MAHA BARATHI ENGINEERING COLLEGE,A.VASUDEVANUR

DEPARTMENT OF ECE

II YEAR/ IV SEM

EC3401-NETWORKS SECURITY

2021-REGULATION

MAHA BARATHI ENGINEERING COLLEGE

NH-79, SALEM-CHENNAI HIGHWAY, A.VASUDEVANUR, CHINNASALEM (TK), KALLAKURICHI (DT) 606 201.
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai
2(f) & 12(B) status of UGC, New Delhi
www.mbec.ac.in | Ph: 04151-256333, 257333 | E-mail: mbec123@gmail.com

Bonafide Certificate

Certified that this is a bonafide record of workdone by Selvan/Selvi……………………………………………………………. ………..

Reg.No.…………………………of………………………………Semester……………………………………………………………….…………………Branch

of …………..Degree Examination in the subject………………………………………

…………………………………………………

Staff In charge **Head of the Department**

Date:

Submitted for Anna University Practical Examination conducted on…………………

Internal Examiner **External Examiner**

CONTENTS

Data Link Layer framing methods for Bit stuffing and Character stuffing

Ex.no:1

Date:

Aim:

To Implement the Data Link Layer framing methods for Bit stuffing and Character stuffing using C language.

Apparatus Required:

- PC
- C software

Theory:

Introduction to bit stuffing framing method used in Data Link layer: The new technique allows data frames to contain an arbitrary number of bits and allows character codes with an arbitrary no of bits per character. Each frame begins and ends with the special bit pattern, 01111110, called a flag byte. Whenever the sender's data link layer encounters five consecutive ones in the data, it automatically stuffs a 0 bit into the outgoing bit stream. This bit stuffing is analogous to character stuffing, in which a DLE is stuffed into the outgoing character stream before DLE in the data.

Program Algorithm:

```
Begin
Step 1: Read frame length n
Step 2: Repeat step (3 to 4) until i<n(:Read values into the input frame (0’s and1’s).
Step 3: initialize I i=0;
Step 4: read a[i] and increment i
Step 5: Initialize i=0, j=0,count =0
Step 6: repeat step (7 to 22) until i<n
Step 7: If a[i] == 1 then
Step 8: b[j] = a[i]
Step 9: Repeat step (10 to 18) until (a[k] =1 and k<n and count <5)
Step 10: Initialize k=i+1;
Step 11: Increment j and b[j]= a[k];
Step 12: Increment count ;
Step 13: if count =5 then
Step 14: increment j,
Step 15: b[j] =0
Step 16: end if
Step 17: i=k;
Step 18: increment k
Step 19: else
Step 20: b[j] = a[i]
Step 21: end if
Step 22: increment I and j
Step 23: print the frame after bit stuffing
Step 24: repeat step (25 to 26) until i< j
Step 25: print b[i]
Step 26: increment i
```

End

Program Code: // BIT Stuffing program

```
#include<stdio.h>
#include<string.h>
void main()
{
int a[20],b[30],i,j,k,count,n;
printf("Enter frame length:");
scanf("%d",&n);
printf("Enter input frame (0's & 1's only):");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
i=0; count=1; j=0;
while(i<n)
{
if(a[i]==1)
{
b[j]=a[i];
for(k=i+1;a[k]==1 && k<n && count<5;k++)
{
j++;
b[j]=a[k];
count++;
if(count==5)
{
j++;
b[j]=0;
}
i=k;
}}
else
{
b[j]=a[i];
}
i++;
j++;
}
printf("After stuffing the frame is:");
for(i=0;i<j;i++)
printf("%d",b[i]);
}
```

Program Output:

```
  Enter frame length:5
  Enter input frame (0's & 1's only):
                                                   1
                                                   1
                                                   1
                                                   1
                                                   1
After stuffing the frame is: 111110
------------------
(Program exited with code: 6)
Press return to continue
```

Character Stuffing Method:

Theory:

Introduction to character stuffing method of framing for Data Link Layer: The framing method gets around the problem of resynchronization after an error by having each frame start with the ASCII character sequence DLE STX and the sequence DLE ETX. If the destination ever losses the track of the frame boundaries all it has to do is look for DLE STX or DLE ETX characters to figure out. The data link layer on the receiving end removes the DLE before the data are given to the network layer. This technique is called character stuffing.

Program Algorithm:

```
Begin
Step 1: Initialize I and j as 0
Step 2: Declare n and pos as integer and a[20],b[50],ch as character
Step 3: read the string a
Step 4: find the length of the string n, i.e n-strlen(a)
Step 5: read the position, pos
Step 6: if pos > n then
Step 7: print invalid position and read again the position, pos
Step 8: endif
Step 9: read the character, ch
Step 10: Initialize the array b, b[0…5] as ’d’, ’l’, ’e’, ’s’, ’t’,’x’ respectively
Step 11: j=6;
Step 12: Repeat step[(13to22) until i<n
Step 13: if i==pos-1 then
Step 14: initialize b array,b[j],b[j+1]…b[j+6] as‘d’, ‘l’, ‘e’ ,’ch, ’d’, ‘l’,‘e’ respectively
Step 15: increment j by 7, i.e j=j+7
Step 16: endif
Step 17: if a[i]==’d’ and a[i+1]==’l’ and a[i+2]==’e’ then
Step 18: initialize array b, b[13…15]=’d’, ‘l’, ‘e’ respectively
Step 19: increment j by 3, i.e j=j+3
Step 20: endif
Step 21: b[j]=a[i]
Step 22: increment I and j;
Step 23: initialize b array,b[j],b[j+1]…b[j+6] as‘d’, ‘l’,‘e’,’e’,‘t’, ‘x’,‘\0’ respectively
Step 24: print frame after stuffing
Step 25: print b
End
```

Program: //Program for Character Stuffing

```
#include<stdio.h>
#include<string.h>
#include<process.h>
void main()
{
int i=0,j=0,n,pos;
char a[20],b[50],ch;
printf("Enter string\n");
scanf("%s",&a);
n=strlen(a);
printf("Enter position\n");
scanf("%d",&pos);
if(pos>n)
{
```

```
printf("invalid position, Enter again :");
scanf("%d",&pos);}
printf("Enter the character\n");
ch=getche();
b[0]='d';
b[1]='l';
b[2]='e';
b[3]='s';
b[4]='t';
b[5]='x';
j=6;
while(i<n)
{
if(i==pos-1)
{
b[j]='d';
b[j+1]='l';
b[j+2]='e';
b[j+3]=ch;
b[j+4]='d';
b[j+5]='l';
b[j+6]='e';
j=j+7;
}
if(a[i]=='d' && a[i+1]=='l' && a[i+2]=='e')
{
b[j]='d';  b[j+1]='l';
b[j+2]='e'; j=j+3;
}
b[j]=a[i];
i++;
j++;
}
b[j]='d';
b[j+1]='l';
b[j+2]='e';
b[j+3]='e';
b[j+4]='t';
b[j+5]='x';
b[j+6]='\0';
printf("\nframe after stuffing:\n");
printf("%s",b);
}
```

Output:

```
Enter string
MLRITM
Enter position  2
Enter the character
frame after stuffing:
dlestxMdldleLRITMdleetx
------------------(Program exited with code: 0) Press return to continue
```

Result:

Thus the Data Link Layer framing methods for Bit stuffing and Character stuffing was implemented using C language.

Implementation of Error Detection / Correction Techniques for LRC, CRC and Hamming code

Ex.no:2

Date:

Aim:

To Implement Error Detection / Correction Techniques for LRC, CRC and Hamming code using C language.

Apparatus Required:

- PC
- C software

CYCLIC REDUNDANCY CHECK(CRC):

```
#include<stdio.h>
char data[20],div[20],temp[4],total[100];
int i,j,datalen,divlen,len,flag=1;
void check();
int main()
{
            printf("Enter the total bit of data:");
            scanf("%d",&datalen);
            printf("\nEnter the total bit of divisor");
            scanf("%d",&divlen);
            len=datalen+divlen-1;
            printf("\nEnter the data:");
            scanf("%s",&data);
   printf("\nEnter the divisor");
   scanf("%s",div);

            for(i=0;i<datalen;i++)
            {
                        total[i]=data[i];
                        temp[i]=data[i];
            }
            for(i=datalen;i<len;i++)                    //padded with zeroes corresponding to divlen
       total[i]='0';
            check();                                                        // check for crc
            for(i=0;i<divlen;i++)                       // append crc output (remainder) at end of temp
                        temp[i+datalen]=data[i];
            printf("\ntransmitted Code Word:%s",temp);
            printf("\n\nEnter the received code word:");
            scanf("%s",total);
            check();
            for(i=0;i<divlen-1;i++)
                        if(data[i]=='1')
                        {
                                    flag=0;
                                    break;
                        }
            if(flag==1)
            printf("\nsuccessful!!");
            else
            printf("\nreceived code word contains errors...\n");
}
void check()
{
    for(j=0;j<divlen;j++)
        data[j]=total[j];
    while(j<=len)
            {
        if(data[0]=='1')                                    // in XOR ans remains as it is except in case
of 1
            for(i = 1;i <divlen ; i++)
   data[i] = (( data[i] == div[i])?'0':'1');
        for(i=0;i<divlen-1;i++)                             // left shift data word by 1 after div
            data[i]=data[i+1];
        data[i]=total[j++];                                 // replace empty right by total
    }
}
```

output:-

```
Enter the total bit of data:7

Enter the total bit of divisor4

Enter the data:1001010

Enter the divisor1011

transmitted Code Word:1001010111

Enter the received code word:1001010100

received code word contains errors...
--------------------------------
```

Hamming code:

```
#include <stdio.h>
#include <math.h>
int input[32];
int code[32];
int ham_calc(int,int);
void main()
{
            int n,i,p_n = 0,c_l,j,k;
            printf("Please enter the length of the Data Word: ");
            scanf("%d",&n);
            printf("Please enter the Data Word:\n");
            for(i=0;i<n;i++)
            {
                        scanf("%d",&input[i]);
            }

            i=0;
```

```
    while(n>(int)pow(2,i)-(i+1))
    {
        p_n++;
        i++;
    }

    c_l = p_n + n;

    j=k=0;
    for(i=0;i<c_l;i++)
    {

        if(i==((int)pow(2,k)-1))
        {
            code[i]=0;
            k++;
        }
        else
        {
            code[i]=input[j];
            j++;
        }
    }
    for(i=0;i<p_n;i++)
    {
        int position = (int)pow(2,i);
        int value = ham_calc(position,c_l);
        code[position-1]=value;
    }
    printf("\nThe calculated Code Word is: ");
    for(i=0;i<c_l;i++)
        printf("%d",code[i]);
    printf("\n");
    printf("Please enter the received Code Word:\n");
    for(i=0;i<c_l;i++)
        scanf("%d",&code[i]);

    int error_pos = 0;
    for(i=0;i<p_n;i++)
    {
        int position = (int)pow(2,i);
        int value = ham_calc(position,c_l);
        if(value != 0)
            error_pos+=position;
    }
    if(error_pos == 1)
        printf("The received Code Word is correct.\n");
    else
        printf("Error at bit position: %d\n",error_pos);
}
int ham_calc(int position,int c_l)
{
    int count=0,i,j;
    i=position-1;
    while(i<c_l)
```

```
        {
                for(j=i;j<i+position;j++)
                {
                        if(code[j] == 1)
                                count++;
                }
                i=i+2*position;
        }
        if(count%2 == 0)
                return 0;
        else
                return 1;
}
```

output:-

```
Please enter the length of the Data Word: 8
Please enter the Data Word:
                                             1
                                             1
                                             0
                                             1
                                             0
                                             0
                                             1
                                             1

The calculated Code Word is: 011110100011
Please enter the received Code Word:
0
1
1
1
1
1
1
0
0
0
1
1
Error at bit position: 6
```

Longitudinal Redundancy Check (LRC):

```
#include #include<stdio.h>
#include<conio.h>
void main()
{
int l1,bit[100],count=0,i,choice;
clrscr();
printf("Enter the length of data stream: ");
scanf("%d",&l1);
printf("\nEnter the data stream ");
```

```
for(i=0;i
 {
scanf("%d",&bit[i]);
if(bit[i]==1)
count=count+1;
}
printf("Number of 1's are %d",count);
printf("\nEnter the choice to implement parity bit");
printf("\n1-Sender side\n2-Receiver side\n");
scanf("%d",&choice);
switch(choice)
{
case 1:
if(count%2==0)
bit[l1]=0;
else
bit[l1]=1;
printf("\nThe data stream after adding parity bit is\n");
for(i=0;i<=l1;i++)
printf("%d",bit[i]);
break;
case 2:
if(count%2==0)
printf("There is no error in the received data stream");
else
printf("There is error in the received data stream");
break;
default:
printf("Invalid choice");
break;
}
getch();
}
```

OUTPUT:

```
Enter the length of data stream: 10
Enter the data stream 1 1 0 1 0 1 1 1 0 1
Number of 1's are 7
Enter the choice to implement parity bit
1-Sender side
2-Receiver side 1
The data stream after adding parity bit is
11010111011
Enter the length of data stream: 10
Enter the data stream 1 1 1 1 1 0 0 0 1 0
Number of 1's are 6
Enter the choice to implement parity bit
1-Sender side
2-Receiver side
2
There is no error in the received data stream
```

Result:

Thus the program of Error Detection / Correction Techniques for LRC, CRC and Hamming code was implemented using C language.

Implementation of Stop and Wait, and Sliding Window Protocols

Ex.no:3

Date:

Aim:

To Implement of Stop and Wait, and Sliding Window Protocols using C language.

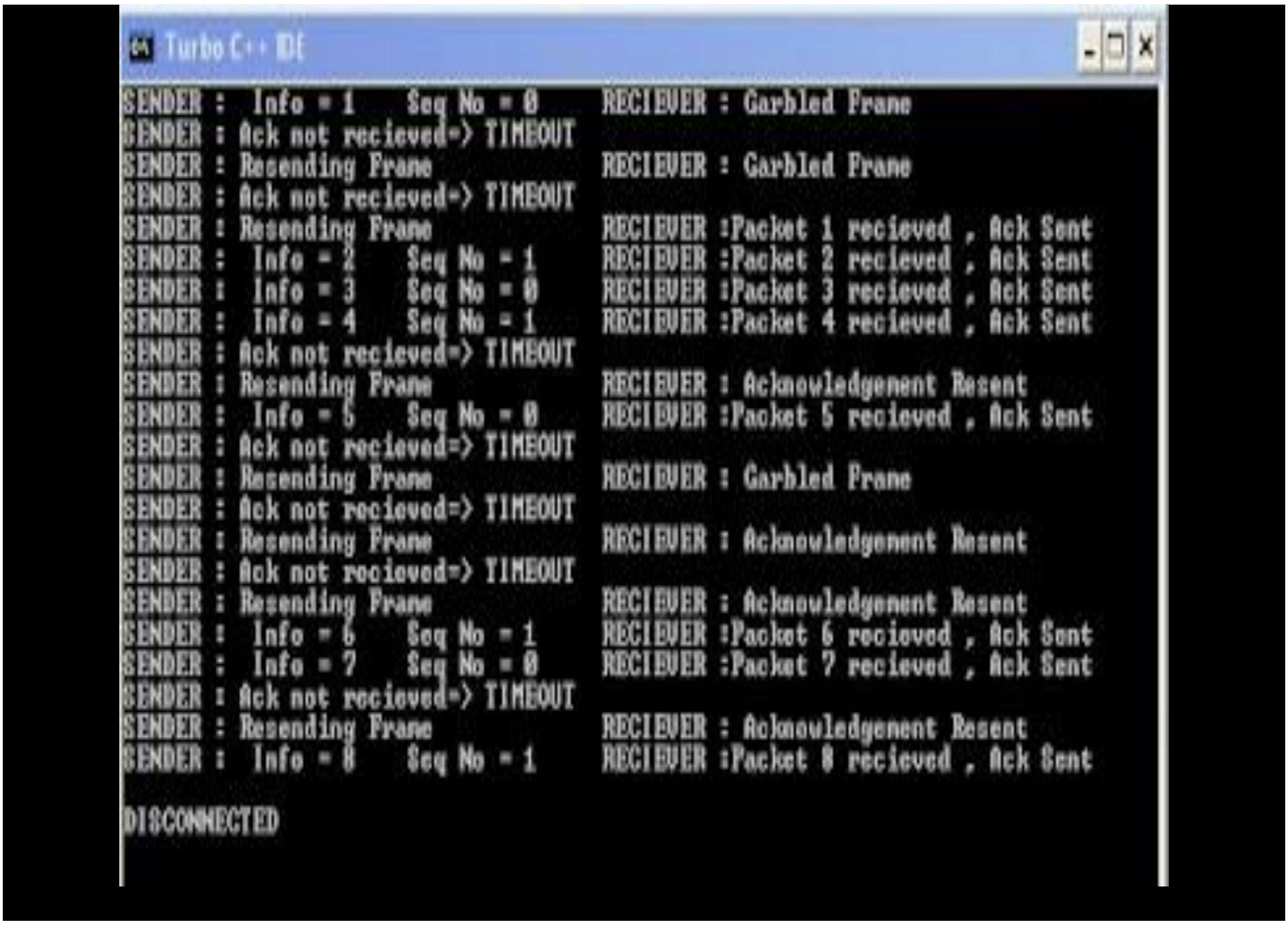
Apparatus Required:

- PC
- C software

Program:

```
#include <stdio.h>
#include <stdlib.h>
#define RTT 4
#define TIMEOUT 4
#define TOT_FRAMES 7
enum {NO,YES} ACK;
int main()
{
int wait_time,i=1;
ACK=YES;
for(;i<=TOT_FRAMES;)
{
if (ACK==YES && i!=1)
{
printf("\nSENDER: ACK for Frame %d Received.\n",i-1);
}
printf("\nSENDER: Frame %d sent, Waiting for ACK...\n",i);
ACK=NO;
wait_time= rand() % 4+1;
if (wait_time==TIMEOUT)
{
printf("SENDER: ACK not received for Frame %d=>TIMEOUT Resending Frame...",i);
}
else
{
sleep(RTT);
printf("\nRECEIVER: Frame %d received, ACK sent\n",i);
printf("-------------------------------------------");
ACK=YES;
i++;
}
}
return 0;
}
```

Stop and Wait Protocol output



```
SENDER :  Info = 1    Seq No = 0     RECIEVER : Garbled Frame
SENDER : Ack not recieved=> TIMEOUT
SENDER : Resending Frame             RECIEVER : Garbled Frame
SENDER : Ack not recieved=> TIMEOUT
SENDER : Resending Frame             RECIEVER :Packet 1 recieved , Ack Sent
SENDER :  Info = 2    Seq No = 1     RECIEVER :Packet 2 recieved , Ack Sent
SENDER :  Info = 3    Seq No = 0     RECIEVER :Packet 3 recieved , Ack Sent
SENDER :  Info = 4    Seq No = 1     RECIEVER :Packet 4 recieved , Ack Sent
SENDER : Ack not recieved=> TIMEOUT
SENDER : Resending Frame             RECIEVER : Acknowledgement Resent
SENDER :  Info = 5    Seq No = 0     RECIEVER :Packet 5 recieved , Ack Sent
SENDER : Ack not recieved=> TIMEOUT
SENDER : Resending Frame             RECIEVER : Garbled Frame
SENDER : Ack not recieved=> TIMEOUT
SENDER : Resending Frame             RECIEVER : Acknowledgement Resent
SENDER : Ack not recieved=> TIMEOUT
SENDER : Resending Frame             RECIEVER : Acknowledgement Resent
SENDER :  Info = 6    Seq No = 1     RECIEVER :Packet 6 recieved , Ack Sent
SENDER :  Info = 7    Seq No = 0     RECIEVER :Packet 7 recieved , Ack Sent
SENDER : Ack not recieved=> TIMEOUT
SENDER : Resending Frame             RECIEVER : Acknowledgement Resent
SENDER :  Info = 8    Seq No = 1     RECIEVER :Packet 8 recieved , Ack Sent

DISCONNECTED
```

Sliding Window Protocols:

Program:

```
#include<stdio.h>
 int main()
{
    int w,i,f,frames[50];
     printf("Enter window size: ");
    scanf("%d",&w);

    printf("\nEnter number of frames to transmit: ");
    scanf("%d",&f);
     printf("\nEnter %d frames: ",f);
     for(i=1;i<=f;i++)
        scanf("%d",&frames[i]);
     printf("\nWith sliding window protocol the frames will be sent in the following manner 
(assuming no corruption of frames)\n\n");
    printf("After sending %d frames at each stage sender waits for acknowledgement sent by the 
receiver\n\n",w);
     for(i=1;i<=f;i++)
    {
        if(i%w==0)
        {
            printf("%d\n",frames[i]);
            printf("Acknowledgement of above frames sent is received by sender\n\n");
        }
        else
            printf("%d ",frames[i]);
    }

    if(f%w!=0)
        printf("\nAcknowledgement of above frames sent is received by sender\n");

    return 0;
}
```

OUTPUT:

```
Enter window size: 3
Enter number of frames to transmit: 5
Enter 5 frames: 12 5 89 4 6
With sliding window protocol the frames will be sent in the following manner (assuming no
corruption of frames)
After sending 3 frames at each stage sender waits for acknowledgement sent by the receiver 12

5 89

Acknowledgement of above frames sent is received by sender 4 6

Acknowledgement of above frames sent is received by sender
```

Result:

Thus the program of Go Back N, and Selective Repeat Protocols was implemented using C language.

Implementation of Go Back N, and Selective Repeat Protocols

Ex.no:4

Date:

Aim:

To Implement of Go Back N, and Selective Repeat Protocols using C language.

Apparatus Required:

- PC
- C software

Theory:

The Go Back-N ARQ or Go Back Automatic Repeat Request is a way to implement sliding window protocol. This protocol is used for flow control in networking and is a part of the data-link layer. The sender's window is of size N and the receiver's window size is always one.The window sizes for Go Back-N are:

1. SENDER WINDOW SIZE (WS)

The WS is equal to N. If we say the protocol is GB-4, then WS = 4. In order to implement pipelining, the window size should always be greater than one. If window size is one, the protocol reduces to stop-and -wait protocol.

2. RECEIVER WINDOW SIZE (WR)

WR is always one in Go Back-N. Now we will understand how Go Back-N actually works with the help of an example. In the diagram below the sender's window has a size of four, or Go Back-4. Assume that we've plenty of sequence numbers, for the sake of explanation. Now the sender has despatched the packets zero, one, two and three. After acknowledging the packets zero and one, the receiver is now awaiting packet two and the sender window has additionally slided to similarly transmit the packets four and five.Now suppose packet two is misplaced within the network. The receiver will discard all the packets which the sender has transmitted after packet two as it's awaiting the packet with sequence number two. On the sender side, for every packet dispatched there is a time-out timer to expire for packet range two. Now from the remaining transmitted packet five the sender will go back to the packet with sequence number two within the current window and transmit all the packets until packet number five. That's why it is referred to as Go Back-N. In the Go Back-N approach the sender has to move back N places from the closing transmitted packet within the unacknowledged window and not from the factor where the packet is misplaced.

3. ACKNOWLEDGEMENTS

There are 2 kinds of acknowledgments, namely:

- Cumulative ACK
- Independent ACK

GBN makes use of cumulative acknowledgement. On the receiver side, it starts an acknowledgment timer when any packet is received, which is constant, and when it expires, it's going to send a cumulative ACK for the range of packets received in the interval of the timer. If the receiver has obtained N packets, then the acknowledgement range might be N+1. A crucial factor is that the ACK timer will now not begin after the expiration of the primary timer, but after the receiver has received a packet. The timer at the sender side must be greater than the ACK timer.

Program:

```
#include<bits/stdc++.h>
#include<ctime>
#define ll long long int
using namespace std;

void transmission(ll & i, ll & N, ll & tf, ll & tt) {
  while (i <= tf) {
    int z = 0;
    for (int k = i; k < i + N && k <= tf; k++) {
      cout << "Sending Frame " << k << "..." << endl;
      tt++;
    }
    for (int k = i; k < i + N && k <= tf; k++) {
      int f = rand() % 2;
      if (!f) {
        cout << "Acknowledgment for Frame " << k << "..." << endl;
        z++;
      } else {
        cout << "Timeout!! Frame Number : " << k << " Not Received" << endl;
        cout << "Retransmitting Window..." << endl;
        break;
      }
    }
    cout << "\n";
    i = i + z;
  }
   }

 int main() {
  ll tf, N, tt = 0;
  srand(time(NULL));
  cout << "Enter the Total number of frames : ";
  cin >> tf;
  cout << "Enter the Window Size : ";
  cin >> N;
  ll i = 1;
  transmission(i, N, tf, tt);
  cout << "Total number of frames which were sent and resent are : " << tt <<
    endl;
  return 0;
}
```

OUTPUT:

```
Enter the Total number of frames : 12
Enter the Window Size : 4
Sending Frame 1…
Sending Frame 2…
Sending Frame 3…
Sending Frame 4…
Timeout!! Frame Number : 1 Not Received
Retransmitting Window…
Sending Frame 1…
Sending Frame 2…
Sending Frame 3…
Sending Frame 4…
Acknowledgment for Frame 1…
Timeout!! Frame Number : 2 Not Received
Retransmitting Window…
Sending Frame 2…
Sending Frame 3…
Sending Frame 4…
Sending Frame 5…
Timeout!! Frame Number : 2 Not Received
Retransmitting Window…

Sending Frame 2…
Sending Frame 3…
Sending Frame 4…
Sending Frame 5…
Acknowledgment for Frame 2…
Acknowledgment for Frame 3…
Acknowledgment for Frame 4…
Timeout!! Frame Number : 5 Not Received
Retransmitting Window…
Sending Frame 5…
Sending Frame 6…
Sending Frame 7…
Sending Frame 8…
Timeout!! Frame Number : 5 Not Received
Retransmitting Window…
Sending Frame 5…
Sending Frame 6…
Sending Frame 7…
Sending Frame 8…
Acknowledgment for Frame 5…
```

```
Timeout!! Frame Number : 6 Not Received
Retransmitting Window…
Sending Frame 6…
Sending Frame 7…
Sending Frame 8…
Sending Frame 9…
Acknowledgment for Frame 6…
Timeout!! Frame Number : 7 Not Received
Retransmitting Window…
Sending Frame 7…
Sending Frame 8…
Sending Frame 9…
Sending Frame 10…
Acknowledgment for Frame 7…
Acknowledgment for Frame 8…
Acknowledgment for Frame 9…
Acknowledgment for Frame 10…
Sending Frame 11…
Sending Frame 12…
Timeout!! Frame Number : 11 Not Received
Retransmitting Window…
Sending Frame 11…
Sending Frame 12…
Timeout!! Frame Number : 11 Not Received
Retransmitting Window…
Sending Frame 11…
Sending Frame 12…
Acknowledgment for Frame 11…
Acknowledgment for Frame 12…
Total number of frames transmitted(sent + resent) are : 38
```

Selective Repeat Protocol:

```
#include<iostream>
#include<stdio.h>
 #include<sys/types.h>
#include<netinet/in.h>
#include<netdb.h>
 #define cls() printf("33[H33[J")
                                        //structure definition for designing the packet.
struct frame
```

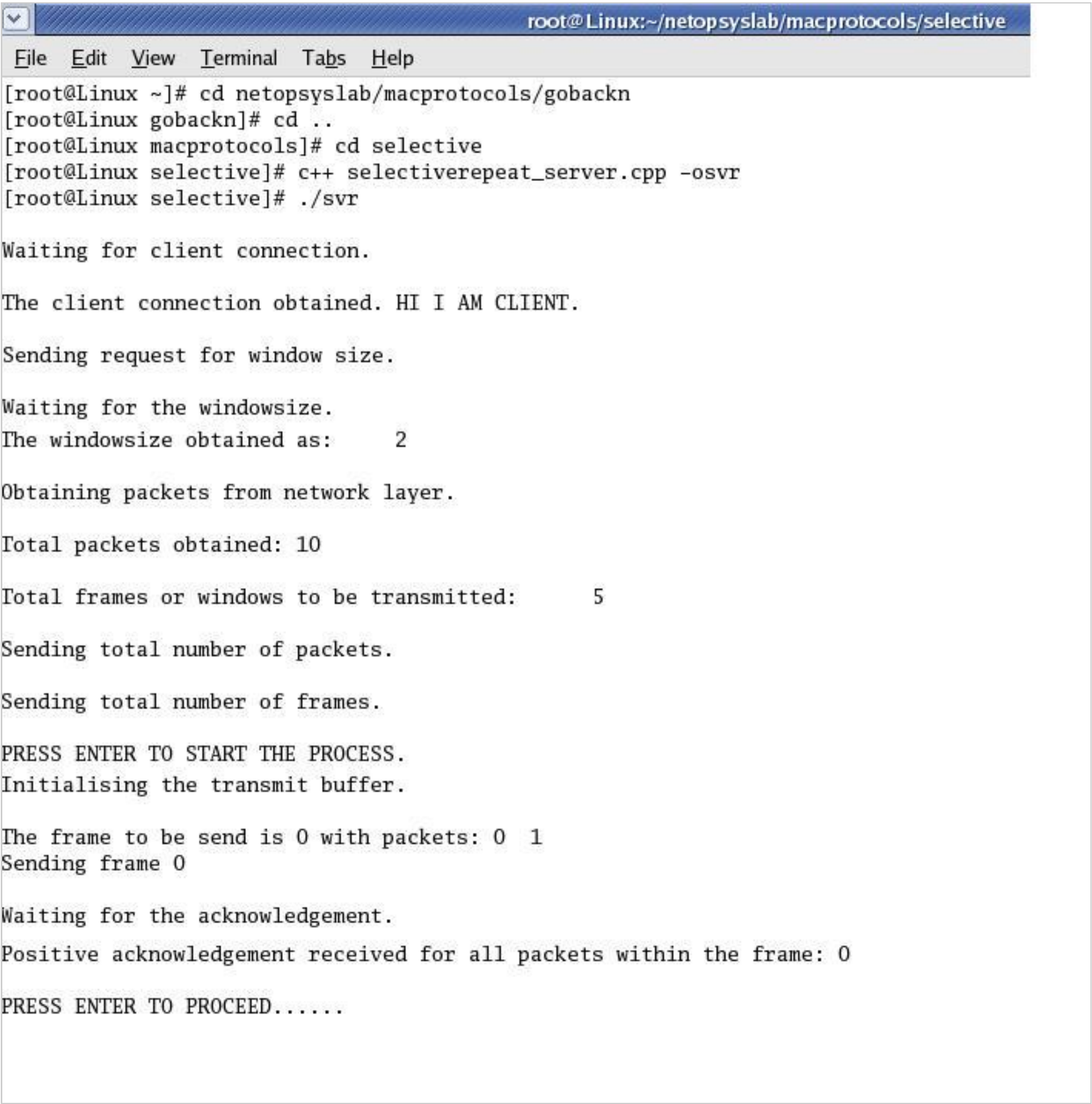
```
{
 int packet[40];
};
                                    //structure definition for accepting the acknowledgement.
struct ack
{
 int acknowledge[40];
};
 int main()
{
 int serversocket;
 sockaddr_in serveraddr,clientaddr;
 socklen_t len;
 int windowsize,totalpackets,totalframes,framessend=0,i=0,j=0,k,l,m,n,repacket[40];
 ack acknowledgement;
 frame f1;
 char req[50];
  serversocket=socket(AF_INET,SOCK_DGRAM,0);
  bzero((char*)&serveraddr,sizeof(serveraddr));
 serveraddr.sin_family=AF_INET;
 serveraddr.sin_port=htons(5018);
 serveraddr.sin_addr.s_addr=INADDR_ANY;
  bind(serversocket,(sockaddr*)&serveraddr,sizeof(serveraddr));
  bzero((char*)&clientaddr,sizeof(clientaddr));
 len=sizeof(clientaddr);
                                                    //connection establishment.
 printf(“\nWaiting for client connection.\n”);
 recvfrom(serversocket,req,sizeof(req),0,(sockaddr*)&clientaddr,&len);
 printf(“\nThe client connection obtained.\t%s\n”,req);
                                                    //sending request for windowsize.
 printf(“\nSending request for window size.\n”);
 sendto(serversocket,”REQUEST FOR WINDOWSIZE.”,sizeof(“REQUEST FOR
WINDOWSIZE.”),0,(sockaddr*)&clientaddr,sizeof(clientaddr));
                                                    //obtaining windowsize.
 printf(“\nWaiting for the windowsize.\n”);
 recvfrom(serversocket,(char*)&windowsize,sizeof(windowsize),0,(sockaddr*)&clientaddr,&len);
 cls();
  printf(“\nThe windowsize obtained as:\t%d\n”,windowsize);
  printf("\nObtaining packets from network layer.\n");
  printf("\nTotal packets obtained:\t%d\n",(totalpackets=windowsize*5));
  printf("\nTotal frames or windows to be transmitted:\t%d\n",(totalframes=5));
                                                    //sending details to client.
 printf("\nSending total number of packets.\n");
 sendto(serversocket,(char*)&totalpackets,sizeof(totalpackets),0,(sockaddr*)&clientaddr,sizeof(clientaddr));
 recvfrom(serversocket,req,sizeof(req),0,(sockaddr*)&clientaddr,&len);
  printf("\nSending total number of frames.\n");
 sendto(serversocket,(char*)&totalframes,sizeof(totalframes),0,(sockaddr*)&clientaddr,sizeof(clientaddr));
 recvfrom(serversocket,req,sizeof(req),0,(sockaddr*)&clientaddr,&len);
  printf("\nPRESS ENTER TO START THE PROCESS.\n");
 fgets(req,2,stdin);
 cls();
  j=0;
 l=0;                                              //starting the process of sending
 while( l<totalpackets)
 {
                                                  //initialising the transmit buffer.
  bzero((char*)&f1,sizeof(f1));
  printf("\nInitialising the transmit buffer.\n");
  printf("\nThe frame to be send is %d with packets:\t",framessend);
                                                    //Builting the frame.
  for(m=0;m<j;m++)
  {
         //including the packets for which negative acknowledgement was received.
   printf("%d  ",repacket[m]);
   f1.packet[m]=repacket[m];
  }

  while(j<windowsize && i<totalpackets)
  {
  printf("%d  ",i);
  f1.packet[j]=i;
  i++;
  j++;
  }
  printf("\nSending frame %d\n",framessend);
```

```
                                            //sending the frame.
 sendto(serversocket,(char*)&f1,sizeof(f1),0,(sockaddr*)&clientaddr,sizeof(clientaddr));
                                            //Waiting for the acknowledgement.
 printf("\nWaiting for the acknowledgement.\n");
 recvfrom(serversocket,(char*)&acknowledgement,sizeof(acknowledgement),0,(sockaddr*)&clientaddr,&
len);
 cls();
                                            //Checking acknowledgement of each packet.
 j=0;
 k=0;
 m=0;
 n=l;
 while(m<windowsize && n<totalpackets)
 {
  if(acknowledgement.acknowledge[m]==-1)
  {
   printf("\nNegative acknowledgement received for packet: %d\n",f1.packet[m]);
   k=1;
   repacket[j]=f1.packet[m];
   j++;
  }
  else
  {
   l++;
  }
  m++;
  n++;
  }
  if(k==0)
 {
 printf("\nPositive acknowledgement received for all packets within the frame: %d\n",framessend);
 }
  framessend++;
 printf("\nPRESS ENTER TO PROCEED……\n");
 fgets(req,2,stdin);
 cls();
  }
   printf("\nAll frames send successfully.\n\nClosing connection with the client.\n");
```

```
close(serversocket);
 }
```

Output


root@Linux:~/netopsyslab/macprotocols/selective
File Edit View Terminal Tabs Help

```
[root@Linux ~]# cd netopsyslab/macprotocols/gobackn
[root@Linux gobackn]# cd ..
[root@Linux macprotocols]# cd selective
[root@Linux selective]# c++ selectiverepeat_server.cpp -osvr
[root@Linux selective]# ./svr

Waiting for client connection.

The client connection obtained. HI I AM CLIENT.

Sending request for window size.

Waiting for the windowsize.
The windowsize obtained as:     2

Obtaining packets from network layer.

Total packets obtained: 10

Total frames or windows to be transmitted:      5

Sending total number of packets.

Sending total number of frames.

PRESS ENTER TO START THE PROCESS.
Initialising the transmit buffer.

The frame to be send is 0 with packets: 0  1
Sending frame 0

Waiting for the acknowledgement.
Positive acknowledgement received for all packets within the frame: 0

PRESS ENTER TO PROCEED......
```

Result:

Thus the program of Go Back N, and Selective Repeat Protocols was implemented using C language.

Implementation of Distance Vector Routing Algorithm

Ex.no:5

Date:

Aim:

To Implement the Distance Vector Routing Algorithm using C language.

Apparatus Required:

- PC
- C software

Program:

```
#include <iostream>

#include <stdio.h>

using namespace std;

struct node {

  int dist[20];

  int from[20];

} route[10];

int main()

{

  int dm[20][20], no;

  cout << "Enter no of nodes." << endl;

  cin >> no;

  cout << "Enter the distance matrix:" << endl;

  for (int i = 0; i < no; i++) {

      for (int j = 0; j < no; j++) {

          cin >> dm[i][j];

          /*  Set distance from i to i as 0 */

          dm[i][i] = 0;

          route[i].dist[j] = dm[i][j];

          route[i].from[j] = j;

      }
```

```
    }
    int flag;
    do {
        flag = 0;
        for (int i = 0; i < no; i++) {
            for (int j = 0; j < no; j++) {
                for (int k = 0; k < no; k++) {
                    if ((route[i].dist[j]) > (route[i].dist[k] + route[k].dist[j])) {
                        route[i].dist[j] = route[i].dist[k] + route[k].dist[j];
                        route[i].from[j] = k;
                        flag = 1;
                    }
                }
            }
        }
    } while (flag);
    for (int i = 0; i < no; i++) {
        cout << "Router info for router: " << i + 1 << endl;
        cout << "Dest\tNext Hop\tDist" << endl;
        for (int j = 0; j < no; j++)
            printf("%d\t%d\t\t%d\n", j+1, route[i].from[j]+1, route[i].dist[j]);
    }
    return 0;
}
```

Output:

(Commands for execution:-

- Open a terminal.
- Change directory to the file location.
- Run g++ filename.cpp
- If there are no errors, run ./a.out **)**

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
```

Result:

Thus the program of Distance Vector Routing Algorithm was implemented using C language.

Implementation of Link State Routing Algorithm

Ex.no:6

Date:

Aim:

To Implement the Link State Routing Algorithm using C language.

Apparatus Required:

- PC
- C software

Theory:

Implement Dijkstra’s algorithm to compute the shortest path. Theory: In order to transfer packets from a source host to the destination host, the network layer must determine the path or route that the packets are to follow. This is the job of the network layer routing protocol. As the heart of any routing protocol is the routing algorithm that determines the path for a packet from source router to destination router. Given a set of router, with links connecting the routers, a routing algorithm finds a good path from source router to destination router. Dijkstra’s method of computing the shortest path is a static routing algorithm. It involves building a graph of the subnet, with each node of the graph representing a router and each arc representing a communication line or a link. To find a route between a pair of routers, the algorithm just finds the shortest path between them on the graph. Dijkstra’s algorithm finds the solution for the shortest path problems only when all the edgeweights are non-negative on a weighted, directed graph. In Dijkstra’s algorithm the metric used for calculation is distance. Each node is labeled with its distance from the source node along the best known path. Initially, no paths are known, so all nodes are labeled with infinity. As the algorithm proceeds and path are found, the labels may change, reflecting better paths. A label may either be tentative or permanent. Initially all nodes are tentative and once it is discovered that the shortest possible path to a node is got it is made permanent and never be changed.

Dijkstra’s Algorithm

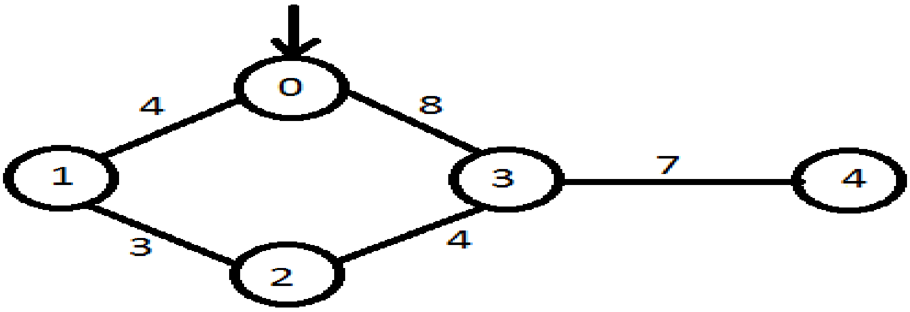
```
            1. Enter cost matrix C[ ][ ]. C[i][j] is the cost of going from vertex i to vertex j.
            If there is noedge between vertices i and j then C[i][j] is infinity.
            2. Array visited[ ] is
                initialized to zero.
                for(i=0;i<n;i++)
                    visited[i]=0;
            3. If the vertex 0 is the source vertex then visited [0] is marked as 1.
            4. Create the distance matrix, by storing the cost of vertices from
            vertex 0 to n-1from the source vertex 0.
                 for(i=1;i<n;i++)
                      distance[i]=cost[0][i];
Initially, distance of source vertex is taken as 0. i.e.
distance[0]=0;5. for(i=1;i<n;i++)
            – Choose a vertex w, such that distance[w] is minimum and visited[w] is 0.
```

Mark visited[w]as 1.
– Recalculate the shortest distance of remaining vertices from the source.
– Only, the vertices not marked as 1 in array visited[ ] should be considered forrecalculation of distance. i.e. for each vertex

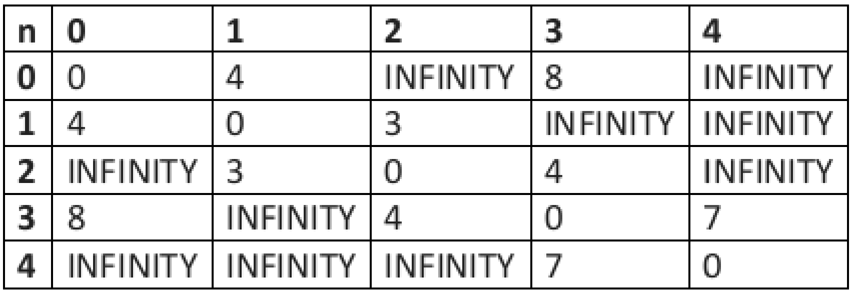
6. Stop the algorithm if, when all the nodes has been marked visited.

Below is an example which further illustrates the Dijkstra's algorithm mentioned.

Consider a weighted graph as shown:



Here 0, 1, 2, 3 and 4 which are inside the circle are nodes of the graph, and the number between them are the distances of the graph. Now using Dijkstra's algorithm we can find the shortest path between initial node and the remaining vertices. For this, the cost matrix of the graph above is,

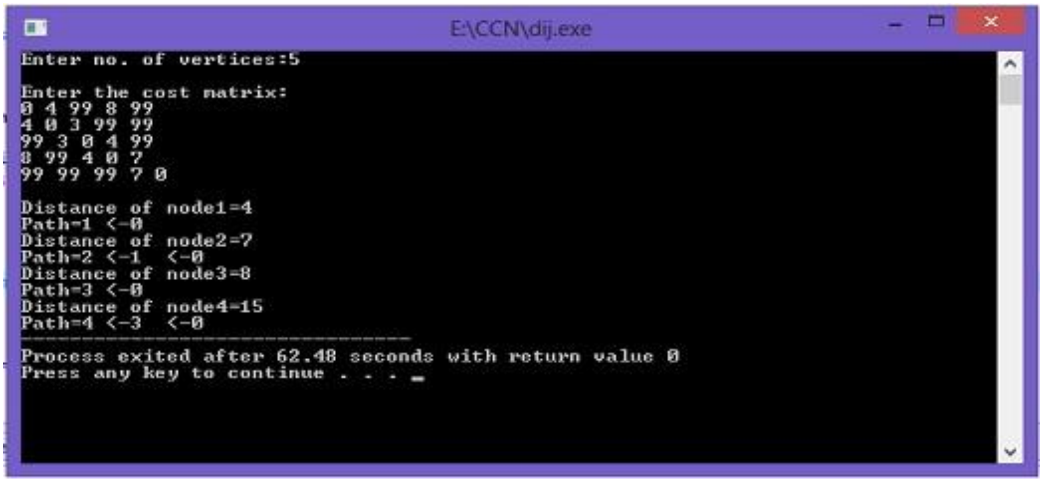


n	**0**	**1**	**2**	**3**	**4**
0	0	4	INFINITY	8	INFINITY
1	4	0	3	INFINITY	INFINITY
2	INFINITY	3	0	4	INFINITY
3	8	INFINITY	4	0	7
4	INFINITY	INFINITY	INFINITY	7	0

Program:

```
#include<stdio.h>
#include<conio.h>
#define INFINITY 99
#define MAX 10
#define startnode 0
voiddijkstra(int cost[MAX][MAX],int n);
int main()
{
int cost[MAX][MAX],i,j,n,u;
printf("Enter no. of vertices:");
scanf("%d",&n);
printf("\nEnter the cost matrix:\n");
for(i=0;i<n;i++)
for(j=0;j<n;j++)
scanf("%d",&cost[i][j]);
dijkstra(cost,n);
return 0;
}
void dijkstra(int cost[MAX][MAX],int n)
{
   int distance[MAX],pred[MAX];
   int visited[MAX],count, mindistance, nextnode, 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);
 }
}
```

Output:



```
Enter no. of vertices:5

Enter the cost matrix:
0 4 99 8 99
4 0 3 99 99
99 3 0 4 99
8 99 4 0 7
99 99 99 7 0

Distance of node1=4
Path=1 <-0
Distance of node2=7
Path=2 <-1  <-0
Distance of node3=8
Path=3 <-0
Distance of node4=15
Path=4 <-3  <-0
--------------------------------
Process exited after 62.48 seconds with return value 0
Press any key to continue . . .
```

Result:

Thus the program of Link State Routing Algorithm was implemented using C language.

Data Encryption and Decryption using Data Encryption Standard Algorithm

Ex.no:7

Date:

Aim:

To Write a program for Data Encryption and Decryption using Data Encryption Standard Algorithm in C language.

Apparatus Required:

- PC
- C software

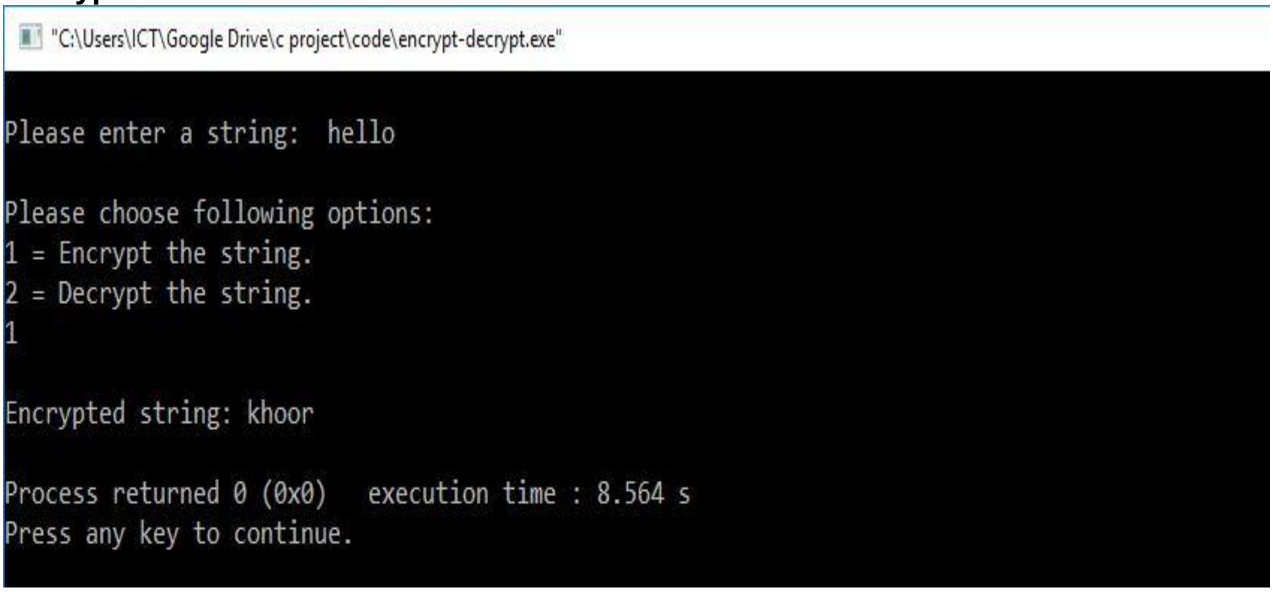
Program:

```
#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


"C:\Users\ICT\Google Drive\c project\code\encrypt-decrypt.exe"

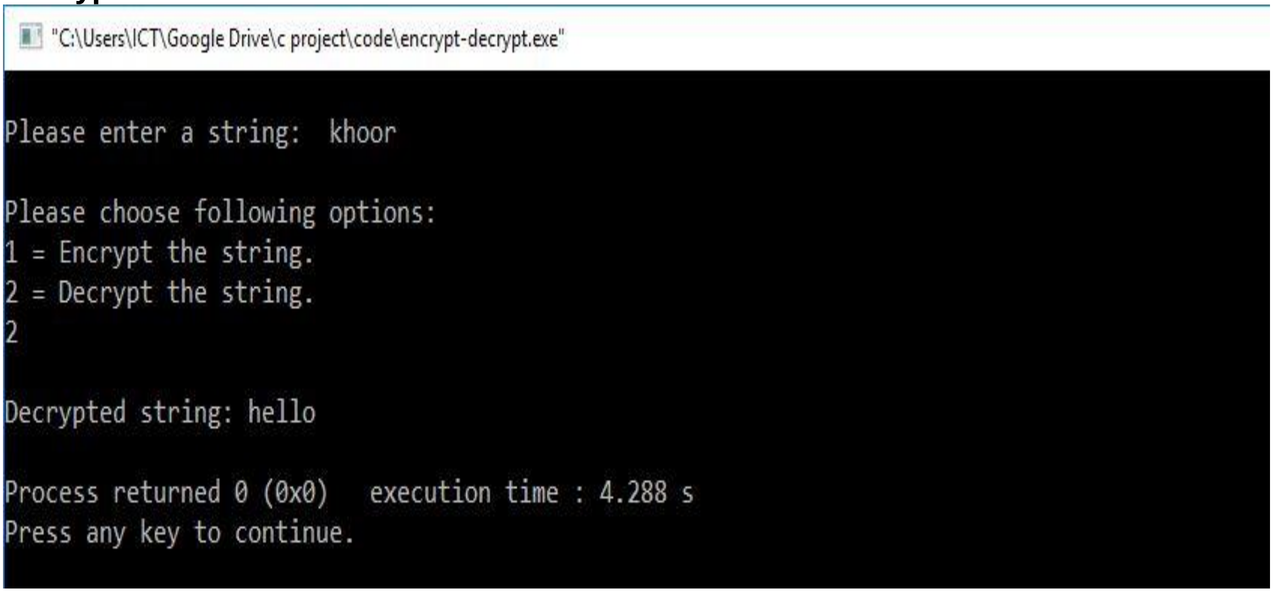
```
Please enter a string:  hello

Please choose following options:
1 = Encrypt the string.
2 = Decrypt the string.
1

Encrypted string: khoor

Process returned 0 (0x0)   execution time : 8.564 s
Press any key to continue.
```

Decryption


"C:\Users\ICT\Google Drive\c project\code\encrypt-decrypt.exe"

```
Please enter a string:  khoor

Please choose following options:
1 = Encrypt the string.
2 = Decrypt the string.
2

Decrypted string: hello

Process returned 0 (0x0)   execution time : 4.288 s
Press any key to continue.
```

Result:

Thus the program of Data Encryption and Decryption using Data Encryption Standard Algorithm was executed.

Data Encryption and Decryption using RSA Algorithm

Ex.no:8

Date:

Aim:

Write a program for simple RSA algorithm to encrypt and decrypt the data.

Apparatus Required:

- Turbo C
- PC

Theory

Cryptography has a long and colorful history. The message 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. The enemy, or intruder, hears and accurately copies down the complete ciphertext. However, unlike the intended recipient, he does not know the decryption key and so cannot decrypt the ciphertext easily. The art of breaking ciphers is called cryptanalysis the art of devising ciphers (cryptography) and breaking them (cryptanalysis) is collectively known as cryptology.There are several ways of classifying cryptographic algorithms. They are generally categorized based on the number of keys that are employed for encryption and decryption, and further defined by their application and use. The three types of algorithms are as follows:

1. Secret Key Cryptography (SKC): Uses a single key for both encryption and decryption. It is also known as symmetric cryptography.
2. Public Key Cryptography (PKC): Uses one key for encryption and another for decryption. It is also known as asymmetric cryptography.
3. Hash Functions: Uses a mathematical transformation to irreversibly "encrypt" information

Public-key cryptography has been said to be the most significant new development in cryptography. Modern PKC was first described publicly by Stanford University professor Martin Hellman and graduate student Whitfield Diffie in 1976. Their paper described a two-key crypto system in which two parties could engage in a secure communication over a non-secure communications channel without having to share a secret key.Generic PKC employs two keys that are mathematically related although knowledge of one key does not allow someone to easily determine the other key. One key is used to encrypt the plaintext and the other key is used to decrypt the ciphertext. The important point here is that it does not matter which key is applied first, but that both keys are required for the process to work. Because pair of keys is required, this approach is also called asymmetric cryptography.In PKC, one of the keys is designated the public key and may be advertised as widely as the owner wants. The other key is designated the private key and is never revealed to another party. It is straight forward to send messages under this scheme.The RSA algorithm is named after Ron Rivest, Adi Shamir and Len Adleman, who invented it in 1977. The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers.

Algorithm

Generate two large random primes, P and Q, of approximately equal size.

Compute N = P x Q

Compute Z = (P-1) x (Q-1).
Choose an integer E, 1 < E < Z, such that GCD (E, Z) = 1

Compute the secret exponent D, 1 < D < Z, such that E x D ≡ 1 (mod Z)
The public key is (N, E) and the private key is (N, D).

Note: The values of P, Q, and Z should also be kept secret.

The message is encrypted using public key and decrypted using private key.

An example of RSA encryption

1. Select primes P=11, Q=3.

2. N = P x Q = 11 x 3 = 33

   Z = (P-1) x (Q-1) = 10 x 2 = 20

3. Lets choose E=3
   Check GCD(E, P-1) = GCD(3, 10) = 1 (i.e. 3 and 10 have no common factors except 1), and check GCD(E, Q-1) = GCD(3, 2) = 1

   therefore GCD(E, Z) = GCD(3, 20) = 1

4. Compute D such that E x D ≡ 1 (mod Z)compute D = E$^{\wedge-1}$ mod Z = 3$^{\wedge-1}$ mod 20
   find a value for D such that Z divides ((E x D)-1)find D such that 20 divides 3D-1.

   Simple testing (D = 1, 2, ...) gives D = 7

   Check: (E x D)-1 = 3.7 - 1 = 20, which is divisible by Z.

5. Public key = (N, E) = 33, 3)
   Private key = (N, D) =(33, 7).
   Now say we want to encrypt the message m = 7, Cipher code = M$^{\wedge E}$ mod N= 7$^{\wedge 3}$ mod 33= 343 mod 33= 13.
   Hence the ciphertext c = 13.

To check decryption we compute Message' = C$^{\wedge D}$ mod N

= 13$^{\wedge 7}$ mod 33

= 7.

Note that we don't have to calculate the full value of 13 to the power 7 here. We can make use of the fact that a = bc mod n = (b mod n).(c mod n) mod n so we can break down a potentially large number into its components and combine the results of easier, smaller calculations to calculate the final value.

Program

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <math.h>
```

```
int mult(unsigned int x, unsigned int y, unsigned int n) {unsigned long int
   k=1;

   int j;

   for (j=1; j<=y; j++) k = (k * x) % n;return
   (unsigned int) k;

}

void main () { char
   msg[100];

   unsigned int pt[100], ct[100], n, d, e, p, q, i;printf("Enter message : ");
   gets(msg);

   //strcpy(pt, msg); for(i=0;i<strlen(msg);i++)

     pt[i]=msg[i];

   n = 253; d = 17; e = 13;

   printf("\nCT = ");

   for(i=0; i<strlen(msg); i++) ct[i] = mult(pt[i], e,n);

   for(i=0; i<strlen(msg); i++) printf("%d ", ct[i]);printf("\nPT = ");

   for(i=0; i<strlen(msg); i++) printf("%c", pt[i]);

   for(i=0; i<strlen(msg); i++) pt[i] = mult(ct[i], d,n) ;

}
```

Output

```
Enter message : alpha

CT = 113 3 129 213 113

PT = alpha
```

Result:

Thus the program for simple encryption and decryption using RSA algorithm was executed.

Implement Client Server model using FTP protocol

Ex.no:9

Date:

Aim:

Write a program for program for FTP client server.

Apparatus Required:

- Turbo C
- PC

ALGORITHM:

SERVER:

```
STEP 1: Start
STEP 2: Declare the variables for the socket
STEP 3: Specify the family, protocol, IP address and port number
STEP 4: Create a socket using socket() function
STEP 5: Bind the IP address and Port number
STEP 6: Listen and accept the client's request for the connection
STEP 7: Establish the connection with the client
STEP 8: Close the socket
STEP 9: Stop
```

CLIENT:

```
STEP 1: Start
STEP 2: Declare the variables for the socket
STEP 3:  Specify the family, protocol, IP address and port number
STEP 4: Create a socket using socket() function
STEP 5: Call the connect() function
STEP 6: Close the socket
STEP 7: Stop
```

SOURCE CODE:

SERVER:

```
#include<stdio.h>
#include<arpa/inet.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5035
#define MAX 60
int i, j, tem;
char buff[4096], t;
FILE *f1;
int main(int afg, char *argv)
{
	int sockfd, newsockfd, clength;
	struct sockaddr_in serv_addr,cli_addr;
	char t[MAX], str[MAX];
	strcpy(t,"exit");
```

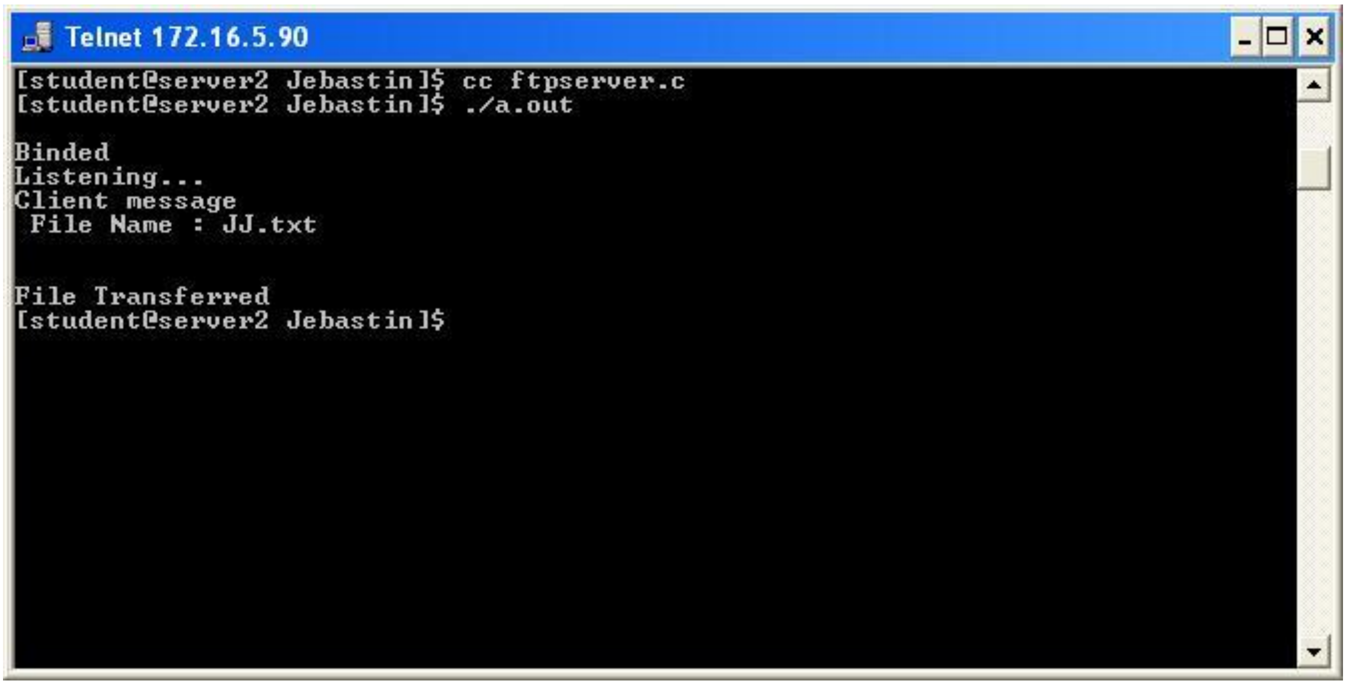
```
    sockfd=socket(AF_INET, SOCK_STREAM,0);
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=INADDR_ANY;
    serv_addr.sin_port=htons(SERV_TCP_PORT);
    printf("\nBinded");
    bind(sockfd,(struct sockaddr*)&serv_addr, sizeof(serv_addr));
    printf("\nListening...");
    listen(sockfd, 5);
    clength=sizeof(cli_addr);
    newsockfd=accept(sockfd,(struct sockaddr*) &cli_addr,&clength);
    close(sockfd);
    read(newsockfd, &str, MAX);
    printf("\nClient message\n File Name : %s\n", str);
    f1=fopen(str, "r");
    while(fgets(buff, 4096, f1)!=NULL)
    {
        write(newsockfd, buff,MAX);
        printf("\n");
    }
    fclose(f1);
    printf("\nFile Transferred\n");
    return 0;
}
```

CLIENT:

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5035
#define MAX 60
int main(int arg,char*argv[])
{
    int sockfd,n;
    struct sockaddr_in serv_addr;
    struct hostent*server;
    char send[MAX],recvline[MAX],s[MAX],name[MAX];
    sockfd=socket(AF_INET,SOCK_STREAM,0);
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
    serv_addr.sin_port=htons(SERV_TCP_PORT);
    connect(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr));
    printf("\nEnter the source file name : \n");
    scanf("%s",send);
    write(sockfd,send,MAX);
    while((n=read(sockfd,recvline,MAX))!=0)
    {
      printf("%s",recvline);
    }
    close(sockfd);
    return 0;
```

}

OUTPUT:
SERVER:

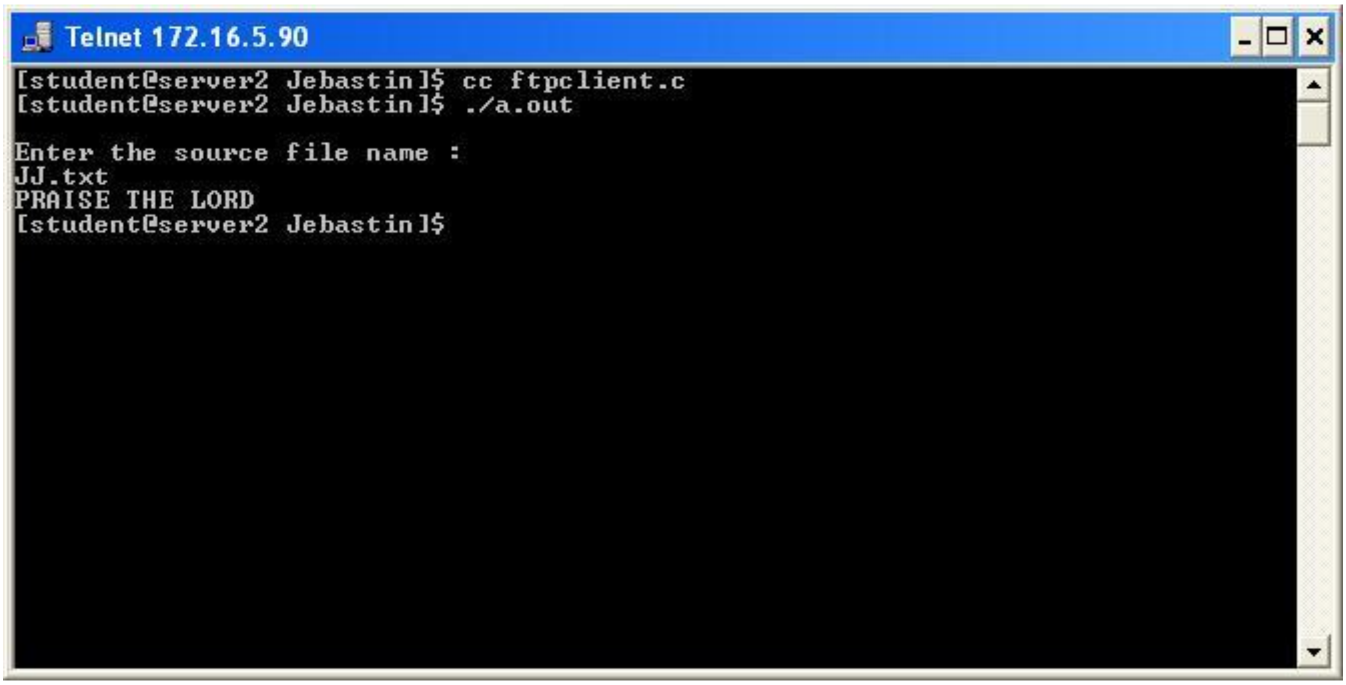


```
[student@server2 Jebastin]$ cc ftpserver.c
[student@server2 Jebastin]$ ./a.out

Binded
Listening...
Client message
 File Name : JJ.txt


File Transferred
[student@server2 Jebastin]$
```

CLIENT:



```
[student@server2 Jebastin]$ cc ftpclient.c
[student@server2 Jebastin]$ ./a.out

Enter the source file name :
JJ.txt
PRAISE THE LORD
[student@server2 Jebastin]$
```

RESULT:

Thus the program for FTP client server was executed and the output was verified.

Ex.no:10 Implement and realize the Network Topology - Star, Bus and Ring

Date:

Aim:

To create scenario and study the performance of Topology -Star, Bus and Ring protocol through simulation.

Apparatus Required:

NS-2
PC

Theory:

Token bus is a LAN protocol operating in the MAC layer. Token bus is standardized as per IEEE 802.4. Token bus can operate at speeds of 5Mbps, 10 Mbps and 20 Mbps. The operation of token bus is as follows: Unlike token ring in token bus the ring topology is virtually created and maintained by the protocol. A node can receive data even if it is not part of the virtual ring, a node joins the virtual ring only if it has data to transmit. In token bus data is transmitted to the destination node only where as other control frames is hop to hop. After each data transmission there is a solicit_successsor control frame transmitted which reduces the performance of the protocol.

Algorithm:

1. Create a simulator object
2. Define different colors for different data flows
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create five nodes that forms a network numbered from 0 to 4
5. Create duplex links between the nodes and add Orientation to the nodes for setting a LAN topology
6. Setup TCP Connection between n(1) and n(3)
7. Apply CBR Traffic over TCP.
8. Schedule events and run the program.

PROGRAM FOR BUS TOPOLOGY:

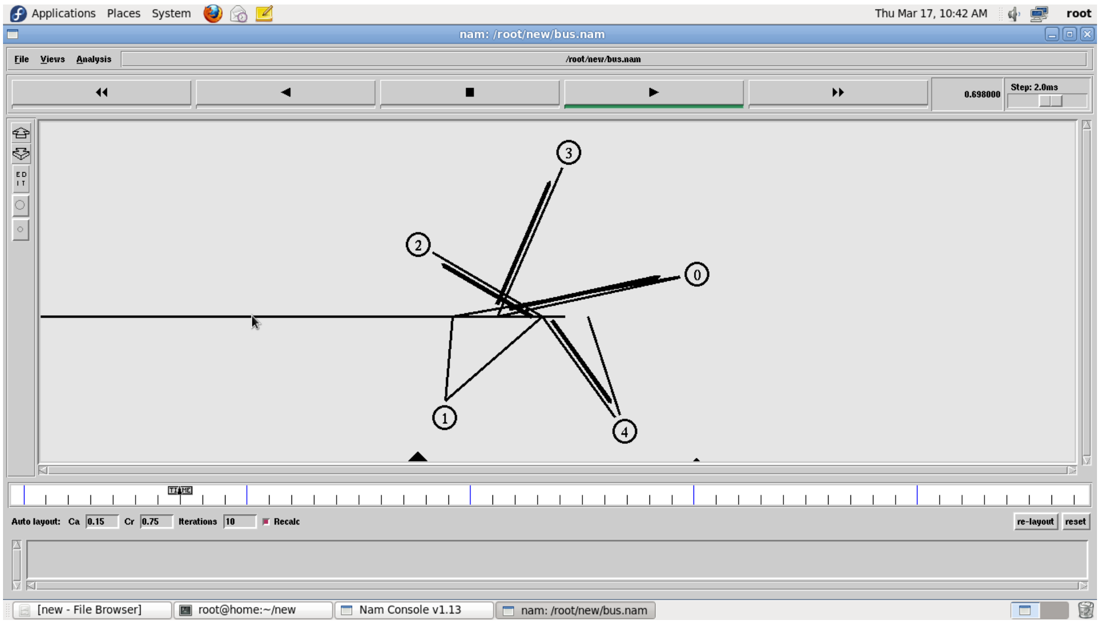
```
#Create a simulator
object set ns [new
Simulator]
#Open the nam trace file set
nf
[open out.namw]
$ns namtrace-all $nf
#Define a 'finish'
procedure proc finish
{}
{
   global ns nf
   $ns flush-trace
   #Close the
   trace file close
   $nf
```

```
  #Executenam on the
  trace file exec nam
  out.nam &

  exit 0

}
#Create five nodes set n0
[$ns node] set n1 [$ns node]
set n2 [$ns node] set n3 [$ns
node] set n4 [$ns node]
#Create Lan between the nodes
set lan0 [$ns newLan "$n0 $n1 $n2 $n3 $n4" 0.5Mb 40ms 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_ 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
```

OUTPUT FOR BUS TOPOLOGY:



Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc

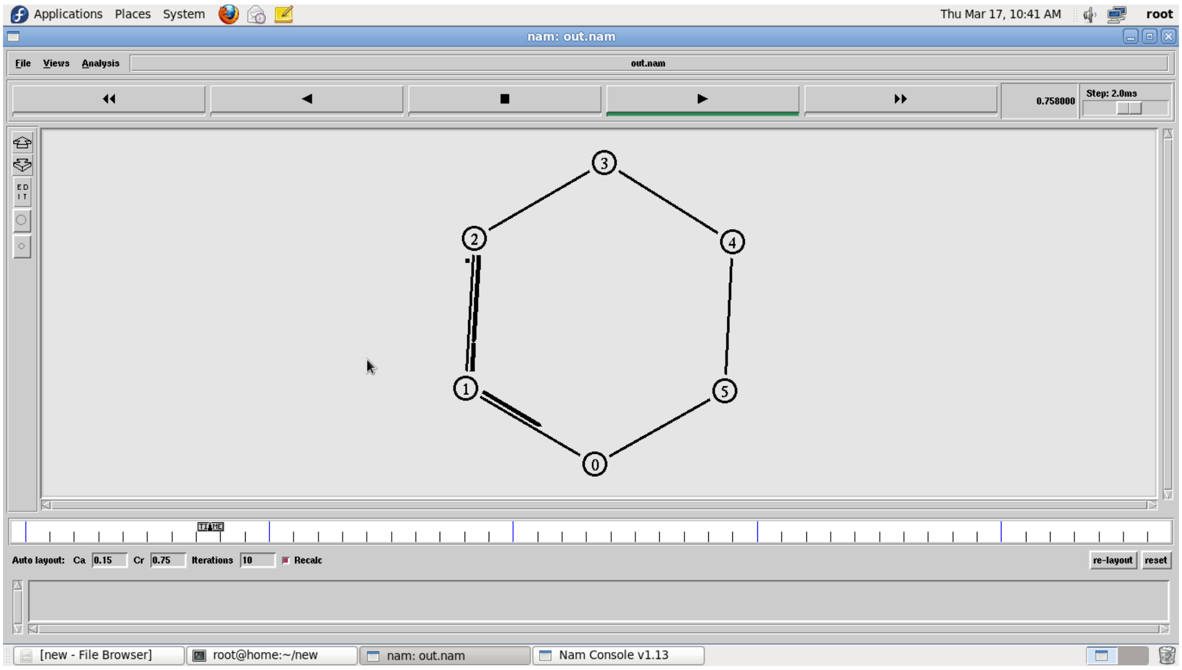
PROGRAM FOR RING PROTOCOL:

```
#Create a simulator
objectset ns [new
Simulator]

#Open the nam
trace fileset nf
[open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedureproc finish
{} {
   global ns nf
   $ns flush-trace
   #Close the
   trace fileclose
   $nf
   #Executenam on the
   trace fileexec nam
   out.nam &
   exit0
}
#Create five nodesset n0
[$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 n0set 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 n3set 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 tcp0set 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
```

RING TOPOLOGY OUTPUT:



Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc

PROGRAM FOR STAR TOPOLOGY:

```
#Create a simulator
objectset ns [new
Simulator]
```

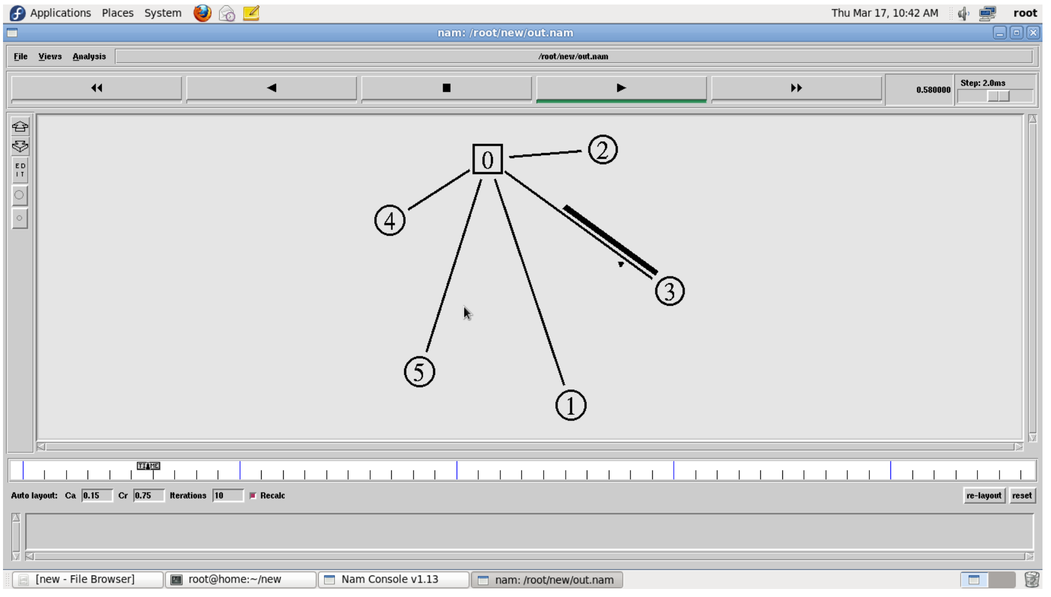
```
#Open the nam
trace file set nf
[open out.nam w]
$ns namtrace-all $nf
#Define a 'finish'
procedure proc finish
{} {
   global ns nf
   $ns flush-trace
   #Close the
   trace file close
   $nf
   #Executenam on the
   trace file exec nam
   out.nam &
   exit0
}

#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]
#Change the shape of center node in a star topology
$n0 shape square
#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 $n0 $n4 1Mb 10ms DropTail
$ns duplex-link $n0 $n5 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
```

STAR TOPOLOGY OUTPUT:



Step: 2.0ms

Auto layout: Ca 0.15 Cr 0.75 Iterations 10 ☒ Recalc

RESULT:

Thus the topology of star, bus and ring was simulated and studied using NS2.

Ex.no:10 IMPLEMENT AND PERFORM THE OPERATION OF CSMA/CD AND CSMA/CA

Date:

AIM:

To create scenario and study the performance of CSMA/CA and CSMA / CD protocol through simulation.

SOFTWARE REQUIREMENTS:

Ns-2

THEORY:

Ethernet is a LAN (Local area Network) protocol operating at the MAC (Medium Access Control) layer. Ethernet has been standardized as per IEEE 802.3. The underlying protocol in Ethernet is known as the CSMA /CD – Carrier Sense Multiple Access / Collision Detection. The working of the Ethernet protocol is as explained below, A node which has data to transmit senses the channel. If the channel is idle then, the data is transmitted. If the channel is busy then, the station defers transmission until the channel is sensed to be idle and then immediately transmitted. If more than one node starts data transmission at the same time, the data collides. This collision is heard by the transmitting nodes which enter into contention phase. The contending nodes resolve contention using an algorithm called Truncated binary exponential back off.

ALGORITHM:

1. Create a simulator object
2. Define different colors for different data flows
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on tracefile.
4. Create six nodes that forms a network numbered from 0 to 5
5. Create duplex links between the nodes and add Orientation to the nodes for setting a LAN topology
6. Setup TCP Connection between n(0) and n(4)
7. Apply FTP Traffic over TCP
8. Setup UDP Connection between n(1) and n(5)
9. Apply CBR Traffic over UDP.
10. Apply CSMA/CA and CSMA/CD mechanisms and study their performance
11. Schedule events and run the program.

PROGRAM:

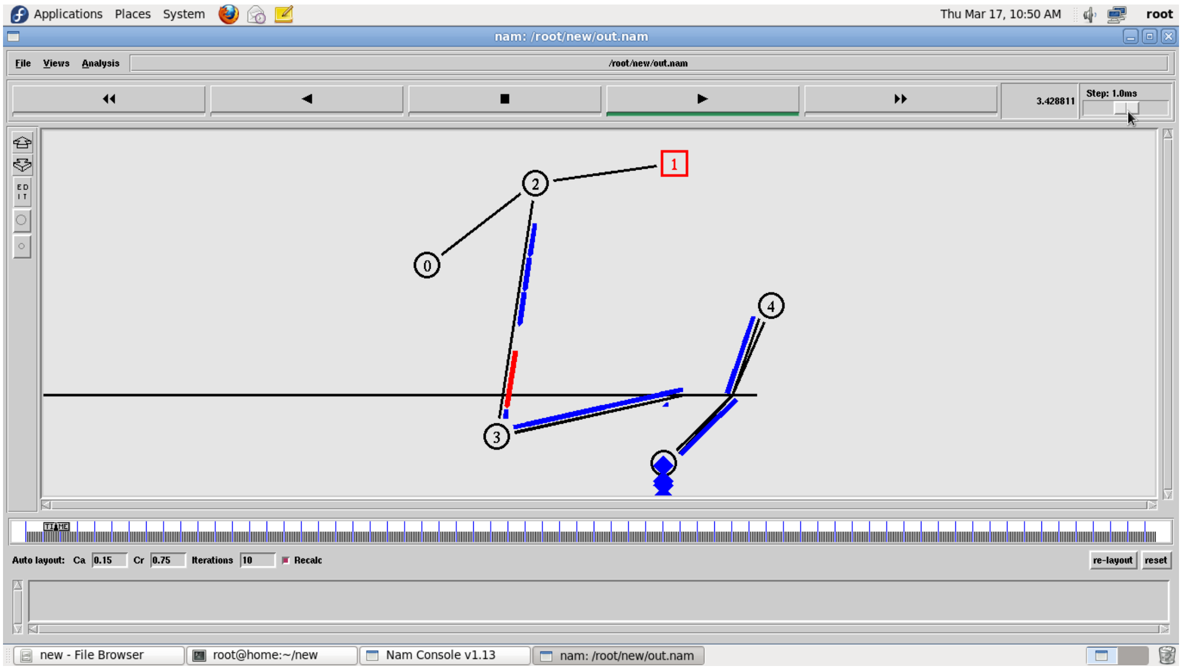
CSMA/CA

```
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-
traceclose
$file1 close
$file2
exec nam
out.nam &exit 0
}
#Create six nodesset
n0 [$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
#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/Ca 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_ 8000
$tcp set packetSize_ 552
#Setup a FTP over TCP
connectionset ftp [new
Application/FTP]
$ftp attach-agent $tcp
$ftp set type_  FTP
#Setup a UDP
connectionset udp
[new Agent/UDP]
$ns attach-agent $n1
$udpset 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_ 1000
$cbr set rate_ 0.01mb
$cbr set random_ false
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
```

```
$ns at 124.0 "$ftp stop"
$ns at 124.5 "$cbr stop"
# next procedure gets two arguments: the
name of the# tcp source node, will be called
here "tcp",
# and the name of output file.
proc plotWindow {tcpSource
file} {global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set
cwnd_] set wnd [$tcpSource
set window_]puts $file
"$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 0.1 "plotWindow $tcp $winfile"
$ns at 5 "$ns trace-annotate
\"packet drop\""# PPP
$ns at 125.0 "finish"
$ns run
```

OUTPUT:



nam: /root/new/out.nam

/root/new/out.nam

3.428811

Step: 1.0ms

Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc

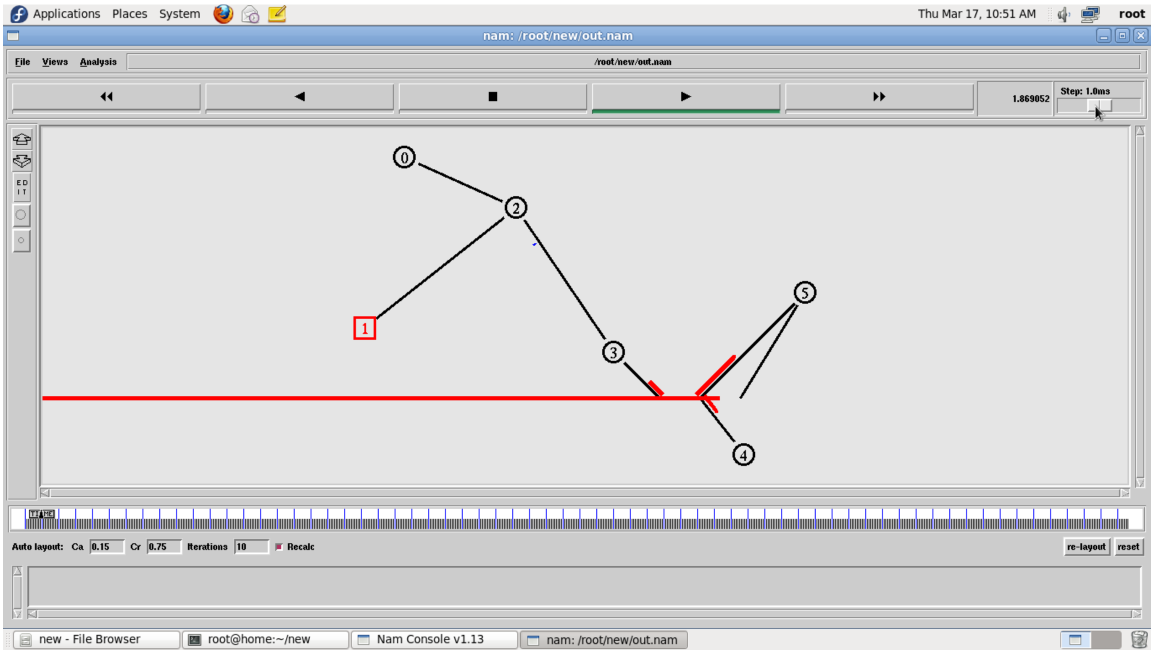
CSMA/CD PROGRAM:

```
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 fileset file2
[open out.nam w]
$ns namtrace-all $file2
#Define a 'finish'
procedureproc finish {}
{
global ns file1 file2
$ns flush-
traceclose
$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
#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_ 8000
$tcp set packetSize_ 552
#Setup a FTP over TCP
connectionset ftp [new
Application/FTP]
$ftp attach-agent $tcp
$ftp set type_  FTP
#Setup a UDP
connectionset udp
[new Agent/UDP]
$ns attach-agent $n1
$udpset 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_ 1000
$cbr set rate_ 0.01mb
$cbr set random_ false
```

```
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 124.0 "$ftp stop"
$ns at 124.5 "$cbr stop"
# next procedure gets two arguments: the
name of the# tcp source node, will be called
here "tcp",
# and the name of output file.
proc plotWindow {tcpSource
file} {global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set cwnd_] set wnd
[$tcpSource set window_]puts $file "$now
$cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 0.1 "plotWindow $tcp $winfile"
$ns at 5 "$ns trace-annotate
\"packet drop\""# PPP
$ns at 125.0 "finish"
$ns run
```

OUTPUT:



Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc

RESULT:

Thus the performance of CSMA/CD and CSMA/CA protocol was studied through simulation.