SAI RAM ENGINEERING COLLEGE

An Autonomous Institution | Affiliated to Anna University & Approved by AICTE, New Delhi

Accredited by NBA and NAAC "A+" | BIS/EOMS ISO 21001 : 2018 and BVQI 9001 : 2015 Certified and NIRF ranked institution

Sai Leo Nagar, West Tambaram, Chennai - 600 044. www.sairam.edu.in

PROSPERITY THROUGH TECHNOLOGY

LAB MANUAL

20ECPL602 NETWORKS LABORATORY

VI Semester ECE

Academic year: 2023-2024

Department
of
Electronics and Communication Engineering

PREFACE

"THE TRUE METHOD OF KNOWLEDGE IS EXPERIMENT"

The true transmission of knowledge really occurs when the student is open to and engaged in the information. With this in mind, this manual is compiled as a preparatory note for the Communication Systems laboratory experiments. Sufficient details have been included to impart self-learning.

Networks laboratory experiments mainly focuses on error detection techniques, different data link layer protocols, network layer protocols. This laboratory also focuses on NS2 and Java simulation of error detection technique, Link layer protocols ,Routing algorithms. Knowledge of Networks laboratory is essential as they are the basics for most of the communication systems.

This manual is intended for the VI semester ECE students . Each experiment is provided with introductory information and procedure to perform the experiment.

The manual has been compiled by Ms. B. Rajalakshmi, Assistant Professor/ECE and Ms. V. Remya, Assistant Professor/ECE Department. It is expected that this will be well received by the students.

Head of the Department **Principal**

INSTITUTION VISION

To emerge as a ”Centre of excellence “ offering Technical Education and Research opportunities of very high standards to students, develop the total personality of the individual and instill high levels of disciple and strive to set global standards, making our students technologically superior and ethically stronger, who in turn shall contribute to the advancement of society and humankind.

INSTITUTION MISSION

We dedicate and commit ourselves to achieve, sustain and foster unmatched excellence in Technical Education. To this end, we will pursue continuous development of infra-structure and enhance state-of-art equipment to provide our students a technologically up-to date and intellectually inspiring environment of learning, research, creativity, innovation and professional activity and inculcate in them ethical and moral values.

INSTITUTION POLICY

We at Sri Sai Ram Engineering College are committed to build a better Nation through Quality Education with team spirit. Our students are enabled to excel in all values of Life and become Good Citizens. We continually improve the System, Infrastructure and Service to satisfy the Students, Parents, Industry and Society.

DEPARTMENT VISION

To emerge as a "centre of excellence" in the field of Electronics and Communication Engineering and to mould our students to become technically and ethically strong to meet the global challenges. The Students in turn contribute to the advancement and welfare of the society.

DEPARTMENT MISSION

M1: To achieve, sustain and foster excellence in the field of Electronics and Communication Engineering.

M2: To adopt proper pedagogical methods to maximize the knowledge transfer.

M3: To enhance the understanding of theoretical concepts through professional society activities

M4: To improve the infrastructure and provide conducive environment of learning and research following ethical and moral values

Program Educational Objectives (PEOs)

To prepare the graduates to:

1. Acquire strong foundation in Engineering, Science and Technology for a successful career in Electronics and Communication Engineering.
2. Apply their knowledge and skills acquired to solve the issues in real world Electronics and Communication sectors and to develop feasible and viable systems.
3. Be receptive to new technologies and attain professional competence through professional society activities.
4. Participate in lifelong learning, higher education efforts to emerge as expert researchers and technologists.
5. Practice the profession with ethics, integrity, leadership and social responsibilities.

Program Specific Outcomes PSO

Electronics and Communication Engineering graduates will be able to:

1. Design, implement and test Electronics and Communication systems using analytical knowledge and applying modern hardware and software tools
2. Develop their skills to solve problems and assess social, environmental issues with ethics and manage different projects in multidisciplinary areas.

COURSE OUTCOMES

On completion of this laboratory course, the student should be able to:

1	Communicate between two desktop computers. (K2)
2	Implement different Protocols such as Stop & Wait, Go back N/Sliding window, Selective repeat. (K2)
3	Study the performance of network with CSMA / CA protocol and compare with CSMA/CD protocols. (K2)
4	Program using Sockets –Client server model, Echo/Ping/Talk commands. (K2)
5	Implement and compare Distance vector and Link state routing algorithms & congestion control algorithm. (K3)
6	Use simulation tool such as NS2/OPNET. (K2)

SYLLABUS

20ECPL602 NETWORK LABORATORY

L	T	P	C
0	0	2	1.5

OBJECTIVES:

- Learn to communicate between two desktop computers.
- Learn to implement the different protocols.
- Be familiar with IP Configuration.
- Be familiar with the various routing algorithms.
- Be familiar with simulation tools.

LIST OF EXPERIMENTS

1. Implementation of Error Detection / Error Correction Techniques
2. Implementation of Stop and Wait Protocol and sliding window
3. Implementation and study of Goback-N and selective repeat protocols
4. Implementation of High Level Data Link Control
5. Implementation of IP Commands such as ping, Traceroute, nslookup.
6. Implementation of IP address configuration.
7. To create scenario and study the performance of network with CSMA / CA protocol and compare with CSMA/CD protocols.
8. Network Topology - Star, Bus, Ring .
9. Implementation of distance vector routing algorithm.
10. Implementation of Link state routing algorithm.
11. Study of Network simulator (NS) and simulation of Congestion Control Algorithms using NS.
12. Implementation of Encryption and Decryption Algorithms using any programming language.

TOTAL: 45 PERIODS

TABLE OF CONTENTS

CONTINUOUS ASSESSMENT

Description	Maximum Marks
Aim / Procedure	**2**
Observation / Tabulation	**2**
Calculation	**2**
Graph & Result	**2**
Viva	**2**
Total	**10**

EXP No:1	ERROR DETECTION/ERROR CORRECTION USING HAMMING CODE
DATE:	

AIM:

To perform the Error Detection /Error Correction using Hamming code in transmitting and receiving a message between the sender and receiver.

REQUIREMENTS:

Operating System : Windows NT/2000/XP
Programming Tool : NetBeans IDE

THEORY:

To provide service to the network layer, the data link layer must use the service provided to it by the physical layer. What the physical layer does is accept a raw bit stream and attempt to deliver it to the destination. If the channel is noisy, as it is for most wireless and some wired links, the physical layer will add some redundancy to its signals to reduce the bit error rate to a tolerable level. However, the bit stream received by the data link layer is not guaranteed to be error free. Some bits may have different values and the number of bits received may be less than, equal to, or more than the number of bits transmitted. It is up to the data link layer to detect and, if necessary, correct errors.

Hamming code is a set of error-correction codes that can be used to detect and correct the errors that can occur when the data is moved or stored from the sender to the receiver.

Redundant bits

Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer. The number of redundant bits can be calculated using the following formula:

$$2^r >= m+r+1$$

Where r=redundant bit, m=data bit.

Parity bits

A parity bit is a bit appended to a data of binary bits to ensure that the total number of1's in the data is even or odd. Parity bits are used for error detection. There are two types of parity bits:

1. **Even parity bit:**

   In the case of even parity, for a given set of bits, the number of 1's is counted. If that count is odd, the parity bit value is set to 1, making the total count of occurrences of 1's an even number. If the total number of 1's in a given set of bits is already even, the parity bit's value is 0.

2. **Odd Parity bit**

In the case of odd parity, for a given set of bits, the number of 1’s is counted. If that count is even, the parity bit value is set to 1, making the total count of occurrences of 1’san odd number. If the total number of 1’s in a given set of bits is already odd, the paritybit’svalueis0.

General Algorithm of Hamming code

The Hamming Code is simply the use of extra parity bits to allow the identification of an error.

Step1: Take a data of arbitrary length. (Ex:1011001)

Step2: Write the bit positions starting from1in binary form (1, 10,11,100, etc.).

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	DI
1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001

Fig1.1BinaryassignmentforHammingcode

Step 3: All the bit positions that area power of 2 are marked as parity bits (1,2,4,8, etc.).

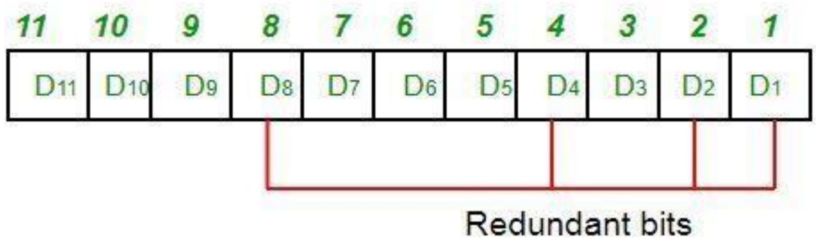


Fig 1.2Redundantbitpositionof Hamming Code

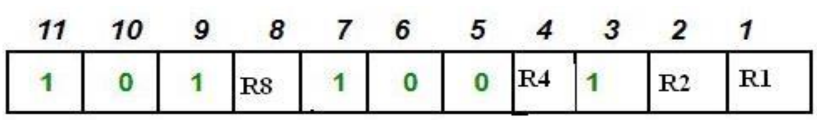


Fig1.3Hamming Code

Step4: All the other bit positions are marked as data bits.

Step5: Each data bit is included in a unique set of parity bits, as determined its bit position in binary form.

a. Paritybit1covers all the bits position whose binary representation includes a. 1 in the least significant position (1,3,5,7,9,11, etc.).

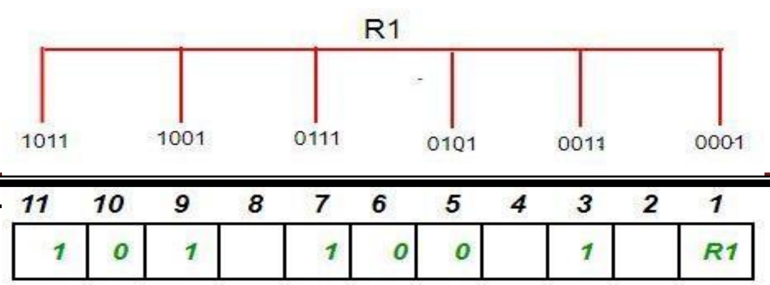


Fig1.4FindingR1paritybit

To find the redundant bit R1, we check for even parity. Since the total number of 1’s in allthebitpositionscorrespondingtoR1isanevennumberthevalueofR1(paritybit’svalue)=0.

b. Parity bit 2 covers all the bits positions whose binary representation includes a 1 in the second position from the least significant bit (2, 3, 6, 7, 10, 11, etc.). Similarly, R4 and R8 is calculated.

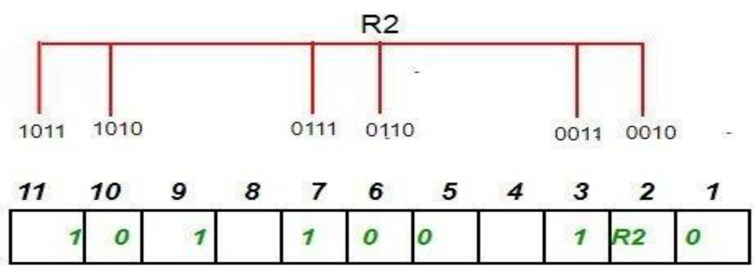


Fig1.5 Finding R2 parity bit

c. Parity bit 4 covers all the bits positions whose binary representation includes a 1 in the third position from the least significant bit (4–7, 12–15, 20–23, etc.).

d. Parity bit 8 covers all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit bits (8–15, 24–31, 40–47, etc.).

e. In general each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.

Step 6: Since we check for even parity set a parity bit to 1 if the total number of ones in the positions it checks is odd.

Step 7: Set a parity bit to 0 if the total number of ones in the positions it checks is even.

11	10	9	8	7	6	5	4	3	2	1
1	0	1	0	1	0	0	1	1	1	0

Fig1.6Generated code

PROCEDURE:

To create an IDE project:

1. Start Net Beans IDE.

2. In the IDE, choose File>New Project, as shown in the figure below.

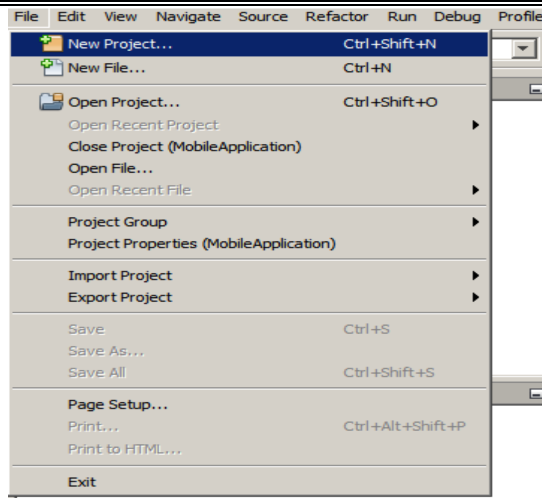


Fig1.7 Open a new file

3. In the NewProjectwizard, expand the Java category and select Java Application as shown in the figure below. Then click Next.

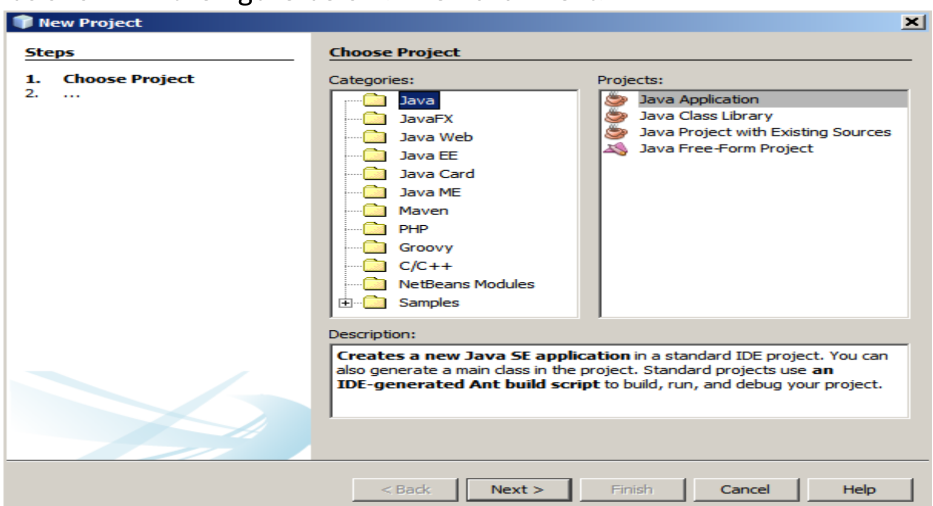


Fig1.8NewProjectwizard

4. In the Name and Location page of the wizard, do the following (as shown in the figure below):

- In the Project Name field, type HelloWorldApp.
- Leave the Use Dedicated Folder for Storing Libraries check box unselected.
- In the Create Main Classfield, typehelloworldapp. HelloWorldApp.

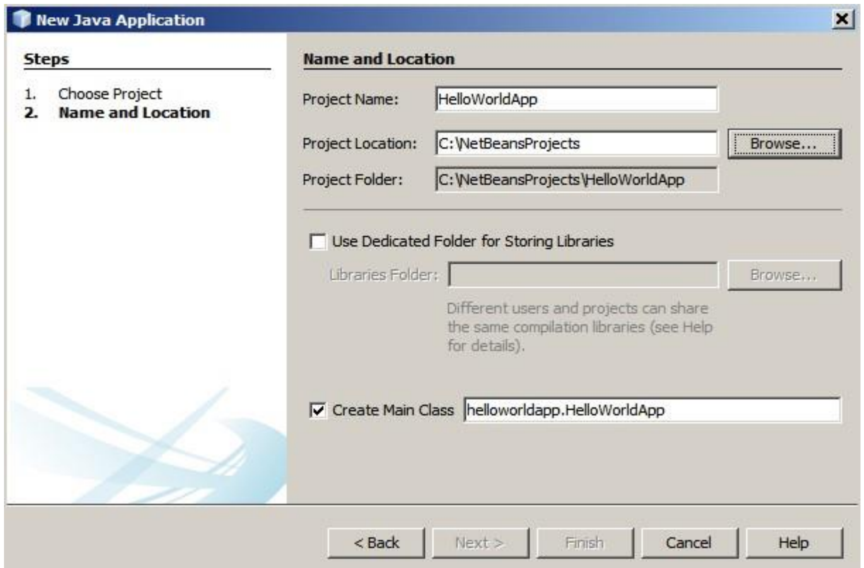


Fig1.9 Name and Location page

5. Click Finish.
6. The project is created and opened in the IDE. You should see the following components:
   - The Projects window, which contains a tree view of the components of the project, including source files, libraries that your code depends on, and soon.

   - The Source Editor window with a file called Hello World App open.
   - The Navigator window, which you can use to quickly navigate between elements within the selected class.

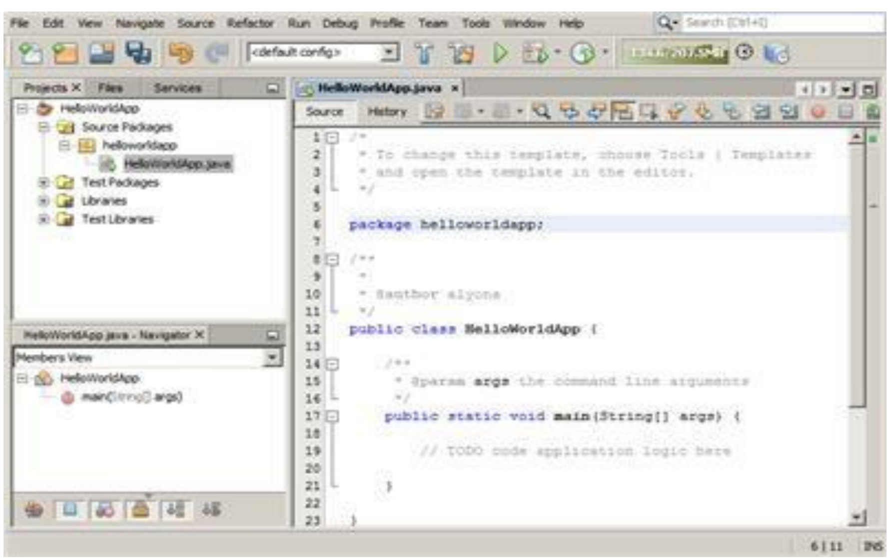


Fig1.10 Navigator window

7. Adding code to the generated source file.

The IDE has created a skeleton main class. Add the "HelloWorld!" message to the skeleton code by replacing the line:

//TO DO code application logic here with the line:

System.out.println(“HelloWorld!”);

8. Save the change by choosing File>Save.
   The file should look something like the following code sample.

```
packagehelloworldapp;

/**
  *
  *@author<yourname>
 */
public class HelloWorldApp{

    /**
     *@paramargsthecommandlinearguments
     */
    public static void main (String[] args)
            {System.out.println("HelloWorld!");
    }

}
```

9. Compiling and Running the Program. When you save a Java source file, the IDE automatically compiles it. In the Properties window, choose the Compiling tab. The Compile on Save checkbox is right at the top. Note that in the Project Properties window you can configure numerous settings for your project: project libraries, packaging, building, running, etc.

10. To run the program: choose Run> run project.

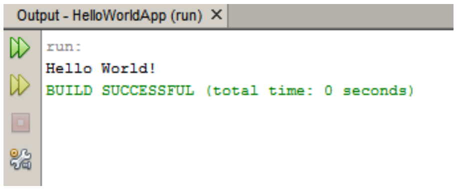


Fig1.11 Output window

PROGRAM:

```
importjava.io. *;

import java. util.*;

class Hamming Code

{

publicstaticvoidmain (String arg [])

{

Scannersc=new Scanner (System.in); System.out.println("Enter the 7-bit data

code");

Int d [] = new int [7];

For (int I =0; I <7; I ++)

{

d [I]= sc.next Int();

}

Int p []=new int[4];

P [0]=d[0]^d[1]^d[3]^d[4]^d[6];

P[1]=d[0]^d[2]^d[3]^d[5]^d[6];

p[2] =d[1]^d[2]^d[3];

p[3]=d[4]^d[5]^d[6];

int c[]=new int[11];

System.out.println("Complete Code Word is");

c[0] =p[0];

c[1]=p[1];

c[2]=d[0];
```

```
c[3]=p[2];
c[4]=d[1];
c[5]=d[2];
c[6]=d[3];
c[7]=p[3];
c[8]=d[4];
c[9]=d[5];
c[10]=d[6];
for (inti=0; i<11; i++)
{
System.out.print(c[i]+"");
}
System.out.println();
System.out.println("Enter the Received code word");
int r[]= new int[11];
for (int i=0; i<11; i++)
{
r[i]=sc.nextInt();
}
int pr [] = new
int[4];int
rd[]=new
int[7];pr[0]=r[0]
;
pr[1]=r[1];
rd[0]=r[2];
```

```
pr[2]=r[3];
rd[1]=r[4];
rd[2]=r[5];
rd[3]=r[6];
pr[3]=r[7];
rd[4]=r[8];
rd[5]=r[9];
rd[6]=r[10];
int s[] = new int[4]; s[0]=pr[0]^rd[0]^rd[1]^rd[3]^rd[4]^rd[6];
s[1]=pr[1]^rd[0]^rd[2]^rd[3]^rd[5]^rd[6];
s[2]=pr[2]^rd[1]^rd[2]^rd[3];
s[3]=pr[3]^rd[4]^rd[5]^rd[6];
int dec=(s[0]*1)+(s[1]*2)+(s[2]*4)+(s[3]*8);
if(dec==0)
System.out.println("Noerror");
else
{
System.out.println("Error is at "+dec);
if(r[dec-1] ==0)
r[dec-1]=1;
else
r[dec-1] =0;
}
System.out.println("Corrected code word is:");
for (int i=0; i<11; i++)
System.out.print(r[i]+"");
```

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

SAMPLEOUTPUT:

Inputdata:1001101Generatedcode:10011100101

Enter the position to alter to check for error detection at the receiver end: 5

Sent code is:10010100101.

Error is at location 5

Corrected code is : 10011100101

Original data sent:1001101

RESULT:

Thus, the error detection and correction is successfully processed for a message transmitted and received between the sender and receiver.

<u>SAMPLEVIVAQUESTIONS</u>:

1. What do you mean by redundancy?
2. Define parity check.
3. What is Hamming code?
4. How many errors can Hamming code detect?
5. Why Hamming codeiscalled7,4 code?
6. What is meant by Checksum?
7. Why Hamming code is used?

EXP. NO:2a	IMPLEMENTATION OF STOP AND WAIT PROTOCOL
DATE:	

AIM:

To implement the stop and wait protocol using java programming and observe the communication between the server and the client.

REQUIREMENTS:

Operating System	:	Windows NT/2000/XP
Programming Tool	:	NetBeans IDE

THEORY:

After having passed a packet to its network layer, the receiver sends a little dummy frame back to the sender which, in effect, gives the sender permission to transmit the next frame. After having sent a frame, the sender is required by the protocol to bide its time until the little dummy (i.e., acknowledgement) frame arrives. This delay is a simple example of a flow control protocol.

Protocols in which the sender sends one frame and then waits for an acknowledgement before proceeding are called **stop-and-wait**. Protocols in which the sender waits for a positive acknowledgement before advancing to the next data item are often called **ARQ** (**Automatic Repeater Quest**) or **PAR** (**Positive Acknowledgement with Retransmission**).

After transmitting a frame, the sender starts the timer running. If it was already running, it will be reset to allow another full timer interval. The interval should be chosen to allow enough time for the frame to get to the receiver, for the receiver to process it in the worst case, and for the acknowledgement frame to propagate back to the sender. Only when that interval has elapsed is it safe to assume that either the transmitted frame or its acknowledgement has been lost, and to send a duplicate. If the timeout interval is set too short, the sender will transmit unnecessary frames. While these extra frames will not affect the correctness of the protocol, they will hurt performance.

After transmitting a frame and starting the timer, the sender waits for something exciting to happen. Only three possibilities exist: an acknowledgement frame arrives undamaged, a damaged acknowledgement frame staggers in, or the timer expires. If a valid acknowledgement comes in, the sender fetches the next packet from its network layer and puts it in the buffer, overwriting the previous packet. It also advances the sequence number. If a damaged frame arrives or the timer expires, neither the buffer nor the sequence number is changed so that a duplicate can be sent. In all cases, the contents of the buffer (either the nextpacket or a duplicate) are then sent.

When a valid frame arrives at the receiver, its sequence number is checked to see if it is a duplicate. If not, it is accepted, passed to the network layer, and an acknowledgement is generated. Duplicates and damaged frames are not passed to the network layer, but they do cause the last correctly received frame to be acknowledged to signal the sender to advance to the next frame or retransmit a damaged frame.

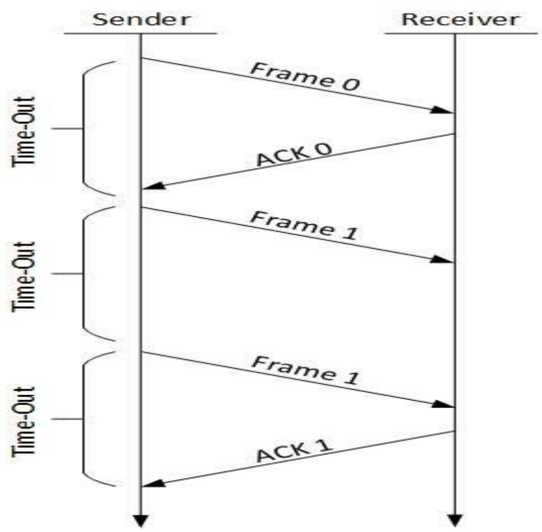


Fig2.1STOP AND WAIT PROTOCOL

ALGORITHM:

Server:

Step1: Start.
Step2: Include the necessary header files.
Step3: Create a class’s ender and declare, initialize the necessary files.
Step4: Establish a connection between server and client.
Step5: Get the frames to be sent from the user.
Step6: After the data is sent, wait for acknowledgement from client.
Step7: If acknowledgement is received, send the next frame. Else resend frames.
Step8: Exit after all the frames are sent.
Step9: Stop.

Client:

Step1: Start.
Step2: Include the necessary header files.
Step3: After the connection is established, receive the frames from the server.
Step4: After receiving the frames, send acknowledgement to the server.
Step 5: Print the data received s**tep6:** Stop.

PROGRAM:
Run the sender and then the receiver

//SENDERPROGRAM

```
importjava.io. *;

importjava.net.*;

import java.util.Scanner;

public class Stop Wait Sender {

    publicstaticvoidmain (String [] args) throws IO Exception {

    String packet, ack, str, msg="";

    int n, i=0, sequence=0;

     Server Socket ss=new Server Socket(2004);

    Sockets =ss.accept();

    Scanner br= newScanner(System.in);

    Scanner msg from Receiver=newScanner(s. getInputStream());

    Print Stream dos = new

    PrintStream(s.getOutputStream());System.out.println("WaitingforConnection..");

    str=msgfromReceiver.nextLine();

    System.out.println("reciver> "+str);

    System.out.println("Enter the data to send .      ");

    packet=br.nextLine();n=packet.length();

    while(i<n+1)

    {
    if(i<n){

            msg=String.valueOf(sequence);

            msg=msg.concat(packet.substring(i,i+1));

        }
```

```
        elseif(i==n){

            msg="end";

            dos.println(msg);

            break;

        }

        dos.println(msg);

        /*

            Changing sequence numbers incedatasent

        */

        sequence=(sequence==0)?1:0; System.out.println("data sent>

        "+msg);ack=msgfromReceiver.nextLine();System.out.println("waitingf

        orack. ...............................................\n");

        if(ack.equals(String.valueOf(sequence))){i++;

            System.out.println("receiver    >"+"packet received\n");

        }

        else {      /*whenever ack lost or wrong ack we change the sequence
        number*/System.out.println("Timeout resending data    \n");
            sequence=(sequence==0)? 1:0;

        }

    }

    System.out.println("All data sent. exiting.");

    s.close();

    }

}
```

//RECEIVERPROGRAM

```
importjava.o.*;

importjava.net. *;
import java. util.Scanner;classStopWaitReceiver{
    publicstaticvoidmain (String[] args)throwsIOException{ int i=0;
    int sequence=0;
    Sockets=new Socket("localhost",2004);
    PrintStream dos = new PrintStream(s.getOutputStream());
    ScannermsgfromSender=newScanner(s.getInputStream());
    System.out.println("Connection established:");
    dos. println("Connected");
    while(true)
    {
    packet=msgfromSender.nextLine();if(!packet.equals("end"))
    {
    if(Integer.valueOf(packet.substring(0,1))==sequence){data+=packet.substring(1);seq
    uence=(sequence==0)?1:0;System.out.println("\n\nreceiver>"+packet);
    }
    else
    {
    System.out.println("\n\receiver>"+packet "duplicate data");

    }
    dos. println (String.valueOf(sequence));
    i++;
```

```
    System.out.println("Data recived="+data);
    }
    else
    {
    dos.println("connection ended...");
    s.close();
    break;
    }
    }
    }
}
```

Sample output:

sender:	**Receiver:**
waiting for connection….	connection established
receiver connected	receiver >01
Enter the data to send :1234	data received =1
data sent>01	receiver>1 2
waiting for ack…	data received=12
receiver >packet received	receiver>0 3
data sent>12	data received=123
waiting for ack…	receiver>1 4
receiver>packet received	data received=1234
data sent>03waiting for ack…	
receiver>packet received data sent>14	
waiting for ack…receiver>packet received All data sent. exiting	

RESULT:

Thus the stop and wait protocol has been implemented and the communication phases between the server and the client verified successfully.

SAMPLE VIVA QUESTIONS:

1. What do you mean by flow control?
2. What do you mean by error control?
3. Define stop and wait ARQ.
4. What do you mean by pipelining, is there any pipelining in error control?
5. List The Layers Of OSI?
6. What are the responsibilities of Data Link Layer?
7. What are the responsibilities of Network Layer?
8. What are the responsibilities of Transport Layer?
9. Routers work at Which OSI Layer?
10. What is the role of the LLC sublayer in Datalink Layer?
11. What is the function of the application layer in Networking?

EXP. NO:2.b	IMPLEMENTATION OF SLIDING WINDOW PROTOCOL
DATE:	

AIM:

To implement the sliding window protocol using java programming and observe the communication between the server and the client.

REQUIREMENTS:

Operating System :Windows NT/2000/XP

Programming Tool : NetBeans IDE

THEORY:

In the previous protocols, data frames were transmitted in one direction only. In most practical situations, there is a need to transmit data in both directions. One way of achieving full-duplex data transmission is to run two instances of one of the previous protocols, each using a separate link for simplex data traffic (in different directions). Each link is then comprised of a “forward’’ channel (for data) and a ‘’reverse’’ channel (for acknowledgements). In both cases the capacity of the reverse channel is almost entirely wasted.

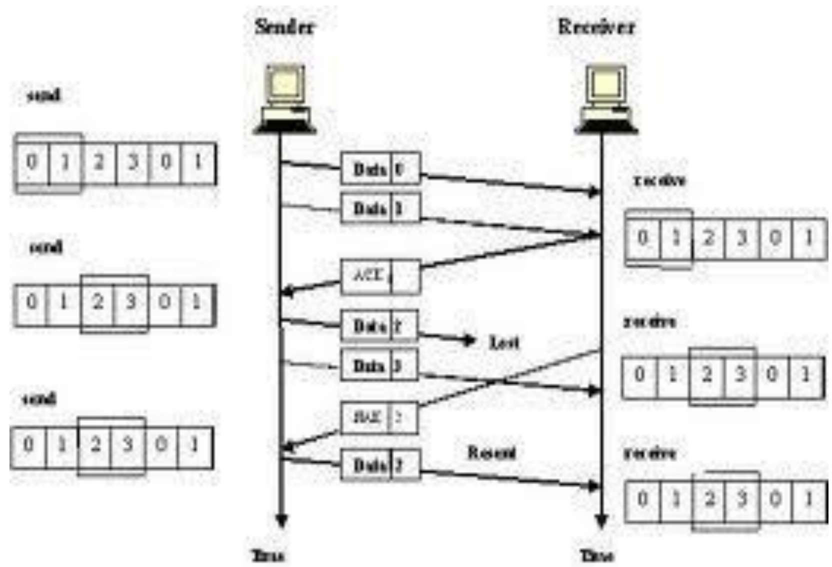


Fig3.1 **SLIDINGWINDOW PROTOCOL**

Although interleaving data and control frames on the same link is a big improvement over having two separate physical links, yet another improvement is possible. When a data frame arrives, instead of immediately sending a separate control frame, the receiver restrains itself and waits until the network layer passes it the next packet. The acknowledgement is attached to the outgoing data frame (using the *ack* field in the frame header).In effect, the acknowledgement gets a free ride on then next outgoing data frame. The technique of temporarily delaying outgoing acknowledgements so that they can be hooked onto the next out going data frame is known as **piggybacking**.

The principal advantage of using piggybacking over having distinct acknowledgement frames is a better use of the available channel bandwidth. The *ack* field in the frame header costs only a few bits, whereas a separate frame would need a header, the acknowledgement, and a checksum. In addition, fewer frames sent generally means a lighter processing load at the receiver .In then Ext protocol to be examined, the piggyback field consists only1 bit in the frame header. It rarely costs more than a few bits. However, piggybacking introduces a complication not present with separate acknowledgements. How long should the data link layer wait for a packet onto which to piggyback the acknowledgement? If the data link layer waits longer than the sender's timeout period, the frame will be retransmitted, defeating the whole purpose of having acknowledgements.

ALGORITHM:

Server:

Step1: Start.
Step2: Include the necessary header files.
Step3: Create a class "slide sender".
Step4: Initialize the string buffer value.
Step5: Indo-while statement, get the number of frames from the user.
Step6: Get then number of messages from the user.
Step7: Within main(),call the functions to implement sliding window protocol.
Step8: Stop.

Client:

Step1: Start.
Step2: Include the necessary header files.
Step3: Create a class "slide receiver".
Step4: Receive the frames from the sender and send the acknowledgement.
Step5: Stop.

PROCEDURE:

To create an IDE project:
1. Start NetBeans IDE.
2. In the IDE, choose File>New Project, asshowninthefigurebelow.
3. In the NewProjectwizard, expand the Java category and select Java Application. Then click Next.
4. In the Name and Location page of the wizard, do the following (as shown in the figure below):

- In the Project Name field, type HelloWorldApp.
- Leave the Use Dedicated Folder for Storing Libraries checkbox unselected.
- In the Create Main Class field, type helloworldapp. HelloWorldApp.

5. Click Finish.
6. The project is created and opened in the IDE. You should see the following components:
   - The Projects window, which contains at review of the components of the project, including source files, libraries that your code depends on, and so on.
   - The Source Editor window with a file called HelloWorldApp open.
   - The Navigator window, which you can use to quickly navigate between elements within the selected class.
7. Adding code to the generated source file.

8. Compiling and Running the Program. In the Properties window, choose the Compiling tab. The Compile on Save checkbox is right at the top. Note that in the Project Properties window you can configure numerous settings for your project: project libraries, packaging, building, running, etc.

9. To run the program: choose Run> run project.

PROGRAM:
Run the sender and then the receiver

//SENDERPROGRAM

```
importjava.io. *;

import java.net. *;

import java. util. Scanner;

public class SW sender

{

publicstaticvoid main (String args []) throws IO Exception

{

Int SWS=8;// Sender window size

int LAR=0;//Sequence number of the last

acknowledgementreceivedintLFS=0;//Sequencenumberofthelastframesent

Server Socket ss=new Server Socket (500); Sockets=ss.accept();
```

```
System.out.println("Type your Message...");
Scanner msg from Receiver=new Scanner(s.getInputStream());
PrintStream p=new PrintStream(s.getOutputStream());while(true)
{
Int i=0, j=0;
Scanner msg to Receiver=newScanner (System.in);
String t=msg to Receiver. nextLine();
if(t.trim().to LowerCase(). equals("quit"))
{
p.println(t);
System. Exit(0);
}
Char c[]=newchar [100];
c=t.toCharArray();
Int sent=1;
while(i<t.length())
{
while(i<t.length()&&i<SWS*sent)
{
If (LFS-LAR<=SWS)
{
p.println(c[i]);
LFS++;
System.out.println("sent="+c[i++]+"Successfully");
}
}
while(j<t.length()&&j<SWS*sent)
```

```
{
Stringt1=msgfromReceiver.nextLine();
System.out.println(t1);
j++;
LAR++;
}
sent++;
System.out.println();
}
}
}
}
```

//RECEIVERPROGRAM

```
importjava.io.*;
importjava.net.*;
import java.util.Scanner;
publicclassSWreceiver
{
publicstaticvoidmain (Stringargs []) throwsIOException
{
intRWS=8;//Receiverwindowsize
int LAF=0;//Sequencenumberoflargest acceptableframeint
LFR=0; //Sequence number of last frame
receivedInetAddressobj=InetAddress.getLocalHost();
```

```
Sockets=newSocket(obj,500);


Scanner msgfromSender = new Scanner(s.getInputStream());
PrintStream p=new PrintStream(s.getOutputStream());while(true)
{
inti=0;
while(i<RWS)
{
String t;
if(LAF-LFR<=RWS)
{
t=msgfromSender.nextLine();
if(t.equals("quit"))System.exit(0);
System.out.println("received"+t+"Successfully");
LFR++;
p.println("Acknowledgement for "+t);
LAF++;
}
i++;
}
System.out.println();
}
}
}
```

RESULT:

Thus the sliding window protocol has been implemented and the communication phases between the server and the client verified successfully.

SAMPLE VIVA QUESTIONS:

1. Define Goback NARQ.
2. Define selective repeat ARQ.
3. In which layer term “frames” is used?
4. In which layer term “packets” is used?
5. In which layer term “segments” is used?
6. Give some example for protocols work at application layer?
7. What is the purpose of the datalink?
8. Which layer provides logical addressing that routers will use for path determination?
9. Which layer specifies voltage, wire speed, and pinout cables and moves bits between devices?
10. Which layer combines bits into bytes and bytes into frames, uses MAC addressing, and provide error detection?
11. What is Piggybacking?

EXP. NO:3	SOCKET PROGRAMMING CLIENT-SERVER MODEL
DATE:	

AIM:

To implement the socket addressing concept using java programming .

REQUIREMENTS:

Operating System	:	Windows NT/2000/XP
Programming Tool	:	NetBeans IDE

THEORY:

Socket programming:

Sockets provide the communication mechanism between two computers using TCP. A client program creates a socket on its end of the communication and attempts to connect that socket to a server.

The following steps occur when establishing a TCP connection between two computers using sockets:

- The server instantiates a Server Socket object, denoting which port number communication is to occur on.
- The server invokes the accept () method of the Server Socket class. This method waits until a client connects to the server on the given port.
- After the server is waiting, a client instantiates a Socket object, specifying the server name and port number to connect to.
- The constructor of the Socket class attempts to connect the client to the specified server and port number. If communication is established, the client now has a Socket object capable of communicating with the server.
- On the server side, the accept () method returns a reference to a new socket on the server that is connected to the client's socket.

After the connections are established, communication can occur using I/O streams. Each socket has both an Output Stream and an Input Stream. The client's Output Stream is connected to the server's Input Stream, and the client's Input Stream is connected to the server's Output Stream.

Server Socket ClassMethods:

The **java.net.ServerSocket** class is used by server applications to obtain a port and listen for client requests.

The Server Socket class has four constructors:

publicServerSocket(intport)throwsIOException	Attempts to create a server socket bound to the specified port. An exception occurs if the port is already bound by another application.
publicServerSocket(intport,intbacklog)throwsIOException	Like the previous constructor, the back log parameter specified show many incoming clients to store in a wait queue.
public ServerSocket(int port,intbacklog,InetAddressaddress)throwsIOException	Like the previous constructor, the Inet Address parameter specifies the local IP address to bind to.
publicServerSocket()throwsIOException	Creates an unbound server socket. When using this constructor, use the bind () method when you are ready to Bind the server socket

Here are some of the common methods of the Server Socket class:

publicintgetLocalPort()	Returns the port that the server socket is listening on.
publicSocketaccept()throwsIOException	Waits for an incoming client. This method blocks until either a client connects to the server on the specified port or the socket times out
public void setSoTimeout(inttimeout)	Sets the time-out value for how long the server socket waits for a client during the accept ().
public voidbind(SocketAddresshost,intbacklog)	Binds the socket to the specified server and port in the Socket Address object.

Socket ClassMethods:

The **java.net. Socket class** represents the socket that both the client and server use to communicate with each other. The client obtains a Socket object by instantiating one, whereas the server obtains a Socket object from the return value of the accept () method.

The Socket class has five constructors that a client uses to connect to a server:

publicSocket(Stringhost,intport)throwsUnknownHostException,IOException.	This method attempts to connect to the specified server at the specified port.
publicSocket(InetAddress host,intport)throwsIOException	This method is identical to the previous constructor, except that the host is denoted by an Inet Address object.
publicSocket(Stringhost,intport,InetAddresslocalAddress, int localPort)throwsIOException.	Connects to the specified host and port, creating a socket on the local host at the specified address and port.
public Socket(InetAddresshost, int port, InetAddresslocalAddress,intlocalPort)throwsIOException.	This method is identical to the previous constructor, except that the host is denoted by an Inet Address object instead of a String
publicSocket()	Creates an unconnected socket. Use the connect () method to connect this socket to a server.

Some method so of interest in the Socket class are listed here.

public voidconnect(SocketAddresshost,int timeout) throwsIOException	This method connects the socket to the specified host. This method is needed only when you instantiated the Socket using the no-argument constructor.
publicInetAddressgetInetAddress()	This method returns the address of the other computer that this socket is connected to.
publicint getPort()	Returns the port the socket is bound to on the remote machine.
publicintgetLocalPort()	Returns the port the socket is bound to on the local machine.
public SocketAddressgetRemoteSocketAddress()	Returns the address of the remote socket.

public InputStreamgetInputStream( )throwsIOException	Returnstheinputstreamofthesocket.Theinputstreamisconnectedtotheoutputstreamof theremote socket.
public OutputStreamgetOutputStream()throwsIOException	Returns the output stream of the socket. The output stream is connected to the input stream of the remote socket
publicvoidclose()throwsIOException	Closes the socket, which makes this Socket object no longer capable of connecting a gain to any server

Process-to-process delivery needs two identifiers, IP address and the port number, at each end to make a connection. The combination of an IP address and a port number is called a socket address. The client socket address defines the client process uniquely just as the server socket address defines the server process uniquely. A transport layer protocol needs a pair of socket addresses: the client socket address and the server socket address. These four pieces of information are part of the IP header and the transport layer protocol header. The IP header contains the IP addresses; the UDP or TCP header contains the port numbers.

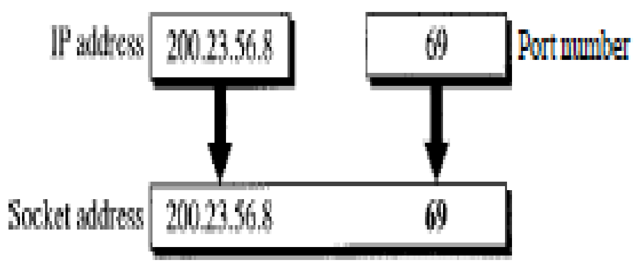


Fig4.1SOCKETPROGRAMMING

The SOCKET primitive creates a new endpoint and allocates table space for it within the transport entity. The parameters of the call specify the addressing format to be used, the type of service desired (e.g., reliable byte stream), and the protocol. A successful SOCKET call returns an ordinary file descript or for use in succeeding calls, the same way an OPEN call on a file does.

Newly created sockets do not have network addresses. These are assigned using the BIND primitive. Once a server has bound an address to a socket, remote clients can connect to it. Thereas on for not having the SOCKET call create an address directly is that some processes care about their addresses (e.g., they have been using the same address for years and everyone knows this address), whereas others do not.

Next comes the LISTEN call, which allocates space to queue incoming calls for the case that several clients try to connect at the same time. In contrast to LISTEN in our first example, in the socket model LISTEN is not a blocking call. To block waiting for an incoming connection, the server executes an ACCEPT primitive. When a segment asking for a connection arrives, the transport entity creates a new socket with the same properties as the original one and returns a file descriptor for it. The server can then fork off a process or thread to handle the connection on the new socket and go back to waiting for the next connection on the original socket. ACCEPT returns a file descriptor, which can be used for reading and writing in the standard way, the same as for files. Now let us look at the client side. Here, too, a socket must first be created using the SOCKET primitive, but BIND is not required since the address used does not matter to the server. The CONNECT primitive blocks the caller and actively starts the connection process. When it completes (i.e., when the appropriate segment is received from the server), the client process is unblocked and the connection is established. Both sides can now use SEND and RECEIVE to transmit and receive data over the full-duplex connection. The standard UNIX READ and WRITE system calls can also be used if none of the special options of SEND and RECEIVE are required.

Connection release with sockets is symmetric. When both sides have executed a CLOSE primitive, the connection is released. Sockets have proved tremendously popular and are the defector standard for abstracting transport services to applications. The socket API is often used with the TCP protocol to provide a connection-oriented service called a **reliable byte stream**, which is simply the reliable bit pipe that we described. However, other protocols could be used to implement this service using the same API. It should all be the same to the transport service users.

The Strength of the socket API is that is can be used by an application for other transport services. For instance, sockets can be used with a connectionless transport service. In this case, CONNECT sets the address of the remote transport peer and SEND and RECEIVE send and receive datagram to and from the remote peer.

ALGORITHM:

SERVER:

Step1: Start.
Step2: Include the necessary header files.
Step3: Create a class "tcp date server".
Step4: Within main (), initialize the variables for server socket, print stream, Buffered reader and inet of type string.
Step5: Within try block, under the while loop, assign "cs=ss.accept" and initialized at print stream objects.
Step6: Display client systems IP address.
Step7: Stop.

CLIENT:

Step1: Start.
Step2: Include the necessary header files.
Step 3: Create a class "tcp dateclient".
Step 4: Read the system time and date from server using read () and write it to the standard output.
Step5: Stop.

PROCEDURE:

To create an IDE project:
1. Start Net Beans IDE.
2. In the IDE, choose File> new Project, as shown in the figure below.
3. In the New Project wizard, expand the Java category and select Java Application. Then click Next.
4. In the Name and Location page of the wizard, do the following (as shown in the figure below):

- In the ProjectNamefield, typeHelloWorldApp.
- Leave the Use Dedicated Folder for Storing Libraries check box unselected.
- In the Create Main Class field, type helloworldapp. HelloWorldApp.

5. Click Finish.
6. The project is created and opened in the IDE. You should see the following components:
   - The Projects window, which contains at review of the components of the project, including source files, libraries that your code depends on, and so on.
   - The Source Editor window with a file called HelloWorldApp open.
   - The Navigator window, which you can use to quickly navigate between elements within the selected class.
7. Adding code to the generated source file.

8. Compiling and Running the Program. In the Properties window, choose the Compiling tab. The Compile on Save checkbox is right at the top. Note that in the Project Properties window you can configure numerous settings for your project: project libraries, packaging, building, running, etc.

9. To run the program: choose Run>run project.

PROGRAM:
Run the server and then the client

//SERVERPROGRAM

```
importjava.io.*;

importjava.net.*;

importjava.util.Scanner;

classtcp1server

{
```

```
Public staticvoid main(String a[])throwsIOException

{

ServerSocketss=newServerSocket(8000);

//Opens the socketSockets=ss.accept();

Scannerin= newScanner(System.in);

PrintStreamdos=newPrintStream(s.getOutputStream());

while(true)

{

System.out.println("enter message to send:");

//Reads the input from the input device

String str=in.nextLine();

dos.println(str);

//checksforendofmessage

if(str.equals("end"))

{

//Closes the socket

s.close();

break;

}

}

}

}
```

//CLIENTPROGRAM

```
importjava.io.*;

importjava.net.*;

importjava.util.Scanner;

classtcp1client
```

```
{
publicstaticvoid main(Stringargs[])throwsIOException
{
Socket s=newSocket("localHost",8000);
//Used to get input from keyword
Scanner in=newScanner(s.getInputStream());
while(true)
{
//Reads the input from keybroad
String str=in.nextLine();
System.out.println("Message Received:"+str);
if(str.equals("end"))
{
//Close the socket
s.close();
break;
}
}
}
}
```

Sample output:

Server:

Enter message to send: HELLO

Client:

Message Received: HELLO

RESULT:

Thus a client server model have been programmed and verified by implementing socket concepts.

SAMPLEVIVAQUESTIONS:

1. What is MAC address?
2. Why IP address is required when we have MAC address?
3. What is meant by port?
4. What is a socket?
5. What is the port no of DNS and Telnet?

.EXP. NO:4.a	SOCKET PROGRAM FOR ECHO CLIENT AND ECHO SERVER COMMANDS
DATE:	

AIM:

To implement the echoclient-servermodel using java programming based on TCP sockets.

REQUIREMENTS:

Operating System : Windows NT/2000/XP
Programming Tool : NetBeans IDE

ALGORITHM:

SERVER:

Step1: Start.
Step2: Include the necessary header files.
Step3: Create a class “echoclient”.
Step4: Connect to the server o initialize request and assign any port number.
Step5: After the server sends message,read the data and write it to the server.
Step 6: Stop.

CLIENT:

Step1: Start.
Step2: Include the necessary header files.
Step3: Create a class “echoserver”.
Step4: Assign any port number as desired.
Step5: On client’s request, establish a connection using accept ().
Step6: Read the message from client and write the message. Close the connection.
Step7: Stop.

PROCEDURE:

To create an IDE project:
1. Start Net Beans IDE.
2. In the IDE, choose File>new Project, as shown in the figure below.
3. In the New Projectwizard, expand the Javacategory and select JavaApplication. Then click Next.
4. In the Name and Location page of the wizard, do the following (as shown in the figure below):

- In the ProjectName field, type HelloWorldApp.
- Leave the Use Dedicated Folder for Storing Libraries check box unselected.
- In there Main Class field, type hello world app. HelloWorldApp.

5. Click Finish.
6. The project is created and opened in the IDE. You should see the following components:
    - The Projects window, which contains at review of the components of the project, including source files, libraries that your code depends on, and so on.
    - The Source Editor window with a file called HelloWorld App open.
    - The Navigator window, which you can use to quickly navigate between elements within the selected class.
7. Adding code to the generated source file.
8. Compiling and Running the Program. In the Properties window, choose the Compiling tab. The Compile on Save checkbox is right at the top. Note that in the Project Properties window you can configure numerous settings for your project: project libraries, packaging, building, running, etc.
9. To run the program: choose Run>run project.

PROGRAM:
Run the server and then the client

//SERVERPROGRAM

```
Import java.io.*;

import java.net.*;

import java.util.Scanner;

public class echoserver

{

publicstaticvoid main(Stringargs[])throwsIOException

{

ServerSocket ss=new ServerSocket(500); Sockets=ss.accept();

System.out.println("Serverisready");

DataInputStreamdis=newDataInputStream(s.getInputStream());

ScannermsgfromClient=newScanner(s.getInputStream());
```

```
PrintStream p=new PrintStream(s.getOutputStream());while(true)
{
String t=msgfromClient.nextLine();
if(t==null)
break;
System.out.println(t);
p.println(t);
}
}
}
```

//CLIENTPROGRAM

```
importjava.io.*;
importjava.net.*;
import java.util.Scanner;
publicclassechoclient
{
public static void main(Stringargs[])throwsIOException
{
Inet Address obj=InetAddress.getLocalHost();Sockets=new Socket(obj,500);
Scannermsg from Server=newScanner(s.getInputStream());
PrintStreamp=new PrintStream(s.getOutputStream());
System.out.println("TYPE YOURMESSAGE TOSERVERANDTYPEQUITTOEXIT");
while(true)
{
Stringt=newScanner(System.in).nextLine();
if(t.equals("quit"))
```

```
{
p.close();
System.exit(0);
}
else
{
p.println(t);
t=msg from Server.nextLine();
System.out.println(t);
}
}
}
}
```

Sample output:

Client:

Type your message to server and type quit to exit12345
12345
quit
Build successfully.

Server:

Sever is ready.12345
Build successfully.

RESULT:

Thus the echo commands on a client server model have been successfully implemented using a socket program.

SAMPLEVIVAQUESTIONS:

1. What is server?
2. What is a client?
3. What is an Echo command?
4. What are the differences between TCP And UDP?

EXP. NO:4.B	SOCKET PROGRAM FOR PING AND TRACE-ROUTE COMMANDS
DATE:	

AIM:

To write a socket program for simulating the ping and trace-route commands and observe the ping information between the server and client.

REQUIREMENTS:

Operating System : Windows NT/2000/XP
Programming Tool : NetBeans IDE

ALGORITHM:

SERVER:

Step1: Start.
Step2: Include the necessary header files.
Step3: Create a class "ping server".
Step4: Initialize the server socket and assign a port number.
Step 5: Read the IP address and TTL of the server system.
Step6: Reply from the server is displayed.
Step7: Stop.

CLIENT:

Step1: Start.
Step2: Include the necessary header files.
Step3: Create a class "Ping client".
Step4: Initialize the client's socket and assign a port number.
Step5: Call function system current Millis ().
Step 6: Read the IP address from the server.
Step7: Stop.

PROCEDURE:

To create an IDE project:
1. Start Net Beans IDE.
2. In the IDE, choose File> New Project, as shown in the figure below.
3. In the New Project wizard, expand the Java category and select JavaApplication. Then click Next.
4. In the Name and Location page of the wizard, do the following (as shown in the figure below):

- In the Project Name field, type HelloWorldApp.
- Leave the Use Dedicated Folder for Storing Libraries checkbox

unselected.

- In the Create Main Class field, type hello worldapp. HelloWorldApp.

5. Click Finish.
6. The project is created and opened in the IDE. You should see the following components:

- The Projects window, which contains at reeview of the components of the project, including sourcefiles, libraries that your code depends on, and so on.
- The Source Editor window with a file called HelloWorldApp open.
- The Navigator window, which you can use to quickly navigate between elements within the selected class.

7. Adding code to the generated sourcefile.
8. Compiling and Running the Program. In the Properties window, choose the Compiling tab. The Compile on Save checkbox is right at the top. Note that in the Project Properties window you can configure numerous settings for your project: project libraries, packaging, building, running, etc.
9. To run the program: choose Run>run project.

PROGRAM:

```
importjava.io.*;

importjava.net.*;

import java.util.Scanner;

classpingserver

{

publicstaticvoidmain(Stringargs[])

{

try

{

String str;

System.out.print("EntertheIPAddresstobePing:");

Scanners1= newScanner(System.in);
```

```
Stringip=s1.nextLine();
RuntimeH=Runtime.getRuntime();
Processp=H.exec("ping "+ip);
Scanners2=newScanner(p.getInputStream());
while((str=s2.nextLine())!=null)
{
System.out.println(""+str);
}
}
Catch (Exceptione)
{
System.out.println(e.getMessage());
}
}
}
```

Note: This is exactly same as the Server - Client (Two way). Run the server and then theclient

//SERVERPROGRAM

```
importjava.io.*;
importjava.net.*;
importjava.lang.*;
importjava.util.Scanner;
classtcp2server
{
publicstaticvoid main(Stringa[])throwsIOException
{
ServerSocketss=newServerSocket(8000);
```

```
//Opens the socketSockets=ss.accept();
PrintStreamdos=newPrintStream(s.getOutputStream());ScannermsgtoSend
=newScanner(System.in);
ScannermsgfromClient=newScanner(s.getInputStream());while(true)
{
System.out.println("enterthemsgtosend:");
//Readstheinput
String str=msgtoSend.nextLine();dos.println(str);
//Checksforendofmessageif(str.equals("end"))
{
//Closesthesocketss.close();
break;
}
String str1=msgfromClient.nextLine();System.out.println("msg
received"+str1);if(str1.equals("end"))
{
ss.close();break;
}
}
}
}
```

//CLIENTPROGRAM

```
importjava.io.*;
importjava.net.*;
importjava.lang.*;
importjava.util.Scanner;
```

```
classtcp2client

{

publicstaticvoid main(Stringa[])throwsIOException

{

//Createsobjectforsocket

Sockets=newSocket("LocalHost",8000);


ScannermsgfromServer=newScanner(s.getInputStream());

ScannermsgtoSend=new Scanner(System.in);


PrintStreamdos=newPrintStream(s.getOutputStream());while(true)

{

//Reads the input from the input deviceString

str=msgfromServer.nextLine();

System.out.println("msgreceived:"+str);

//Checksforendofmessage

if(str.equals("end"))

{

//Closesthesocket

s.close();

break;

}

System.out.println("enter the msg to send:");

//Reads the message to send

String str1=msgtoSend.nextLine();

dos.println(str1);

//Checksforendofmessage
```

```
if(str1.equals("end"))
{
//Closesthesocket
s.close();
break;
}
}
}
}
```

Sample output:

Enter the IP address to be ping: 172.16.18.40 pinging 172.16.18.40 with 32 bytes of data.

Replyfrom172.16.18.40: bytes=32time<1ms TTL=64

Replyfrom172.16.18.40: bytes=32time=1ms TTL=64

Replyfrom172.16.18.40: bytes=32time<1ms TTL=64

Pingstatics for 172.16.18.40:

packets: sent=4, received =4, lost= 0 Approximate round trip times in milliseconds minimum=0ms, maximum=1ms, average=0ms No line found build successfully.

RESULT:

Thus the ping and trace-route commands on a client server model have been successfully implemented using a socket program.

EXP. NO:5	ENCRYPTION AND DECRYPTION
DATE:	

AIM:

To implement encryption and decryption process using the RC4algorithm.

REQUIREMENTS:

Operating System : Windows NT/2000/XP
Programming Tool : NetBeans IDE Encrypting/Decrypting Engine and the User Interaction Tool.

THEORY:

The messages to be encrypted, known as the **plaintext**, are transformed by a function that is parameterized by a **key**. The output of the encryption process, known as the **ciphertext**, is then transmitted, often by messenger or radio. We assume that the enemy, or **intruder**, hears and accurately copies down the complete ciphertext. However, unlike the intended recipient, he does not know what the decryption key is and so cannot decrypt the ciphertext easily. Sometimes the intruder can not only listen to the communication channel (passive intruder) but canal sore cord messages and play them back later, inject his own messages, or modify legitimate messages before they get to the receiver (active intruder). The art of breaking ciphers, known as **cryptanalysis**, and the art of devising them (cryptography) are collectively known as **cryptology**.

It will often be useful to have a notation for relating plaintext, ciphertext, and keys. We will use $C = EK(P)$ to mean that the encryption of the plaintext P using key K gives the ciphertext C. Similarly, $P = DK(C)$ represents the decryption of C to get the plaintext again. It then follows that $DK(EK(P))= P.$

This notation suggests that E and D are just mathematical functions, which they are. The only tricky part is that both are functions of two parameters, and we have written one of the parameters (the key) as a subscript, rather than as an argument, to distinguish it from the message. A fundamental rule of cryptography is that one must assume that the crypt analyst knows the methods used for encryption and decryption. In other words, the cryptanalyst knows how the encryption method, E, and decryption, $D.$

From the cryptanalyst's point of view, the cryptanalysis problem has three principal variations. When he has a quantity of ciphertext and no plaintext, he is confronted with the **ciphertext-only** problem. The cryptograms that appear in the puzzle section of newspapers pose this kind of problem. When the cryptanalyst has some matched ciphertext and plaintext, the problem is called the **known plaintext** problem. Finally, when the cryptanalyst has the ability to encrypt pieces of plaintext of his own choosing, we have the **chosen plaintext** problem.

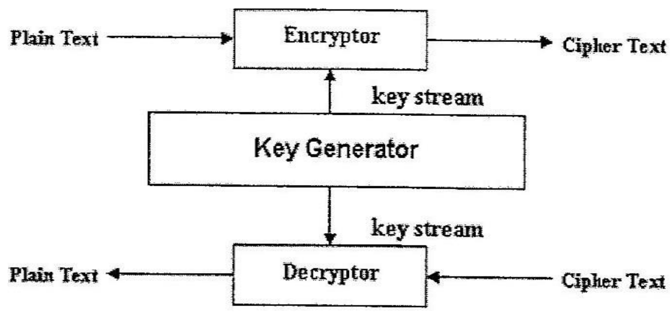


Fig5.1**ENCRYPTION AND DECRYPTION**

The SBox area displays the values from 0 to 255 filled in ascending order. Depending upon the key size (maximum size is restricted to five 8 bit characters resulting in 40 bits), the key is repeated as many times as necessary to fill out 256 entries which are displayed as K Box values.

The initial permutation of SBox involves starting with SBox[0] and going through to SBox[255], and for each SBox[i], swapping SBox[i] with another byte in SBox, according tothe scheme dictated by KBox[i]. The SBox still contains all the numbers from 0 through 255 but in permuted order according to the key value chosen since only swapping took place. After this the input key and the KBox is no longer used.

For each SBox[i], swap another byte in SBox according to the scheme generated by the current SBox configuration. In every cycle, the derived key ‘k’ will be used for encryption or decryption of each input text byte T. This process continues for every text byte and afterreaching256times, the process starts over again.

The *Steps* panel displays the steps carried out in each process of algorithm execution. The interface allows cut and paste such that the cipher text obtained after encryption can be cut and paste in the Text(T) field for trying out decryption. This takes care of the problem of keying in the ciphertext through keyboard.

ALGORITHM:

Step1: Initialize a substitution box SBox and the key box KBox.
Step2: Produce the initial permutation of SBox.
Step 3: Generate the stream of bits to encrypt or decrypt with the input text.
Step4: XOR the derived key ‘k’ and the input text T to get the cipher text

or original text.

PROCEDURE:

To create an IDE project:

1. Start NetBeans IDE.
2. In the IDE, choose File>New Project, as shown in the figure below.
3. In the NewProjectwizard, expand the Javacategory and select JavaApplication. Then click Next.
4. In the Name and Location page of the wizard, do the following (as shown in the figure below):
    - In the Project Name field, type HelloWorldApp.
    - Leave the Use Dedicated Folder for Storing Libraries checkbox unselected.
    - In the Create MainClassfield, type helloworldapp. HelloWorldApp.
5. Click Finish.
6. The project is created and opened in the IDE. You should see the following components:
    - The Projects window, which contains a tree view of the components of the project, including sourcefiles, libraries that your code depends on, and so on.
    - The Source Editor window with a file called HelloWorldApp open.
    - The Navigator window, which you can use to quickly navigate between elements within the selected class.
7. Adding code to the generated source file.
8. Compiling and Running the Program. In the Properties window, choose the Compiling tab. The Compile on Save checkbox is right at the top. Note that in the Project Properties window you can configure numerous settings for your project: project libraries, packaging, building, running, etc.
9. To run the program: choose Run>run project.

PROGRAM:

```
RC4_KSA(input:key_String )
{
Restrictkey_Stringtoamaximumof5characters(40bits)                Initialize
        state_table[0:255]=0
```

```
        Initialize key_table[0:255]=repeatingblocksofkeystring.j=0;
        For i= 0 to 255 do
        for j=(j+state_table[i] +key_table[i])            mod256Swap
        (State_table[i],state_table[j])
        End for
        Output: State_table [0:255]
    }
    RC4_PRNG (input:state_table[0:255]
    i=j=0;
    do (for length of plaintext)i=(i+1) mod256;
    j= (j+State_table [i]+State_table [j])
    Swap (State_table [i]+ State_table[j]
    mod 256RC4_Enc(input : plain_text [current_byte], state_table[t])
    End do;
```

Sample output:

Encrypted string b c d e f g
Decrypted string a b c de f

RESULT:

Thus the encryption and decryption processes have been implemented using the RC4algorithm.

<u>SAMPLEVIVAQUESTIONS</u>:

1. What is cryptography?
2. How do you classify cryptographic algorithms?
3. What is public key?
4. What is private key?
5. What are key, ciphertext and plaintext?

EXP. NO:6	STUDY OF NETWORK SIMULATOR NS 2.35
DATE:	

AIM:

To study the concept of Network Simulator (NS2.35)to develop a tcl (tool command language) script which simulate a simple topology.

REQUIREMENT

Operating System: REDHAT LINUX
Programming tool: Network Simulator (NS2.35)

THEORY

NS-2 is an event driven packet level network simulator developed as part of the VINT project (Virtual Internet Testbed). This was a collaboration of many institutes including UC Berkeley, AT&T, XEROX PARC and ETH. Version 1 of NS was developed in 1995 and with version 2 released in 1996. Version 2 included a scripting language called Object-oriented Tool Command Language Tcl (OTcl). It is an open source software package available for both Windows32 and Linux platforms.

NS-2 has many uses including:

- To evaluate the performance of existing network protocols.
- To evaluate new network protocols before use.
- To run large scale experiments not possible in real experiments.
- To simulate a variety of IP networks

To use NS-2, a user programs in the OTcl script language. An OTcl script will do the following.

- Initiates an event scheduler.
- Sets up the network topology using the network objects.
- Tells traffic sources when to start/stop transmitting packets through the event scheduler.

A user can add OTcl modules toNS-2 by writing a new object class in OTcl. These then have to be compiled together with the original source code.

Another major component of NS besides network objects is the event scheduler. An *event* in NS is a packet ID that is unique for a packet with scheduled time and the pointer to an object that handles the event.
The event scheduler inNS-2 performs the following tasks

- Organises the simulation timer.

- Fire sevents in the event queue.
- Invokes network components in the simulation.

Depending on the user's purpose for an OTcl simulation script, simulation results are stored a strace files, which can be loaded f or analysis by an external application:

1. AN Am tracefile (file.nam) for use with the NetworkAnimatorTool
2. A Tracefile (file.tr) for use with XGraphor Trace Graph.

HOW TO WRITE NS2 PROGRAM:

To write a 'template' that is use for all of the first Tcl scripts. You can write the Tcl scripts in any text editor and call this first example 'example1.tcl'.

1. First create a simulator object. This is done with the command

setns[newSimulator]

2. Now open a file for writing that is going to be used for the nam trace data.

setnf [openout.namw]
$nsnamtrace-all$nf

The first line opens the file 'out.nam' for writing and gives it the file handle 'nf'. The second line tells the simulator object that it is created above to write all simulation data that is going to be relevant for nam into this file.

3. The next step is to add a 'finish' procedure that closes the trace file and starts nam.

procfinish {}
{
global nsnf
$nsflush-trace

close$nf
exec namout.nam&exit0
}

4. The next line tells the simulator object to execute the 'finish' procedure after 5.0 seconds of simulation time.

$nsat5.0"finish"
$nsrun

The last line finally starts the simulation

5. Save the file now and try to run it with 'ns example1.tcl'. You are going to get an error message like 'nam: empty trace file out.nam' though, because until now we haven't defined any objects (nodes, links, etc.)or events.

2. Create node sandlink:

To define a very simple topology with two nodes that are connected by a link the following two lines define the two nodes.

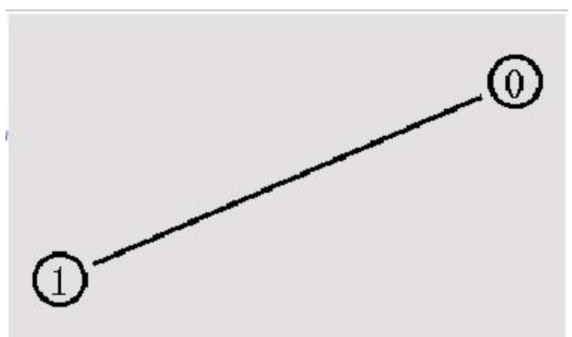
set Client1 [$ns node]set Router1 [$ns node]setn0[$nsnode]
setn1[$nsnode]

A new node object is created with the command' $nsnode'. The above code creates two nodes and assigns them to the handles 'n0'and'n1'.

The next line connects the two nodes.

$nsduplex-link$n0$n1 1Mb10msDropTail

This line tells the simulator object to connect the nodes n0 and n1 with a duplex link with the bandwidth 1Megabit, a delay of 10ms and a Drop Tail queue. Now save your file and start the script with 'ns example1.tcl'. nam will be started automatically and see an output that resembles the picture below



.

Fig6.1. Create nodes and links using NS2

To create orientation

$nsduplex-link-op $C1 $ROU1 orient down

$nsduplex-link-op$C2 $ROU1orient down-right

$nsduplex-link-op $C3 $ROU1 orient down-left

$nsduplex-link-op $C4 $ROU2orientup

$nsduplex-link-op $R1 $ROU1 orientup

$nsduplex-link-op$R2 $ROU1 orientup-right

$nsduplex-link-op$R3$ROU1orientup-left

$nsduplex-link-op $R4 $ROU3 orient down

$nsduplex-link-op$ROU1$ROU2orientdown-right

$nsduplex-link-op$ROU3$ROU2orientdown-right

To Configurenodes

$Endserver1shapehexagon
$Router1shapesquare
$Router2shapesquare

To describecolor

$nscolor1dodgerblue
$nscolor2red
$nscolor3cyan
$nscolor4 green
$nscolor5yellow
$nscolor 6black
$ns color7magenta
$nscolor 8gold
$nscolor9red

To set identification colors to router-links

$ns duplex-link-op$n1$n2colorcyan
$nsduplex-link-op $ROU1$ROU3 colorcyan

3. Sending data:

The next step is to send some data from node n0 to node n1. In ns, data is always being sent from one 'agent' to another. So the next step is to create an agent object that sends data from node n0, and another agent object that receives the data on node n1.
#Create a UDP agent and attach it to node n0setudp0[new Agent/UDP]

$nsattach-agent$n0$udp0
#Createa CBRtraffic sourceand attachit
toudp0setcbr0[newApplication/Traffic/CBR]
$cbr0setpacketSize_500
$cbr0set interval_ 0.005
$cbr0attach-agent$udp0

These lines create a UDP agent and attach it to the node n0, then attach a CBR traffic generated to the UDP agent. CBR stands for 'constant bit rate'. Line 7 and 8 should be self-explaining. The packet Size is being set to 500 bytes and a packet will be sent every 0.005 seconds(i.e.200packets per second).

The next lines create a Null agent which acts as traffic sink and attach it to node n1.

setnull0[newAgent/Null]
$nsattach-agent $n1 $null0

Now the two agents have to be connected with each other.

$nsconnect $udp0 $null0

And now we have to tell the CBR agent when to send data and when to stop sending. Note: It’s probably best to put the following lines just before the line'$nsat5.0 "finish"'.

$nsat 0.5"$cbr0 start"
$nsat4.5"$cbr0 stop"

Now you can save the file and start the simulation again. Click on the 'play' button in the nam window, and see that after 0.5 simulation seconds, node 0 starts sending data packets to node. You might want to slow nam down then with the 'Step' slider.

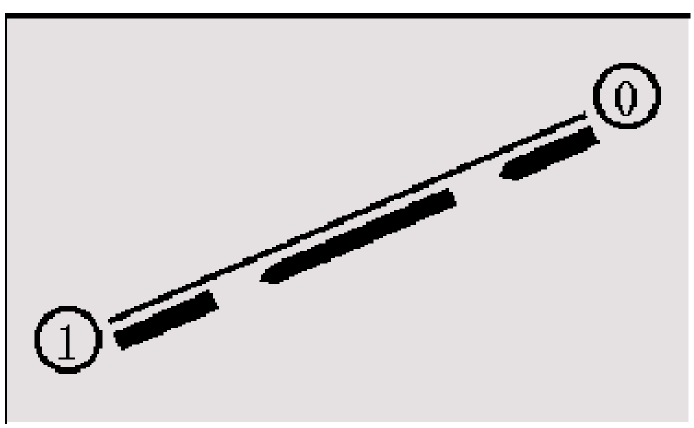


Fig 6.2FlowofPackets usingNS2

4. Creating TCP agent

Create TCP agent and attach it to the node

settcp0[newAgent/TCP]
$nsattach-agent$n0$tcp0
Setnull0[newAgent/TCPSink]
$nsattach-agent $n1 $null0

•Connect the agents
$nsconnect $tcp0$null0

PROCEDURE:

1. Open Redhat LinuxOS. Type the username and password.
2. Click Roothome→file→Openterminal.
3. Now vieditor window will open. Type ged it followed by filename with extension.tcl.

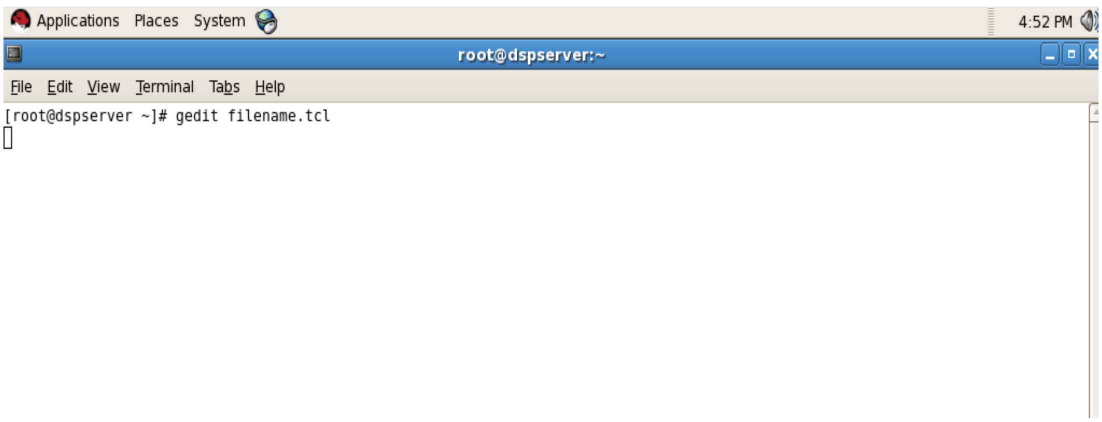


Fig 6.3vieditorwindow

4. Now the new ged it file will open. Type the program and save it.

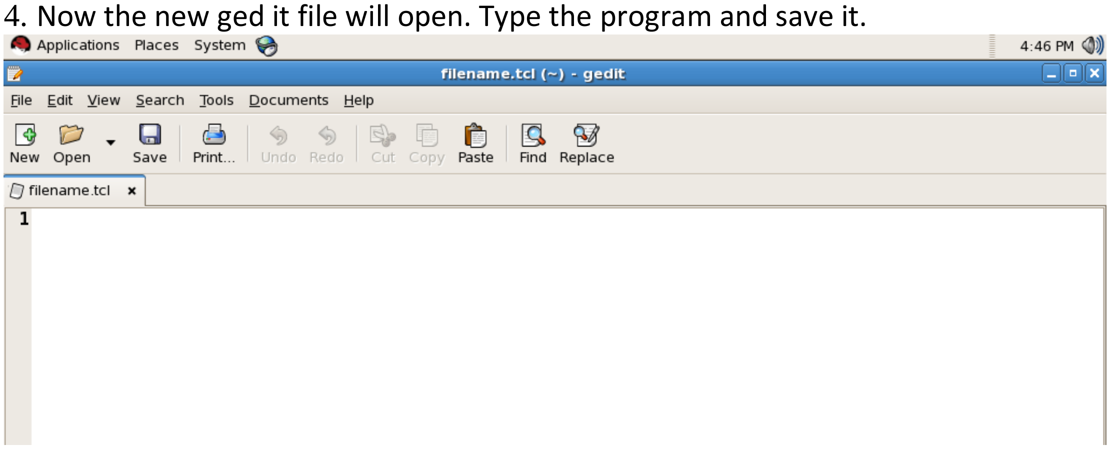


Fig6.4 ged it window

5. Run the saved file with the command “nsfilename.tcl” in vieditor window.

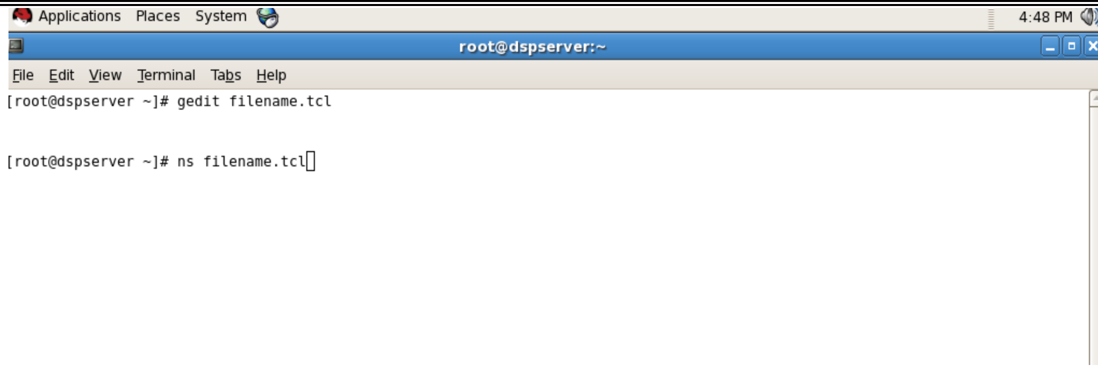


Fig 6.5vieditorwindow

6. If any errors, edit the min the vieditor and rerun it again.
7. The output is displayed in the Nam console.

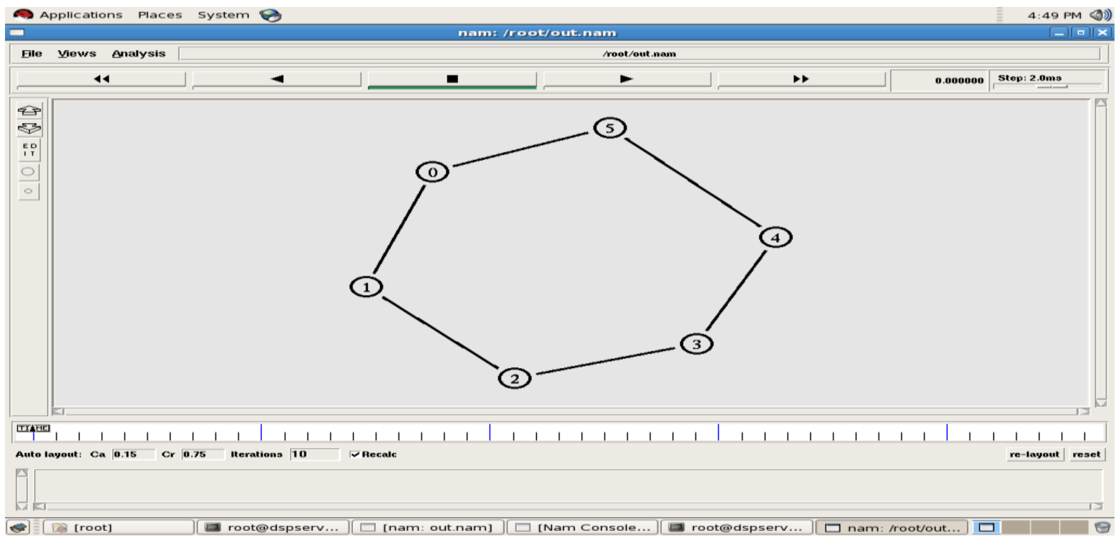


Fig6.6Animationwindow

8. Click on the play button and view the flow of packets.

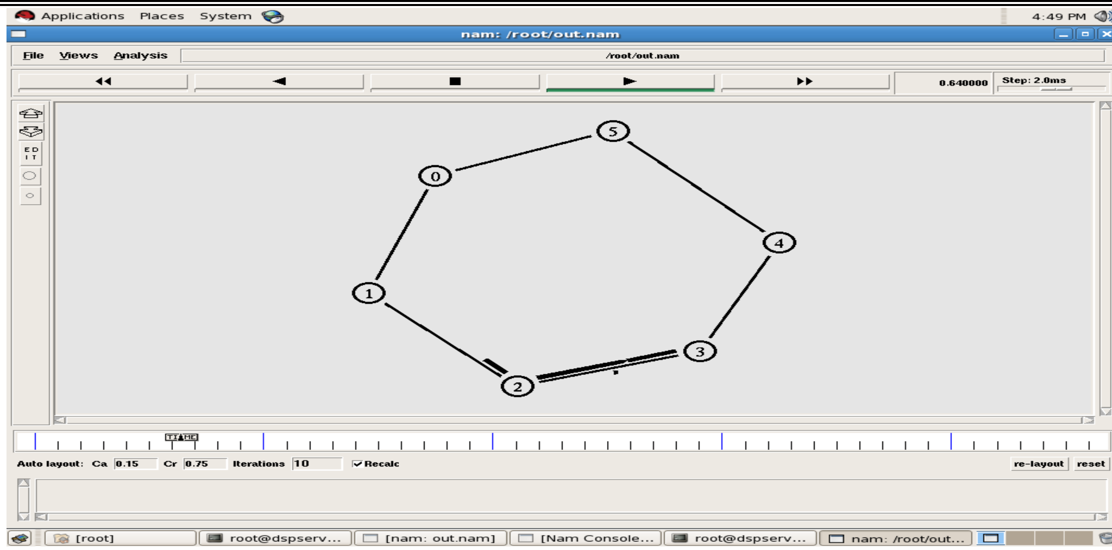


Fig6.7 Animation window

RESULT:-

Thus the concept of Network Simulator (NS2.35) is studied.

EXP. NO:6.b	SIMULATION OF SIMPLE NETWORK USING NS2
DATE:	

AIM

To build and simulate as implement network using NetworkSimulator2.

REQUIREMENTS

Operating system: REDHAT Linux

Programming tool: Network Simulator (NS-2.35)

THEORY

In a two-node system, both the nodes are connected to each other. They both exchangeinformation between them. One would be the source and other would-be destination.

PROCEDURE

1. Open a terminal and type the TCL file in the Vim editor with the command “vifilename.tcl”and save it.
2. Run the saved file with the command “ns filename.tcl”.
3. If any errors, edit them in the Vim editor and run it again.
4. The output is displayed in the NAM console.

PROGRAM

#Create a simulator objectset ns [new simulator] #Open the nam trace file set

nf [open out.nam W]

$ ns namtrace-all $nf #define a “finish” procedureProc finish {}

{

Globalnsnf

$nsflush-trace #Close the trace file

Close$nf

```
#Execute name on the trace file

Exec namout.nam&

Exit0

}

#Create 2 nodesSetn0[$nsnode]Setn1[$nsnode]

#Create a duplex link between the nodes

$nsduplex-link$n0$n11Mb10msDroptail

#Create a UDP agent and attach it to node no Set udp0 [newagent/UDP]

$nsattach–agent$n0 $udp0

#Create a CBR traffic source and attach it to UDP0 Set cbr0 [new application/traffic/CBR]

$cbr0setpacketrsize-500

$cbr0setinterval-0.01

$cbr0 attach –agent $udp0 #Call the “finish” Procedure

$ns at 5.0 “finish” #Run the simulation

$nsrun
```

NAMVIEW

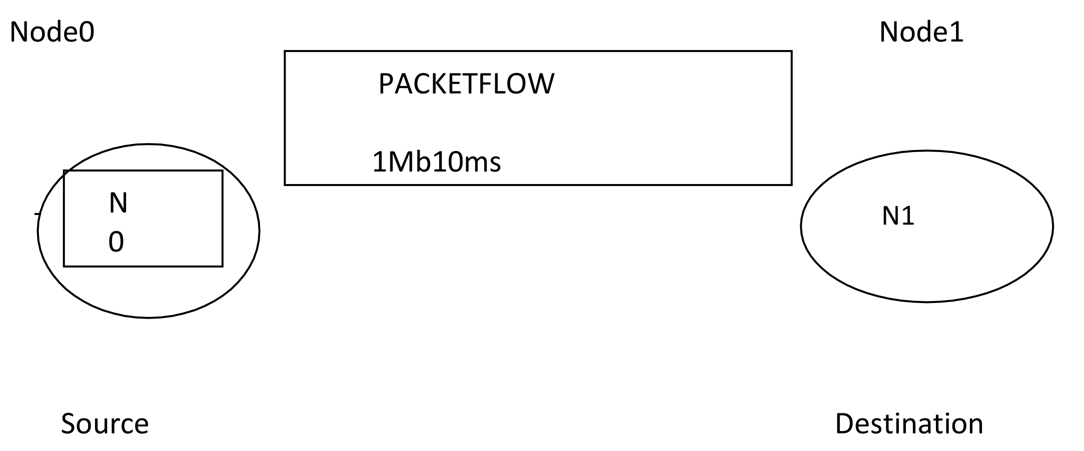


Fig 6.8 Implementation of simple network using ns2

RESULT

Thus, a simple network has been implemented using Network Simulator (NS2.35).

EXP. NO:7	NETWORK TOPOLOGY-BUS, MESH AND RING
DATE:	

AIM:

To build and simulate a network under different topologies like bus, mesh and ring.

REQUIREMENTS:

Operating System	:	Windows NT/2000/XP or LINUX
Programming Tool	:	Network Simulator (NS2)

THEORY:

The term *physical topology* refers to the way in which a network is laid out physically. Two or more devices connect to a link; two or more links form a topology. The topology of a network is the geometric representation of the relationship of all the links and linking devices (usually called nodes) to one another. There are four basic topologies possible: mesh, star, bus, and ring.

Mesh Topology In a mesh topology, every device has a dedicated point-to-point link to every other device. The term *dedicated* means that the link carries traffic only between the two devices it connects. To find the number of physical links in a fully connected mesh network with *n* nodes, we first consider that each node must be connected to every other node. Node1

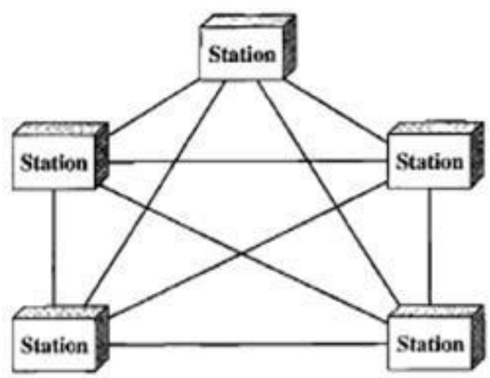


Fig.7.1**MeshTopology**

must be connected to *n* - I nodes, node 2 must be connected to *n* – 1 nodes, and finally node *n* must be connected to *n* - 1 nodes. We need $n(n-1)$ physical links. However, if each physical link allows communication in both directions (duplex mode), we can divide the number of links by 2. In other words, we can say that in a mesh topology, we need

$$n(n-1)/2$$

duplex-mode links.

A mesh offers several advantages over other network topologies. First, the use of dedicated links guarantees that each connection can carry its own data load, thus eliminating the traffic problems that can occur when links must be shared by multiple devices. Second, a mesh topology is robust. If one link becomes unusable, it does not incapacitate the entire system. Third, there is the advantage of privacy or security. When every message travels along a dedicated line, only the intended recipient sees it. Physical boundaries prevent other users from gaining access to messages. Finally, point-to-point links make fault identification and fault isolation easy. Traffic can be routed to avoid links with suspected problems. This facility enables the network manager to discover the precise location of the fault and aids in finding its cause and solution.

The main disadvantages of a mesh are related to the amount of cabling and the number of I/O ports required. First, because every device must be connected to every other device, installation and reconnection are difficult. Second, the sheer bulk of the wiring can be greater than the available space (in walls, ceilings, or floors) can accommodate. Finally, the hardware required to connect each link (I/O ports and cable) can be prohibitively expensive. For these reasons a mesh topology is usually implemented in a limited fashion, for example, as a backbone connecting the main computers of a hybrid network that can include several other topologies.

One practical example of a mesh topology is the connection of telephone regional offices in which each regional office needs to be connected to every other regional office.

Bus Topology : The preceding examples all describe point-to-point connections. A **bus topology,** on the other hand, is multipoint. One long cable act as a **backbone** to link all the devices in a network. Nodes are connected to the bus cable by drop lines and taps. A drop line isa connection running between the device and the main cable. A tap is a connector that either splices into the main cable or punctures the sheathing of a cable to create a contact with the metallic core. As a signal travels along the backbone, some of its energy is transformed into heat. Therefore, it becomes weaker and weaker as it travels farther and farther. For this reason there is a limit on the number of tapsa bus can support and on the distance between those taps.

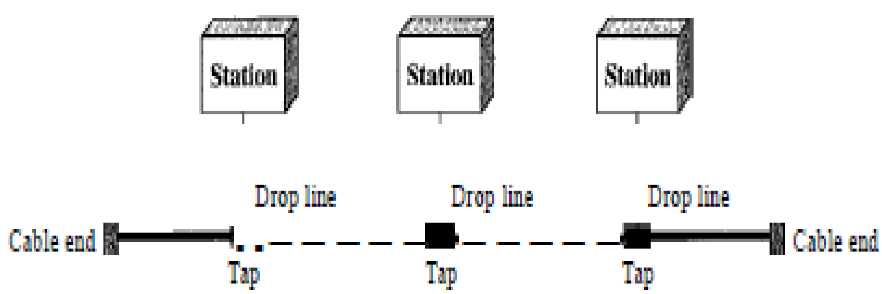


Fig.7.2. **A bus Topology connecting three stations**

Advantages of a bus topology include ease of installation. Backbone cable can be laid along the most efficient path, then connected to the nodes by drop lines of various lengths. In this way, a bus uses less cabling than mesh or star topologies. In a star, for example, four network devices in the same room require four lengths of cable reaching all the way to the hub. In a bus, this redundancy is eliminated. Only the backbone cable stretches through the entire facility. Each drop line has to reach only as far as the nearest point on the back bone.

Disadvantagesincludedifficultreconnectionandfaultisolation.Abusisusuallydesigned to be optimally efficient at installation. It can therefore be difficult to add new devices. Signal reflection at the taps can cause degradation in quality. This degradation can be controlled by limiting the number and spacing of devices connected toa given length of cable. Adding new devices may therefore require modification or replacement of the backbone.

In addition, a fault or break in the bus cable stops all transmission, even between devices on the same side of the problem. The damaged area reflects signals back in the direction of origin, creating noise in both directions. Bus topology was the one of the first topologies use

the design of early local area networks. Ethernet LANs can use a bus topology, but they are less popular now for reasons.

Ring Topology: In a ring topology, each device has a dedicated point-to-point connection with only the two devices on either side of it. A signal is passed along the ring in one direction, from device to device, until it reaches its destination. Each device in the ring incorporates a repeater. When a device receives a signal intended for another device, its repeater regenerates the bits and passes them along. A ring is relatively easy to install and reconfigure. Each device is linked to only its immediate neighbors (either physically or logically). To add or delete a device requires changing only two connections. The only constraints are media and traffic considerations (maximum ring length and number of devices). In addition, fault isolation is simplified.

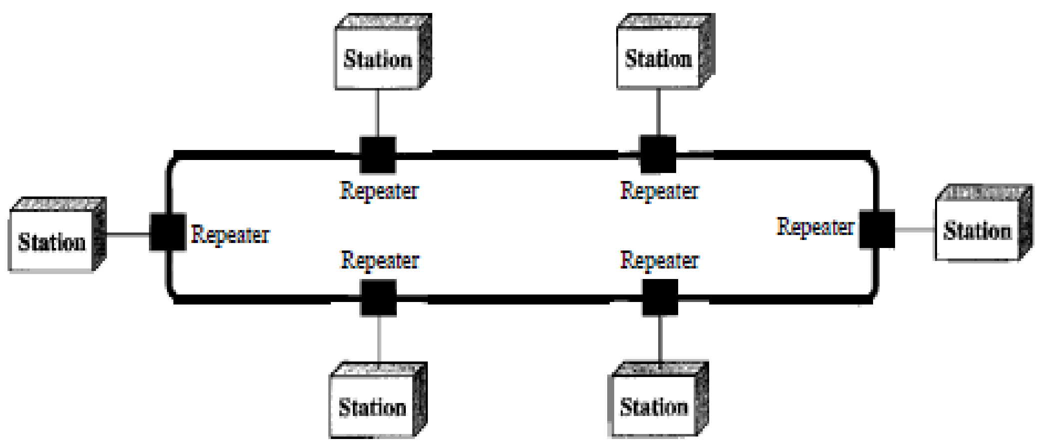


Fig.7.3**A Ring Topology connecting six stations**

Generally, in a ring, a signal is always circulating. If one device does not receive a signal within a specified period, it can issue an alarm. The alarm alerts the network operator to the problem and its location. However, unidirectional traffic can be a disadvantage. In a simple ring, a break in the ring (such as a disabled station) can disable the entire network. This weakness can be solved by using a dual ring or a switch capable of closing off the break.

Ring topology was prevalent when IBM introduced its local-area network Token Ring.

Today, the need for higher-speed LANs has made this topology less popular.

PROGRAM:

MESH TOPOLOGY

```
#Create a simulator object #

set ns [new Simulator] #Open the nam trace file#

setnf[openout.namw]

$ns namtrace-all $nf #Define a 'finish' procedure#

Proc finish {} {

global ns nf

   $ns flush-trace #Close the trace file#

   close $nf

   #Execute nam on the tracefile #

   exec namout.nam&

   exit0

}

#Create four nodes#

setn0[$nsnode]setn1[$nsnode]setn2[$nsnode]setn3[$nsnode]

#Create links between the nodes

$nsduplex-link $n0$n1 1Mb10msDropTail

$nsduplex-link$n0$n21Mb10msDropTail

$nsduplex-link$n0$n31Mb10msDropTail

$nsduplex-link$n1$n21Mb10msDropTail

$nsduplex-link$n1$n31Mb10msDropTail

$ns duplex-link $n2 $n3 1Mb 10ms DropTail #Create a TCP agent and attach it to

node n1# set tcp0 [newAgent/TCP]

$tcp0setclass_1

$nsattach-agent$n1 $tcp0

#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3#
```

```
setsink0[new Agent/TCPSink]

$nsattach-agent$n3$sink0
#Connect the traffic sources with the trafficsink#

$nsconnect$tcp0$sink0

#Create a CBR traffic source and attach it to tcp0#

set cbr0 [new Application/Traffic/CBR]

$cbr0setpacketSize_500

$cbr0setinterval_0.01

$cbr0attach-agent$tcp0

#Schedule events for the CBR agents

$nsat0.5"$cbr0start"

$nsat4.5 "$cbr0stop"

#Callthefinishprocedureafter 5 seconds of simulation time

$nsat5.0"finish"#Run the simulation

$nsrun
```

BUSTOPOLOGY

```
#Create a simulator object

set ns[newSimulator] #Open the nam trace file set nfv[open out.nam w]

$ns namtrace-all $nf #Define a 'finish' procedure proc finish {}{

   Global nsnf

   $nsflush-trace #Close the trace file close $nf

   #Executenamonthetracefile#

   exec nam out.nam&

   exit0

}

#Create four nodes

setn0[$nsnode] setn1 [$nsnode] set n2[$nsnode] set n3[$nsnode] set
```

```
n4[$nsnode]

#Create LAN between the nodes

Set lan0 [$nsnewLan"$n0$n1$n2$n3$n4"0.5Mb 40msLLQueue/ Drop Tail MAC/
Csma/ CdChannel]

#Create aTCP agent and attach it to node n0 set tcp0 [newAgent/TCP]

$tcp0setclass_1

$nsattach-agent$n1 $tcp0

#CreateaTCP Sink agent(a trafficsink) for TCP and attach it to node n3

setsink0[new Agent/TCPSink]

$nsattach-agent$n3$sink0

#Connect the traffic sources with the traffic sink

$nsconnect$tcp0$sink0

# Create a CBR traffic source and attach it to tcp0 set cbr0

[newApplication/Traffic/CBR]

$cbr0setpacketSize_500

$cbr0setinterval_0.01

$cbr0attach-agent$tcp0

#Schedule events for the CBR agents

$nsat0.5"$cbr0start"

$nsat4.5 "$cbr0stop"
#Call the finish procedure after 5 seconds of simulation time

$nsat5.0"finish"#Run the simulation

$nsrun
```

RINGTOPOLOGY

```
#Create a simulator object set ns[new Simulator] #Open the nam trace file set nf

[openout.namw]

$ns namtrace-all $nf #Define a 'finish' procedure

proc finish {}{
```

```
    Global nsnf
    $nsflush-trace #Close the tracefile
    close $nf
    #Execute nam on the trace file
    exec namout.nam&
    exit0
}
#Create four nodes
setn0 [$nsnode] setn1 [$nsnode] setn2 [$nsnode] setn3 [$nsnode] setn4 [$nsnode] setn5
[$nsnode]
#Create links between the nodes
$nsduplex-link$n0$n11Mb10msDropTail
$nsduplex-link$n1$n21Mb10msDropTail
$nsduplex-link$n2$n31Mb10msDropTail
$nsduplex-link$n3$n41Mb10msDropTail
$nsduplex-link$n4$n51Mb10msDropTail
$nsduplex-link$n5$n01Mb10msDropTail#Create a TCP agent and attach it to
node n0settcp0[newAgent/TCP]
$tcp0setclass_1
$nsattach-agent$n1 $tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node
n3setsink0[new Agent/TCPSink]
$nsattach-agent$n3$sink0
#Connectthetrafficsourceswiththetrafficsink
$nsconnect$tcp0$sink0
# Create a CBR traffic source and attach it to tcp0 set cbr0
```

[newApplication/Traffic/CBR]

$cbr0setpacketSize_ 500

$cbr0setinterval_0.01

$cbr0attach-agent$tcp0

#ScheduleeventsfortheCBRagents

$nsat0.5"$cbr0start"

$nsat4.5 "$cbr0stop"

#Callthefinishprocedureafter 5 seconds of simulation time

$nsat5.0"finish"#Run the simulation

$nsrun

Program output:

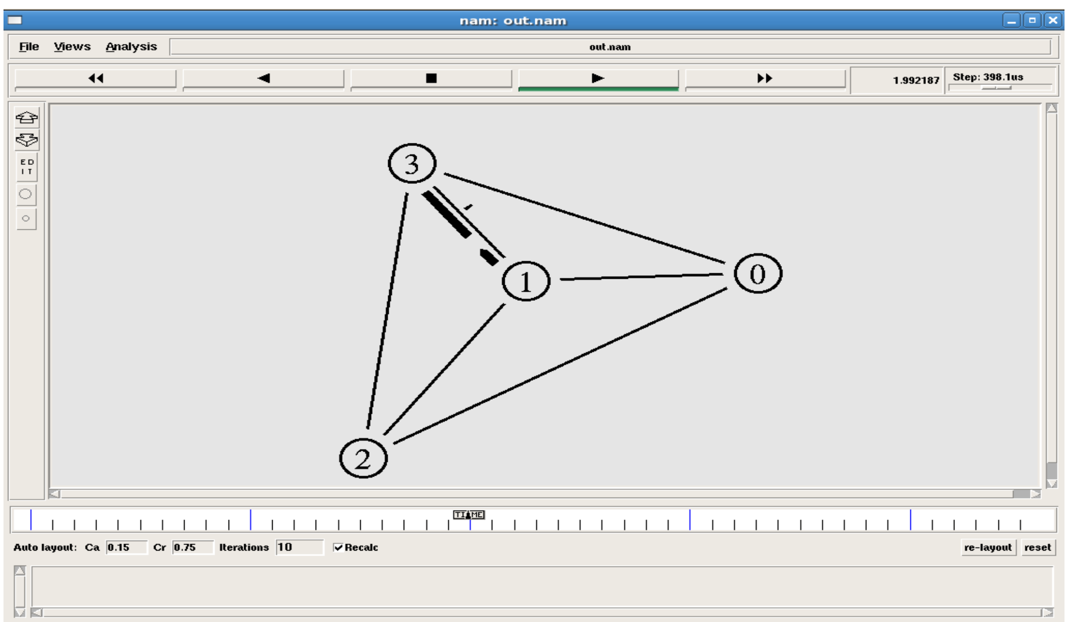


Fig7.4.Meshtopology

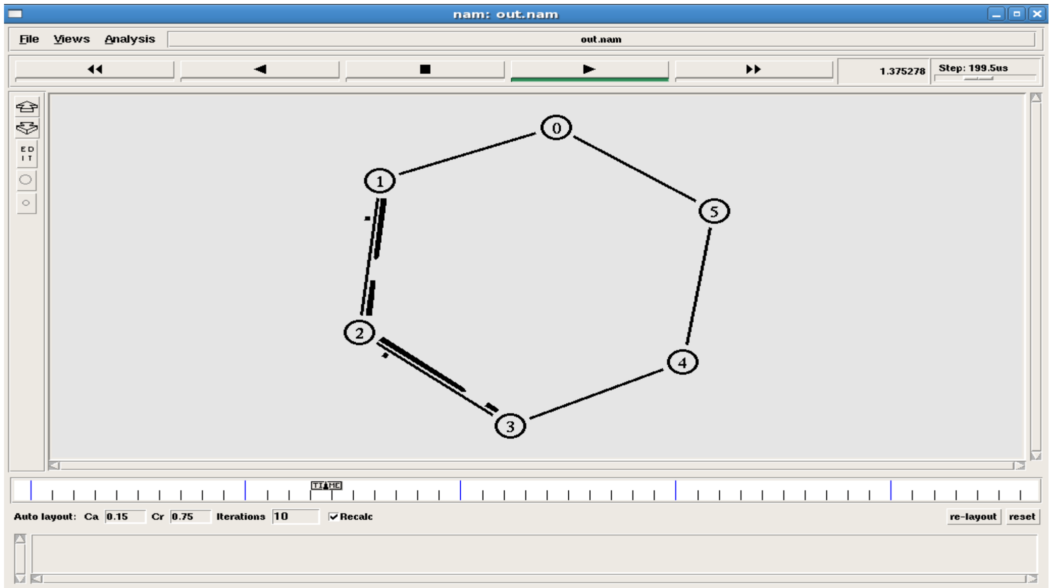


Fig7.5. Ring topology

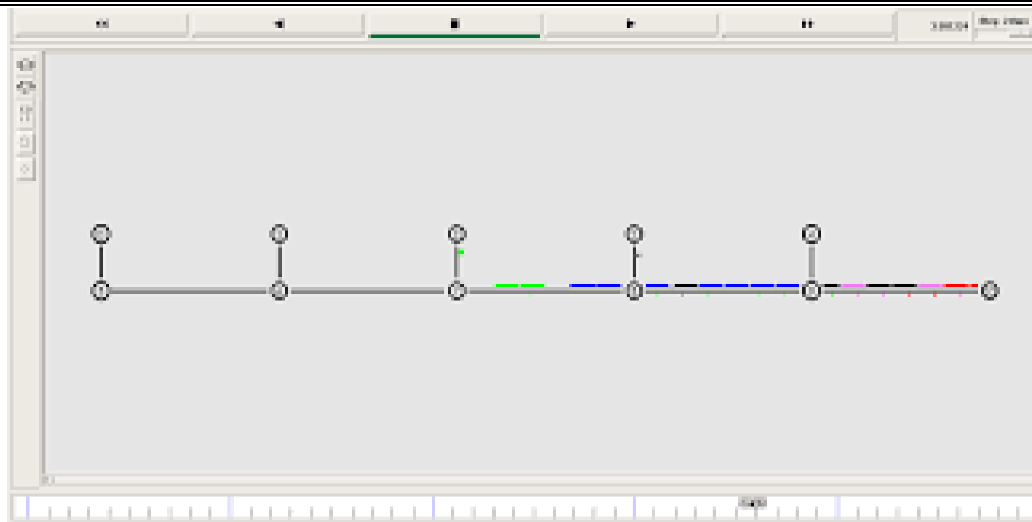


Fig.7.6. Bus topology

RESULT:

Thus the network topologies like mesh, bus and ring have been implemented using network simulator.

SAMPLEVIVAQUESTIONS:

1) What is a Network?
2) What is a Node?
3) What is Network Topology?
4) What are the different types of a network?
5) Few important terminologies we come across networking concepts?
6) Explain the characteristics of networking?
7) How many types of modes are used in data transferring through networks?
8) Name the different types of network topologies.
9) What are the advantages and disadvantages of bus topology?
10) What are the advantages and disadvantages of ring topology?
11) What are the advantages and disadvantages of mesh topology?
12) What are the differences between TCP And UDP?
13) Which service use both TCP and UDP?
14) What is the port no of SMTP and pop3?
15) Which one is reliable– TCP or UDP?
16) What is the port number of FTP (data)and FTP?
17) What is the way to establish a TCP connection?

EXP. NO: 8	IMPLEMENTATION OF DISTANCE VECTOR ROUTING ALGORITHM
DATE:	

AIM:

To simulate the distance vector routing algorithm using NS2 simulator.

REQUIREMENTS:

Operating System	:	Windows NT/2000/XP or LINUX
Programming Tool	:	Network Simulator(NS2)

THEORY:

Computer networks generally use dynamic routing algorithms that are more complex than flooding, but more efficient because they find shortest paths for the current topology. Two dynamic algorithms in particular, distance vector routing and link state routing, are the most popular.

Distance Vector Routing

A **distance vector routing** algorithm operates by having each router maintain a table(i.e., a vector) giving the best known distance to each destination and which link to use to get there. These tables are updated by exchanging information with the neighbors. Eventually, every router knows the best link to reach each destination.

The distance vector routing algorithm mis sometimes called by other names, most commonly the distributed **Bellman-Ford** routing algorithm, after there searchers who developed it (Bellman, 1957; and Ford and Fulkerson, 1962). It was the original ARPANET routing algorithm and was also used in the Internet under the name RIP.

In distance vector routing, each router maintains a routing table indexed by, and containing one entry for each router in the network. This entry has two parts: the preferred outgoing line to use for that destination and an estimate of the distance to that destination. The distance might be measured as the number of hops or using another metric, as we discussed for computing shortest paths.

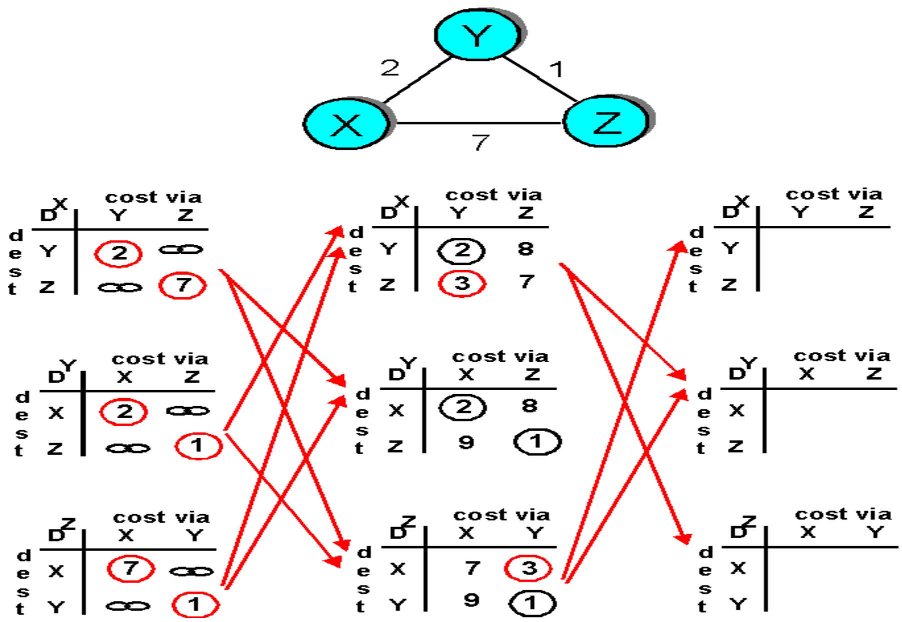


Fig.8.1 **Distance Vector Routing**

The router is assumed to know the ''distance'' to each of its neighbors. If the metric is hops, the distance is just one hop. If the metric is propagation delay, the router can measure it directly with special ECHO packets that the receiver just timestamps and sends back as fast as it can. As an example, assume that delay is used as a metric and that the router knows the delay to each of its neighbors. Once every *T* msec, each router sends to each neighbor a list of its estimated delays to each destination. It also receives a similar list from each neighbor. Imagine that one of these tables has just come in from neighbor *X*, with *Xi* being *X*'s estimate of how long it takes to get to router *i*.

If the router knows that the delay to *X* is *m* msec, it also knows that it can reach router *I* via *X* in *Xi* + *m* msec. By performing this calculation for each neighbor, a router can find out which estimate seems the be stand use that estimate and the corresponding link in its new routing table. Note that the old routing table is not used in the calculation.

The Count-to-InfinityProblem

The settling of routes to best paths across the network is called **convergence**. Distance vector routing is useful as a simple technique by which routers can collectively compute shortest paths, but it has a serious drawback in practice: although it converges to the correct answer, it may do so slowly.

ALGORITHM:

Step1: Start.
Step 2: Create a simulator object.
Step3: Configure the simulator to use dynamic routing.
Step 4: Open the nam.trace file.
Step 5: Open the output file.
Step6: Define the finish procedure.
Step7: Close the trace file.

Step8: Call x-graph to display the result.
Step 9: Create 7 nodes.
Step10: Create links between the nodes.
Step 11: Create a UDP agent to attach the node.
Step 12: Create a CBR traffic router and attach.
Step 13: Create a Null agent to the traffic sink.
Step 14: Connect the traffic source to the sink.
Step 15: Schedule the events for CBR agent.
Step16: Stop.

PROGRAM:

```
Set ns [newSimulator]

$nsrtprotoDV

$ns macType MAC/802_3setnf[opendistance.namw]

$nsnamtrace-all$nf

setf0 [opendistance.tr w]

$nstrace-all$f0proc finish {} {globalnsf0nf

$ns flush-traceclose $f0

close$nf

execnamdistance.nam&exit0

}

setn0[$nsnode]setn1[$nsnode]setn2[$nsnode]setn3[$nsnode]setn4[$nsnod

e]setn5[$nsnode]setn6[$nsnode]

$nsduplex-link$n0$n11Mb10msDropTail

$nsduplex-link$n1$n21Mb10msDropTail

$nsduplex-link$n2$n31Mb10msDropTail

$nsduplex-link$n3$n41Mb10msDropTail

$nsduplex-link$n4$n51Mb10msDropTail

$nsduplex-link$n5$n61Mb10msDropTail

$nsduplex-link$n6 $n01Mb10msDropTailsetudp0[new Agent/TCP]

$nsattach-agent$n1$udp0

setcbr0[newApplication/Traffic/CBR]
```

```
$cbr0setpacketsize_500

$cbr0setinterval_0.005

$cbr0attach-agent$udp0

setnull0[newAgent/TCPSink]

$nsattach-agent$n3$null0

$nsconnect$udp0$null0

$nsat0.05"$cbr0 start"

$nsrtmodel-at1.0down$n1$n2

$nsrtmodel-at2.0up$n1$n2

$nsat4.5 "$cbr0stop"

$nsat5.0"finish"

$nsrun
```

RESULT:

Thus the distance vector routing algorithm has been implemented and simulated using NS2 simulator.

SAMPLEVIVAQUESTIONS:

1. What is Routing?
2. What is the difference between static and dynamic routes?
3. What is a routing protocol?
4. What are the three classes of Routing Protocols?
5. How do Distance vector routing Protocol function?
6. How do Distance vector routing Protocol keep track of any changes to the Internetwork?
7. What is Split Horizon?
8. What is Convergence?
9. What is Route Poisoning?
10. What is Hold-down timer?
11. What is Link-state Routing Protocol?
12. What is Metric?
13. What is Hop Count?
14. What is Convergence?
15. What is Converged Network Topology?

EXP. NO:9	**SIMULATION OF LINK STATE ROUTING ALGORITHM**
DATE:	

AIM:

To simulate the link state routing algorithm using NS2 simulator.

REQUIREMENTS:

Operating System	:	Windows NT/2000/XP or LINUX
Programming Tool	:	Network Simulator (NS2)

THEORY:

Link State Routing

Distance vector routing was used in the ARPANET until 1979, when it was replaced by link state routing. The primary problem that caused its demise was that the algorithm often took too long to converge after the network topology changed (due to the count-to-infinity problem). Consequently, it was replaced by an entirely new algorithm, now called **link state routing**. Variants of link state routing called IS-IS and OSPF are the routing algorithms that are most widely used inside large networks and the Internet today. The idea behind link state routing is simple and can be stated as five parts.

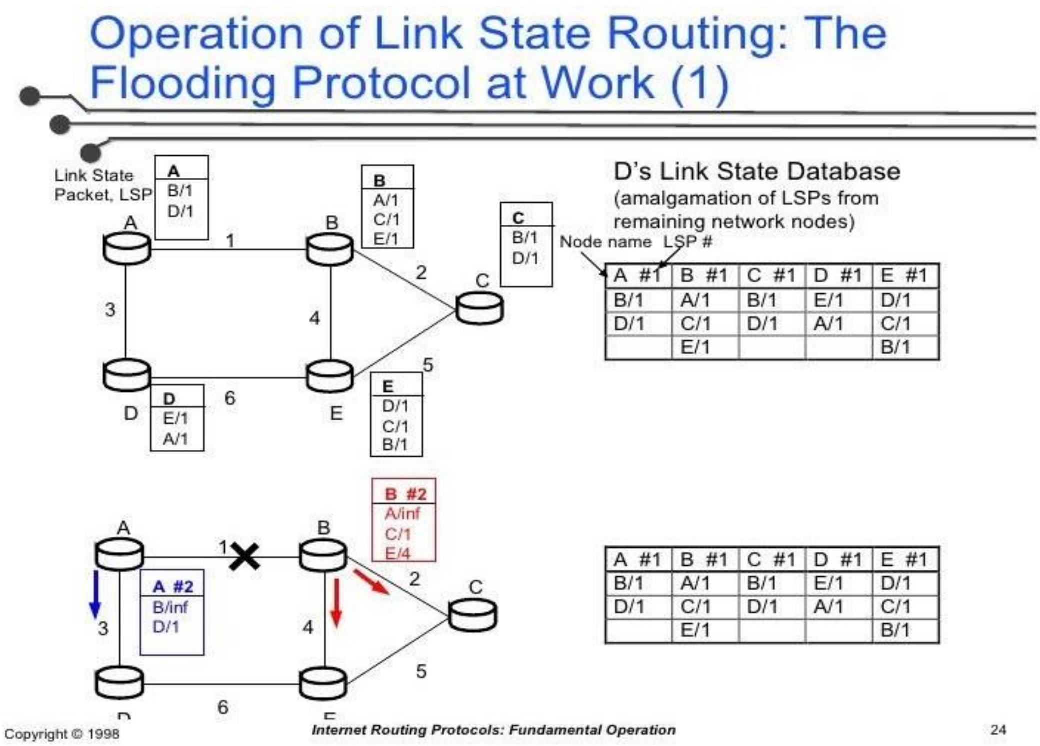


Fig.9.1 Link State Routing

Each router must do the following things to make it work:

1. Discover its neighbors and learn their network addresses.
2. Set the distance or cost metric to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to and receive packets from all other routers.
5. Compute the shortest path to every other router.

In effect, the complete topology is distributed to every router. Then Dijkstra's algorithm can be run at each router to find the shortest path to every other router. Link state routing is widely used in actual networks, so a few words about some example protocols are in order. Many ISPs use the **IS-IS** (**Intermediate System-Intermediate System**) link state protocol (Oran, 1990). It was designed for an early network called DEC net, later adopted by ISO for use with the OSI protocols and then modified to handle other protocols as well, most notably, IP. **OSPF** (**Open Shortest Path First**) is the other main link state protocol. It was designed by IETF several years after IS-IS and adopted many of the innovations designed for IS-IS. These innovations include a self-stabilizing method of flooding link state updates, the concept of a designated router on a LAN, and the method of computing and supporting path splitting and multiple metrics. Consequently, there is very little difference between IS-IS and OSPF. The most important difference is that IS-IS can carry information about multiple network layer protocols at the same time (e.g., IP, IPX, and AppleTalk). OSPF does not have this feature, and it is an advantage in large multiprotocol environments.

ALGORITHM:

Step1: Start.
Step 2: Create a simulator object.
Step3: Configure the simulator to use dynamic routing.
Step 4: Open the nam trace file.
Step 5: Open the output file.
Step6: Define the finish procedure.
Step7: Close the trace file.
Step8: Call x-graph to display the result.
Step 9: Create 7 nodes.
Step10: Create links between the nodes.
Step 11: Create a UDP agent to attach the node.
Step 12: Create a CBR traffic router and attach.
Step 13: Create a Null agent to the traffic sink.
Step 14: Connect the traffic source to the sink.
Step 15: Schedule the events for CBR agent.
Step16: Stop.

PROGRAM:

```
Set ns[newSimulator]

$nsrtprotoLS

setnf[openlinkstate.namw]

$nsnamtrace-all$nf
```

```
setnt[openlinkstate.trw]

$nstrace-all$ntproc finish {} {globalnsnfnt

$ns flush-traceclose $nf

close$nt

exec namlinkstate.nam&exit0

}

setn0[$nsnode]setn1[$nsnode]setn2[$nsnode]setn3[$nsnode]setn4[$nsnode]set

n5[$nsnode]setn6[$nsnode]

$nsduplex-link$n0$n11Mb10msDropTail

$nsduplex-link$n1$n21Mb10msDropTail

$nsduplex-link$n2$n31Mb10msDropTail

$nsduplex-link$n3$n41Mb10msDropTail

$nsduplex-link$n4$n51Mb10msDropTail

$nsduplex-link$n5$n61Mb10msDropTail

$nsduplex-link$n6$n01Mb10msDropTailsetudp0[new Agent/UDP]

$nsattach-agent$n0$udp0

setcbr0[newApplication/Traffic/CBR]

$cbr0setpacketsize_500

$cbr0setinterval_0.005

$cbr0attach-agent$udp0setnull0[newAgent/Null]

$nsattach-agent$n3$null0

$nsconnect$udp0$null0

$nsat0.5"$cbr0 start"

$nsrtmodel-at1.0down$n1$n2

$nsrtmodel-at2.0up$n1$n2

$nsat4.5 "$cbr0stop"

$nsat5.0"finish"
```

$nsrun

Distance Vector Routing and Link State Routing

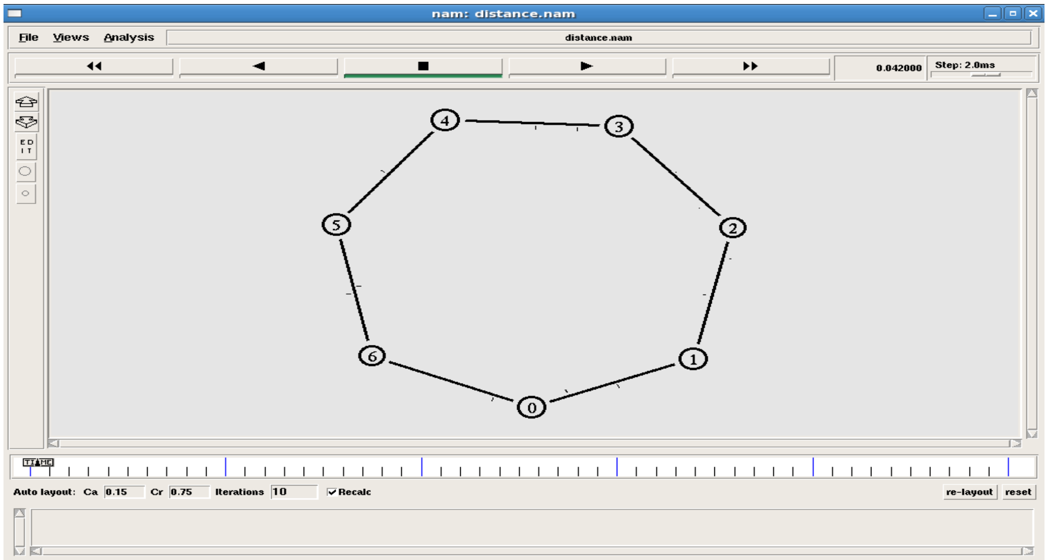


Fig9.2.Initialization

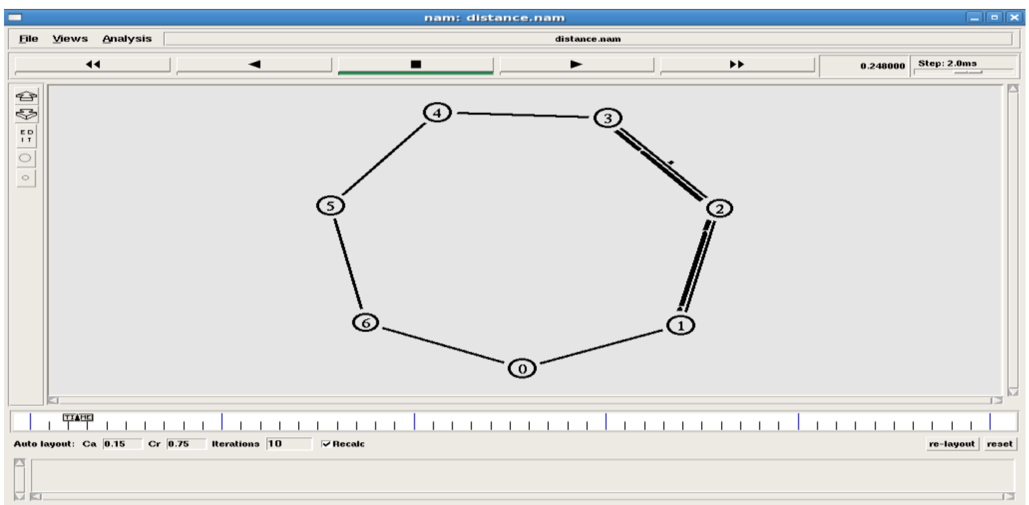


Fig.9.3Packetforwarding

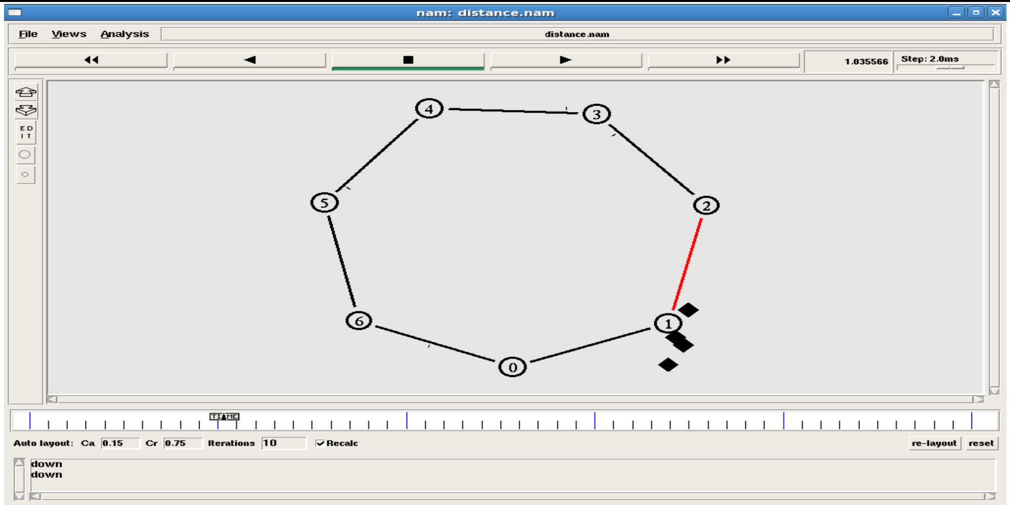


Fig.9.4. Link down between node1 and 2

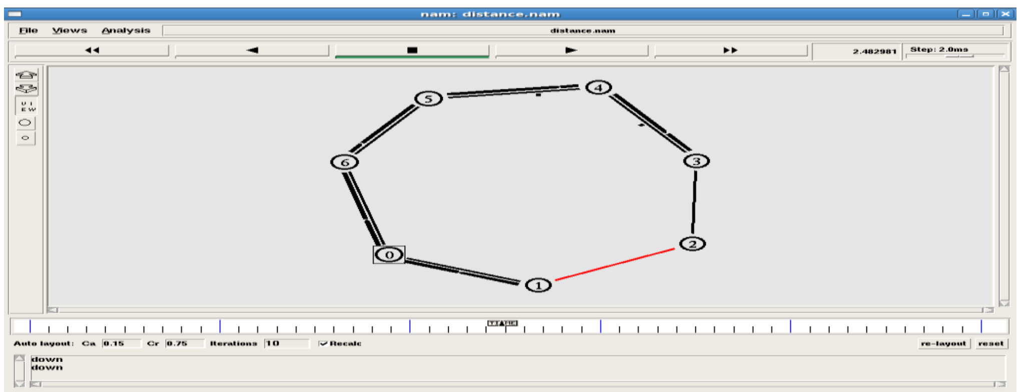


Fig.9.5Packets rerouted

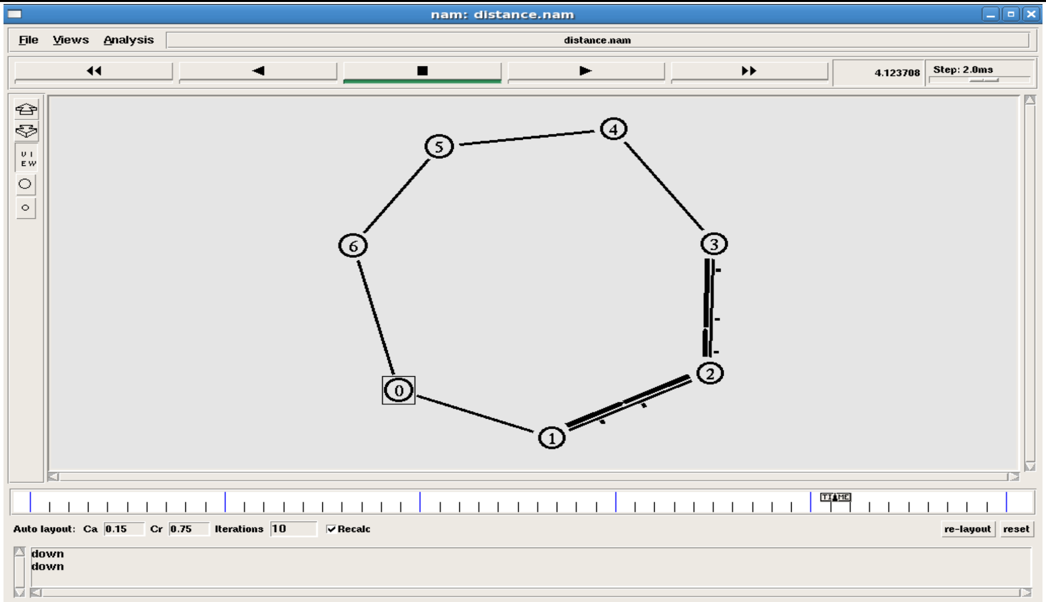


Fig.9.6Linkup between node1and 2

RESULT:

Thus the links state routing algorithm has been implemented and simulated using NS2simulator.

<u>SAMPLEVIVAQUESTIONS</u>:

1. How do you classify routing algorithms? Give examples for each.
2. What are drawbacks in distance vector algorithm?
3. How routers update distances to each of its neighbor?
4. How do you overcome count to infinity problem?
5. What is difference between distance vector and link-state routing protocols

EXP.NO:10	STUDY AND IMPLEMENTATION OF GOBACK-N AND SELECTIVE REPEAT PROTOCOLS
DATE:	

AIM:

To implement the Go Back-N and Selective Repeat protocols and study the performance parameters.

REQUIREMENTS:

LAN Trainer Kit, Patch Chords, Personal Computer and required simulator.

THEORY:

GOBACK-N

Until now we have made the tacit assumption that the transmission time required for a frame to arrive at the receiver plus the transmission time for the acknowledgement to come back is negligible. Sometimes this assumption is clearly false. In these situations, the long round-trip time can have important implications for the efficiency of the bandwidth utilization. The problem described here can be viewed because of the rule requiring a sender to wait for an acknowledgement before sending another frame. If we relax that restriction, much better efficiency can be achieved.

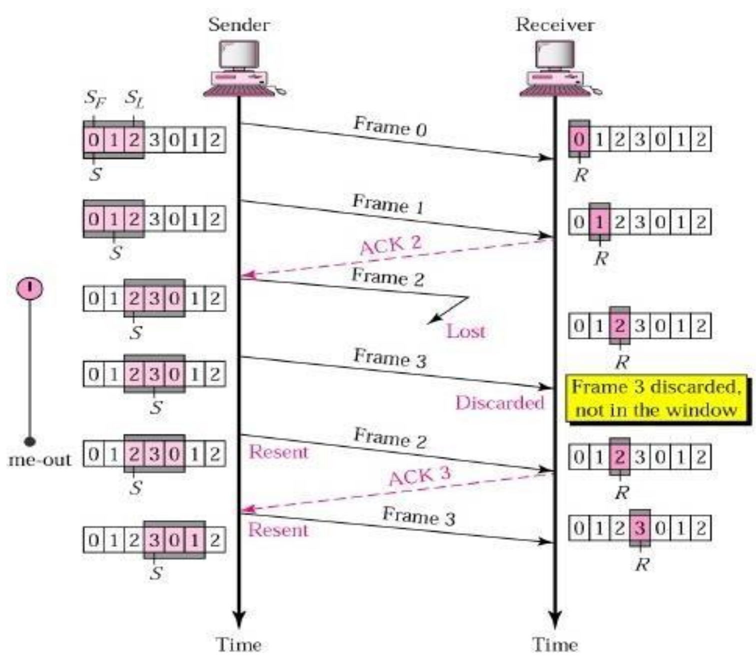


Fig10.1**GOBACK-N**

Basically, the solution lies in allowing the sender to transmit up to *w* frames before blocking, instead of just1. With a large enough choice of *w* the sender will be able to continuously transmit frames since the acknowledgements will arrive for previous frames before the window becomes full, preventing the sender from blocking To find an appropriate value for *w* we need to know how many frames can fit inside the channel as they propagate from sender to receiver. This capacity is determined by the bandwidth in bits/sec multiplied by

the one-way transit time, or the **bandwidth-delay product** of the link. We can divide this quantity by the number of bits in a frame to express it as a number of frames. Call this quantity *BD*. Then *w* should be set to 2*BD* + 1. Twice the bandwidth-delay is the number of frames that can be outstanding if the sender continuously sends frames when the round-trip time to receive an acknowledgement is considered. The ''+1'' is because an acknowledgement frame will not be sent until after a complete frame is received. For smaller window sizes, the utilization of the link will be less than 100% since the sender will be blocked sometimes. We can write the utilization as the fraction of time that the sender is not blocked:

$$\textbf{link utilization} \leq w/1 + 2BD$$

This value is an upper bound because it does not allow for any frame processing time and treats the acknowledgement frame as having zero length, since it is usually short. The equation shows the need for having a large window *w* whenever the bandwidth-delay product is large. If the delay is high, the sender will rapidly exhaust its window even for a moderate bandwidth, as in the satellite example. If the bandwidth is high, even for a moderate delay the sender will exhaust its window quickly unless it has a large window (e.g., a 1-Gbps link with 1-msec delay holds 1 megabit). With stop-and-wait for which *w* = 1, if there is even one frame's worth of propagation delay the efficiency will be less than 50%. This technique of keeping multiple frames in flight is an example of **pipelining**. Pipelining frames over an unreliable communication channel raises some serious issues. Selective repeat is often combined with having the receiver send a negative acknowledgement (NAK) when it detects an error, for example, when it receives a checksum error or a frame out of sequence. NAKs stimulate retransmission before the corresponding timer expires and thus improve performance.

PROCEDURE:

With BER:

1. Set the BER value to 10^{-6} in NEU.
2. Set the time-out value to a constant value and proceed as earlier.
3. Calculate the throughput using the formula,

   Throughput, X={Successfully transmitted packets* Packet Length*8} / {Data rate*Duration of the experiment}
4. Plot the graph for BERVs throughput.

Without BER:

6. Click on the "SlwinGBN"/"SelectiveRepeat" icon from the desktop on both the PCs.

7. Click configure button in the window of both PCs.
8. Set the IPD, number of packets etc.
9. Download the driver to the NIU using boot command.
10. Run the experiment by clicking "RUN-START" from each application.
11. Note down the value of successfully transmitted packets.
12. Vary the time-out and get various values of throughput.
13. Plot the graph for time-out Vs throughput.

OBSERVATION:

PacketLength=1000bytes Datarate =8Kbps Duration=30 sec

GOBACK-NProtocol:

WithoutBER

Time-out (ms)	Successfully Transmitted	Throughput (X)
1500		
2000		
3000		
4000		
4500		

WithBER

BER	Successfully Transmitted	Throughput (X)
10^{-6}		
10^{-5}		
10^{-4}		
10^{-3}		
10^{-2}		

SelectiveRepeatProtocol:

WithoutBER

Time-out (ms)	Successfully Transmitted	Throughput (X)
1500		
2000		
3000		
4000		
4500		

WithBER

BER	Successfully Transmitted	Throughput (X)
10^{-6}		
10^{-5}		
10^{-4}		
10^{-3}		
10^{-2}		

RESULT:

Thus the Goback N and Selective Repeat protocols have been implemented and the performance parameters were studied.

SAMPLE VIVA QUESTIONS:

1. Which layer is responsible for keeping the data from different applications separate on the network?
2. Which layer segments and resembles data into a data stream?
3. Which layer provides the physical transmission of the data and handles error notification, network topology, and flow control?
4. Which layer manages device addressing, tracks the location of devices on the network, and determine the best way to move data?
5. Mac address works on which layer?
6. What are the differences of mac sub layer and LLC sublayer?
7. Which layer is responsible for converting data packets from the data link layer into electrical signals?
8. At which layer is routing implemented, enabling connections and path selection between two end systems?
9. Which layer define show data is formatted, presented, encoded, and converted for use on the network?
10. What is the difference between flow control and error control?

EXP. NO:11

DATE:

IMPLEMENTATION ANDSTUDY OF CSMA/CD PROTOCOL

AIM:

To create a scenario and implement CSMA/CD protocol and study the performance parameters.

REQUIREMENTS:

LAN Trainer Kit, Patch Chords, Personal Computer and required simulator.

THEORY:

Carrier Sensed Multiple Access (CSMA): CSMA is a network access method used on shared network topologies such as Ethernet to control access to the network. Devices attached to the network cable listen (carrier sense) before transmitting. If the channel is in use, devices wait before transmitting. MA (Multiple Access) indicates that many devices can connect to and share the same network. All devices have equal access to use the network when it is clear.

CSMA protocol was developed to overcome the problem found in ALOHA i.e. to minimize the chances of collision, to improve the performance. CSMA protocol is based on the principle of 'carrier sense'. The station senses the carrier or channel before transmitting a frame. It means the station checks the state of channel, whether it is idle or busy. There are three different type of CSMA Protocols

(i) I-persistent CSMA
(ii) Non-Persistent CSMA
(iii) p-persistent CSMA

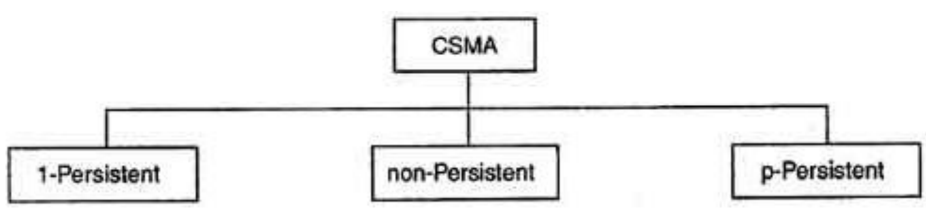


Fig11.1.Different TypeofCSMAProtocols

(i) I-persistent CSMA

- In this method, station that wants to transmit data continuously senses the channel to check whether the channel is idle or busy.
- If the channel is busy, the station waits until it becomes idle.

• When the station detects an idle-channel, it immediately transmits the frame with probability1.Hence it is called p-persistent CSMA.

Even if propagation delay time is zero, collision will still occur. If two stations became ready in the middle of third station's transmission, both stations will wait until the transmission of first station ends and then both will begin their transmission exactly simultaneously. This will also result in collision.

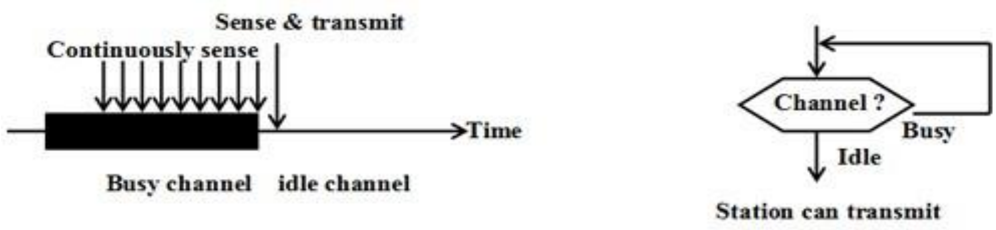


Fig11.2.I-persistent CSMA

(ii) Non-persistent CSMA

• In this scheme, if a station wants to transmit a frame and it finds that the channel is busy(some other station is transmitting) then it will wait for fixed interval of time.

• After this time, it again checks the status of the channel and if the channel is. Free it will transmit.

• A station that has a frame to send senses the channel.

• If the channel is idle,it sends immediately.

• If the channel is busy, it waits a random amount of time and then senses the channel again.

Advantage of non-persistent:

• It reduces the chance of collision because the stations wait a random amount of time. It is unlikely that two or more stations will wait for same amount of time and will retransmit at the same time.

Disadvantage of non-persistent:

• It reduces the efficiency of network because the channel remains idle when there maybe stations with frames to send. This is due to the fact that the stations wait a random amount of time after the collision.

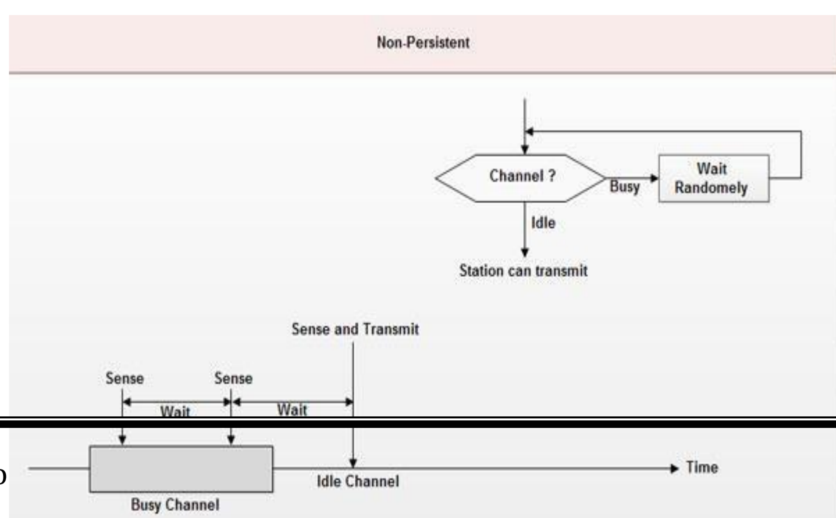


Fig11.2.Non-persistent CSMA

(iii) p-persistent

- This method is used when channel has time slots such that the time slot duration is equal to or greater than the maximum propagation delay time
- Whenever a station becomes ready to send, it senses the channel.
- If channel is busy, station waits until next slot.
- If channel is idle, it transmits with a probability p.
- With the probability q=l-p, the station then waits for the beginning of the next time

Advantage of p-persistent:

• It reduces the chance of collision and improves the efficiency of the network.

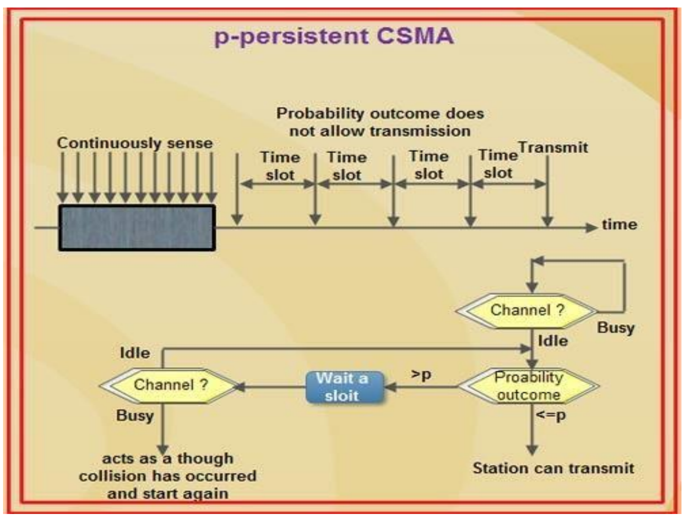


Fig11.3.p-persistent

PROCEDURE:

1. Click on the MAC experiment I contwice from the desktop on both the PCs.
2. Click Configuration button in thewindow on both PCs.
3. Set the IPD to 40 ms.
4. Download the driver to the NIU using boot button command for both PCs.
5. Run the application by clicking the RUN-START from each application.
6. View the statistics window for the result, only transmitted packets and collision count are taken for calculation in MAC

experiments.

7. Note down the reading when the experiment says it has stopped after specified duration.
8. Repeat the above steps from 1to7 and take readings choosing a range of “G” and plot the graph between “X” and “G”.

Formula Used:

Throughput,

X = {(Sum of successfully transmitted packets/4) * Packet Length*8}/ {Data rate*Duration} Offered Load,

G={(Sum of offered load in all 4nodes/ 4)*Packet Length*8} / {Data rate*Duration}

Average,

D=Sumofdelayof4nodes/4

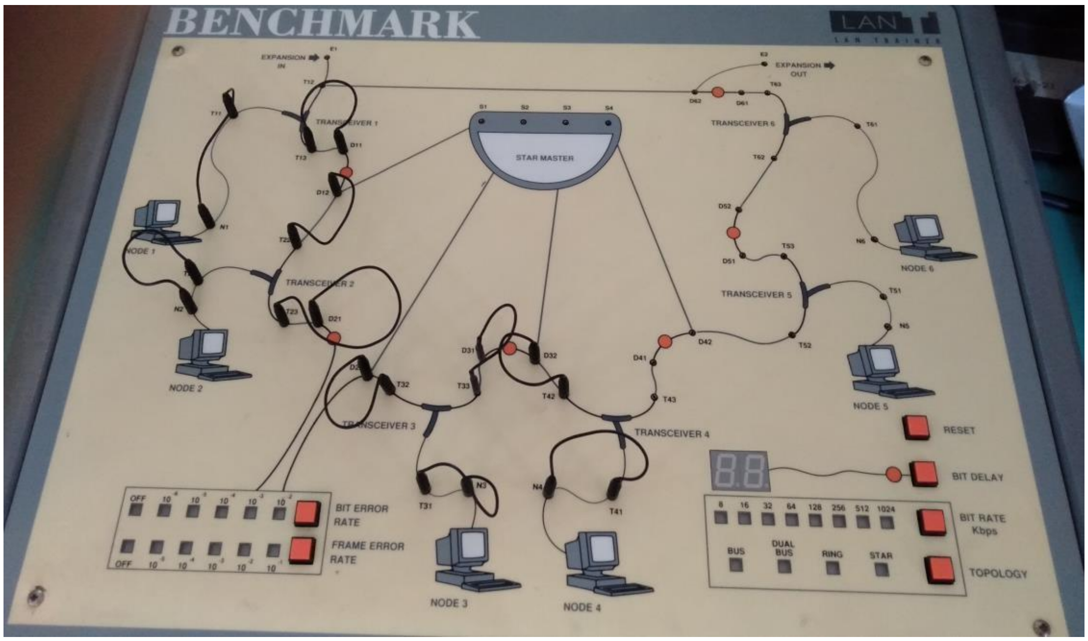


Fig:11.4 LANtrainerkit

OBSERVATION:

IPD (ms)	Node1	Node2 to Node4	Offered Load(G)	Throughput (X)	
	Transmitted Packets	Collided Packets	Successfully Transmitted		
40					
100					
200					
400					
800					
1000					
2000					
4000					

RESULT:

Thus a scenario for the CSMA/CD protocol have been created and implemented and its performance studied.

SAMPLEVIVAQUESTIONS:

1. What is CSMA/CD?
2. What is CSMA/CA?
3. Explain procedure for CSMA/CD.
4. Explain procedure for CSMA/CA.

EXP. NO:12	PERFORMANCE ANALYSIS OF WIRELESS LAN (CSMA/ CA)
DATE:	

AIM:

To study the Wireless LAN protocol and to analyze its throughput under different transmission power levels and increase in number of clients.

REQUIREMENTS:

PCs (with FTP Server), Windows PC with USB interface (to plug the Wireless adapters) and Net Stumbler software.

THEORY:

A **wireless LAN** (**WLAN**) is a wireless computer network thatl inks two or more devices using wireless communication to form a local area network (LAN) within a limited area. All components that can connect into a wireless medium in a network are referred to as stations (STA). All stations are equipped with wireless network interface controllers(WNICs).Wireless stations fall into two categories: wireless access points, and clients. Access points(APs), normally wireless routers, are base stations for the wireless network. They transmit and receive radiofrequencies for wireless enabled devices to communicate with. Wireless clients can be mobile devices. The basic service set (BSS) is a set of all stations that can communicate with each other at PHY layer. Every BSS has an identification (ID) called the BSSID, which is the MAC address of the access point servicing the BSS.

There are two types of BSS: Independent BSS (also referred to as IBSS), and infrastructure BSS. An independent BSS (IBSS) is an ad hoc network that contains no accesspoints, which means they cannot connect to any other basic service set.

An extended service set (ESS) is a set of connected BSSs. Access points in an ESS are connected by a distribution system. Each ESS has an ID called the SSID which is a 32-byte(maximum) character string.

A distribution system (DS) connects access points in an extended service set. The concept of a DS can be used to increase network coverage through roaming between cells. D Scan be wired or wireless. Current wireless distribution systems are mostly based on WDS or MESH protocols, though other systems are in use.

The IEEE 802.11 has two basic modes of operation: **infrastructure** and **ad hoc** mode. In ad hoc mode, mobile units transmit directly peer-to-peer. In infrastructure mode, mobile units communicate through an access point that serves as a bridge to other networks.

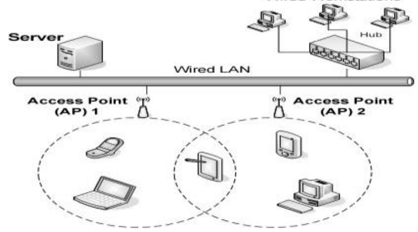
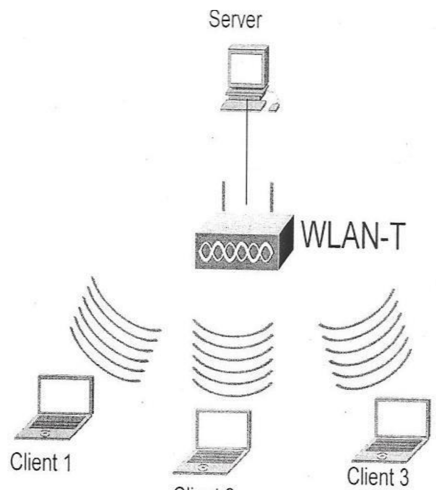


Fig:12.1 Architecture of WLAN



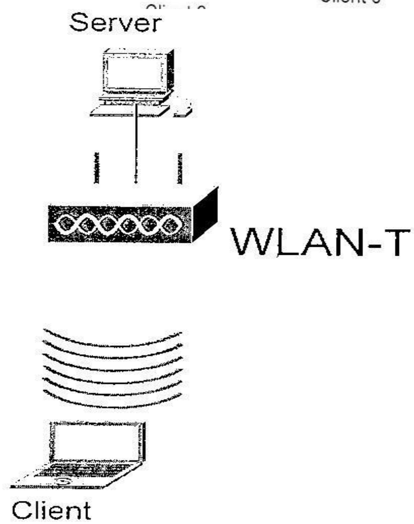


Fig12.2: Throughput measurement using multiple clients and for various input power.

PROCEDURE:

No. of clients Vs Throughput:

1. Configure the WLAN-Tto802.11g mode.
2. Connect the FTP server to WLAN-T as shown in the figure.
3. Start the FTP in client1.

(Use the same file throughout this experiment for through put measurement)

4. Note down the throughput displayed in client1 and tabulate.
5. Initiate FTP session with server using two clients simultaneously. The same file is to be retrieved in both the clients.

(Choose a suitable location for the clients such that all of them show excellent reception power)

6. Note down the through put displayed in client1 and tabulate.
7. Repeat step5 and 6 by increasing the number of clients.
8. Plot a graph for No.of clients Vs throughput.

Power Level Vs Throughput:

1. Configure WLAN-T for 802.11g mode.
2. Configure the client as shown in the figure and initiate a FTP session.

(The Client should be kept in a suitabled is tance from the WLAN-T where the effect of change in the TX powerlevel of the WLAN-T is felt by the client)

3. Keep the power level of WLAN-T to the full level.
4. Measure the signal strength using NetStumbler software and tabulate.
5. Run the FTP session in the Client as mentioned in the previous experiment.
6. Note down the throughput and tabulate.
7. Now vary the transmit power level of the WLAN-T to 50%, 25%, 10% and 5%andnotedownthesignalstrength&throughputforeachpowerlevel andtabulate.
8. Plot a graph for Powerlevel Vs throughput.

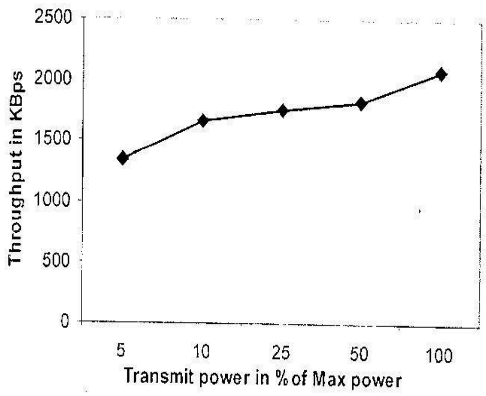
OBSERVATION:

No. of clients Vs Throughput:

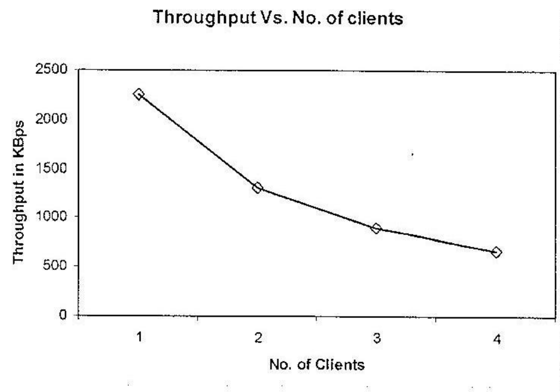
No.of clients	Throughput (Kbps)
1	
2	
3	

PowerLevel Vs Throughput:

PowerLevel	Throughput (Kbps)
5%	
10%	
25%	
50%	
100%	



Transmit power Vs throughput



No. of clients Vs Throughput

Fig12.3: Sample Graph of Performance analysis of wireless LAN

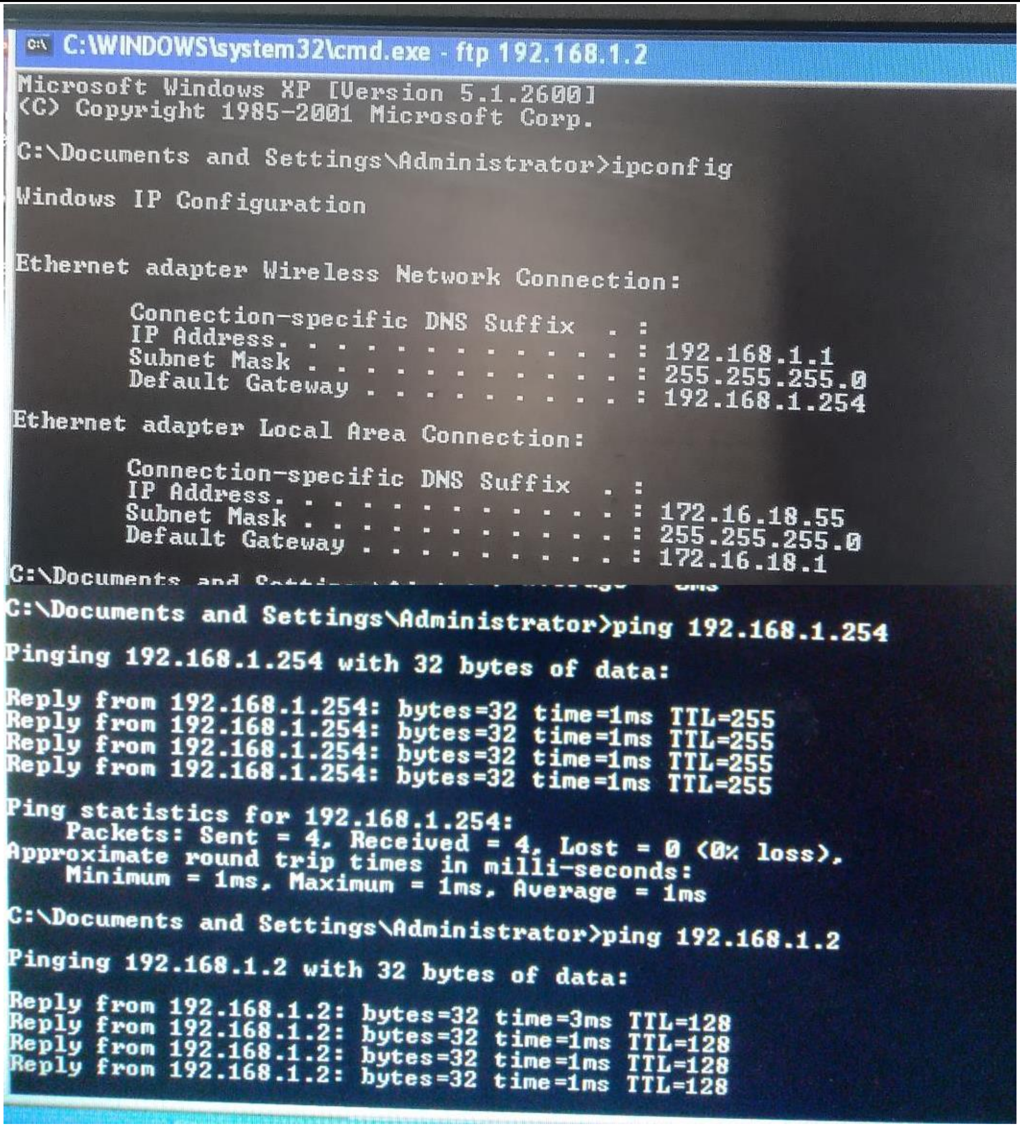


Fig12.4:Wireless LAN

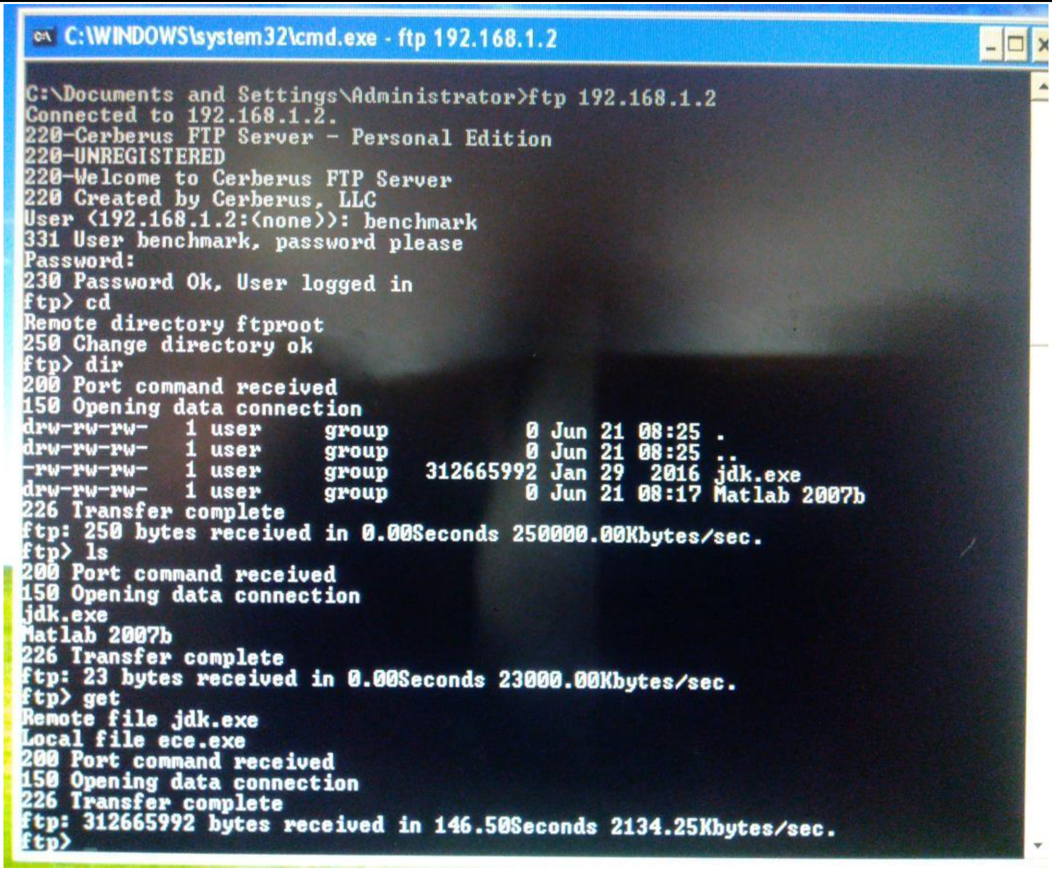


Fig12.5Wireless LANoutputwindow

RESULT:

Thus the Wireless LAN protocol is studied and its throughput performance over no.of clients and powerlevels have been analysed.

<u>SAMPLE VIVA QUESTIONS</u>:

1. What happens when you type a URL in web browser?
2. What is DHCP, how does it work?
3. What is ARP, how does it work?

EXP. NO:13	STUDY OF HDLC PROTOCOL
DATE:	

AIM

To study HDLC (High-level Data Link Control), a transmission protocol**.**

THEORY

HDLC **-** Short for High-level Data Link Contr**ol**, a transmission protocol used at the datalink layer (layer 2) of the OSI seven layer model for data communications. The HDLC protocol embeds information in a data frame that allows devices to **c**ontrol data flow and correct errors**'** is a bit oriented protocol that supports both half-duplex and full-duplex communication overpoint to point & multipoint link.

For any HDLC communications session, one station is designated primary and the other secondary. A session can use one of the following connection modes, which determine how the primary and secondary stations interact.

- **Normal unbalanced**: The secondary station responds only to the primary station.
- **Asynchronous**: The secondary station can initiate a message.
- **Asynchronous balanced**: Both stations send and receive over its part of a

duplex line. This mode isused for X.25 packet-switching networks.

The Link Access Procedure-Balanced (LAP**-**B)and Link Access Procedure D-channel(LAP-D)protocols are subsets of HDLC**.**

LAPB is a bit-oriented synchronous protocol that provides complete data transparency in a full-duplex point-to-point operation. It supports a peer-to-peer link in that neither end of the link plays the role of the permanent master station. HDLC NRM, on the other hand, has a permanent primary station with one or more secondary stations.

HDLC LAPB is a very efficient protocol, which requires a minimum of overhead to ensure flow control, error detection and recovery. If data is flowing in both directions (fullduplex), the data frames themselves carry all the information required to ensure data integrity.

The concept of a frame window is used to send multiple frames before receiving confirmation that the first frame has been correctly been received. This means that data can continue to flow in situations where there may be long but this kind of situation occurs, for instance in satellite communication.

Types of Frames in HDLC

HDLC defines three types of frames:

1. Information frames(I-frame)
2. Supervisory frame(S-frame)
3. Unnumbered frame(U-frame)

1. Information frames

• I-frames carry user's data and control information about user's data.

• I-frame carries user data in the information field.

• TheI-frame format is shown in diagram.

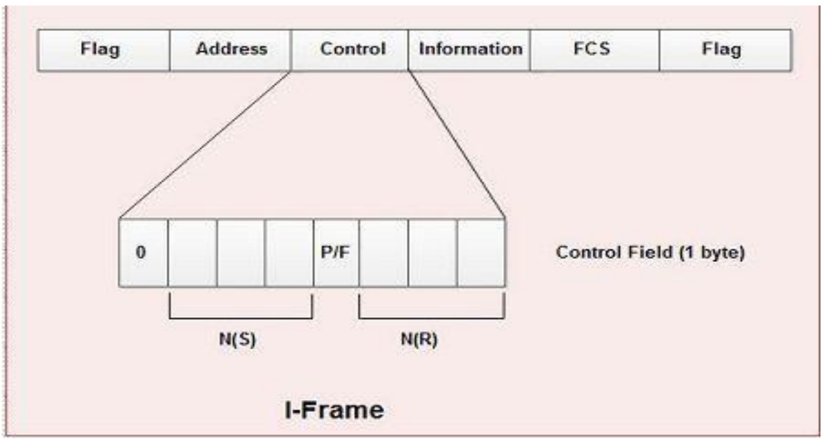


Fig13.1. HDLC I-Frame

• The first bit of control field is always zero, *i.e.* the presence of zero at this place indicates that it is I-frame.

• Bit number 2, 3 & 4 in control field is called N(S) that specifies the sequence number of the frame. Thus it specifies the number of the frame that is currently being sent. Since it is a 3. bitfield, only eight sequence numbers are possible 0,1,2,3,4,5,6,7(000to111).

• Bit number 5 in control field is P/F i.e. Poll/Final and is used for these two purposes. It has, meaning only when it is set i.e. when P/F=1.It can represent the following two cases.

(i) It means poll when frame is sent by a primary station to secondary (when address field contains the address of receiver).

(ii) It means final when frame is sent by secondary to a primary (when the address field contains the address of the sender).

• Bitnumber6,7, and8 in control field specifies N(R) i.e. the sequence number of the frame expected inreturn in two-way communication.

If last frame received was error-free then N(R) number will be that of the next frame is sequence. If the last frame was not received correctly, the N(R) number will be the number of the damaged frame, asking for its retransmission.

2. Supervisory frame

• S-frame carries control information, primarily datalink layerflow and errorcontrols.

• It does not contain information field.

• The format of S-frame is shown in diagram.

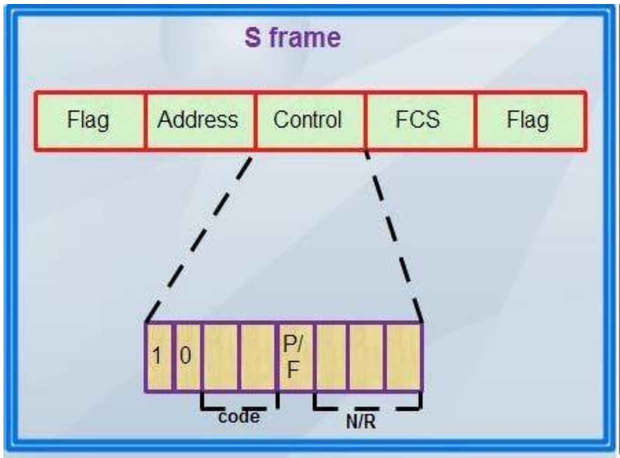


Table: Types of S-frame

Code	Command
00	RR Receive Ready
01	REJ Reject
10	RNR Receive Not Ready
11	SREJ Selective Reject

Fig13.2.HDLC

S-Frame

• The first two bits in the control field of S-frame are always 10.

• Then there is a bit code field that specifies four types of S-frame with combination 00,01,10,11as shown in table:-

1. RR, Receive Ready-used to acknowledge frames when noI-frames are available to piggy back the acknowledgement.
2. REJ Reject-used by the receiver to send a NAK when error has occurred.
3. RNR Receive Not Ready-used for flow control.
4. SREJ Selective Reject-indicates to the transmitter that it should retransmit the frame indicated in the N(R) subfield.

• There is no N(S) field in control field of S-frame as S-frames do not transmit data.

• *P/F* bit is the fifth bit and serves the same purpose as discussed earlier.

• Last three bits in control field indicates N(R) *i.e.* they correspond to the ACK or NAK value.

3. Unnumbered frame

• U-frames are reserved for system management and information carried by them is used for managing the link

• U-frames are used to exchange session management and control information between the two connected devices.

- Information field in U-frame does not carry user information rather, it carries system management information.
- The frame format of U-frame is shown in diagram.
- U-frame is identified by the presence of 11 in the first- and second-bit position in control field.
- These frames do not contain N(S) or N(R) in control field.

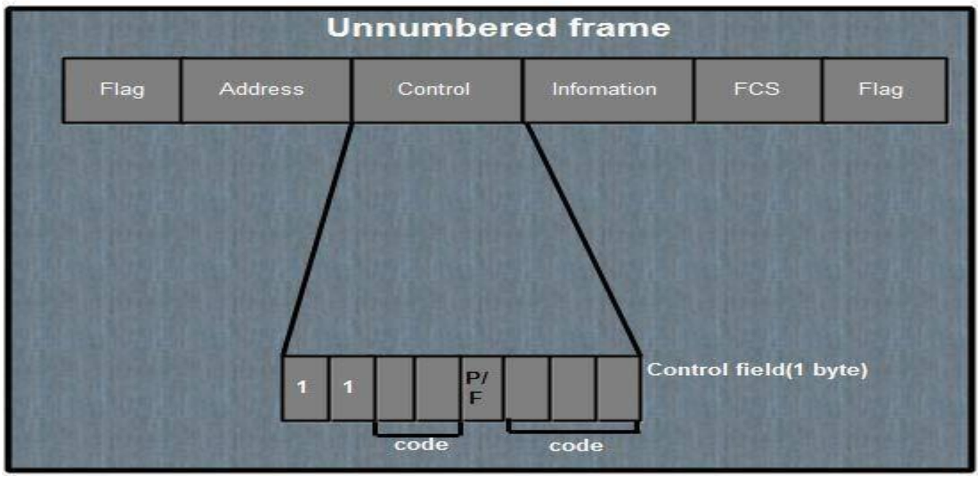


Fig13.2. HDLC Frame

- U-frame contains two code fields, one two hit and other three bit.
- These five bit scan create upto 32 different U-frames.
- .*P/F* bit in control field has same purpose in V-frame as discussed earlier.

Protocol Structure -HDLC: High Level DataLink Control Flag-The value of the flag is always (0x7E).

Address field - Defines the address of the secondary station which is sending the frame or the destination of the frame sent by the primary station. It contains Service Access Point (6bits), a Command/Response bit to indicate whether the frame relates to information frames (I-frames) being sent from the node or received by the node, and an address extension bit which is usually set to true to indicate that the address is of length one byte. When set to false it indicates an additional byte follows.

Extended address - HDLC provides another type of extension to the basic format. The address field may be extended to more than one byte by agreement between the involved parties.

Control field - Serves to identify the type of the frame. In addition, it includes sequence numbers, control features and error tracking according to the frame type.

FCS - The Frame Check Sequence (FCS) enables a high level of physical error control by allowing the integrity of the transmitted frame data to be checked.

Bit Stuffing:

To fill bit frames, the position where the new bits are stuffed is communicated to the receiving end of the data link. The receiver removes the extra bits to return the bit streams to their original bit rate. This is used when a communication protocol requires a fixed frame size. Bits are inserted to make the frame size equal to the defined frame size.

Bits stuffing also works to limit the number of consecutive bits of the same value included in the transmitted data for run-length limited coding. This procedure includes a bit of the opposite value after the maximum allowed number of consecutive bits of the same value.

For instance, if a number of zero bits are transmitted consecutively, the receiving end loses synchronization because a lot of time has passed without voltage sensing. Using bit stuffing, sets of bits beginning with the number one are stuffed into streams of zeros at specific intervals. The receiver does not require any extra information regarding the bit location when the extra bits are removed. Such bit stuffing is done to ensure reliable data transmission and ensure that transmissions start and end at the right places, among other purposes.

A standard HDLC packet begins and ends with 01111110. To make sure this sequence doesn't appear again before the end of the packet, a0 is inserted after every five consecutive1s.

RESULT:

Thus the HDLC protocol is studied.

SAMPLEVIVAQUESTIONS:

1. What is HDLC?
2. What is I-frame?
3. What is S-Frame?
4. What is Bit-Stuffing?
5. What is Link Control?