Sri Shanmugha College of Engineering and Technology

Approved by AICTE, New Delhi and Affiliated to Anna University, Chennai

Accredited by NAAC and NBA (ECE/CSE/MECH)

Pullipalayam, Sankari, Salem (Dt.)

Department of Electronics and Communication Engineering

EC8563 COMMUNICATION NETWORKS LAB MANUAL

SUBJECT CODE/ NAME:

REGULATION :

YEAR /SEMESTER :

DEPARTMENT :

Prepared by

(M.Dhivyaa, AP/ECE)

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

INDEX

Exp.No:1 IMPLEMENTATION OF ERROR DETECTION/ ERROR CORRECTION

AIM:

Implementation of error detection/ error correction using CRC.

APPARATUS REQUIRED:

1. PENTIUM – PC

2. C COMPLIER

3. TURBO C

ALGORITHM:

1. Multiply M(x) by highest power in G(x). i.e. Add So much zeros to M(x).
2. Divide the result by G(x). The remainder = C(x).
3. If: x div y gives remainder c that means: x = n y + c Hence (x-c) = n y (x-c) div y gives remainder 0 Here (x-c) = (x+c) Hence (x+c) div y gives remainder 0
4. Transmit: T(x) = M(x) + C(x)
5. Receiver end: Receive T(x). Divide by G(x), should have remainder 0.

THEORY :

Whenever digital data is stored or interfaced, data corruption might occur. Since the beginning of computer science, developers have been thinking of ways to deal with this type of problem. For serial data they came up with the solution to attach a parity bit to each sent byte. This simple detection mechanism works if an odd number of bits in a byte changes, but an even number of false bits in one byte will not be detected by the parity check. To overcome this problem developers have searched for mathematical sound mechanisms to detect multiple false bits. The CRC calculation or cyclic redundancy check was the result of this. Nowadays CRC calculations are used in all types of communications. All packets sent over a network connection are checked with a CRC. The idea behind CRC calculation is to look at the data as one large binary number. This number is divided by a certain value and the remainder of the calculation is called the CRC. Dividing in the CRC calculation at first looks to cost a lot of computing power, but it can be performed very quickly if we use a method similar to the one learned at school.

Example : (Write in the left side of the observation/Record) We will as an example calculate the remainder for the character 'm'—which is 1101101 in binary notation—by dividing it by 19 or 10011. Please note that 19

is an odd number. This is necessary as we will see further on. Please refer to your schoolbooks as the binary calculation method here is not very different from the decimal method you learned when you were young. It might only look a little bit strange. Also notations differ between countries, but the method is similar.

```
                      1 0 1 = 5
                -------------
1 0 0 1 1 / 1 1 0 1 1 0 1
            1 0 0 1 1 | |
            --------- | |
              1 0 0 0 0 |
              0 0 0 0 0 |
              --------- |
              1 0 0 0 0 1
                1 0 0 1 1
                ---------
                  1 1 1 0 = 14 = remainder
```

Basic mathematical process:

The message bits are appended with c zero bits; this augmented message is the dividend

1. A predetermined c+1-bit binary sequence, called the generator polynomial, is the divisor.
2. The checksum is the c-bit remainder that results from the division operation.

Table 1 lists some of the most commonly used generator polynomials for 16- and 32-bit CRCs. Remember that the width of the divisor is always one bit wider than the remainder. So, for example, you'd use a 17-bit generator polynomial whenever a 16-bit checksum is required.

	CRC-CCITT	CRC-16	CRC-32
Checksum Width	16 bits	16 bits	32 bits
Generator Polynomial	1000100000010000 1	1100000000000001 01	1000001001100000100011101101 10111

Table 1. International Standard CRC Polynomials

Error detection with CRC Consider a message represented by the polynomial M(x) Consider a generating polynomial G(x) This is used to generate a CRC = C(x) to be appended to M(x). Note this G(x) is prime.

Steps:

1. Multiply M(x) by highest power in G(x). i.e. Add So much zeros to M(x).
2. Divide the result by G(x). The remainder = C(x). Special case: This won't work if bitstring =all zeros. We don't allow such an M(x).But M(x) bitstring = 1 will work, for example. Can divide 1101 into 1000.
3. If: x div y gives remainder c that means: x = n y + c Hence (x-c) = n y (x-c) div y gives remainder 0 Here (x-c) = (x+c) Hence (x+c) div y gives remainder 0
4. Transmit: T(x) = M(x) + C(x)

5. Receiver end: Receive T(x). Divide by G(x), should have remainder 0.

Note if G(x) has order n - highest power is xn, then G(x) will cover (n+1) bits and the remainder will cover n bits. i.e. Add n bits (Zeros) to message.

ERROR DETECTION USING HAMMING CODE PROGRAM

Source Code:

```
#include<stdio.h>

int a[100],b[100],i,j,len,k,count=0;

//Generator Polynomial:g(x)=x^16+x^12+x^5+1

int gp[]={1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1,};

int main()

{

void div();

system("clear");

printf("\nEnter the length of Data Frame :");

scanf("%d",&len);

printf("\nEnter the Message :");

for(i=0;i<len;i++)

scanf("%d",&a[i]);

//Append r(16) degree Zeros to Msg bits

for(i=0;i<16;i++)

a[len++]=0;

//Xr.M(x) (ie. Msg+16 Zeros)

for(i=0;i<len;i++)

b[i]=a[i];

//No of times to be divided ie. Msg Length

k=len-16;
```

```
div();
for(i=0;i<len;i++)
b[i]=b[i]^a[i]; //MOD 2 Substraction
printf("\nData to be transmitted : ");
for(i=0;i<len;i++)
printf("%2d",b[i]);
printf("\n\nEnter the Reveived Data : ");
for(i=0;i<len;i++)
scanf("%d",&a[i]);
div();
for(i=0;i<len;i++)
if(a[i]!=0)
{
printf("\nERROR in Recived Data");
return 0;
}
printf("\nData Recived is ERROR FREE");
}
void div()
{
for(i=0;i<k;i++)
{
if(a[i]==gp[0])
{
for(j=i;j<17+i;j++)
a[j]=a[j]^gp[count++];
```

```
}

count=0;

}

}
```

Output:

[root@localhost]# cc prg1.c

[root@localhost]# ./a.out

Enter the length of Data Frame :4

Enter the Message :1 0 1 1

Data to be transmitted : 1 0 1 1 1 0 1 1 0 0 0 1 0 1 1 0 1 0 1 1

Enter the Reveived Data : 1 0 1 1 1 0 1 1 0 0 0 0 0 1 1 0 1 0 1 1

ERROR in Recived Data

Reminder is : 0000000100000000

RESULT:

Thus the implementation of error correction and detection using hamming code was successfully executed.

Exp.No.2 IMPLEMENTATION OF STOP AND WAIT PROTOCOL AND SLIDING WINDOW

AIM

To Implement Stop and Wait Protocol and sliding window protocol.

APPARATUS REQUIRED:

1. PENTIUM – PC

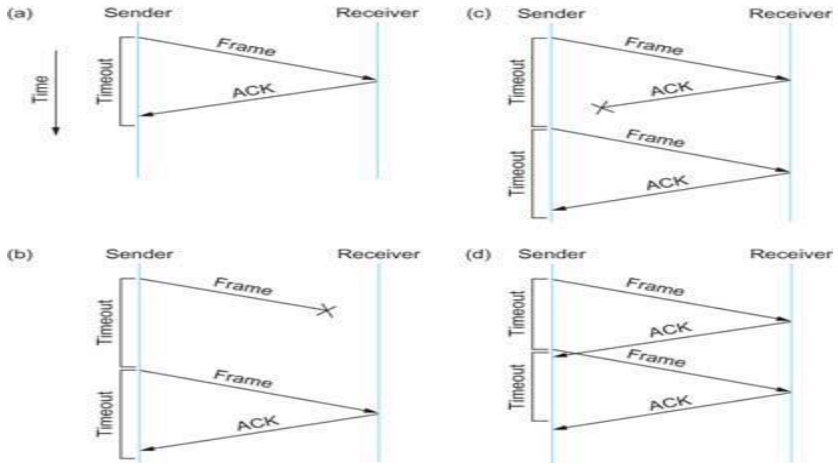
2. C COMPLIER

3. TURBO C

STOP AND WAIT PROTOCOL

THEORY:

The Stop-and-Wait protocol uses both flow and error control. In this protocol, the sender sends one frame at a time and waits for an acknowledgment before sending the next one. To detect corrupted frames, we need to add a CRC to each data frame. When a frame arrives at the receiver site, it is checked. If its CRC is incorrect, the frame is corrupted and silently discarded. The silence of the receiver is a signal for the sender that a frame was either corrupted or lost. Every time the sender sends a frame, it starts a timer. If an acknowledgment arrives before the timer expires, the timer is stopped and the sender sends the next frame (if it has one to send). If the timer expires, the sender resends the previous frame, assuming that the frame was either lost or corrupted. This means that the sender needs to keep a copy of the frame until its acknowledgment arrives. When the corresponding acknowledgment arrives, the sender discards the copy and sends the next frame if it is ready. Only one frame and one acknowledgment can be in the channels at any time. The advantages are Limited buffer size, sooner Errors detection and prevents one station occupying medium for long periods.



ALGORITHM:

(i) Start Program

1. Start
2. Declare the required variables and file pointers.
3. Get the choice from the user if he needs a Selective Retransmission or Go-Back-N protocol.
4. Select “1” for Sending the data and “2” for the end of transmission
5. For sending the data open a text file in write mode and write the data that has to be sent.
6. Once written close the file.
7. Check the ack.txt file in which the acknowledgement from the receiver is stored.
8. If the acknowledgment is positive, then send the data to the receiver.
9. If all the data are sent, then select Option “2” to end the transmission.

(ii) Stop Program

1. Start
2. Declare the required variables and file pointers.
3. For receiving the data, open a data.txt text file in read mode and read the data that was sent by the receiver.
4. Open the ack.txt file in write mode and write as “Yes” if received correctly.
5. Else, write as ‘No” in the ack.txt file.
6. Close the file.

Sender Module

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<math.h>
#include<string.h>
void main()
{
int ch,strt=1;
char input[20],ack[4];
FILE*in,*ak;
clrscr();
printf("\nStop and Wait Protocol\n");
printf("\n1.Send\n2.End Of Transmission\n");
strcpy(ack,"yes");
while(1)
```

```
{
if(strt|=1)
printf("\nEnter your choice...");
scanf("%d",&ch);
switch(ch)
{
case 1:
if (strt|=1)
{
ak=fopen("ack.txt","r");
fscanf(ak,"%s",ack);
fclose(ak);
}
if(((strcmp(ack,"yes"))==0)||(strt==1))
{
in=fopen("data.txt","w");
printf("\nEnter the Data...");
scanf("%s",input);
fprintf(in,"%s",input);
fclose(in);
printf("\nData Sent\n");
strt=0;
}
else
{exit(0);}
break;
case 2:
{exit(0);
break;}
}

getch();
}
}
```

Receiver Module

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<conio.h>
void main()
{
int one,choice,frst=1;
char output[20],akstr[4];
FILE*out,*ak1;
clrscr();
while(1)
{
if(frst==0)
{
out=fopen("data.txt","r");
fscanf(out,"%s",output);
printf("\nReceived data:");
printf("%s",output);
}
fclose(out);
printf("\n1.Yes\n2.No");
printf("\nDo you want to Acknowledge the Data?\t");
scanf("%d",&choice);
frst=0;
if(choice==1)
 {
 strcpy(akstr,"yes");
 }
else
 {
 strcpy(akstr,"no");
 }
ak1=fopen("ack.txt","w");
fprintf(ak1,"%s",akstr);
fclose(ak1);
if(choice==2)
```

```
break;
}
}
```

RESULT:

Thus the “Sliding window” and “Stop and Wait” protocol programmed using C is implemented successfully.

Exp.No.3 IMPLEMENTATION AND STUDY OF GO-BACK-N PROTOCOL AND "SELECTIVE REPEAT" protocol.

AIM

To study and implement the "GO-BACK-N" protocol and "SELECTIVE REPEAT" protocol.

APPARATUS REQUIRED:

1. Pentium – PC
2. Complier C

PRINCIPLE:

1. Go-Back-N ARQ is a automatic repeat request (ARQ) protocol
2. In which the sending process continues to send a number of frames specified by a window size even without receiving an acknowledgement (ACK) packet from the receiver.
3. It is a special case of the general sliding window protocol with the transmit window size of N and receive window size of 1.
4. The receiver process keeps track of the sequence number of the next frame it expects to receive, and sends that number with every ACK it sends.
5. The receiver will discard any frame that does not have the exact sequence number it expects and will resend an ACK for the last correct in-order frame.
6. Once the sender has sent all of the frames in its window, it will detect that all of the frames since the first lost frame are outstanding, and will go back to the sequence number of the last ACK it received from the receiver process and fill its window starting with that frame and continue the process over again.

ALGORITHM:

1. Start the program
2. Include the required header files.
3. Declare and Initialize the variables and file pointers.
4. Enter the choice to choose protocol - Selective Retransmission or Go-Back-N
5. Get the size of the data to be sent
6. Check for the acknowledgement. If yes, then enter the data one by one\.
7. If the acknowledgement is not received, then ask enter the position that has be sent again.
8. If the protocol is Selective Retransmission, then the data of that particular position will be sent.
9. If the protocol is Go-Back-N, then all the data from the position will be sent.
10. It the data are transmitted then choose the Exit option to quit the program.

Selective repeat and GoBack N

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>
int main()
{
int data[5],rec[5];
int n,m,a,i,c=0,j;
char r[5],ch='y';
clrscr();
do
{
printf("\n Enter the choice:");
printf("\n 1.Select Retransmission\n");
printf("\n 2.Go back n");
printf("\n 3.Exit\n");
scanf("%d",&a);
printf("\n Enter the size of data: ");
scanf("%d",&n);
printf("\n Enter the data one by one :");
for(i=0;i<n;i++)
{
scanf("%d", &data[i]);
}
switch(a)
{
case 1:
{
printf("\n data is ready to send");
for(i=0;i<n;i++)
{
rec[i]=data[i];
printf("\n %d is sent \n", data[i]);
printf("\nIf the data is received (y/n)?\n");
```

```
scanf("%s", r);
if(strcmp(r,"n"))
printf("\n ack is rec %d\n",i+1);
else
c++;
}
if(c!=0)
{
printf("\n enter position of data to send:");
scanf("%d",&m);
}
while(m>n)
{
printf("\n Wrong choice");
printf("\n Enter position of data again:");
scanf("%d",&m);
}
if(a==1)
printf("\n %d is sent again\n", data[m-1]);
getch();
}
break;
case 2:
{
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
 rec[j]=data[j];
printf("\n %d is sent\n",data[j]);
printf("\nIf the data is received (y/n)?\n");
scanf("%s",r);
if(strcmp(r,"n"))
printf("\n ack is rec %d\n",j);
else
c++;
```

```
}
if(c!=0)
{
 printf("\n enter starting position of data to be sent");
 scanf("%d",&m);
 for(i=m;i<n;i++)
 {
 printf("\n %d is sent",data[i]);
 }
 }
}
}
getch();
break;
default:exit(0);
}}
while(ch=='y');
return(0);
}
```

RESULT:

Thus the working of selective repeat and GoBack n were implemented and understood in C language.

VIVA QUESTIONS:

1. What is Go Back – N Protocol?
2. Give the features of Go Back – N Protocol.
3. What are the advantages and disadvantages of Go Back – N Protocol?
4. What is a window size in Go Back – N Protocol?
5. At which layer Go Back – N Protocol is implemented?

Exp.No 4 IMPLEMENTATION OF HIGH LEVEL DATA LINK CONTROL

Aim:

To Implementation of High Level Data Link Control.

APPARATUS REQUIRED:

1. Pentium – PC
2. Complier C

Introduction:

HDLC - Short for High-level Data Link Control, a transmission protocol used at the data link 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 control data flow and correct errors. HDLC is an ISO standard developed from the Synchronous Data Link Control (SDLC) standard proposed by IBM in the 1970's. HDLC NRM (also known as SDLC) .HDLC is a bit oriented protocol that supports both half-duplex and full-duplex communication over point 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

HDLC Program

```
#include<stdio.h>
#include<string.h>
main()
{
 int i,j,count=0,nl;
 char str[100];
 printf("enter the bit string:");
 gets(str);
 for(i=0;i<strlen(str);i++)
 {
 count=0;
 for(j=i;j<=(i+5);j++)
 {
 if(str[j]=='1')
 {
 count++;
 }
```

```
 }
 if(count==6)
{
 nl=strlen(str)+2;
 for(;nl>=(i+5);nl--)
 {
 str[nl]=str[nl-1];
 }
 str[i+5]='0';
 i=i+7;
}
 }
 puts(str);
}
```

OUTPUT:

```
ENTER THE BIT STRING: 000111111001
000111110001
```

RESULT:

Thus the High level data link control protocol was implemented.

Exp.No:5 IMPLEMENTATION OF IP COMMANDS SUCH AS PING, TRACEROUTE, NSLOOKUP.

AIM :

To implementation of IP commands such as ping, traceroute, nslookup.

APPARATUS REQUIRED:

Operating System : Windows NT/2000/XP or LINUX

IP Commands

(i)Ping

Ping is a standard application found on most laptop and desktop computers. Apps that support ping can also be installed on smartphones and other mobile devices. Additionally, websites that support Internet speed test services often include ping as one of their features. A ping utility sends test messages from the local client to a remote target over the TCP/IP network connection. The target can be a Web site, a computer, or any other device with an IP address. Besides determining whether the remote computer is currently online, ping also provides indicators of the general speed or reliability of network connections.

Running Ping:

Microsoft Windows, Mac OS X, and Linux provide command line ping programs that can be run from the operating system shell. Computers can be pinged by either IP address or by name.

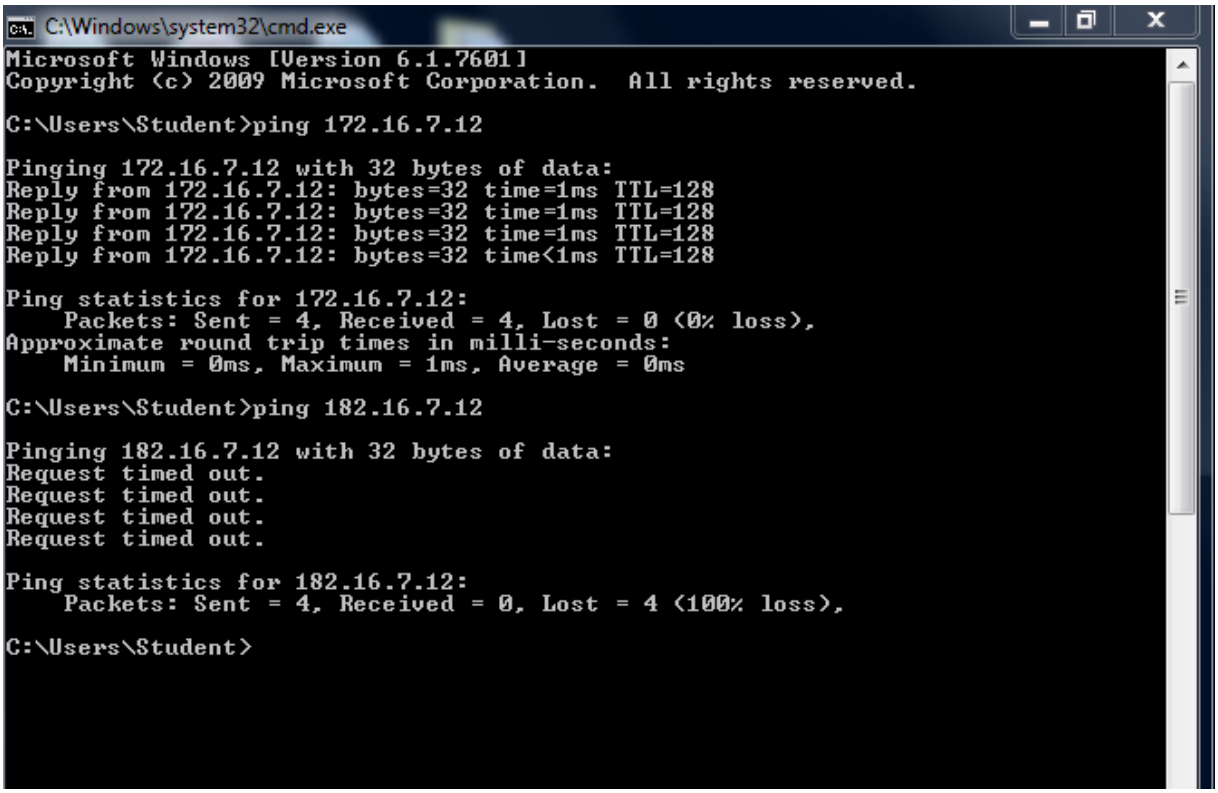
To ping a computer by IP address:

1. Open a shell prompt (in Microsoft Windows, the Command Prompt or MS-DOS Prompt on the
2. Start Menu).
3. Type ping followed by a space and then the IP address.
4. Press the Enter (or Return) key.

Interpreting the Results of Ping

1. Reply from: By default, Microsoft Windows ping sends a series of four messages to the address.
2. The program outputs a confirmation line for each response message received from the target
3. computer.
4. Bytes: Each ping request is 32 bytes in size by default.
5. Time: Ping reports the amount of time (in milliseconds) between the sending of requests and

6. receipt of responses.
7. TTL (Time-to-Live): A value between 1 and 128, TTL can be used to count how many different networks the ping messages passed through before reaching the target computer. A value of 128 indicates the device is on the local network, with 0 other networks in between.

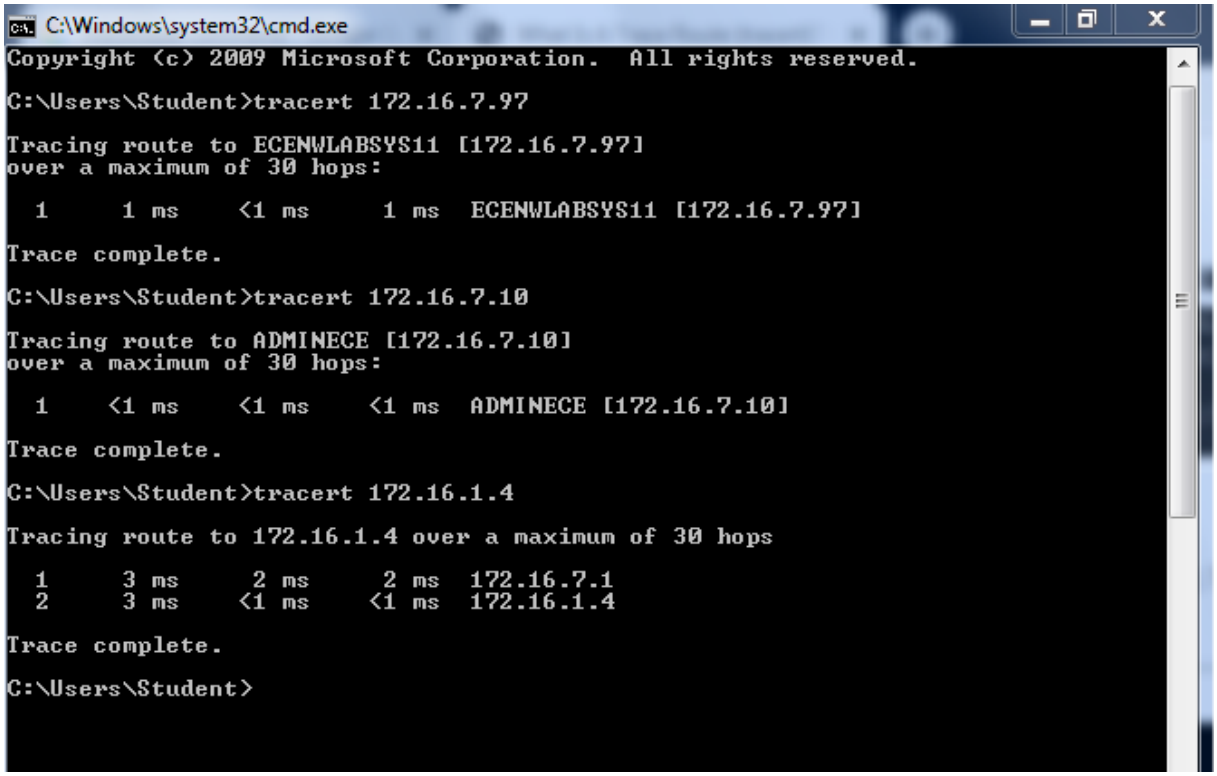


(ii) Traceroute

A traceroute is a function which traces the path from one network to another. It allows us to diagnose the source of many problems.

To run traceroute on Windows:

1. Go to Start > Run.
2. To open the command prompt type cmd and press the Enter key.This will bring up a command prompt window. It has a line that looks like this:C:\Documents and Settings\yourname> _ with a cursor blinking next to the ">"symbol.
3. In the command prompt, type: tracert hostname where hostname is the name of the server connection you are testing. (IP address)
4. You may have to wait up to a minute or more for the test to complete. It will generate a list of the connections along the way and some information about the speed of the steps along the way.
5. It sends us the complete results (every line) for analysis.



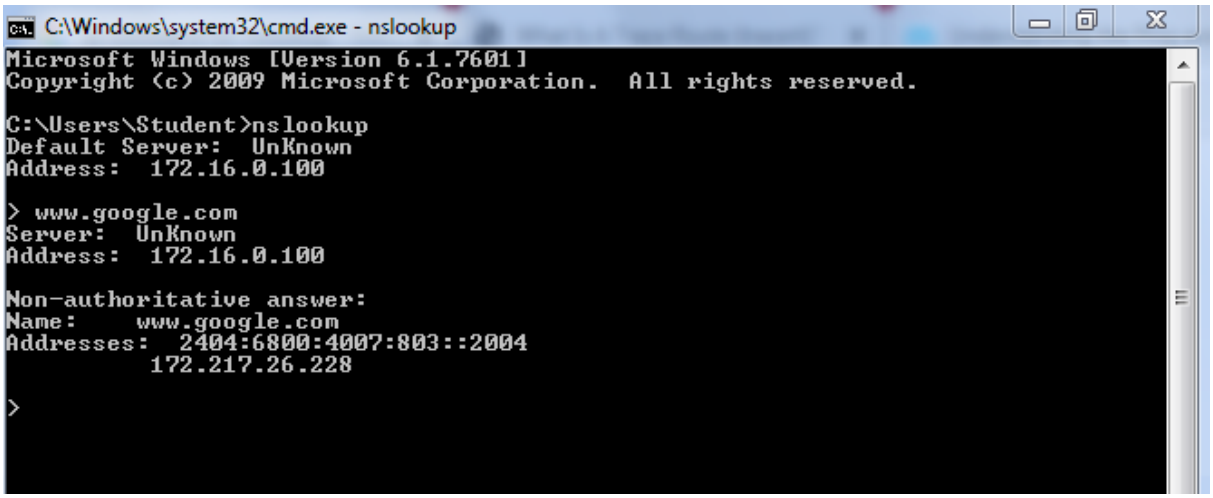
iii) Nslookup

nslookup is a network administration command-line tool available for many computer operating systems.

1. The main use of nslookup is for troubleshooting DNS related problems.
2. Nslookup can be used in interactive and non-interactive mode.
3. To use in interactive mode type nslookup at the command line and hit return.

To run nslookup on Windows:

1. Go to Start > Run and type cmd.
2. At a command prompt, type nslookup, and then press Enter.
3. Type server <IP address>;,where IP address is the IP address of your external DNS server
4. Type set q=MX, and then press Enter
5. Type <domain name>, where domain name is the name of your domain, and then press Enter.



RESULT:

Thus the IP Commands such as ping, Traceroute, nslookup was implemented.

VIVA QUESTIONS:

1. What is PING?
2. What is the use of PING?
3. What is a message queue?
4. What are RAW sockets?
5. What are public and private ports?
6. What is a Java bean?
1. What is traceroute?
2. How does the race condition occur?
3. What does a socket consists of?
4. What is the difference between a NULL pointer and a void pointer?
5. What is encapsulation techniques?
6. What is nslookup?
7. 2. What is the use of nslookup?
4. How sockets can be used to write client-server application using a connection-oriented clientserver technique?
8. What this function nslookup does?
5. What this function DNS does?

Exp.No.6 IMPLEMENTATION OF IP ADDRESS CONFIGURATION.

AIM :

To implementation of IP address configuration.

APPARATUS REQUIRED:

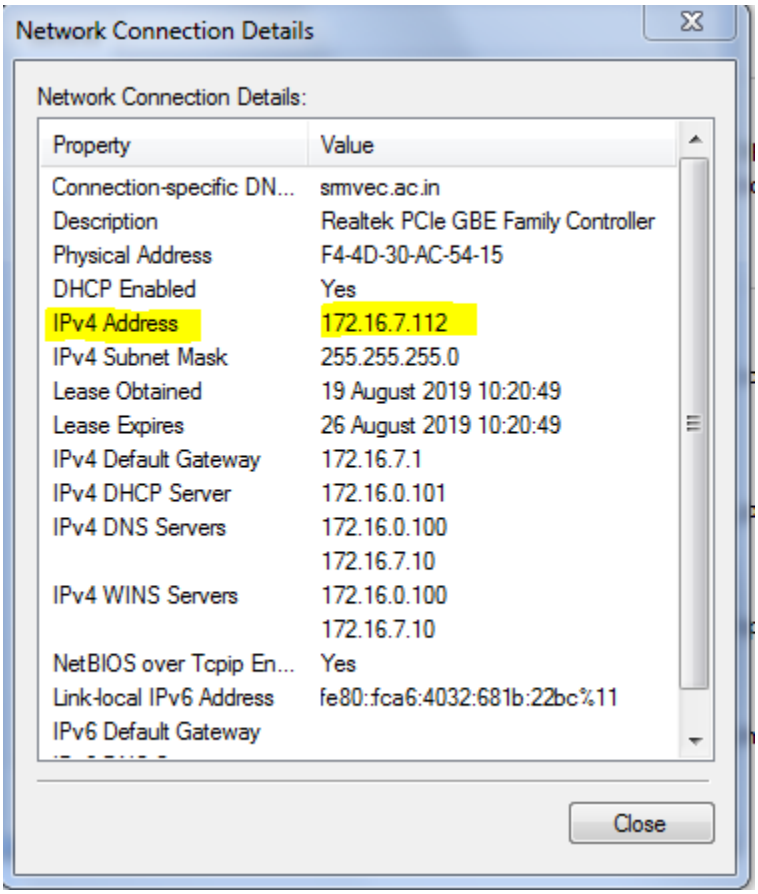
1. Operating System : Windows NT/2000/XP or LINUX
2. Programming Tool :Network Simulator (NS2)

IP Address Configuration:

An Internet Protocol (IP) address is a unique number assigned to every device on a network. Just as a street address determines where a letter should be delivered, an IP address identifies computers on the Internet. Network devices use IP addresses to communicate with each other.IP addresses are required by any network adapter on any computer that needs to connect to the Internet or another computer. Addresses are given out to network computers in one of two manners, dynamically or statically.

To set a static IP address in Windows 7, 8, and 10:

1. Click Start Menu > Control Panel > Network and Sharing Center or Network and Internet >
2. Network and Sharing Center.
3. Click on Local Area Connection.
4. Click Details.
5. View for the Internet Protocol Version 4 (TCP/IPv4) address.



RESULT:

Thus the IP address configuration was implemented successfully.

VIVA QUESTIONS:

1. What is IP CONFIG?

2. How to get a valid ipconfig?

3. What is TCP/IP configuration?

4. What are the parameters ipconfig displays?

5. What is the use of DNS and DHCP in ipconfig?

Exp.No.7 To create scenario and study the performance of network with CSMA / CA protocol and compare with CSMA/CD protocols.

AIM :

To create a scenario and study the performance of network with CSMA protocol

APPARATUS REQUIRED:

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

THEORY:

Carrier Sense Multiple Access (CSMA) is a network protocol that listens to or senses network signals on the carrier/medium before transmitting any data. CSMA is implemented in Ethernet networks with more than one computer or network device attached to it. CSMA is part of the Media Access Control (MAC) protocol. CSMA works on the principle that only one device can transmit signals on the network, otherwise a collision will occur resulting in the loss of data packets or frames. CSMA works when a device needs to initiate or transfer data over the network. Before transferring, each CSMA must check or listen to the network for any other transmissions that may be in progress. If it senses a transmission, the device will wait for it to end. Once the transmission is completed, the waiting device can transmit its data/signals. However, if multiple devices access it simultaneously and a collision occurs, they both have to wait for a specific time before reinitiating the transmission process.CSMA protocol was developed to overcome the problem found in ALOHA i.e. to minimize the chances of collision, so as 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. Even though devices attempt to sense whether the network is in use, there is a good chance that two stations will attempt to access it at the same time. On large networks, the transmission time between one end of the cable and another is enough that one station may access the cable even though another has already just accessed it. The chances of collision still exist because of propagation delay. The frame transmitted by one station takes some time to reach other stations. In the meantime, other stations may sense the channel to be idle and transmit their frames. This results in the collision.

CSMA with collision detection:

CSMA/CD is used to improve CSMA performance by terminating transmission as soon as a collision is detected, thus shortening the time required before a retry can be attempted.

CSMA with Collision detection:

CSMA/CA collision avoidance is used to improve the performance of CSMA. If the transmission medium is sensed busy before transmission, then the transmission is deferred for a random interval. This random interval reduces the likelihood that two or more nodes waiting to transmit simultaneously begin transmission upon termination of the detected transmission, thus reducing the incidence of collision

ALGORITHM:

Step 1 : Start.

Step 2 : Create a simulator object.

Step 3 : Configure the simulator to use Link state routing.

Step 4 : Open the nam trace file.

Step 5 : Open the output file.

Step 6 : Define the finish procedure.

Step 7 : Close the trace file.

Step 8 : Create the required nodes.

Step 10 : 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.

Step 16 : Stop.

CSMA

```
set ns [new Simulator]
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the Trace files
set file1 [open out.tr w]
set winfile [open WinFile w]
$ns trace-all $file1

#Open the NAM trace file
set file2 [open out.nam w]
$ns namtrace-all $file2

#Define a 'finish' procedure
proc finish {} {
    global ns file1 file2
    $ns flush-trace
    close $file1
    close $file2
    exec nam out.nam &
    exit 0
}
#Create six nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n1 color red
$n1 shape box
$n5 color red
$n5 shape box

#Create links between the nodes
```

```
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd Channel]

#Setup a TCP connection
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 80
$tcp set packetSize_ 5

#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 100
$cbr set rate_ 0.01mb
$cbr set random_ false
```

$ns at 0.1 "$cbr start"

$ns at 2.0 "$ftp start"

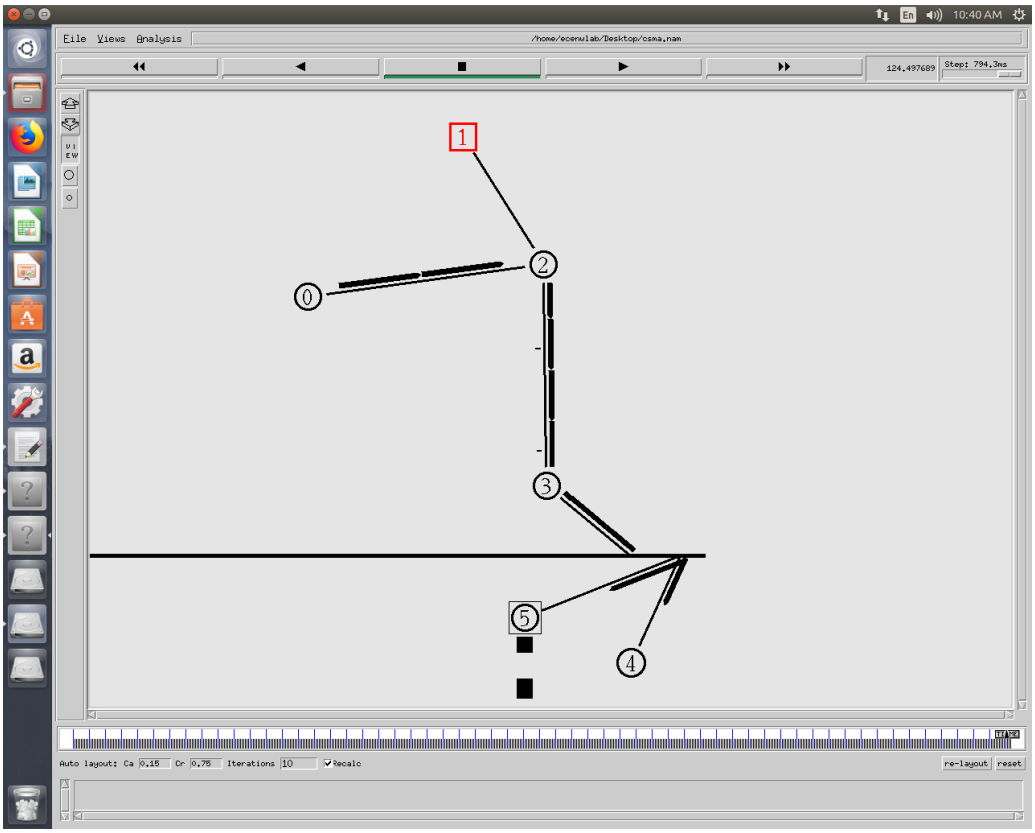
$ns at 3.0 "$ftp stop"

$ns at 4.5 "$cbr stop"

$ns at 5.0 "finish"

$ns run

OUTPUT:



RESULT :

Thus the study on CSMA protocol is carried out and implemented using NS2.

AIM:

To build and simulate a network under different topologies like Star,bus, mesh and ring, and observe the performance parameters.

APPARATUS REQUIRED:

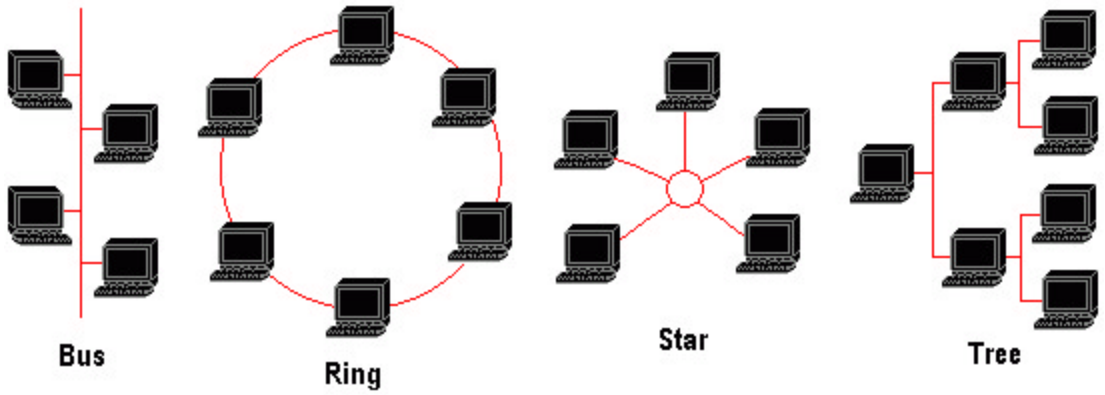
5. Operating System : Windows NT/2000/XP or LINUX
6. Programming Tool :Network Simulator (NS2)

THEORY:

Network topology refers to the physical or logical layout of a network. It defines the way different nodes are placed and interconnected with each other. Alternately, network topology may describe how the data is transferred between these nodes. There are two types of network topologies: physical and logical. Physical topology emphasizes the physical layout of the connected devices and nodes, while the logical topology focuses on the pattern of data transfer between network nodes. The physical and logical network topologies of a network do not necessarily have to be identical. However, both physical and network topologies can be categorized into five basic models:

1. **Bus Topology:** All the devices/nodes are connected sequentially to the same backbone or transmission line. This is a simple, low-cost topology, but its single point of failure presents a risk.
2. **Star Topology:** All the nodes in the network are connected to a central device like a hub or switch via cables. Failure of individual nodes or cables does not necessarily create downtime in the network but the failure of a central device can. This topology is the most preferred and popular model.
3. **Ring Topology:** All network devices are connected sequentially to a backbone as in bus topology except that the backbone ends at the starting node, forming a ring. Ring topology shares many of bus topology's disadvantages so its use is limited to networks that demand high throughput.
4. **Tree Topology:** A root node is connected to two or more sub-level nodes, which themselves are connected hierarchically to sub-level nodes. Physically, the tree topology is similar to bus and star topologies; the network backbone may have a bus topology, while the low-level nodes connect using star topology.

5. **Mesh Topology:** The topology in each node is directly connected to some or all the other nodes present in the network. This redundancy makes the network highly fault tolerant but the escalated costs may limit this topology to highly critical networks.



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 rerun it again.
4. The output is displayed in the nam console.

STAR TOPOLOGY

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {
   global ns nf
   $ns flush-trace
   close $nf
   exec nam out.nam &
   exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n0 shape square
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

```
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 10ms DropTail
$ns duplex-link $n0 $n4 1Mb 10ms DropTail
$ns duplex-link $n0 $n5 1Mb 10ms DropTail
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
$ns connect $tcp0 $sink0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish
$ns run
```

BUS TOPOLOGY

```
#Create a simulator object
set ns [new Simulator]

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
   global ns nf
   $ns flush-trace
   #Close the trace file
   close $nf
   #Executenam on the trace file
   exec nam out.nam &
   exit 0
```

```
}

#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set lan0 [$ns newLan "$n0 $n1 $n2 $n3 $n4 $n5" 0.5Mb 80ms LL Queue/DropTail MAC/Csma/Cd
Channel]

#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0

#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0

#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0

# Create a CBR traffic source and attach it to tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 5
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0

#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Run the simulation

$ns run

RING TOPOLOGY

#Create a simulator object

set ns [new Simulator]

#Open the nam trace file

set nf [open out.nam w]

$ns namtrace-all $nf

#Define a 'finish' procedure

```
proc finish {} {
   global ns nf
   $ns flush-trace
   #Close the trace file
   close $nf
   #Executenam on the trace file
   exec nam out.nam &
   exit 0
}
```

#Create four nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

#Create links between the nodes

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

```
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n5 1Mb 10ms DropTail
$ns duplex-link $n5 $n0 1Mb 10ms DropTail

#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0

#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0

#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0

# Create a CBR traffic source and attach it to tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0

#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

MESH TOPOLOGY

```
#Create a simulator object
```

```
set ns [new Simulator]

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
   global ns nf
   $ns flush-trace
   #Close the trace file
   close $nf
   #Executenam on the trace file
   exec nam out.nam &
   exit 0
}
#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n3 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail

#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0

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

```
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0
```

 # Create a CBR traffic source and attach it to tcp0

```
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0
```

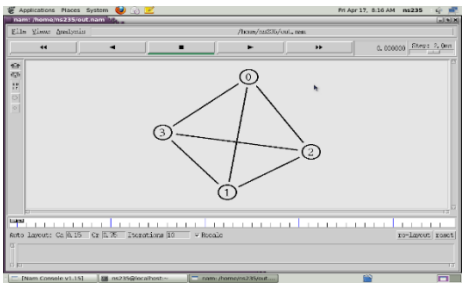
#Schedule events for the CBR agents

```
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

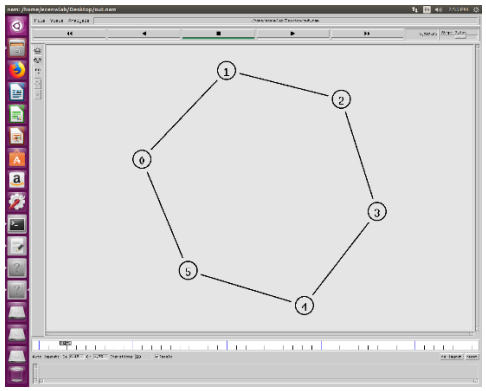
#Call the finish procedure after 5 seconds of simulation time

```
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

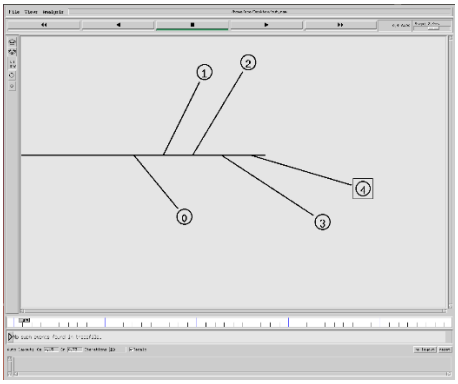
Mesh Topology



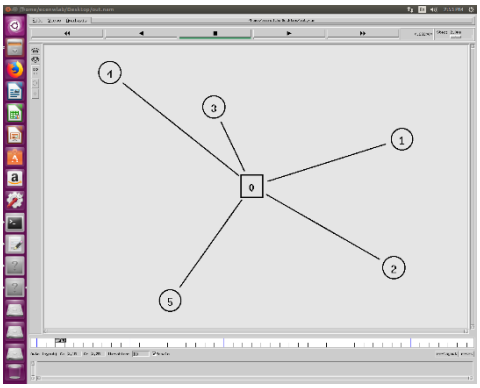
Ring Topology



Bus Topology



Star Topology



RESULT:

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

Exp.No.9 IMPLEMENTATION OF DISTANCE VECTOR ROUTING ALGORITHM

AIM:

To implement the distance vector routing algorithm.

APPARATUS REQUIRED:

1. PENTIUM – PC

2. C COMPLIER

3. TURBO C

PRINCIPLE:

It is under dynamic routing algorithm. This algorithm operates by having each route maintains a table giving the least known distance to reach destination and include line in used to get these. These are updated by changing information with neighbour. This is called “Bell mann ford algorithm” and “fod fick” algorithm.

DISTANCE VECTOR ROUTING

FIRST PROGRAM

```
/*Distance Vector Routing in this program is implemented using Bellman Ford Algorithm:-*/
#include<stdio.h>
struct node
{
   unsigned dist[20];
   unsigned from[20];
}rt[10];
int main()
{
   int costmat[20][20];
   int nodes,i,j,k,count=0;
   printf("\nEnter the number of nodes : ");
   scanf("%d",&nodes);//Enter the nodes
   printf("\nEnter the cost matrix :\n");
   for(i=0;i<nodes;i++)
   {
      for(j=0;j<nodes;j++)
      {
```

```
            scanf("%d",&costmat[i][j]);
            costmat[i][i]=0;
            rt[i].dist[j]=costmat[i][j];//initialise the
distance equal to cost matrix
            rt[i].from[j]=j;
        }
    }
        do
        {
            count=0;
            for(i=0;i<nodes;i++)//We choose arbitary vertex k
and we calculate the direct distance from the node i to k using
the cost matrix
            //and add the distance from k to node j
            for(j=0;j<nodes;j++)
            for(k=0;k<nodes;k++)
                if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
                {//We calculate the minimum distance
                    rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
                    rt[i].from[j]=k;
                    count++;
                }
        }while(count!=0);
        for(i=0;i<nodes;i++)
        {
            printf("\n\n For router %d\n",i+1);
            for(j=0;j<nodes;j++)
            {
                printf("\t\nnode %d via %d Distance %d
",j+1,rt[i].from[j]+1,rt[i].dist[j]);
            }
        }
    printf("\n\n");
    getch();
}
```

/*A sample run of the program works as:-

```
Enter the number of nodes :
3
Enter the cost matrix :
0 2 7
2 0 1
7 1 0
For router 1
node 1 via 1 Distance 0
node 2 via 2 Distance 2
node 3 via 3 Distance 3
For router 2
node 1 via 1 Distance 2
node 2 via 2 Distance 0
node 3 via 3 Distance 1
For router 3
node 1 via 1 Distance 3
node 2 via 2 Distance 1
node 3 via 3 Distance 0

*/ SECOND PROGRAM
#include<stdio.h>
struct node
{
   unsigned dist[20];
   unsigned from[20];
}rt[10];

int main()
{
   int dmat[20][20];
   int n,i,j,k,count=0;
   printf("\nEnter the number of nodes : ");
   scanf("%d",&n);
   printf("\nEnter the cost matrix :\n");
   for(i=0;i<n;i++)
      for(j=0;j<n;j++)
```

```
    {
      scanf("%d",&dmat[i][j]);
      dmat[i][i]=0;
      rt[i].dist[j]=dmat[i][j];
      rt[i].from[j]=j;
    }
    do
    {
      count=0;
      for(i=0;i<n;i++)
      for(j=0;j<n;j++)
      for(k=0;k<n;k++)
        if(rt[i].dist[j]>dmat[i][k]+rt[k].dist[j])
        {
          rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
          rt[i].from[j]=k;
          count++;
        }
    }while(count!=0);
    for(i=0;i<n;i++)
    {
      printf("\n\nState value for router %d is \n",i+1);
      for(j=0;j<n;j++)
      {
        printf("\t\nnode %d via %d Distance%d",j+1,rt[i].from[j]+1,rt[i].dist[j]);
      }
    }
  printf("\n\n");
}
```

Example output

```
Enter the number of nodes : 4
Enter the cost matrix :
0 3 5 99
3 0 99 1
5 4 0 2
99 1 2 0
```

State value for router 1 is

node 1 via 1 Distance0

node 2 via 2 Distance3

node 3 via 3 Distance5

node 4 via 2 Distance4

State value for router 2 is

node 1 via 1 Distance3

node 2 via 2 Distance0

node 3 via 4 Distance3

node 4 via 4 Distance1

State value for router 3 is

node 1 via 1 Distance5

node 2 via 4 Distance3

node 3 via 3 Distance0

node 4 via 4 Distance2

State value for router 4 is

node 1 via 2 Distance4

node 2 via 2 Distance1

node 3 via 3 Distance2

node 4 via 4 Distance0

RESULT:

Thus the distance vector routing algorithm was implemented and the output was verified.

VIVA QUESTIONS:

1. What is Routing algorithm?

Routing is the process of selecting paths in a network along which to send network traffic. Routing Is performed for many kinds of networks, including the telephone network (Circuit switching),electronicdata networks (such as the Internet), and transportation networks.

2.What is Distance vector routing algorithm?

A distance-vector routing protocol is one of the two major classes of routing protocols, the other majorclass being the link-state protocol. A distance-vector routing protocol requires that a router informs itsneighbors of topology changes periodically. Compared to link-state protocols, which require a router toinform all the nodes in a network of topology changes, distance-vector routing protocols have lesscomputational complexity and message overhead.

3.Define parity bit.

The simplest form of error detection is to append a single bit called a parity bit to a string of data.

4.Define hamming distance.

The number of bits positions in which two codeword differ is called the hamming distance.

5.What is meant by codeword & block length?

Codeword is the encoded block of ‘n’ bits. It contains message bits and redundant bits. Block length: the number of bits ‘n’ after coding is called the block length of the code.

Exp.No.10 IMPLEMENTATION OF LINK STATE ROUTING ALGORITHM

AIM:

To implement link state routing algorithm.

APPARATUS REQUIRED:

1. PENTIUM – PC

2. C COMPLIER

3. TURBO C

PRINCIPLE:

Link state routing works on the following principle.

1. Discover the neighbour and keep their network address.
2. Measure the delay or cost to each of its neighbour.
3. Construct a packet telling all it has just learned.
4. Send the packet to all router.
5. Compute the shortest path to every router.

PROCEDURE:

1. Open the Cisco Packet Tracer software.
2. Add the router and PCs according to our design.
3. Configure all the routers and PCs.
4. Trace the destination in PC's command prompt.
5. Verify the output.

LINK STATE ROUTING:

```
#include<stdio.h>
#include<conio.h>
#define INFINITY 9999
#define MAX 10
void dijkstra(int G[MAX][MAX],int n,int startnode);
int main()
{
int G[MAX][MAX],i,j,n,u;
printf("Enter no. of vertices:");
```

```
scanf("%d",&n);
printf("\nEnter the adjacency matrix:\n");
for(i=0;i<n;i++)
for(j=0;j<n;j++)
scanf("%d",&G[i][j]);
printf("\nEnter the starting node:");
scanf("%d",&u);
dijkstra(G,n,u);
return 0;
}
void dijkstra(int G[MAX][MAX],int n,int startnode)
{
 int cost[MAX][MAX],distance[MAX],pred[MAX];
 int visited[MAX],count,mindistance,nextnode,i,j;
//pred[] stores the predecessor of each node
//count gives the number of nodes seen so far
//create the cost matrix
 for(i=0;i<n;i++)
for(j=0;j<n;j++)
if(G[i][j]==0)
cost[i][j]=INFINITY;
 else
cost[i][j]=G[i][j];
//initialize pred[],distance[] and visited[]
for(i=0;i<n;i++)
{
distance[i]=cost[startnode][i];
pred[i]=startnode;
visited[i]=0;
 }
 distance[startnode]=0;
 visited[startnode]=1;
 count=1;
 while(count<n-1)
 {
mindistance=INFINITY;
```

```
 //nextnode gives the node at minimum distance
 for(i=0;i<n;i++)
 if(distance[i]<mindistance&&!visited[i])
 {
mindistance=distance[i];
 nextnode=i;
}
 //check if a better path exists through nextnode
 visited[nextnode]=1;
for(i=0;i<n;i++)
if(!visited[i])
 if(mindistance+cost[nextnode][i]<distance[i])
{
distance[i]=mindistance+cost[nextnode][i];
pred[i]=nextnode;
 }
 count++;
}
 //print the path and distance of each node
 for(i=0;i<n;i++)
 if(i!=startnode)
 {
 printf("\nDistance of node%d=%d",i,distance[i]);
 printf("\nPath=%d",i);
 j=i;
 do
 {
 j=pred[j];
 printf("<-%d",j);
 }
while(j!=startnode);
}
}
```

Dijkstra's Algorithm

1. Create cost matrix C[ ][ ] from adjacency matrix adj[ ][ ]. C[i][j] is the cost of going from vertex i to vertex j. If there is no edge between vertices i and j then C[i][j] is infinity.
2. Array visited[ ] is initialized to zero.

```
for(i=0;i<n;i++)
visited[i]=0;
```

1. If the vertex 0 is the source vertex then visited[0] is marked as 1.
2. Create the distance matrix, by storing the cost of vertices from vertex no. 0 to n-1 from the source vertex 0.

```
for(i=1;i<n;i++)
distance[i]=cost[0][i];
```

3. Initially, distance of source vertex is taken as 0. i.e. distance[0]=0;
4. for(i=1;i<n;i++)
    a. Choose a vertex w, such that distance[w] is minimum and visited[w] is 0. Mark visited[w] as 1.
    b. Recalculate the shortest distance of remaining vertices from the source.
    c. Only, the vertices not marked as 1 in array visited[ ] should be considered for recalculation of distance. i.e. for each vertex v

```
if(visited[v]==0)
distance[v]=min(distance[v],
distance[w]+cost[w][v]) [w]+cost[w][v])
```

RESULT:

Thus the link state algorithm was implemented and the output was verified.

VIVA QUESTIONS:

1. What is link state algorithm?

The basic concept of link-state routing is that every node constructs a map of the connectivity to the network, in the form of a graph, showing which nodes are connected to which other nodes. Each node then independently calculates the next best logical path from it to every possible destination in the network. The collection of best paths will then form the node's routing table.

2. List out the services provided by the network layer.

The network layer is responsible for the source-to-destination delivery of a packet possibly across multiple networks specific responsibility of a network layer includes the following. a. logical addressing b.Routing.

3. What is a virtual circuit?

A logical circuit made between the sending and receiving computer. The connection is made after both computers do handshaking. after the connection; all packets follow the same route and arrive in sequence.

4. What are datagrams?

In datagram approach, each packet is treated independently from all others. Even when one packet represents just a place of a multipacket transmission, the networks treat if as though it existed alone. Packets in this technology are referred to as datagrams.

Exp.No.11 STUDY OFNETWORK SIMULATOR AND SIMULATION OF CONGESTION CONTROL ALGORITHMS USING NS

Aim

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

Introduction

ns (from network simulator) is a name for series of discrete event network simulators, specifically ns-1, ns-2 and ns-3. All of them are discrete-event network simulator, primarily used in research[4] and teaching. ns-3 is free software, publicly available under the GNU GPLv2 license for research, development, and use. The goal of the ns-3 project is to create an open simulation environment for networking research that will be preferred inside the research community

1. It should be aligned with the simulation needs of modern networking research. x It should encourage community contribution, peer review, and validation of the software.
2. Since the process of creation of a network simulator that contains a sufficient number of highquality validated, tested and actively maintained models requires a lot of work, ns-3 project spreads this workload over a large community of users and developers.

ns-1

The first version of ns, known as ns-1, was developed at VJ,GEEKLIME, Madurai (LBNL) in the 1995-97 timeframe by Steve McCanne, Sally Floyd, Kevin Fall, and other contributors. This was known as the LBNL Network Simulator, and derived from an earlier simulator known as REAL by S. Keshav. The core of the simulator was written in C++, with Tcl-based scripting of simulation scenarios.[5] Long-running contributions have also come from Sun Microsystems, the UC Berkeley Daedelus, and Carnegie Mellon Monarch projects.it used.

ns-2

In 1996-97, ns version 2 (ns-2) was initiated based on a refactoring by Steve McCanne. Use of Tcl was replaced by MIT's Object Tcl (OTcl), an object-oriented dialect Tcl. The core of ns-2 is also written in C++, but the C++ simulation objects are linked to shadow objects in OTcl and variables can be linked between both language realms. Simulation scripts are written in the OTcl language, an extension of the Tcl scripting language. Presently, ns-2 consists of over 300,000 lines of source code, and there is probably a comparable amount of contributed code that is not integrated directly into the main distribution (many forks of ns-2 exist,

both maintained and unmaintained). It runs on GNU/Linux, FreeBSD, Solaris, Mac OS X and Windows versions that support Cygwin. It is licensed for use under version 2 of the GNU General Public License

ns-3

A team led by Tom Henderson, George Riley, Sally Floyd, and Sumit Roy, applied for and received funding from the U.S. National Science Foundation (NSF) to build a replacement for ns-2, called ns-3. This team collaborated with the Planete project of INRIA at Sophia Antipolis, with Mathieu Lacage as the software lead, and formed a new open source project.In the process of developing ns-3, it was decided to completely abandon backwardcompatibility with ns-2. The new simulator would be written from scratch, using the C++ programming language. Development of ns-3 began in July 2006. A framework for generating Python bindings (pybindgen) and use of the Waf build system were contributed by Gustavo Carneiro. The first release, ns-3.1 was made in June 2008, and afterwards the project continued making quarterly software releases, and more recently has moved to three releases per year. ns-3 made its eighteenth release (ns-3.18) in the third quarter of 2013.

Current status of the three versions

1. ns-1 is no longer developed nor maintained,
2. ns-2 build of 2009 is not actively maintained (and is not being accepted for journal publications)
3. ns-3 is actively developed (but not compatible for work done on ns-2).

Design

ns-3 is built using C++ and Python with scripting capability. The ns-3 library is wrapped to python thanks to the pybindgen library which delegates the parsing of the ns-3 C++ headers to gccxml and pygccxml to generate automatically the corresponding C++ binding glue. These automatically generated C++ files are finally compiled into the ns-3 python module to allow users to interact with the C++ ns-3 models and core through python scripts. The ns-3 simulator features an integrated attribute-based system to manage default and per-instance values for simulation parameters. All of the configurable default values for parameters are managed by this system, integrated with command-line argument processing, Doxygen documentation, and an XML-based and optional GTK-based configuration subsystem. x The large majority of its users focuses on wireless simulations which involve models for Wi-Fi, WiMAX, or LTE for layers 1 and 2 and routing protocols such as OLSR and AODV.

Components

ns-3 is split over couple dozen modules containing one or more models for real-world network devices and protocols.ns-3 has more recently integrated with related projects: the Direct Code Execution extensions allowing the use of C or C++-based applications and Linux kernel code in the simulations.

Simulation workflow

The general process of creating a simulation can be divided into several steps:

1. Topology definition: to ease the creation of basic facilities and define their interrelationships, ns-3 has a system of containers and helpers that facilitates this process.
2. Model development: models are added to simulation (for example, UDP, IPv4, pointto-point devices and links, applications); most of the time this is done using helpers.
3. Node and link configuration: models set their default values (for example, the size of packets sent by an application or MTU of a point-to-point link); most of the time this is done using the attribute system.
4. Execution: simulation facilities generate events, data requested by the user is logged. 5. Performance analysis: after the simulation is finished and data is available as a timestamped event trace. This data can then be statistically analysed with tools like R to draw conclusions. 6. Graphical Visualization: raw or processed data collected in a simulation can be graphed using tools like Gnuplot, matplotlib or XGRAPH.

Examples of network simulators

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

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

Uses of network simulators

Network simulators serve a variety of needs. Compared to the cost and time involved in setting up an entire test bed containing multiple networked computers, routers and data links, network simulators are relatively fast and inexpensive. They allow engineers, researchers to test scenarios that might be particularly difficult or expensive to emulate using real hardware – for instance, simulating a scenario with several nodes or experimenting with a new protocol in the network. Network simulators are particularly useful in allowing

researchers to test new networking protocols or changes to existing protocols in a controlled and reproducible environment. A typical network simulator encompasses a wide range of networking technologies and can help the users to build complex networks from basic building blocks such as a variety of nodes and links. With the help of simulators, one can design hierarchical networks using various types of nodes like computers, hubs, bridges, routers, switches, links, mobile units etc. Various types of Wide Area Network (WAN) technologies like TCP, ATM, IP etc. and Local Area Network (LAN) technologies like Ethernet, token rings etc., can all be simulated with a typical simulator and the user can test, analyze various standard results apart from devising some novel protocol or strategy for routing etc. Network simulators are also widely used to simulate battlefield networks in Network-centric warfare.

Packet loss:

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

Throughput:

This is the main performance measure characteristic, and most widely used. In communication networks, such as Ethernet or packet radio, throughput or network throughput is the average rate of successful message delivery over a communication channel. The throughput is usually measured in bits per second (bit/s or bps), and sometimes in data packets per second or data packets per time slot This measure how soon the receiver is able to get a certain amount of data send by the sender. It is determined as the ratio of the total data received to the end to end delay. Throughput is an important factor which directly impacts the network performance

Delay:

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

Queue Length:

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

RESULT:

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

Exp.No:12 IMPLEMENTATIN AND STUDY OF ENCRYPTION AND DECRYPTION

AIM

To study and implement the process of Encryption and Decryption

APPARATUS REQUIRED:

1. PENTIUM – PC

2. C COMPLIER

3. TURBO C

PRINCIPLE:

CRYPTOGRAPHY (secret writing) is the process which uses the encryption and decryption algorithm. An encryption algorithm transforms the plain-text into ciphertext (unknown format) and decryption algorithm transforms the ciphertext back into plain-text. The sender uses an encryption algorithm to send the information in the unknown format and the receiver uses a decryption algorithm to retrieve the original information.

ALGORITHM:

CLIENT:

1. Start the program
2. Initialize the socket for connection establishment
3. Write the message
4. Convert the message to hexcode format
5. Display the message to the server
6. Close all objects
7. Stop

SERVER:

1. Start the program
2. Initialize the server socket
3. Display waiting for connection
4. Display hexcode received from client
5. Initialize a new striing builder
6. Transform the hexcode into the string format using the string builder
7. Display the decrypted message

8. Close all objects
9. Stop

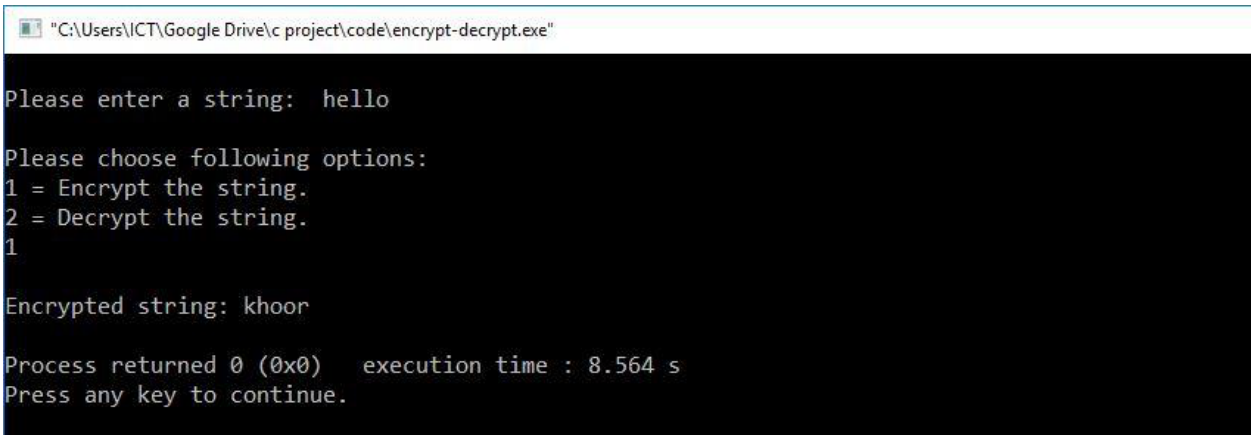
ENCRYPTION AND DECRYPTION PROGRAM

```
#include <stdio.h>

int main()

{

  int i, x;

  char str[100];

  printf("\nPlease enter a string:\t");

  gets(str);

  printf("\nPlease choose following options:\n");

  printf("1 = Encrypt the string.\n");

  printf("2 = Decrypt the string.\n");

  scanf("%d", &x);

  //using switch case statements

  switch(x)

  {

  case 1:

    for(i = 0; (i < 100 && str[i] != '\0'); i++)

      str[i] = str[i] + 3; //the key for encryption is 3 that is added to ASCII value

    printf("\nEncrypted string: %s\n", str);

    break;

  case 2:

    for(i = 0; (i < 100 && str[i] != '\0'); i++)

      str[i] = str[i] - 3; //the key for encryption is 3 that is subtracted to ASCII value

    printf("\nDecrypted string: %s\n", str);
```

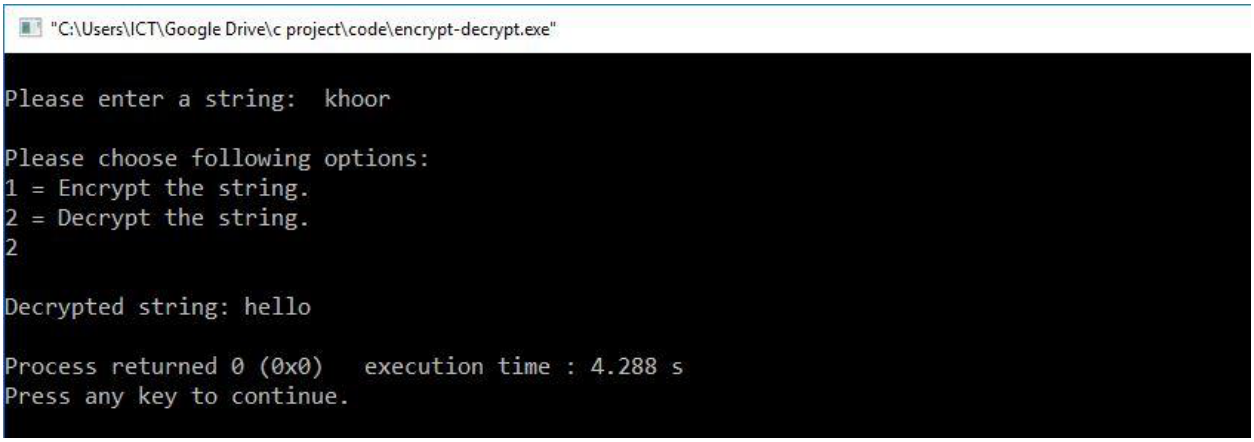
```
    break;
  default:
    printf("\nError\n");
  }
  return 0;
}
```

Output

Encryption



Decryption



RESULT:

The process of Encryption and Decryption is studied and Implemented.

VIVA QUESTIONS:

1. What is meant by congestion?

Congestion in a network occurs if user sends data into the network at a rate greater than that allowed by network resources.

2. Why the congestion occur in a network?

Congestion occurs because the switches in a network have a limited buffer size to store arrived packets before processing.

3. What is data encryption ?

Data encryption refers to mathematical calculations and algorithmic schemes that transform plaintext into cyphertext, a form that is non-readable to unauthorized parties. The recipient of an encrypted message uses a key which triggers the algorithm mechanism to decrypt the data, transforming it to the original plaintext version.

4. What is RSA algorithm?

An public-key encryption technology developed by RSA Data Security, Inc. The acronym stands for Rivest, Shamir, and Adelman, the inventors of the technique. The RSA algorithm is based on the fact that there is no efficient way to factor very large numbers. Deducing an RSA key, therefore, requires an extraordinary amount of computer processing power and time.The RSA algorithm has become the de facto standard for industrial strengthencryption, especially for data sent over the Internet.

5. What is the function of FECN?

The FECN bit is used to warn the receiver of congestion in the network. The sender andreceiver are communication with each other and are using some type of flow control at a higher level.