incoming packet is sent through every outgoing link except the one it arrived on.Flooding is used in bridging and in systems such as Usenet and peer-to-peer file sharing and as part of some routing protocols, including OSPF, DVMRP, and those used in ad-hoc wireless networks.There are generally two types of flooding available, Uncontrolled Flooding and Controlled Flooding.Uncontrolled Flooding is the fatal law of flooding. All nodes have neighbours and route packets indefinitely. More than two neighbours creates a broadcast storm.

Controlled Flooding has its own two algorithms to make it reliable, SNCF (Sequence Number Controlled Flooding) and RPF (Reverse Path Flooding). In SNCF, the node attaches its own address and sequence number to the packet, since every node has a memory of addresses and sequence numbers. If it receives a packet in memory, it drops it immediately while in RPF, the node will only send the packet forward. If it is received from the next node, it sends it back to the sender.

Algorithm

There are several variants of flooding algorithm. Most work roughly as follows:

1. Each node acts as both a transmitter and a receiver.
2. Each node tries to forward every message to every one of its neighbours except thesource node.

This results in every message eventually being delivered to all reachable parts of the network.

Algorithms may need to be more complex than this, since, in some case, precautions have to be taken to avoid wasted duplicate deliveries and infinite loops, and to allow messages to eventually expire from the system. A variant of flooding called *selective flooding* partially addresses these issues by only sending packets to routers in the same direction. In selective flooding the routers don't send every incoming packet on every line but only on those lines which are going approximately in the right direction.

Advantages

- f a packet can be delivered, it will (probably multiple times).
- Since flooding naturally utilizes every path through the network, it will also use theshortest path.
- This algorithm is very simple to implement.

Disadvantages

- Flooding can be costly in terms of wasted bandwidth. While a message may only have one destination it has to be sent to every host. In the case of a ping flood or a denial of service attack, it can be harmful to the reliability of a computer network.
- Messages can become duplicated in the network further increasing the load on the networks bandwidth as well as requiring an increase in processing complexity to disregard duplicate messages.
- Duplicate packets may circulate forever, unless certain precautions are taken:
- Use a hop count or a time to live count and include it with each packet. This value should take into account the number of nodes that a packet may have to pass through on the way to its destination.
- Have each node keep track of every packet seen and only forward each packet once
- Enforce a network topology without loops

iii . Distance vector

In computer communication theory relating to packet-switched networks, a **distance- vector routing protocol** is one of the two major classes of routing protocols, the other major class being the link-state protocol. Distance-vector routing protocols use the Bellman–Fordalgorithm, Ford–Fulkerson algorithm, or DUAL FSM (in the case of Cisco Systems's protocols) to calculate paths.

A distance-vector routing protocol requires that a router informs its neighbors of topologychanges periodically. Compared to link-state protocols, which require a router to inform all the nodes in a network of topology changes, distance-vector routing protocols have less computational complexity and message overhead.

The term *distance vector* refers to the fact that the protocol manipulates *vectors* (arrays) of distances to other nodes in the network. The vector distance algorithm was the original ARPANET routing algorithm and was also used in the internet under the name of RIP (Routing Information Protocol).

Examples of distance-vector routing protocols include RIPv1 and RIPv2 and IGRP.

Method

Routers using distance-vector protocol do not have knowledge of the entire path to a destination. Instead they use two methods:

1. Direction in which router or exit interface a packet should be forwarded.
2. Distance from its destination

Distance-vector protocols are based on calculating the direction and distance to any link in a network. "Direction" usually means the next hop address and the exit interface. "Distance" is a measure of the cost to reach a certain node. The least cost route between any two nodes is the route with minimum distance. Each node maintains a vector (table) of minimum distance to every node. The cost of reaching a destination is calculated using various route metrics. RIP uses the hop count of the destination whereas IGRP takes into account other information such as node delay and available bandwidth.

Updates are performed periodically in a distance-vector protocol where all or part of a router's routing table is sent to all its neighbors that are configured to use the same distance-vector routing protocol. RIP supports cross-platform distance vector routing whereas IGRP is a Cisco Systems proprietary distance vector routing protocol. Once a router has this information it is able to amend its own routing table to reflect the changes and then inform its neighbors of the changes. This process has been described as ‗routing by rumor' because routers are relying onthe information they receive from other routers and cannot determine if the information is actually valid and true. There are a number of features which can be used to help with instability and inaccurate routing information.

EGP and BGP are not pure distance-vector routing protocols because a distance-vector protocol calculates routes based only on link costs whereas in BGP, for example, the local route preference value takes priority over the link cost.

Count-to-infinity problem

The Bellman–Ford algorithm does not prevent routing loops from happening and suffers from the **count-to-infinity problem**. The core of the count-to-infinity problem is that if A tells B that it has a path somewhere, there is no way for B to know if the path has B as a part of it. To see the problem clearly, imagine a subnet connected like A–B–C–D–E–F, and let the metric between the routers be "number of jumps". Now suppose that A is taken offline. In the vector-update-process B notices that the route to A, which was distance 1, is down – B does not receive the vector update from A. The problem is, B also gets an update from C, and C is still not aware of the fact that A is down – so it tells B that A is only two jumps from C (C to B to A), which is false. This slowly propagates through the network until it reaches infinity (in which case the algorithm corrects itself, due to the relaxation property of Bellman–Ford).

RESULT

Thus The Perform a case study about the different routing algorithms to select the network path with its optimum and economical during data transfer was completed .