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
Accredited by NAAC and Recognized under section 2(f) & 12(B) status of UGC, New Delhi

www.mbec.ac.in | Ph: 04151-256333, 257333 | E-mail: mbec123@gmail.com

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

EC3401- NETWORKS AND SECURITY

II Year/ IV Semester B.E ECE

Regulation 2021
(As Per Anna University, Chennai syllabus)

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
Accredited by NAAC and Recognized under section 2(f) & 12(B) status of UGC, New Delhi

www.mbec.ac.in | Ph: 04151-256333, 257333 | E-mail: mbec123@gmail.com

---

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

NAME: **COURSE: B.E-ECE**

REG.NO.: **SEMESTER:IV**

This is to certify that the bonafide record of work done by the student in the **EC3401-NETWORKS AND SECURITY** in the Department of Electronics and Communication Engineering of Maha Barathi Engineering College during the Academic year 2023-24.

Faculty In-Charge **Head of the Department**
Date:

Submitted for the Practical Examination held on…….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

CONTENT

Ex. No:1	**Implement the Data Link Layer Framing methods,**
Date:	**i) Bit stuffing, (ii) Character stuffing**

Aim:
To implement the Data Link Layer framing methods using C program.

Software Used:
PC with C software

Algorithm
Step 1: Open Turbo C Software.
Step 2: Type and Save the program.
Step3: Compile the program and Check for Errors.
Step 4: Run the program.
Step 5: Enter the input and Check for corresponding output.
Step 6: Stop the Program.

Program
i) Bit Stuffing

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
  int x[50], y[50], n;
  int i, j, k, count = 1;
  printf("Enter size of a bit string:");
  scanf("%d", &n);
  printf("Enter the bit string(0's &1's):");
  for (i = 0; i < n; i++)
  {
    scanf("%d", &x[i]);
  }
  i = 0;
  j = 0;
  while (i < n)
  {
    if (x[i] == 1)
    {
      y[j] = x[i];
     //count is less than 5 as 0 is inserted after every
     //5 consecutive 1's

      for (k = i + 1; x[k] == 1 && k < n && count < 5; k++)
      {
        j++;
        y[j] = x[k];
        count++;
        if (count == 5)
        {
          j++;
          y[j] = 0;
        }
        i = k;
      }
    }
    else
    {
      y[j] = x[i];
    }
```

```
    i++;
    j++;
  }
 //Displaying final result
  printf("Result of Bit Stuffing:");
  for (i = 0; i < j; i++)
  {
    printf("%d", y[i]);
  }

  getch();
}
```

Output:

```
Enter size of a bit string:10
Enter the bit string(0's & 1's):1 0 0 1 1 1 1 1 1 0
Result of Bit Stuffing:10011111010
```

ii) Character Stuffing

```
#include<stdio.h>
#include<string.h>
void main()
{
        char a[30], fs[50] = " ", t[3], sd, ed, x[3], s[3], d[3], y[3], ds[50] = " " ;
        int i,len;
        printf("Enter characters to be stuffed:");
        scanf("%s", a);
        printf("\nEnter a character that represents starting delimiter:");
        scanf(" %c", &sd);
        printf("\nEnter a character that represents ending delimiter:");
        scanf(" %c", &ed);
        x[0] = s[0] = s[1] = sd;
        x[1] = s[2] = '\0';
        y[0] = d[0] = d[1] = ed;
        d[2] = y[1] = '\0';
        strcat(fs, x);
        for(i=0; i < strlen(a); i++)
        {
            t[0] = a[i];
            t[1] = '\0';
            if(t[0] == sd)
                strcat(fs, s);
            else if(t[0] == ed)
                strcat(fs, d);
            else
                strcat(fs, t);
        }
        strcat(fs, y);
        printf("\n After stuffing:%s", fs);
      //Character De stuffing from Stuffing
     len=strlen(fs);
       for(i=2;i<len-1;i++)
            {
                        fs[i-1]=fs[i];
            }
            fs[i-1]='\0';
            printf("\n After De stuffing:%s", fs);
          }
```

Output:

Enter characters to be stuffed:goodday
Enter a character that represents starting delimiter:e
Enter a character that represents ending delimiter:q
After stuffing: egooddayq
After De stuffing: goodday

Result:

Thus the data link layer framing methods was implemented successfully using C Program.

Ex. No:2	**Implementation of Error Detection / Correction Techniques**
Date:	**i) LRC, (ii) CRC, (iii) Hamming code**

Aim:
To implement the error detection and correction techniques using C program.

Software Used:
PC with C software

Algorithm
Step 1: Open Turbo C Software.
Step 2: Type and Save the program.
Step3: Compile the program and Check for Errors.
Step 4: Run the program.
Step 5: Enter the input and Check for corresponding output.
Step 6: Stop the Program.

Program
i) LRC

```
#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.
```

ii) 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...
```

iii) Hamming Code

```
#include<stdio.h>
void main()
{
        int data[7],rec[7],i,c1,c2,c3,c;
        printf("this works for message of 4bits in size \nenter message bit one by one:  ");
        scanf("%d%d%d%d",&data[0],&data[1],&data[2],&data[4]);
        data[6]=data[0]^data[2]^data[4];
        data[5]=data[0]^data[1]^data[4];
        data[3]=data[0]^data[1]^data[2];
        printf("\nthe encoded bits are given below: \n");
        for (i=0;i<7;i++) {
                printf("%d ",data[i]);
        }
        printf("\nenter the received data bits one by one: ");
        for (i=0;i<7;i++) {
                scanf("%d",&rec[i]);
        }
        c1=rec[6]^rec[4]^rec[2]^rec[0];
        c2=rec[5]^rec[4]^rec[1]^rec[0];
        c3=rec[3]^rec[2]^rec[1]^rec[0];
        c=c3*4+c2*2+c1 ;
        if(c==0) {
                printf("\ncongratulations there is no error: ");
        } else {
                printf("\nerron on the postion: %d\nthe correct message is \n",c);
                if(rec[7-c]==0)
                                        rec[7-c]=1; else
                                        rec[7-c]=0;
                for (i=0;i<7;i++) {
                        printf("%d ",rec[i]);
                }
        }
}
```

OUTPUT:

```
This works for message of 4bits in size
Enter message bit one by one:  1
0
0
1
The encoded bits are given below:
1 0 0 1 1 0 0
Enter the received data bits one by one: 1
0
1
0
1
0
1
Congratulations there are no error.
```

Result:

Thus the error detection and correction technique was implemented using C program.

Ex. No:3	**Implementation of Stop and Wait, and Sliding Window Protocols**
Date:	

Aim:
To implement Stop and wait, Sliding Window protocols using C program.

Software Used:
PC with C software

Algorithm
Step 1: Open Turbo C Software.
Step 2: Type and Save the program.
Step3: Compile the program and Check for Errors.
Step 4: Run the program.
Step 5: Enter the input and Check for corresponding output.
Step 6: Stop the Program.

Program

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int k,time,win=2,i2=0,frame=0,a[20],b[20],i,j,s,r,ack,c,d;
int send(int,int);
int receive();
int checsum(int *);
main()
{
int i1=0,j1=0,c1;
printf("Enter the frame size\n");
scanf("%d",&frame);
printf("Enter the window size\n");
scanf("%d",&win);
j1=win;
for(i=0;i<frame;i++)
{
a[i]=rand();
}
k=1;
while(i1<frame)
{
if((frame-i1)<win)
j1=frame-i1;
printf("\n\ntransmit the window no %d\n\n",k);
c1=send(i1,i1+j1);
ack=receive(i1,i1+j1,c1);
if (ack!=0)
{
printf("\n\n1.Selective window\n");
printf("2.Go back N\n");
scanf("%d",&ack);
switch(ack)
{
case 1:
printf("\n\n\t Selective window \t\nEnter the faulty frame no\n");
scanf("%d",&i2);
printf("\n\n Retransmit the frame %d \n",i2);
send(i2,i2+1);
break;
case 2:
printf("\n\n\t Go back n\t\n\n");
```

```
printf("\nRetransmit the frames from %d to %d\n",i1,i1+j1);
send(i1,i1+j1);
break;
}
}
 i1=i1+win;
 k++;
 }
 }
 int send(c,d)
 {
 int t1;
 for(i=c;i<d;i++)
 {
 b[i]=a[i];
 printf("frame %d is sent\n",i);
 }
 s=checsum(&a[c]);
 return(s); }
int receive(c,d,c2)
 int c2;
 {
 r=checsum(&b[c]);
 if(c2==r)
 {
 return(0);
 }
else
return(1);
}
int checsum(int *c)
{
int sum=0;
for(i=0;i<win;i++)
sum=sum^(*c);
return sum;
}
```

Output:

```
Enter the frame size
5
Enter the window size
2
Transmit the window no 1
Frame 0 is sent
Frame 1 is sent
Transmit the window no 2
Frame 2 is sent
Frame 3 is sent
Transmit the window no 3
Frame 4 is sent.
```

Result:

Thus the stop and wait, sliding window protocol was implemented using C program.

Ex. No:4	**Implementation of Go back-N and Selective Repeat Protocols**
Date:	

Aim:
To implement Stop and wait, Sliding Window protocols using C program.

Software Used:
PC with C software

Algorithm
Step 1: Open Turbo C Software.
Step 2: Type and Save the program.
Step3: Compile the program and Check for Errors.
Step 4: Run the program.
Step 5: Enter the input and Check for corresponding output.
Step 6: Stop the Program.

Program
i) Go back – N

```
#include<stdio.h>
int main()
{
        int windowsize,sent=0,ack,i;
        printf("enter window size\n");
        scanf("%d",&windowsize);
        while(1)
        {
                for( i = 0; i < windowsize; i++)
                        {
                        printf("Frame %d has been transmitted.\n",sent);
                        sent++;
                        if(sent == windowsize)
                        break;
                        }
                printf("\nPlease enter the last Acknowledgement received.\n");
                        scanf("%d",&ack);

                        if(ack == windowsize)
                                break;
                        else
                                sent = ack;
        }
return 0;
}
```

Output:

```
enter window size
5
Frame 0 has been transmitted.
Frame 1 has been transmitted.
Frame 2 has been transmitted.
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Please enter the last Acknowledgement received.
3
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Please enter the last Acknowledgement received.
```

ii) Selective Repeat Protocols

```
#include<stdio.h>
void main()
{
        int windowsize,sent=0,ack,i;
        printf("enter window size\n");
        scanf("%d",&windowsize);
                for( i = 0; i < windowsize; i++)
                        {
                                printf("Frame %d has been transmitted.\n",sent);
                                sent++;
                                if(sent == windowsize)
                                        break;
                        }
        printf("\nPlease enter the damaged  frame.\n");
                        scanf("%d",&ack);
                        printf("Frame %d has been transmitted.\n",ack);
}
```

OUTPUT:

```
Enter window size
5
Frame 0 has been transmitted.
Frame 1 has been transmitted.
Frame 2 has been transmitted.
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Please enter the damaged frame.
3
Frame 3 has been transmitted.
```

Result:

Thus the Go back-N and Selective repeat protocol was implemented using C language.

Ex. No:5	**Implementation of Distance Vector Routing algorithm (Routing Information Protocol) (Bellman-Ford)**
Date:	

Aim:
To implement Distance vector routing algorithm with Routing Information Protocol that uses Bellman-Ford algorithm using C program.

Software Used:
PC with C software

Algorithm
Step 1: Open Turbo C Software.
Step 2: Type and Save the program.
Step3: Compile the program and Check for Errors.
Step 4: Run the program.
Step 5: Enter the input and Check for corresponding output.
Step 6: Stop the Program.

Program

```
#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();
}
```

Output

```
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 distance vector algorithm with Routing Information Protocol that uses Bellman-Ford algorithm was implemented and executed using C program.

Ex. No:6	**Implementation of Link State Routing algorithm (Open Shortest Path First) with 5 nodes (Dijkstra's)**
Date:	

Aim:
To implement Link State routing algorithm with Open shortest Path First that uses Dijkstra's algorithm using C program.

Software Used:
PC with C software

Algorithm
Step 1: Open Turbo C Software.
Step 2: Type and Save the program.
Step3: Compile the program and Check for Errors.
Step 4: Run the program.
Step 5: Enter the input and Check for corresponding output.
Step 6: Stop the Program.

Program

```
#include <stdio.h>
#include <string.h>
int main()
{
int count,src_router,i,j,k,w,v,min;
int cost_matrix[100][100],dist[100],last[100];
int flag[100];
 printf("\n Enter the no of routers");
scanf("%d",&count);
printf("\n Enter the cost matrix values:");
for(i=0;i<count;i++)
{
for(j=0;j<count;j++)
{
 printf("\n%d->%d:",i,j);
scanf("%d",&cost_matrix[i][j]);
if(cost_matrix[i][j]<0)cost_matrix[i][j]=1000;
}
}
 printf("\n Enter the source router:");
scanf("%d",&src_router);
for(v=0;v<count;v++)
{
flag[v]=0;
last[v]=src_router;
dist[v]=cost_matrix[src_router][v];
}
flag[src_router]=1;
for(i=0;i<count;i++)
{
min=1000;
for(w=0;w<count;w++)
{
if(!flag[w])
if(dist[w]<min)
{
v=w;
min=dist[w];
}
}
flag[v]=1;
for(w=0;w<count;w++)
{
```

```
if(!flag[w])
if(min+cost_matrix[v][w]<dist[w])
{
dist[w]=min+cost_matrix[v][w];
last[w]=v;
}
}
}
for(i=0;i<count;i++)
{
 printf("\n%d==>%d:Path taken:%d",src_router,i,i);
w=i;
while(w!=src_router)
{
 printf("\n<--%d",last[w]);w=last[w];
}
 printf("\n Shortest path cost:%d",dist[i]);
}
}
```

Ouput:

```
Enter the no of routers3
Enter the cost matrix values:
0->0:1
0->1:1
0->2:1
1->0:1
1->1:0
1->2:1
2->0:0
2->1:1
2->2:1
Enter the source router:2
2==>0:Path taken:0
<--2
 Shortest path cost:0
2==>1:Path taken:1
<--2
 Shortest path cost:1
2==>2:Path taken:2
 Shortest path cost:1
```

Result:

Thus Link State routing algorithm with Open shortest Path First that uses Dijkstra's algorithm was implemented C program.

Ex. No:7	**Data encryption and decryption using Data Encryption Standard algorithm**
Date:	

Aim:

To implement Data encryption and decryption using Data Encryption Standard algorithm.

Software Used:

PC with C software

Algorithm

Step 1: Open Turbo C Software.
Step 2: Type and Save the program.
Step3: Compile the program and Check for Errors.
Step 4: Run the program.
Step 5: Enter the input and Check for corresponding output.
Step 6: Stop the Program.

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:

```
Please enter a string:    hello
Please choose following options:
1 = Encrypt the string.
2 = Decrypt the string.
1
Encrypted string: khoor
2
```

Result:

Thus Data encryption and decryption was implemented Data Encryption Standard algorithm.

Ex. No:8	**Data encryption and decryption using RSA (Rivest, Shamir and Adleman) algorithm**
Date:	

Aim:
To implement Data encryption and decryption using RSA algorithm.

Software Used:
PC with C software

Algorithm
Step 1: Open Turbo C Software.
Step 2: Type and Save the program.
Step3: Compile the program and Check for Errors.
Step 4: Run the program.
Step 5: Enter the input and Check for corresponding output.
Step 6: Stop the Program.

Program

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
 long int p, q, n, t, flag, e[100], d[100], temp[100], j, m[100], en[100], i;
char msg[100];
int prime(long int);
void ce();
long int cd(long int);
void encrypt();
void decrypt();

void main()
{
   printf("\nENTER FIRST PRIME NUMBER\n");
   scanf("%d", &p);
   flag = prime(p);
   if (flag == 0)
   {
      printf("\nWRONG INPUT\n");

      exit(1);
   }
   printf("\nENTER ANOTHER PRIME NUMBER\n");
   scanf("%d", &q);
   flag = prime(q);
   if (flag == 0 || p == q)
   {
      printf("\nWRONG INPUT\n");
      exit(1);
   }
   printf("\nENTER MESSAGE\n");
   fflush(stdin);
   scanf("%s", msg);
   for (i = 0; msg[i] != NULL; i++)
      m[i] = msg[i];
   n = p * q;
   t = (p - 1) * (q - 1);
   ce();
   printf("\nPOSSIBLE VALUES OF e AND d ARE\n");
   for (i = 0; i < j - 1; i++)
      printf("\n%ld\t%ld", e[i], d[i]);
   encrypt();
```

```
    decrypt();
}
int prime(long int pr)
{
    int i;
    j = sqrt(pr);
    for (i = 2; i <= j; i++)
    {
        if (pr % i == 0)
            return 0;
    }
    return 1;
}
void ce()
{
    int k;
    k = 0;
    for (i = 2; i < t; i++)
    {
        if (t % i == 0)
            continue;
        flag = prime(i);
        if (flag == 1 && i != p && i != q)
        {
            e[k] = i;
            flag = cd(e[k]);
            if (flag > 0)
            {
                d[k] = flag;
                k++;
            }
            if (k == 99)
                break;
        }
    }
}
long int cd(long int x)
{
    long int k = 1;
    while (1)
    {
        k = k + t;
        if (k % x == 0)
            return (k / x);
    }
}
void encrypt()
{
    long int pt, ct, key = e[0], k, len;
    i = 0;
    len = strlen(msg);
    while (i != len)
    {
        pt = m[i];
        pt = pt - 96;
        k = 1;
        for (j = 0; j < key; j++)
        {
            k = k * pt;
            k = k % n;
        }
        temp[i] = k;
```

```
        ct = k + 96;
        en[i] = ct;
        i++;
    }
    en[i] = -1;
    printf("\nTHE ENCRYPTED MESSAGE IS\n");
    for (i = 0; en[i] != -1; i++)
        printf("%c", en[i]);
}
void decrypt()
{
    long int pt, ct, key = d[0], k;
    i = 0;
    while (en[i] != -1)
    {
        ct = temp[i];
        k = 1;
        for (j = 0; j < key; j++)
        {
            k = k * ct;
            k = k % n;
        }
        pt = k + 96;
        m[i] = pt;
        i++;
    }
    m[i] = -1;
    printf("\nTHE DECRYPTED MESSAGE IS\n");
    for (i = 0; m[i] != -1; i++)
        printf("%c", m[i]);
}
```

Output:

```
ENTER FIRST PRIME NUMBER
         3
ENTER ANOTHER PRIME NUMBER
         7
ENTER MESSAGE
         hello
POSSIBLE VALUES OF e AND d ARE
  5        5
 11        11
THE ENCRYPTED MESSAGE IS
      hqcco
THE DECRYPTED MESSAGE IS
        Hello
```

Result:

Thus Data encryption and decryption was implemented RSA algorithm.

Ex. No:9	**Implement Client Server model using FTP protocol**
Date:	

Aim:
To implement Client Server model using FTP protocol.

Software Used:
PC with C software

Algorithm
Step 1: Open Turbo C Software.
Step 2: Type and Save the program.
Step3: Compile the program and Check for Errors.
Step 4: Run the program.
Step 5: Enter the input and Check for corresponding output.
Step 6: Stop the Program.
Program

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(intafg, char *argv)
{
      intsockfd, newsockfd, clength;
      structsockaddr_inserv_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,(structsockaddr*)&serv_addr, sizeof(serv_addr));
      printf("\nListening...");
      listen(sockfd, 5);
      clength=sizeof(cli_addr);
      newsockfd=accept(sockfd,(structsockaddr*) &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");
      }
```

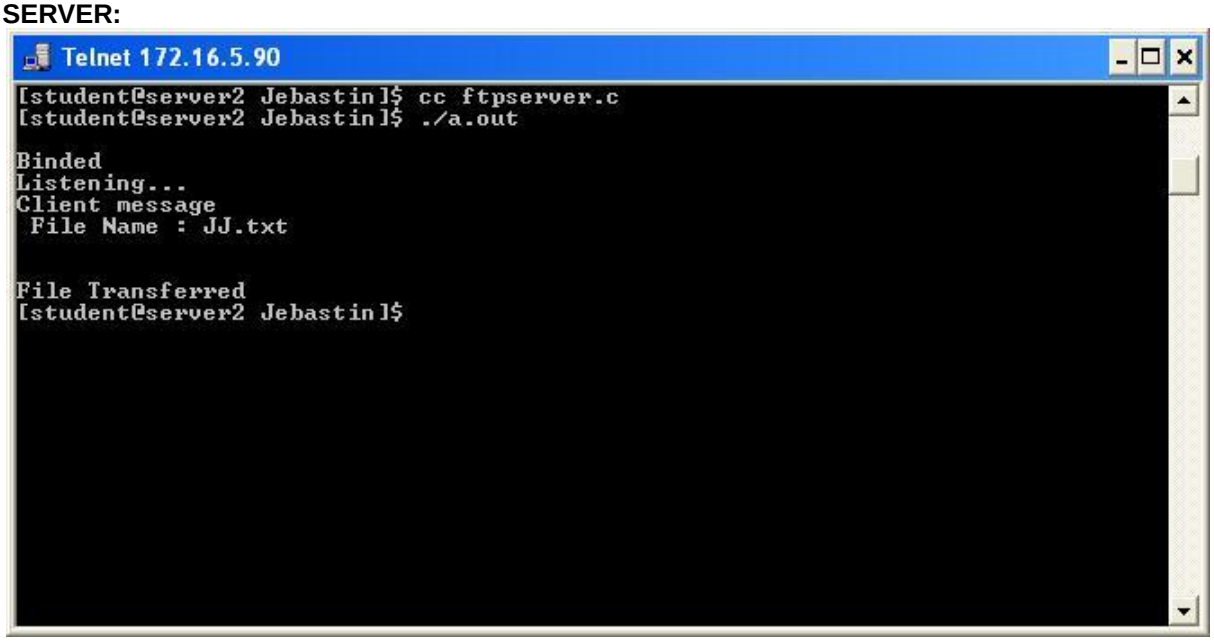
```
    fclose(f1);
    printf("\nFile Transferred\n");
    return 0;
}
```

CLIENT:

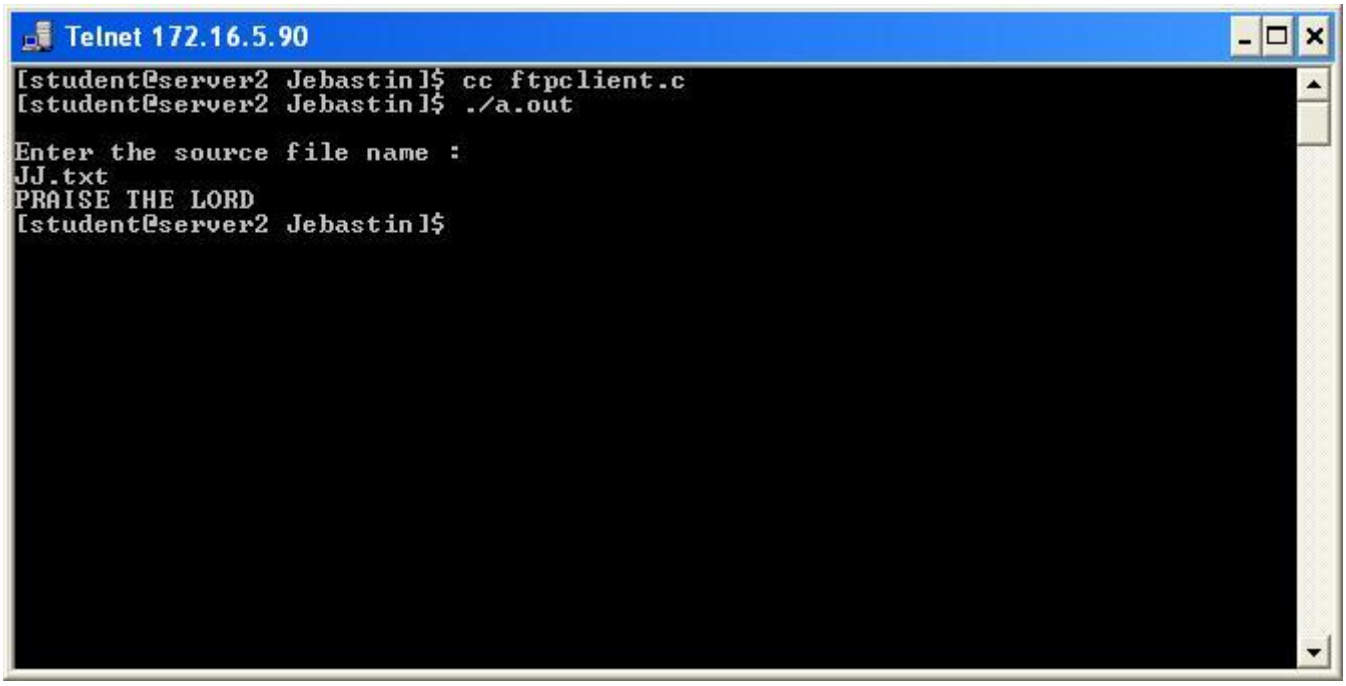
```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5035
#define MAX 60
int main(intarg,char*argv[])
{
    intsockfd,n;
    structsockaddr_inserv_addr;
    structhostent*server;
    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,(structsockaddr*)&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:



CLIENT:



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. Tokenbus can operate at speeds of 5Mbps, 10 Mbps and 20 Mbps. The operation of token bus is as follows: Unliketoken ring in token bus the ring topology is virtually created and maintained by the protocol. A node can receivedata even if it is not part of the virtual ring, a node joins the virtual ring only if it has data to transmit. In tokenbus data is transmitted to the destination node only where as other control frames is hop to hop. After each datatransmission there is a solicit_successsor control frame transmitted which reduces the performance of theprotocol.

Algorithm:

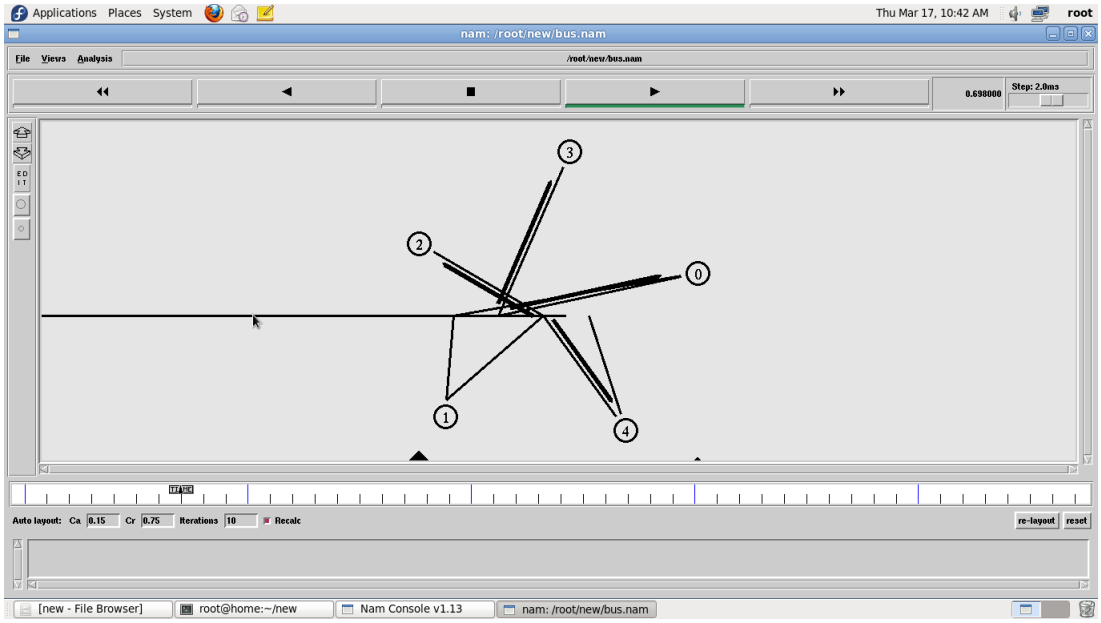
1. Create a simulator object
2. Define different colors for different data flows
3. Open a namtrace file and define finish procedure then close the trace file, and execute namtrace file.
4. Create five nodes that forms a network numbered from 0to 4
5. Create duplex links between the nodes and add Orientation 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
objectset
ns[newSimulator]
#Open the nam trace
filesetnf
[openout.namw]
$nsnamtrace-all$nf
#Define a
'finish'
procedurepr
oc finish{}
{
   globalnsnf
   $ns flush-
   trace#Closethetrace
   fileclose$nf
   #Executenam
   onthetracefile
   exec
   namout.nam&
   exit0
}
```

```
#Create five nodesset
n0 [$ns node]set n1
[$ns node]set n2 [$ns
node]set n3 [$ns
node]setn4[$nsnode]
#CreateLanbetweenthe nodes
setlan0[$nsnewLan"$n0$n1$n2$n3$n4" 0.5Mb40ms LLQueue/DropTailMAC/Csma/CdChannel]
#CreateaTCPagentand
attachittonoden0set
tcp0[newAgent/TCP]
$tcp0 setclass_ 1
$nsattach-agent$n1$tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and
attach it to node n3set sink0[new Agent/TCPSink]
$nsattach-agent$n3$sink0
#Connectthetraffic sources withthetrafficsink
$nsconnect$tcp0$sink0
# Create a CBR traffic source
and attach it to tcp0set
cbr0[newApplication/Traffic/CBR
]
$cbr0 setpacketSize_ 500
$cbr0 setinterval_ 0.01
$cbr0attach-agent$tcp0
#ScheduleeventsfortheCBR agents
$nsat0.5 "$cbr0start"
$nsat4.5"$cbr0stop"
#Callthefinishprocedure after 5secondsofsimulationtime
$nsat5.0"finish"
#Runthesimulation
$nsrun
```

OUTPUT FOR BUS TOPOLOGY:



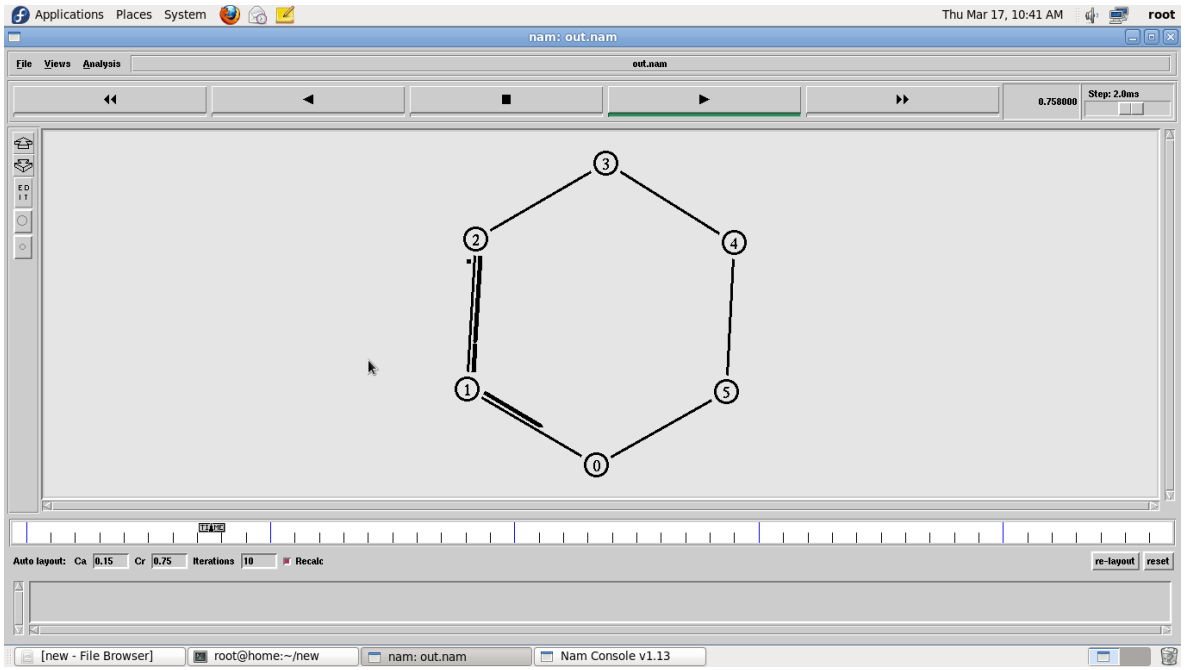
PROGRAM FOR RING PROTOCOL:

```
#Create a
simulator
objectset
ns[newSim
ulator]

#Open
the nam
trace
filesetnf[o
penout.na
mw]
$nsnamtrace-all$nf
#Define a 'finish' procedureproc
finish{} {
   globalnsnf
   $ns flush-
   trace#Closethetracefile
   close$nf
   #Executenam
   onthetracefile
   exec
   namout.nam&
   exit0
}
#Create five nodesset
n0 [$ns node]set n1
[$ns node]set n2 [$ns
node]set n3 [$ns
node]set n4 [$ns
node]setn5[$nsnode]
#Createlinksbetweenthenodes
$nsduplex-link$n0 $n11Mb10msDropTail
$nsduplex-link$n1 $n21Mb10msDropTail
$nsduplex-link$n2 $n31Mb10msDropTail
$nsduplex-link$n3 $n41Mb10msDropTail
$nsduplex-link$n4 $n51Mb10msDropTail
$nsduplex-link$n5 $n01Mb10msDropTail
#CreateaTCPagentand
attachittonoden0set
tcp0[newAgent/TCP]
$tcp0 setclass_ 1
$nsattach-agent$n1$tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and
attach it to node n3set sink0[new Agent/TCPSink]
$nsattach-agent$n3$sink0
#Connectthetrafficsourceswiththetrafficsink
$nsconnect$tcp0$sink0
# Create a CBR traffic source
and attach it to tcp0set
cbr0[newApplication/Traffic/CBR
]
$cbr0setpacketSize_ 500
$cbr0 setinterval_ 0.01
$cbr0attach-agent$tcp0
#ScheduleeventsfortheCBR agents
$nsat0.5 "$cbr0start"
$nsat4.5"$cbr0stop"
#Callthefinishprocedure after 5secondsofsimulationtime
$nsat5.0"finish"
```

#Runthesimulation
$nsrun

RING TOPOLOGY OUTPUT:



PROGRAM FOR STAR TOPOLOGY:

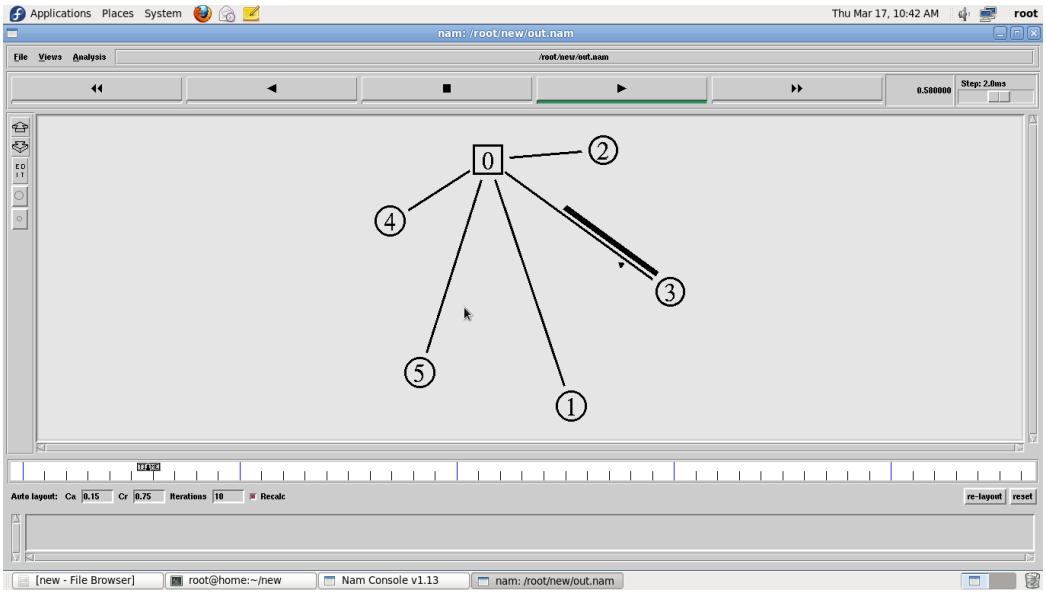
```
#Create a simulator
objectsetns [new Simulator]
#Open the nam
trace
filesetnf[openout.na
mw]
$nsnamtrace-all$nf
#Define a
'finish'
procedurepr
oc finish{} {
   globalnsnf
   $ns flush-
   trace#Closethetracefileclose$
   nf
   #Executenam
   on the trace
   fileexec
   namout.nam&
   exit0
}

#Create          six
nodesset  n0  [$ns
node]set  n1  [$ns
node]set  n2  [$ns
node]set  n3  [$ns
node]set  n4  [$ns
node]setn5[$ns
```

```
node]
#Changetheshapeofcenternodeinastartopology
$n0shapesquare
#Createlinksbetweenthenodes
$nsduplex-link$n0 $n11Mb10msDropTail
$nsduplex-link$n0 $n21Mb10msDropTail
$nsduplex-link$n0 $n31Mb10msDropTail
$nsduplex-link$n0 $n41Mb10msDropTail
$nsduplex-link$n0 $n51Mb10msDropTail

#CreateaTCPagentand
attachittonoden0settcp0[ne
wAgent/TCP]
$tcp0 setclass_ 1
$nsattach-agent$n1$tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and
attach it to node n3setsink0[new Agent/TCPSink]
$nsattach-agent$n3$sink0
#Connectthetraffic sourceswiththetrafficsink
$nsconnect$tcp0$sink0
# Create a CBR traffic source
and attach it to
tcp0setcbr0[newApplication/Traff
ic/CBR]
$cbr0setpacketSize_ 500
$cbr0 setinterval_ 0.01
$cbr0attach-agent$tcp0
#ScheduleeventsfortheCBR agents
$nsat0.5"$cbr0start"
$nsat4.5"$cbr0stop"
#Callthefinishprocedure after 5secondsofsimulationtime
$ns at 5.0 "finish"#Runthesimulation
$nsrun
```

STAR TOPOLOGY OUTPUT:



RESULT:

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

Ex. No:11	**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 and PC

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 explainedbelow, 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 thenimmediately transmitted. If more than one node starts data transmission at the same time, the data collides. Thiscollision is heard by the transmitting nodes which enter into contention phase. The contending nodes resolvecontention usingan algorithmcalledTruncated binaryexponentialbackoff.

ALGORITHM:

1. Createasimulatorobject
2. Definedifferent colorsfordifferentdataflows
3. Opena namtracefileanddefinefinishprocedurethenclosethetracefile,andexecutenamontracefile.
4. Create sixnodesthatformsanetworknumberedfrom0to5
5. Createduplexlinksbetween thenodesandaddOrientationtothe nodesfor settingaLANtopology
6. SetupTCPConnectionbetweenn(0) andn(4)
7. ApplyFTPTrafficoverTCP
8. SetupUDPConnectionbetweenn(1)andn(5)
9. ApplyCBRTraffic over UDP.
10. ApplyCSMA/CAandCSMA/CDmechanismsandstudytheirperformance
11. Scheduleeventsandruntheprogram.

PROGRAM:

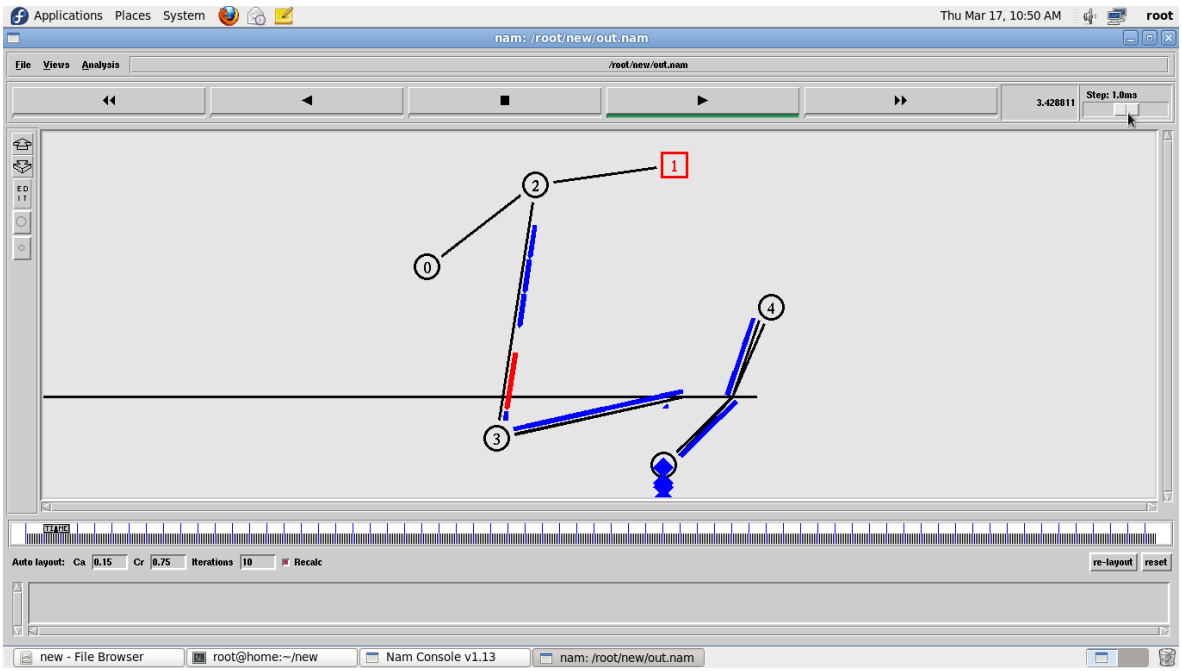
CSMA/CA

```
setns[newSimulator]
#Definedifferentcolorsfordataflows(forNAM)
$nscolor1 Blue
$ns color 2 Red#OpentheTracefiles
setfile1[openout.trw]
setwinfile[openWinFilew]
$nstrace-all $file1
#Open the
NAM trace
filesetfile2[op
enout.namw]
$ns namtrace-
all
```

```
$file2#Define a
'finish'
procedureprocf
inish {}{
globalnsfile1file2
$ns flush-traceclose
$file1close$file2
execna
mout.na
m&exit
0
}
#Create    six
nodesset   n0
[$ns
node]set    n1
[$ns
node]set    n2
[$ns
node]set    n3
[$ns
node]set    n4
[$ns
node]set
n5[$nsnode]
$n1 colorred
$n1 shapebox
#Createlinksbetweenthenodes
$nsduplex-link$n0 $n22Mb 10ms DropTail
$ns duplex-link$n1 $n22Mb10ms DropTail
$nssimplex-link$n2$n30.3Mb100msDropTail
$nssimplex-link$n3$n20.3Mb100msDropTail
setlan[$nsnewLan"$n3$n4$n5"0.5Mb40ms LLQueue/DropTailMAC/Csma/Ca Channel]
SetupaTCPconnection
settcp[newAgent/TCP/Newreno]
$nsattach-agent $n0$tcp
setsink[newAgent/TCPSink/DelAck]
$nsattach-agent $n4$sink
$nsconnect$tcp $sink
$tcpsetfid_1
$tcpsetwindow_8000
$tcpsetpacketSize_ 552
#Setup a FTP over
TCP
connectionsetftp[new
Application/FTP]
$ftpattach-agent$tcp
$ftp
settype_FTP
#Setup a
UDP
connectionse
tudp[newAge
nt/UDP]
$ns attach-
agent $n1
$udpsetnull[n
ewAgent/Null
]
$nsattach-agent $n5$null
```

```
$nsconnect$udp$null
$udp setfid_2
#Setup a CBR over UDP
connectionsetcbr[newApp
lication/Traffic/CBR]
$cbrattach-agent$udp
$cbrsettype_ CBR
$cbrsetpacket_size_ 1000
$cbrsetrate_0.01mb
$cbrsetrandom_ false
$nsat 0.1"$cbrstart"
$nsat 1.0"$ftpstart"
$nsat124.0"$ftpstop"
$nsat124.5"$cbrstop"
# next procedure gets two arguments:
the name of the#tcpsource node,
willbecalled here"tcp",
#andthenameof outputfile.
procplotWindow
{tcpSource file}
{globalns
settime 0.1
set now[$ns now]
setcwnd [$tcpSource
set cwnd_]set wnd
[$tcpSource set
window_]puts
$file"$now$cwnd"
$nsat[expr$now+$time]"plotWindow$tcpSource$file"}
$nsat 0.1"plotWindow$tcp $winfile"
$ns at 5 "$ns trace-annotate
\"packet drop\""# PPP
$nsat 125.0"finish"
$ns run
```

OUTPUT:

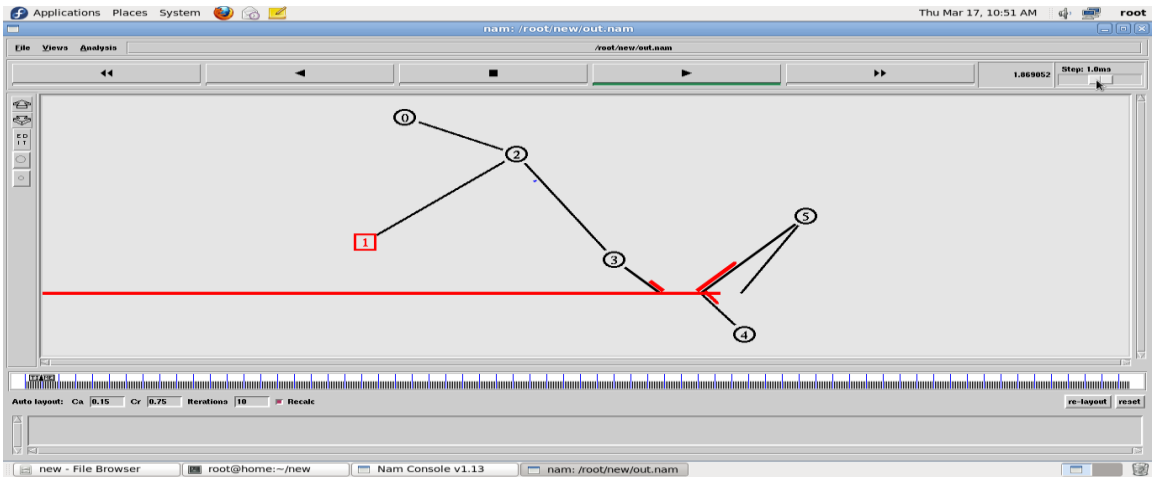


CSMA/CD PROGRAM:

```
setns[newSimulator]
#Definedifferentcolorsfordataflows(forNAM)
$nscolor1 Blue
$ns color 2
Red#OpentheTracefiles
setfile1[openout.trw]
setwinfile[openWinFilew]
$nstrace-all $file1
#Open the
NAM trace
filesetfile2[op
enout.namw]
$ns namtrace-all
$file2#Define a 'finish'
procedureprocfinish {}{
globalnsfile1file2
$ns flush-traceclose
$file1close$file2
execnamout.nam&exit 0
}
#Create six nodesset n0 [$ns
node]set n1 [$ns node]set n2 [$ns
node]set n3 [$ns node]set n4 [$ns
node]set n5[$nsnode]
$n1 colorred
$n1 shapebox
#Createlinksbetweenthenodes
$nsduplex-link$n0 $n22Mb 10ms DropTail
$nsduplex-link$n1 $n22Mb 10ms DropTail
$nssimplex-link$n2$n30.3Mb100msDropTail
$nssimplex-link$n3$n20.3Mb100msDropTail
setlan[$nsnewLan"$n3$n4 $n5"0.5Mb 40msLL Queue/DropTailMAC/Csma/CdChannel]
SetupaTCPconnection
settcp[newAgent/TCP/Newreno]
$nsattach-agent $n0$tcp
setsink[newAgent/TCPSink/DelAck]
$nsattach-agent $n4$sink
$nsconnect$tcp $sink
$tcpsetfid_1
$tcpsetwindow_8000
$tcpsetpacketSize_ 552
#Setup a FTP over
TCP
connectionsetftp[new
Application/FTP]
$ftpattach-agent $tcp
$ftp
settype_FTP
#Setup a
UDP
connectionse
tudp[newAge
nt/UDP]
$ns attach-
agent $n1
$udpsetnull[n
ewAgent/Null
]
```

```
$nsattach-agent$n5$null
$nsconnect$udp$null
$udp setfid_2
#Setup a CBR over UDP
connectionsetcbr[newApp
lication/Traffic/CBR]
$cbrattach-agent$udp
$cbrsettype_ CBR
$cbrsetpacket_size_ 1000
$cbrsetrate_0.01mb
$cbrsetrandom_ false
$nsat 0.1"$cbrstart"
$nsat 1.0"$ftpstart"
$nsat124.0"$ftpstop"
$nsat124.5"$cbrstop"
# next procedure gets two arguments:
the name of the#tcpsource node,
willbecalled here"tcp",
#andthenameof outputfile.
procplotWindow
{tcpSource file}
{globalns
settime 0.1
set now[$ns now]
setcwnd [$tcpSource set cwnd_]set wnd
[$tcpSource set window_]puts
$file"$now$cwnd"
$nsat[expr$now+$time]"plotWindow$tcpSource$file"}
$nsat 0.1"plotWindow$tcp $winfile"
$ns at 5 "$ns trace-annotate
\"packet drop\""# PPP
$nsat 125.0"finish"
$ns run
```

OUTPUT:



RESULT:

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