

**BANGALORE INSTITUTE OF TECHNOLOGY**

*Affiliated to Visvesvaraya Technological University*

K.R.Road, V.V.Puram, Bangalore-560004

**Department of  
INFORMATION SCIENCE & ENGINEERING**

**CGI Programming Laboratory**

**2007-2008**

**B S SANTOSH**  
**Dept of ISE**  
**BIT, Bangalore**  
**2007-2008**





**BANGALORE INSTITUTE OF TECHNOLOGY**  
*Affiliated to Visvesvaraya Technological University*  
K.R.Road, V.V.Puram, Bangalore-560004

**Department of  
INFORMATION SCIENCE & ENGINEERING**

**CERTIFICATE**

*This is to certify that the Networks Laboratory has been satisfactorily completed by  
.....bearing the USN ..... as prescribed by the **Visvesvaraya  
Technological University**, during the VII semester of the academic year 20...-....*

**Head of the Department**  
**R. Nagaraja**  
*Dept. of Information Science  
And Engineering. BIT.*

**Vani V**  
*(Teacher in charge)*

**M Chetana**  
*(Teacher in charge)*

# INDEX

Prg.no.	PROGRAM	Page.no.
<b><u>PART A</u></b>		
<b>1.</b>	Simulate a three nodes point - to - point networks with a duplex links between them .Set the queue size and vary the band width and find the number of packets dropped	<b>1</b>
<b>2.</b>	Simulate a four node point - to - point network , and connect the links as follows: n0 - n2 , n1 - n2 and n2 - n3 . Apply TCP agent between n0 - n3 and UDP n1 - n3 . Apply relevant applications over TCP and UDP agents changing the parameters and determine the number of packets send by TCP/UDP	<b>4</b>
<b>3.</b>	Simulate the different types of Internet traffic such as FTP,TELNET over a network and analyze the throughput	<b>5</b>
<b>4.</b>	Simulate the transmission of ping messages over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion	<b>7</b>
<b>5.</b>	Simulate an Ethernet LAN using N nodes (6 - 10) , change error rate and data rateand compare throughput	<b>8</b>
<b>6.</b>	Simulate an Ethernet LAN using 'n' nodes and set multiple traffic nodes and determine collisions across different nodes	<b>9</b>
<b>7.</b>	Simulate an Ethernet LAN using 'n' nodes and set multiple traffic nodes and plot congestion window for different source and destination	<b>11</b>
<b>8.</b>	Simulate simple BSS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets	<b>12</b>
<b><u>PART B</u></b>		
<b>1.</b>	Write a C/C++ program for error detecting code: CRC-CCITT and CRC-16	<b>13</b>
<b>2.</b>	Write a C/C++ program for frame sorting technique used in buffer	<b>16</b>
<b>3.</b>	Write a C/C++ program for Distance Vector Algorithm to find suitable path for transmission	<b>18</b>
<b>4.</b>	WRITE A C/C++ PROGRAM FOR SPANNING TREE ALGORITHM TO FIND LOOPLESS PATH WITH 6 TO 10 NODES	<b>24</b>
<b>5.</b>	Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the content of the requested file if present	
<b>6.</b>	Implement the above program using message queues or FIFO as IPC channels	
<b>7.</b>	Write a program for simple RSA algorithm to encrypt and decrypt the data key generation algorithm RSA	
<b>8A.</b>	Write a program to generate the HAMMING CODE for the given data	
<b>8B.</b>	Write a program Detection and Correction OF errors in HAMMING CODE	
<b>9.</b>	Write a C/C++ program for congestion control using Leaky Bucket Algorithm	

## **In Networks Laboratory we have 2 parts of exercises,**

### **1. Simulation**

For this we use software called NCTUns. It provides us the provision of simulating network between computers and to analyze how network communication is carried, it also provides the provision to analyze various network parameters like collision, dropped packets, throughput.

### **2. C/C++ programming**

Here we implement basic algorithms that are used extensively used in network communication like, Spanning tree algorithm, Distance vector algorithm, Hamming, RSA encryption algorithm.

## **Hardware and Software Requirements**

### **Hardware Requirements**

- **Processor** : Pentium 3 or higher
- **RAM** : 512MB or more
- **Hard Disk** : 16GB or more( there should be enough space to hold both Linux and Windows)

### **Software Requirements**

- **Operating System** : Windows, Linux
- **Compiler**: Turbo C\C++ or Borland C\C++
- **Simulation Software**: NCTUns

## **Few words about NCTUns**

A high-fidelity and extensible network simulator and emulator. NCTUns directly uses the real-life Linux's TCP/IP protocol stack to generate high-fidelity simulation results. By using the novel kernel re-entering simulation methodology, a real-life UNIX (e.g., FreeBSD or Linux) kernel's protocol stack is directly used to generate high-fidelity simulation results.

## **Support for various network protocols**

NCTUns simulates Ethernet-based IP networks with fixed nodes and point-to-point links. It simulates IEEE 802.11 (b) wireless LAN networks, including both the ad-hoc and infrastructure modes. It simulates GPRS cellular networks. It simulates optical networks, including traditional circuit switching optical network and more advanced optical burst switching (OBS) networks. It simulates IEEE 802.11(b) wireless mesh networks, IEEE 802.11(e) QoS networks, tactical and active mobile ad hoc networks, and wireless networks with directional and steerable antennas. NCTUns 4.0 simulates four new important networks.

## **How to install NCTUns(NCTUns4.0)**

- Linux should be installed in computer
- Login as root user
- Copy prescribed version of NCTUns
- Change working directory to the directory where you have placed NCTUns
- Unzip NCTUns by entering command “tar zxvf NCTUns-allinone-linux-2.6.21.5-f7.20070928.tar”.
- After unzipping go into unzipped directory and type “./install.sh” in command prompt
- Set environmental variables properly, make SELinux disabled. To do these operations instructions are given at end of installation.

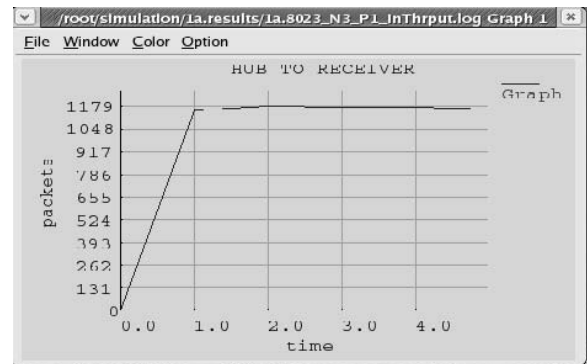
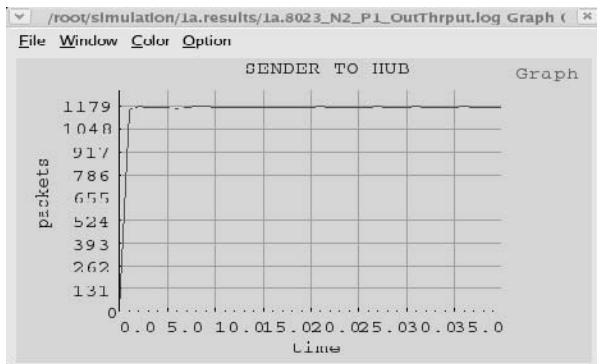
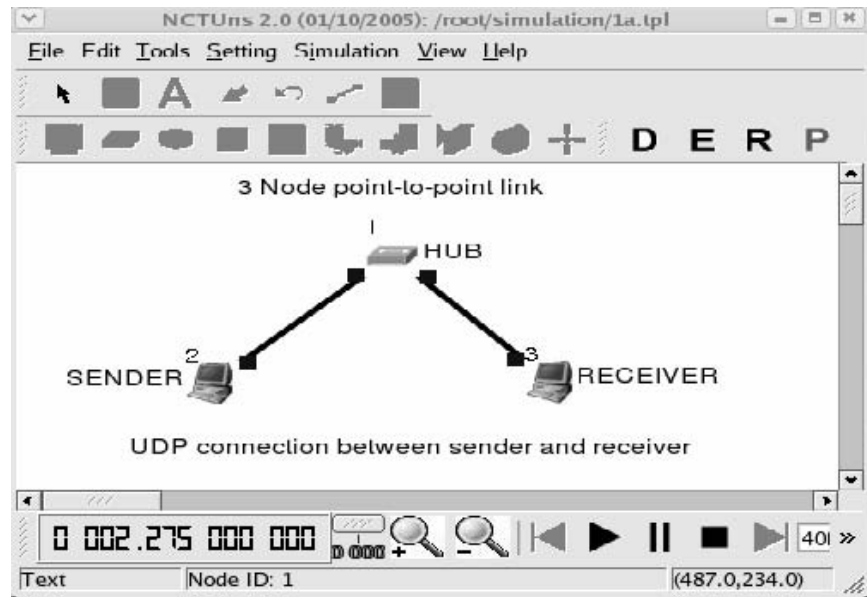
## **To work with NCTUns,**

- Restart the computer and login to NCTUns kernel
- Create a user as we cannot work as root in NCTUns
- Login as new user
- To use NCTUns user should have root user privileges, become root user by using command “su”
- Run dispatcher “/usr/local/nctuns/bin/dispatcher”
- Run coordinator “/usr/local/nctuns/bin/coordinator”
- Run nctuns client “/usr/local/nctuns/bin/nctunsclient”
- Draw the topology in draw mode
- Save it, and edit the parameters
- Run the simulation
- Results of simulation are available in results folder created in user’s directory. Users can open those log files and see what happened in the scenario. Nctuns also provision of drawing graphs, with this facility users can analyze graphs of network parameters like various throughputs, congestion.

## PART-A (Simulation)

1. Simulate a three nodes point - to - point networks with a duplex links between them .Set the queue size and vary the band width and find the number of packets dropped

Using a HUB:



Commands Used:

stg -u 1024 40 1.0.1.2 (At the sender's end)

rtg -u -w log1 (At the receiver's end)

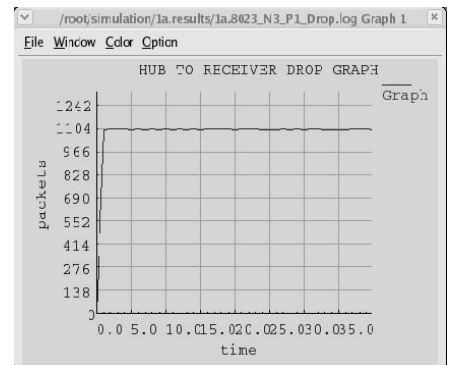
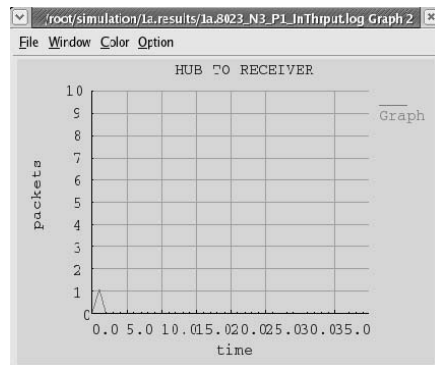
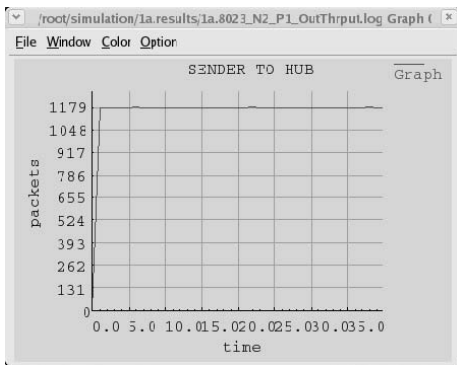
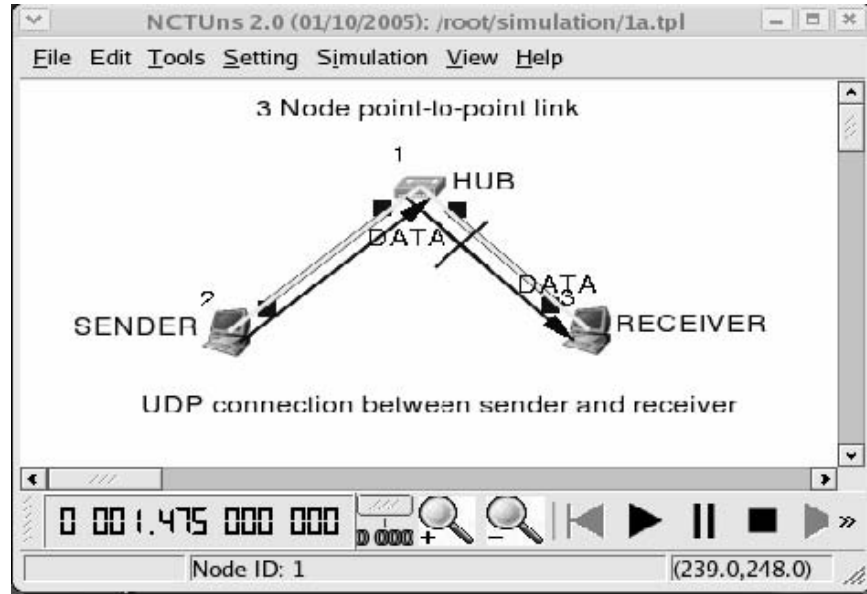
Queue size (fixed) 50

Bandwidth at sender's end 10 Mbps , at receiver's end 10 Mbps

Sender's throughput = 1179

Receiver's throughput = 1179

Bandwidth at sender's end 10 Mbps , at receiver's end 8 Mbps  
 Sender's throughput = 1179  
 Receiver's throughput ~0  
 Receiver's collision and drop = 1100



Using Switch:

Commands used :

step -p 7000 -l 1024 1.0.1.2 (At the sender's end.)

rtcp -p 7000 -l 1024 (At the receiver's end.)

Bandwidth at Sender's end 10 Mbps , at Receiver's end 10 Mbps

Sender's throughput = 1190

Receiver's throughput = 1190

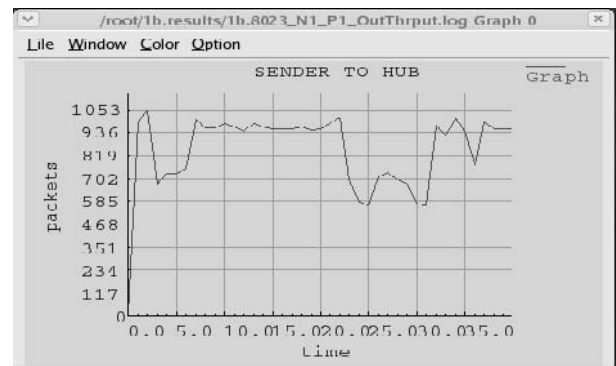
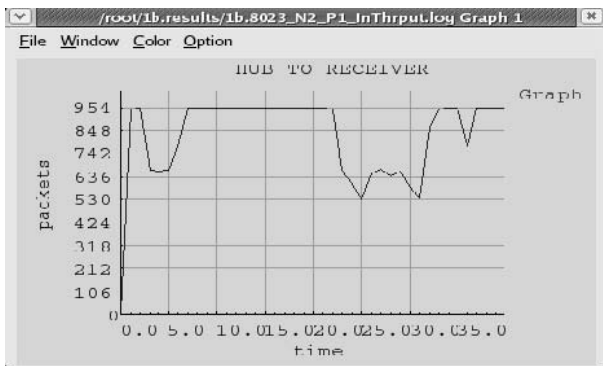
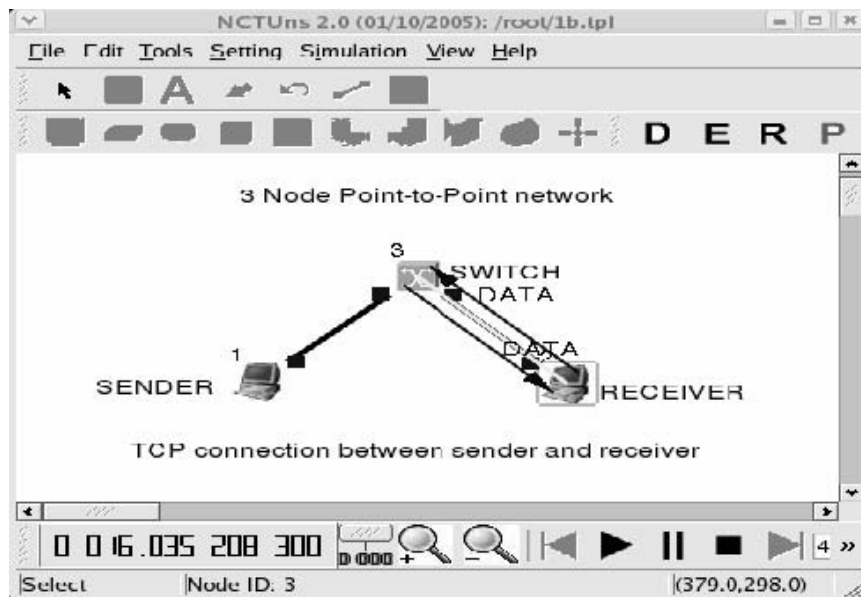
Collision and drop ~0

Bandwidth at Sender's end 10Mbps , at the receiver's end 8Mbps

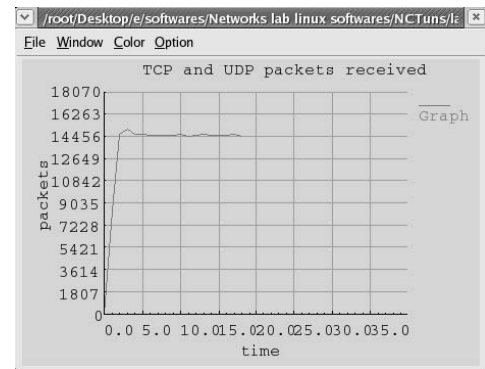
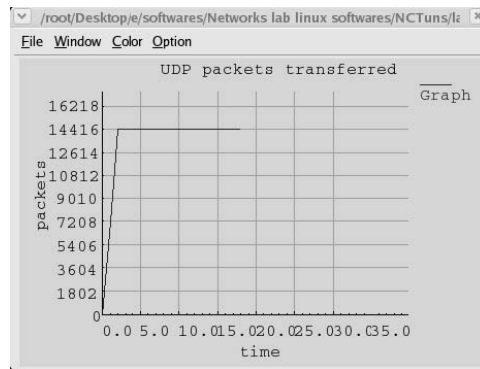
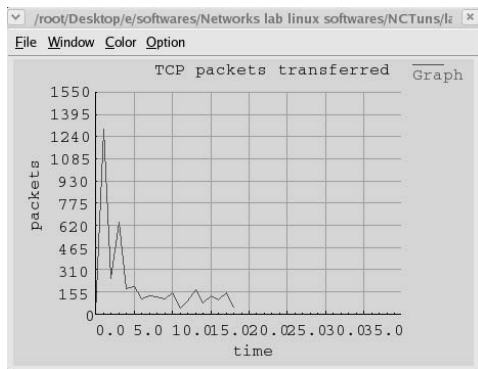
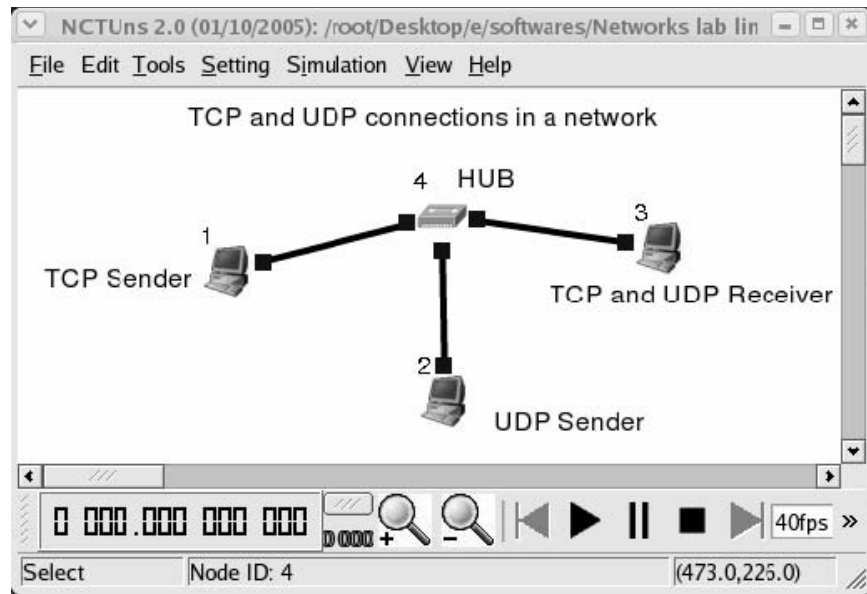
Sender's throughput = 585-1053

Receiver's throughput = 530-954

Collision and Drop = ~0



2. Simulate a four node point - to - point network , and connect the links as follows: n0 - n2 , n1 - n2 and n2 - n3 . Apply TCP agent between n0 - n3 and UDP n1 - n3 . Apply relevant applications over TCP and UDP agents changing the parameters and determine the number of packets send by TCP/UDP.



Commands used:

- stg -u 1400 40 1.0.1.3 (At the UDP sender)
- rtg -u -w log1 (At the receiver)
- rtcp -p 7000 -l 1024 (At the receiver)
- stcp -p 7000 -l 1024 (At the TCP sender)

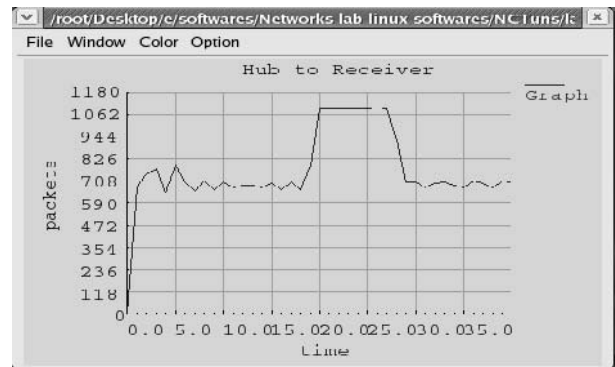
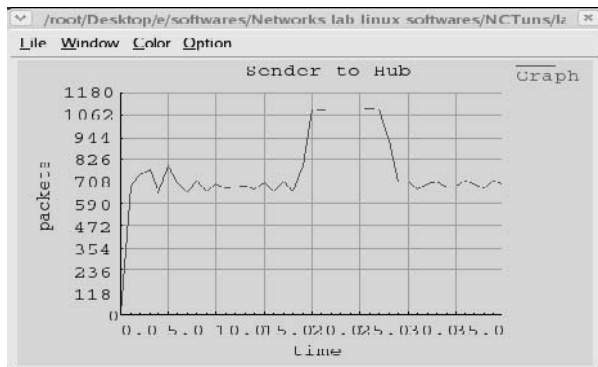
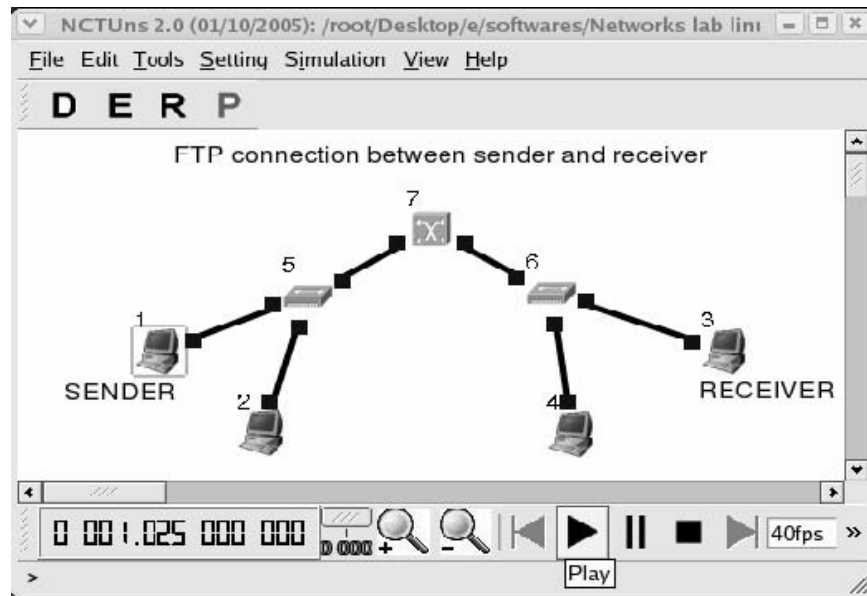
Bandwidth of the network 1000Mbps.

Average no of TCP packets transferred = varying

Average no of UDP packets transferred = 14416

3. Simulate the different types of Internet traffic such as FTP, TELNET over a network and analyze the throughput.

FTP:



Commands Used:

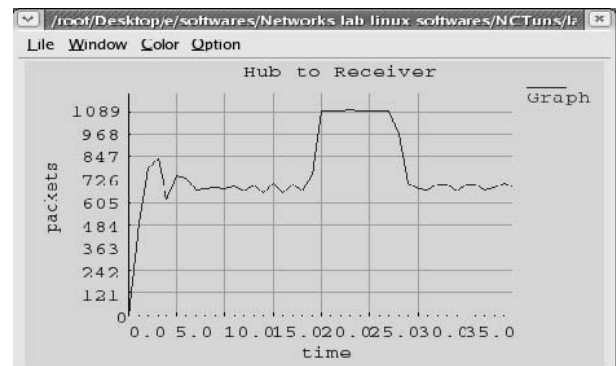
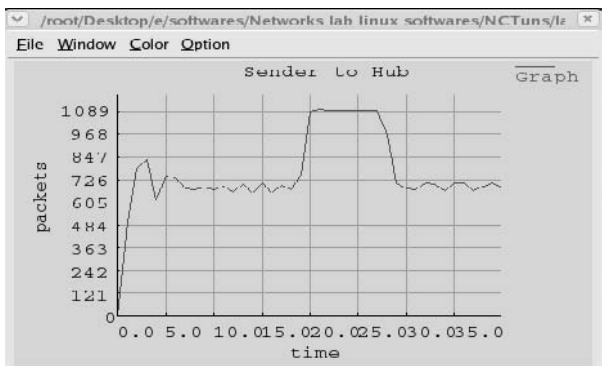
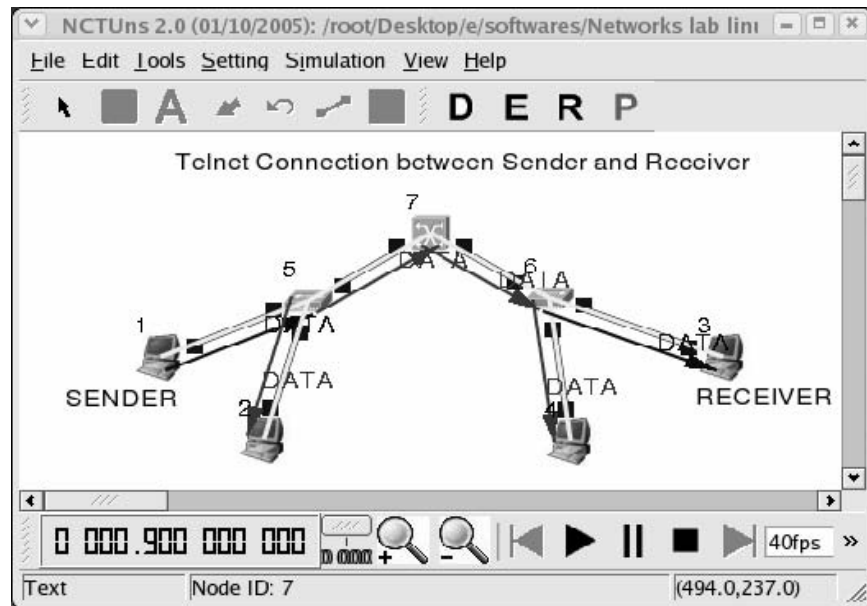
stp -p 21 -l 1024 1.0.1.6 (At the sender's end)

rtc -p 21 -l 1024 (At the receiver's end)

Sender's throughput = 620-1070

Receiver's throughput = 620-1070

Telnet:



Commands Used:

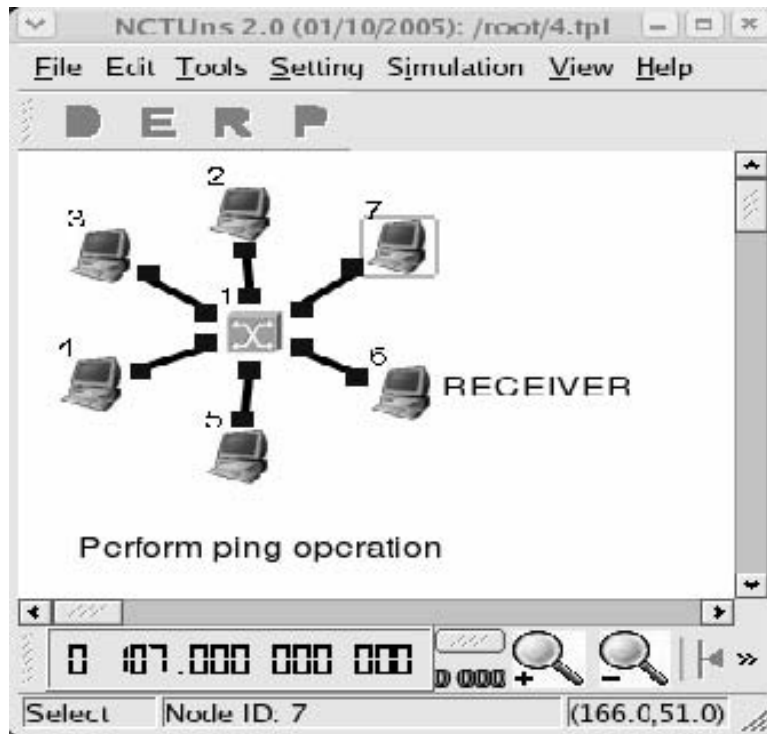
stp -p 23 -l 1024 1.0.1.6 (At the sender's end)

rtc -p 23 -l 1024 (At the receiver's end)

Sender's throughput = 605-1089

Receiver's throughput = 605-1089

4. Simulate the transmission of ping messages over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.



```

rlogin
64 bytes from 1.0.1.5: icmp_seq=58 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=59 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=60 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=61 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=62 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=63 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=64 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=65 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=66 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=67 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=68 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=69 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=70 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=71 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=72 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=73 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=74 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=75 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=76 ttl=64 time=0.3 ms

--- 1.0.1.5 ping statistics ---
77 packets transmitted, 54 packets received, 29% packet loss
round-trip min/avg/max = 0.3/0.3/0.5 ms
%

```

```

rlogin
64 bytes from 1.0.1.5: icmp_seq=22 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=23 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=24 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=25 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=26 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=27 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=28 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=29 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=30 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=31 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=32 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=33 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=34 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=35 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=36 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=37 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=38 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=39 ttl=64 time=0.3 ms
64 bytes from 1.0.1.5: icmp_seq=40 ttl=64 time=0.3 ms

--- 1.0.1.5 ping statistics ---
41 packets transmitted, 37 packets received, 9% packet loss
round-trip min/avg/max = 0.3/0.3/0.5 ms
%

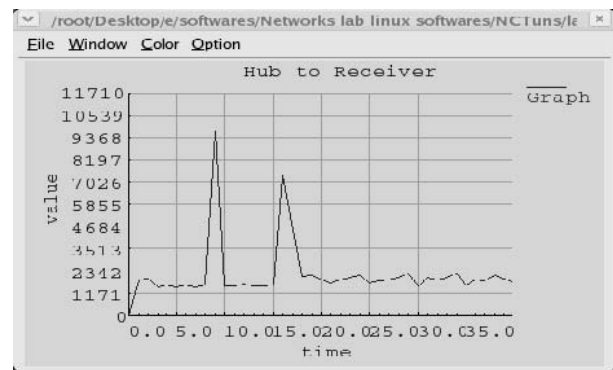
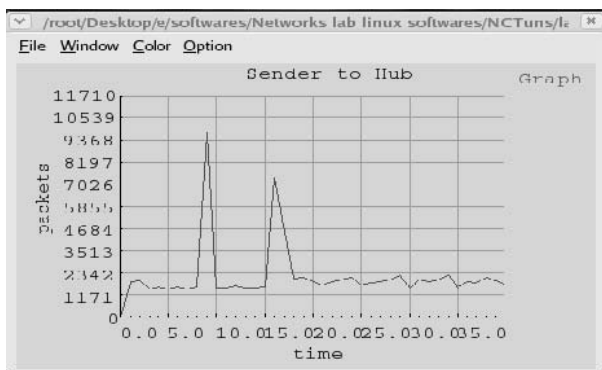
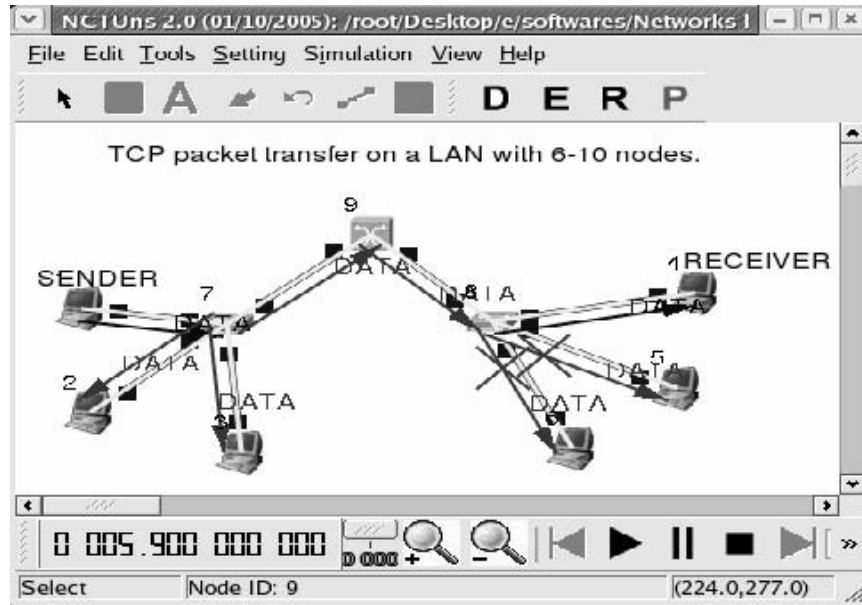
```

### Commands Used

Create UDP connection from all 5 nodes to the receiver.

Ping the receiver from other nodes , then terminate and check the ping statistics.

5. Simulate an Ethernet LAN using N nodes (6 - 10) , change error rate and data rate and compare throughput



Commands Used:

stp -p 7000 -l 1024 1.0.1.6 (At the sender's end)

rtcp -p 7000 -l 1024 (At the receiver's end)

For six nodes:

Initial error rate:0, Initial data rate: 10Mbps

Sender's throughput = 654-1091, Receiver's throughput = 654-1091

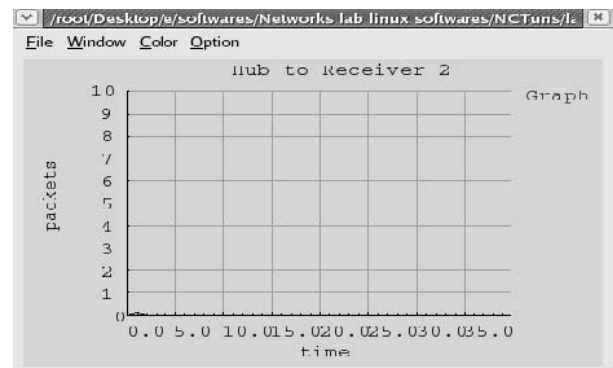
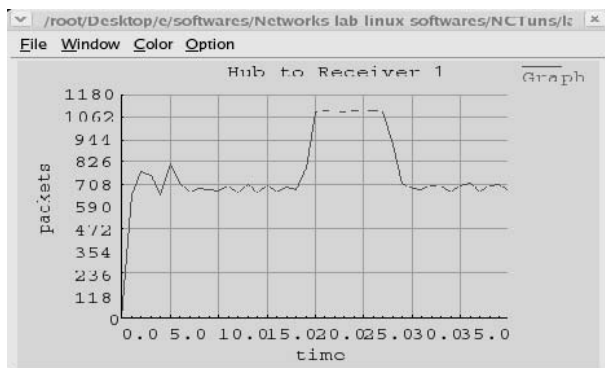
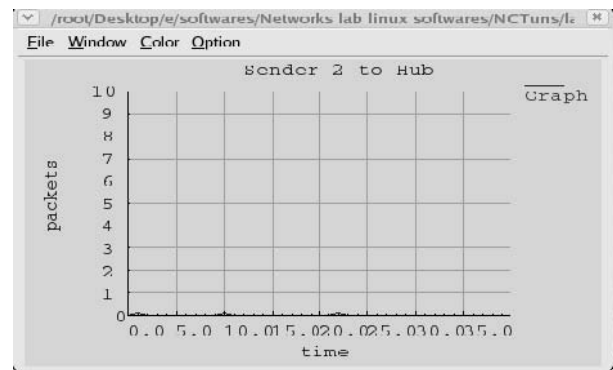
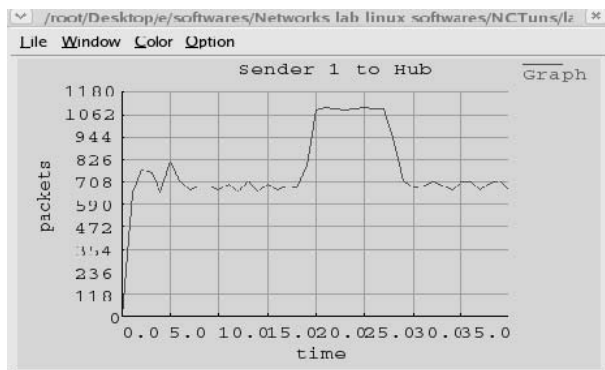
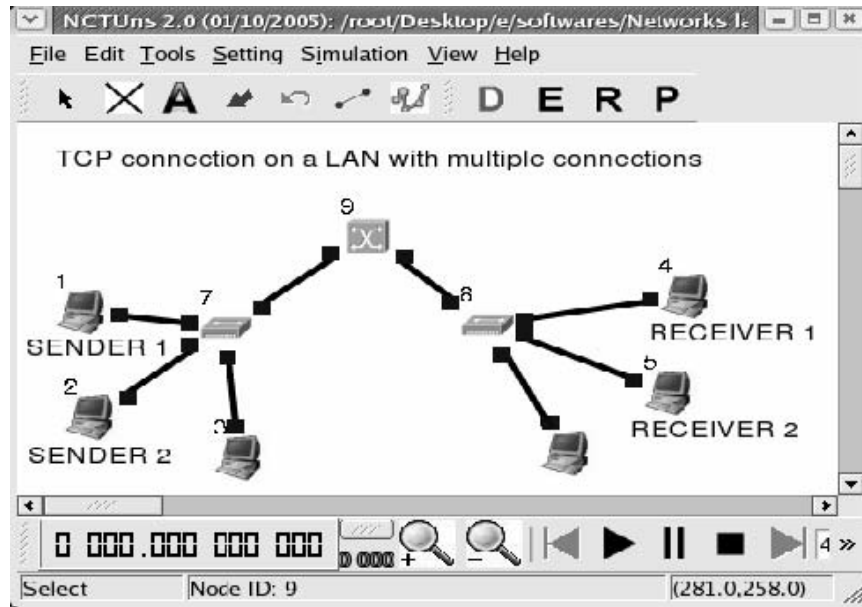
Changed error rate: 1.0, Changed data rate: 10Mbps

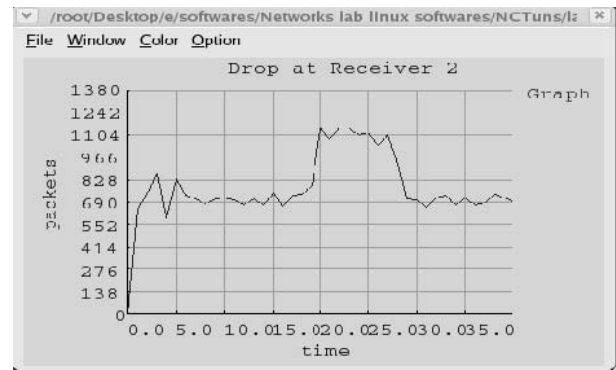
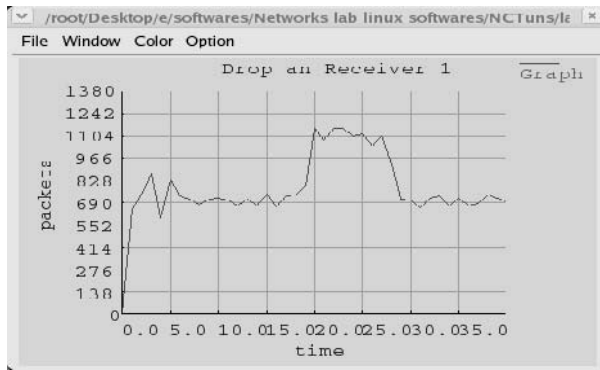
Sender's throughput: 654-1091, Receiver's throughput = 654-1091

Error rate: 1.0,Data rate : 100Mbps

Sender's throughput = 1547-9765, Receiver's throughput = 1547-9765

6. Simulate an Ethernet LAN using 'n' nodes and set multiple traffic nodes and determine collisions across different nodes.





### Commands Used:

```

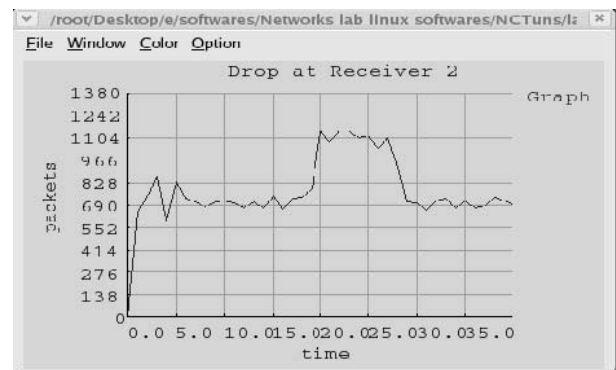
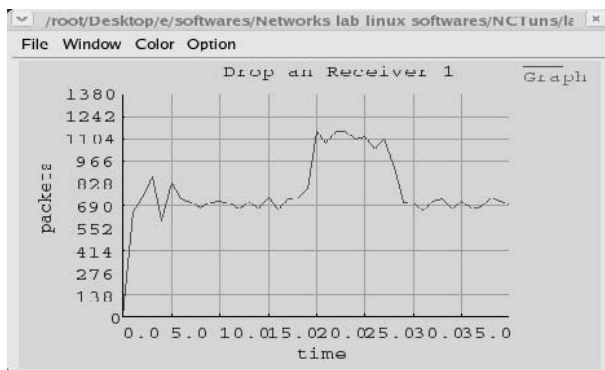
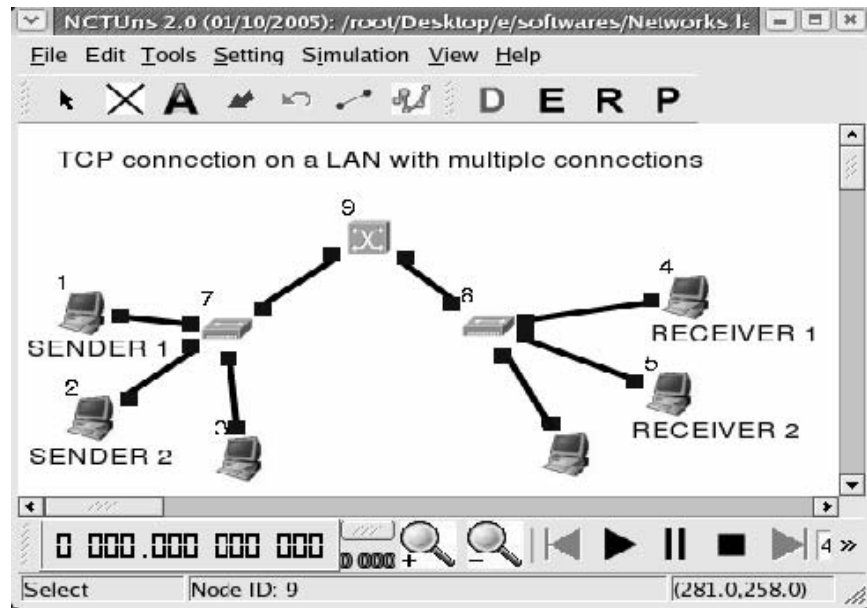
step -p 7000 -l 1024 1.0.1.4 (At sender no 1)
step -p 7001 -l 1024 1.0.5 (At sender no 2)
rtcp -p 7000 -l 1024 (At receiver no 1)
rtcp -p 7001 -l 1024 (At receiver no 2)

```

Drops at Receiver no 1 = 580-1104

Drops at Receiver no 2 = 580-1104

7. Simulate an Ethernet LAN using 'n' nodes and set multiple traffic nodes and plot congestion window for different source and destination.



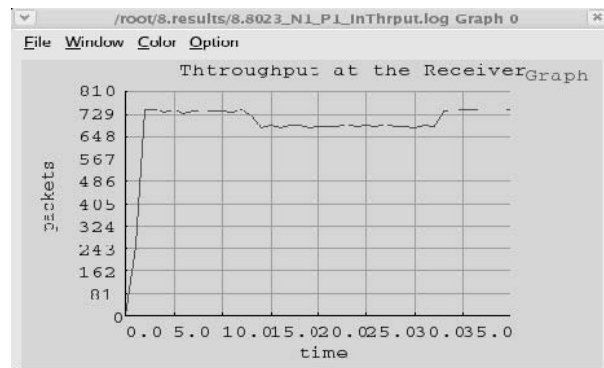
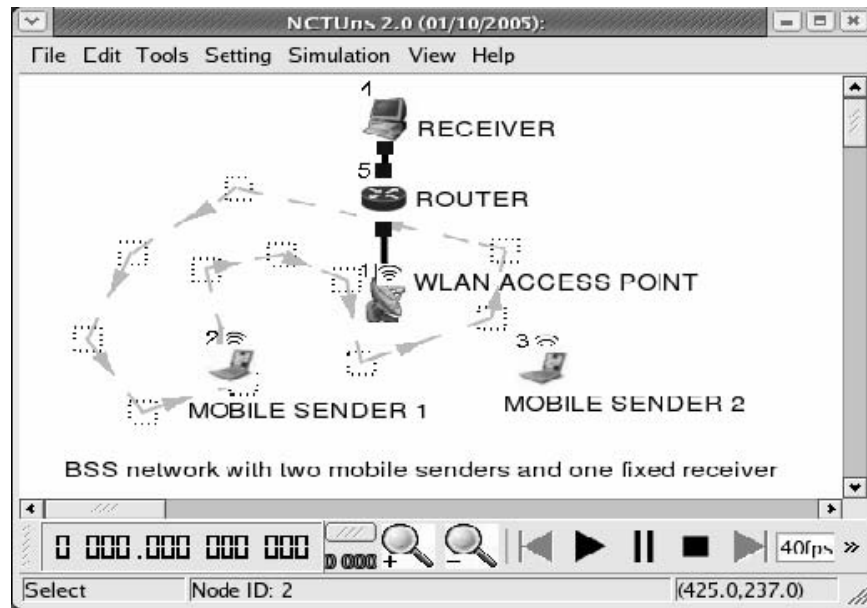
Commands Used:

```
step -p 7000 -l 1024 1.0.1.4 (At sender no 1)
step -p 7001 -l 1024 1.0.5 (At sender no 2)
rtcp -p 7000 -l 1024 (At receiver no 1)
rtcp -p 7001 -l 1024 (At receiver no 2)
```

Drops at Receiver no 1 = 580-1104

Drops at Receiver no 2 = 580-1104

8. Simulate simple BSS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets



Commands Used:

Use a different subnet for the mobile hosts say 1.0.2.0

Use 1.0.2.1 as the gateway for the mobile network.

Use 1.0.2.2 as the IP address for mobile node 1

Use 1.0.2.3 as the IP address for mobile node 2

`ttcp -t -u -s -p 8000 1.0.1.1` (At mobile node 1)

`ttcp -t -u -s -p 8001 1.0.1.1` (At mobile node 2)

`ttcp -r -u -s -p 8000` (At fixed node)

`ttcp -r -u -s -p 8001` (At fixed node)

Receiver's throughput = 660-740

## PART B

### PROGRAM 1

Write a C/C++ program for error detecting code: CRC-CCITT and CRC-16

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<conio.h>
int main()
{
    char rem[50],a[50],s[50],c,msg[50];
    char gen[]="10001000000000101";
    int i,genlen,t,j,flag=0,k,n;
    printf("\nGenerator polynomial is CRC-CCITT:%s",gen);
    genlen=strlen(gen);
    k=genlen-1;
    printf("\nEnter the message:");
    n=0;
    while((c=getchar())!='\n')
    {
        msg[n]=c;
        n++;
    }
    msg[n]='\0';
    for(i=0;i<n;i++)
        a[i]=msg[i];
    for(i=0;i<k;i++)
        a[n+i]='0';
        a[n+k]='\0';
    printf("\nMessage polynomial appended with zero's:");
    puts(a);
    for(i=0;i<n;i++)
    {
        if(a[i]=='1')
        {
            t=i;
```

```

        for(j=0;j<=k;j++)
        {
            if(a[t]==gen[j])
                a[t]='0';
            else
                a[t]='1';
                t++;
        }
    }
}

for(i=0;i<k;i++)
    rem[i]=a[n+i];
rem[k]='\0';

printf("\nThe checksum appended:");

puts(rem);
printf("\nMessage with checksum appended:");
for(i=0;i<n;i++)
    a[i]=msg[i];
for(i=0;i<k;i++)
    a[n+i]=rem[i];
a[n+k]='\0';
puts(a);
n=0;
printf("\nEnter the received polynomial:");
while((c=getchar())!='\n')
{
    s[n]=c;
    n++;
}
s[n]='\0';
for(i=0;i<n;i++)
{
    if(s[i]=='1')
    {
        t=i;
        for(j=0;j<=k;j++,t++)
        {
            if(s[t]==gen[j])
                s[t]='0';
            else
                s[t]='1';
        }
    }
}

for(i=0;i<k;i++)
    rem[i]=s[n+i];
rem[k]='\0';

for(i=0;i<k;i++)
{
    if(rem[i]=='1')
        flag=1;
}

```

```
if(flag==0)
printf("\nThe received polynomial is error free\n");
else
printf("\nThe received polynomial has error\n");
getch();
return 0;
}
```

## OUTPUT :

```
Generator polynomial is CRC-CCITT:10001000000000101
Enter the message:101
```

```
Message polynomial appended with zero's:1010000000000000000
```

```
The checksum appended:0101000000010001
```

```
Message with checksum appended:1010101000000010001
```

```
Enter the received polynomial:1010101000000010001
```

```
The received polynomial is error free
```

```
Generator polynomial is CRC-CCITT:10001000000000101
Enter the message:101
```

```
Message polynomial appended with zero's:1010000000000000000
```

```
The checksum appended:0101000000010001
```

```
Message with checksum appended:1010101000000010001
```

```
Enter the received polynomial:1010101000000010011
```

```
The received polynomial has error
```

## PROGRAM 2

**Write a C/C++ program for frame sorting technique used in buffer**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

struct frame{
int fslno;
char finfo[20];
};

struct frame arr[10];
int n;

void sort()
{
    int i,j,ex;
    struct frame temp;
    for(i=0;i<n;i++)
    {
        ex=0;
        for(j=0;j<n-i-1;j++)
            if(arr[j].fslno>arr[j+1].fslno)
            {
                temp=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp;
                ex++;
            }

        if(ex==0) break;
    }
}
```

```

void main()
{
    int i;
    clrscr();
    printf("\n Enter the number of frames \n");
    scanf("%d",&n);
    for(i=0;i<n;i++)
        {
            arr[i].fslno=random(50);
            printf("\n Enter the frame contents for sequence number
%d\n",arr[i].fslno);
            scanf("%s",arr[i].finfo);
        }
    sort();
    printf("\n The frames in sequence \n");
    for(i=0;i<n;i++)
        printf("\n %d\t%s \n",arr[i].fslno,arr[i].finfo);
    getch();
}

```

## OUTPUT

Enter the number of frames:10

Enter the frame contents for sequence number 3  
institute

Enter the frame contents for sequence number 25  
Of

Enter the frame contents for sequence number 8  
Technology

Enter the frame contents for sequence number 28  
is

Enter the frame contents for sequence number 19  
a

Enter the frame contents for sequence number 33  
very

Enter the frame contents for sequence number 14  
good

Enter the frame contents for sequence number 41  
college

Enter the frame contents for sequence number 45  
did

Enter the frame contents for sequence number 1  
Bangalore

The frames in sequence

1 bangalore  
3 institute  
8 technology  
14 good  
19 a  
25 of  
28 is  
33 very  
41 college

### PROGRAM 3

Write a C/C++ program for distance vector algorithm to find suitable path for transmission

```
#include<stdlib.h>
#define NUL 1000
#define NODES 10

struct node
{
    int t[NODES][3];
};

struct node n[NODES];
typedef struct node NOD;

int main()
{
    void init(int,int);
    void inp(int,int);
    void caller(int,int);
    void opl(int,int,int);
    void find(int,int);
    int i,j,x,y,no;
    clrscr();
    do{
        printf("\n Enter the no of nodes required:");
        scanf("%d",&no);
    }while(no>10||no<0);
    for(i=0;i<no;i++)
    {
        init(no,i);
        inp(no,i);
    }
    printf("\nThe configuration of the nodes after initalization is
as follows:");
    for(i=0;i<no;i++)
        opl(no,i,0);
    for(j=0;j<no;j++)
    {
        for(i=0;i<no;i++)
            caller(no,i);
    }
    printf("\nThe config of the nodes after the comp of the paths is as
follows:");

    for(i=0;i<no;i++)
        opl(no,i,1);
    while(1)
    {
        printf("\n Enter 0 to exit or any other key to find the
shortest path:");
        scanf("%d",&j);
```

```

        if(!j)
        break;
        clrscr();
        do{
            printf("\n Enter the nodes btn which path is to be found
:");
            scanf("%d%d",&x,&y);
        }while((x<0||x>no) && (y<0||y>no));
        printf("\nThe most suitable route from node %d to %d is as
follows\n",x,y);
        find(x,y);
        printf("%d",y);
        printf("\nThe leng of the shortest path btn node %d & %d is %d",x,y,n[x-
1].t[y-1][2]);
    }
}

void init(int no,int x)
{
    int i;
    for(i=0;i<no;i++)
    {
        n[x].t[i][1]=i;
        n[x].t[i][2]=999;
        n[x].t[i][3]=NUL;
    }
    n[x].t[x][2]=0;
    n[x].t[x][3]=x;
}

void inp(int no,int x)
{
    int i;
    printf("\nEnter the dists from the nodes %d to other
node...",x+1);
    printf("\nPls enter 999 if there is no direct \n");
    for(i=0;i<no;i++)
    {
        if(i!=x)
        {
            do
            {
                printf("\n Enter dist to node %d=",i+1);
                scanf("%d",&n[x].t[i][2]);
            }while(n[x].t[i][2]<0|| n[x].t[i][2]>999);
            if(n[x].t[i][2]!=999)
                n[x].t[i][3]=i;
        }
    }
}

void caller(int no,int x)
{
    void compar(int,int,int);
    int i;
    for(i=0;i<no;i++)
    {

```

```

        if(n[x].t[i][2]!=999 && n[x].t[i][2]!=0)
        {
            compar(x,i,no);
        }
    }
}

void compar(int x,int y,int no)
{
    int i,z;
    for(i=0;i<no;i++)
    {
        z=n[x].t[y][2]+n[y].t[i][2];
        if(n[x].t[i][2]>z)
        {
            n[x].t[i][2]=z;
            n[x].t[i][3]=y;
        }
    }
}

void op1(int no,int x,int z)
{
    int i,j;
    printf("\n The routing table for node no %d is as follows",x+1);
    printf("\n\n\t\t\tDESTINATION\tDISTANCE\tNEXT_HOP");
    for(i=0;i<no;i++)
    {
        if((!z && n[x].t[i][2]>=999) ||(n[x].t[i][2]>=(999*no)))
            printf("\n\t\t\t %d \tNO LINK \t NO HOP",n[x].t[i][1]+1);
        else
            if(n[x].t[i][3]==NUL)
                printf("\n\t\t\t %d \t %d \t NO OP",n[x].t[i][1]+1,n[x].t[i][2]);
            else
                printf("\n\t\t\t %d \t %d \t %d",n[x].t[i][1]+1,n[x].t[i][2],n[x].t[i][3]+1);
    }
    getch();
}

void find(int x,int y)
{
    int i,j;
    i=x-1;
    j=y-1;

    printf("%d-->",x);
    if(n[i].t[j][3]!=j)
    {
        find(n[i].t[j][3]+1,y);
        return;
    }
}

```

## OUTPUT

Enter the dists from the nodes 1 to other node...  
Pls enter 999 if there is no direct

/\* Enter dist to node 2=3

Enter dist to node 3=8

Enter dist to node 4=6

Enter dist to node 5=999

Enter the dists from the nodes 2 to other node...  
Pls enter 999 if there is no direct

Enter dist to node 1=3

Enter dist to node 3=999

Enter dist to node 4=7

Enter dist to node 5=5

Enter the dists from the nodes 3 to other node...  
Pls enter 999 if there is no direct

Enter dist to node 1=8

Enter dist to node 2=999

Enter dist to node 4=9

Enter dist to node 5=999

Enter the dists from the nodes 4 to other node...  
Pls enter 999 if there is no direct

Enter dist to node 1=6

Enter dist to node 2=7

Enter dist to node 3=9

Enter dist to node 5=1

Enter the dists from the nodes 5 to other node...  
Pls enter 999 if there is no direct

Enter dist to node 1=999

Enter dist to node 2=5

Enter dist to node 3=999

Enter dist to node 4=1

The configuration of the nodes after initialization is as follows:

The routing table for node no 1 is as follows

DESTINATION	DISTANCE	NEXT_HOP
1	0	1
2	3	2
3	8	3
4	6	4
5	NO LINK	NO HOP

The routing table for node no 2 is as follows

DESTINATION	DISTANCE	NEXT_HOP
1	3	1
2	0	2
3	NO LINK	NO HOP
4	7	4
5	5	5

The routing table for node no 3 is as follows

DESTINATION	DISTANCE	NEXT_HOP
1	8	1
2	NO LINK	NO HOP
3	0	3
4	9	4
5	NO LINK	NO HOP

The routing table for node no 4 is as follows

DESTINATION	DISTANCE	NEXT_HOP
1	6	1
2	7	2
3	9	3
4	0	4
5	1	5

The routing table for node no 5 is as follows

DESTINATION	DISTANCE	NEXT_HOP
1	NO LINK	NO HOP
2	5	2
3	NO LINK	NO HOP
4	1	4
5	0	5
5	1	5

The config of the nodes after the completion of the paths is as follows:

The routing table for node no 1 is as follows

DESTINATION	DISTANCE	NEXT_HOP
1	0	1
2	3	2
3	8	3
4	6	4
5	7	4

The routing table for node no 2 is as follows

DESTINATION	DISTANCE	NEXT_HOP
1	3	1
2	0	2
3	11	1
4	6	5
5	5	5

The routing table for node no 3 is as follows

DESTINATION	DISTANCE	NEXT_HOP
1	8	1
2	11	1
3	0	3
4	9	4
5	10	4

The routing table for node no 4 is as follows

DESTINATION	DISTANCE	NEXT_HOP
1	6	1
2	6	5
3	9	3
4	0	4
5	1	5

The routing table for node no 5 is as follows

DESTINATION	DISTANCE	NEXT_HOP
1	7	4
2	5	2
3	10	4
4	1	4
5	0	5

Enter 0 to exit or any other key to find the shortest path:

Enter the nodes btn which path is to be found :1 5

The most suitable route from node 1 to 5 is as follows

1-->4-->5

The length of the shortest path btn node 1 & 5 is 7

Enter 0 to exit or any other key to find the shortest path:0

## PROGRAM 4

WRITE A C/C++ PROGRAM FOR SPANNING TREE ALGORITHM TO  
FIND LOOPLESS PATH WITH 6 TO 10 NODES

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 20
#define INFINITY 999
#include<conio.h>

enum boolean {FALSE,TRUE};
void prim(int c[][MAX],int t[MAX],int n);
int mincost=0;          //initialize minimum cost to 0

int main()
{
    int n,c[MAX][MAX],t[2*(MAX-1)];
    int i,j;
    clrscr();
    printf("\nTo find min path spanning tree");
    printf("\nEnter no of nodes:");
    scanf("%d",&n);
    printf("\nEnter the cost adjacency matrix");
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            scanf("%d",&c[i][j]);          //distance between the nodes
    prim(c,t,n);                          //algorithm to find the minimum
spanning tree
    for(i=0;i<2*(n-1);i+=2)
        printf("\n(%d %d)",t[i]+1,t[i+1]+1);          //nodes of the
spanning tree
    printf("\nmincost=%d",mincost);
    getch();
    return 0;
}

//using prim's algorithm for finding shortest path
void prim(int c[][MAX],int t[MAX],int n)
{
    int i,j;
    enum boolean v[MAX];
    int k,s,min,v1,v2;
    for(i=0;i<n;i++)
        v[i]=FALSE;
    v[0]=TRUE;
    k=0;
    t[k]=1;
    s=0;
    k++;
    while(k<n)
    {
```

```

min=INFINITY;
for(i=0;i<n;i++)
    for(j=1;j<n;j++)
        //while there is no path b/w any 2 nodes and dist is less
than minimum
        if(v[i]==TRUE && v[j]==FALSE && c[i][j]<min)
        {
            min=c[i][j];
            v1=i;
            v2=j;
        }
mincost=mincost+min; //add the mindist to the mincost
if(min==INFINITY)
{
    printf("graph disconnected");
    exit(0);
}
v[v2]=TRUE;
k++;
t[s++]=v1;
t[s++]=v2;
}
}

```

## OUTPUT:

```

to find min path spanning tree
enter no of nodes:10

enter the cost adjacency matrix
0 1 999 999 999 999 999 999 9 999
1 0 2 999 999 999 999 999 999 999
999 2 0 3 999 999 999 999 999 12
999 999 3 0 4 999 999 999 999 999
999 999 999 4 0 5 999 999 999 11
999 999 999 999 5 0 6 999 999 999
999 999 999 999 999 6 0 7 999 13
999 999 999 999 999 999 7 0 8 999
9 999 999 999 999 999 999 8 0 10
999 999 12 999 11 999 13 999 10 0

(1 2)
(2 3)
(3 4)
(4 5)
(5 6)
(6 7)
(7 8)
(8 9)
(9 10)
mincost=46

```

## Program 5

Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the content of the requested file if present

```
//server 5b.c

#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<stdlib.h>
int main()
{
    int cs,ns,fd,n;
    int bufsize=1024;
    char *buffer=malloc(bufsize);
    struct sockaddr_in address;
    char fname[255];
    address.sin_family=AF_INET;
    address.sin_port=htons(15000);
    address.sin_addr.s_addr=INADDR_ANY;

    cs=socket(AF_INET,SOCK_STREAM,0);

    bind(cs,(struct sockaddr *)&address,sizeof(address));

    listen(cs,3);

    ns=accept(cs,(struct sockaddr *)NULL,NULL);

    recv(ns,fname,255,0);

    fd=open(fname,O_RDONLY);
    n=read(fd,buffer,bufsize);
    send(ns,buffer,n,0);

    close(ns);
    return close(cs);
}
```

```

//client 5b.c

#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<stdlib.h>
int main(int argc,char **argv)
{
    int cs,n;
    int bufsize=1024;
    char *buffer=malloc(bufsize);
    char fname[255];
    struct sockaddr_in address;

    address.sin_family=AF_INET;
    address.sin_port=htons(15000);
    inet_pton(AF_INET,argv[1],&address.sin_addr);

    cs=socket(AF_INET,SOCK_STREAM,0);

    connect(cs,(struct sockaddr *)&address,sizeof(address));

    printf("\nEnter filename: ");scanf("%s",fname);
    send(cs,fname,255,0);

    while((recv(cs,buffer,bufsize,0))>0)
    printf("%s",buffer);
    printf("\nEOF\n");
    return close(cs);
}

```

## Output :

```

/*FIRST TERMINAL
cc shwetha.c
./a.out 5073
SERVER:waiting for client

SECOND TERMINAL
cc shwel.c
./client 127.0.01 5073
client online!
server online!
client:Enter path with filename
data.txt
client:displaying contents of data.txt
finally over

FIRST TREMINAL
cc shwetha.c
./a.out 5073
SERVER:Waiting for client SERVER:data.txt
SERVER:data.txt found!  transferring the contents

```

## Program 6

Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the content of the requested file if present. Implement the above program using message queues or FIFO as IPC channels

```
//server6.c

#include<stdio.h>
#include<stdlib.h>
#include<errno.h>
#include<string.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>

#define FIFO1 "fifo1"
#define FIFO2 "fifo2"

int main()
{ char p[100],f[100],c[100];
  int num,num2,f1,fd,fd2;
  mknod(FIFO1,S_IFIFO|0666,0);
  mknod(FIFO2,S_IFIFO|0666,0);
  printf("\n\nSERVER ONLINE.....\n\n\n");
  fd = open(FIFO1,O_RDONLY);
  printf("\n\nwaitin for request.....\n\n");
  while(1)
    { if((num=read(fd,p,100))==-1)
      perror("\n\nREAD error\n\n");
      else
        { p[num]='\0';
          if((f1 = open(p,O_RDONLY)) < 0)
            { printf("\n\nServer : %s not found ",p);
              exit(1); }
          else
            { printf("\nServer : %s is
found.transferring",p);
              stdin = fdopen(f1,"r");
              if(fgets(c,1024,stdin)!=NULL)
                { fd2 = open(FIFO2,O_WRONLY);
                  if(num2=write(fd2,c,strlen(c))==-1)
                    perror("\n\ntransfer error
.....\n\n");
                  else
                    printf("\nServer transfer
complete!!\n\n");
                }
            }
          else
            perror("\n\nRead Error!! \n\n");
        }
    }
}
```

```

        exit(1);
    }
}

}

//client6.c

#include<stdio.h>
#include<stdlib.h>
#include<errno.h>
#include<string.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>

#define FIFO1 "fifo1"
#define FIFO2 "fifo2"

int main()
{ char p[100],f[100],c[300];
  int num,num2,f1,fd,fd2;
  mknod(FIFO1,S_IFIFO|0666,0);
  mknod(FIFO2,S_IFIFO|0666,0);
  printf("\n\nWaiting for SERVER.....\n\n\n");
  fd = open(FIFO1,O_WRONLY);
  printf("\n\n\nServer online !! ENTER the path.....\n\n\n");

  while(scanf("%s",&p) && (!feof(stdin)))
    { if((num=write(fd,p,strlen(p)))==-1)
      perror("\n\nWrite Error");
      else
        { printf("\n\nWaiting for reply.... \n\n");
          fd2 = open(FIFO2,O_RDONLY);
          if((num2=read(fd2,c,300))==-1)
            perror("\n\nTransfer error.... \n\n");
          else
            { printf("\nFILE received.Showin
contents..\n");
              if(fputs(c,stdout)==EOF)
                perror("Print Error...\n\n");
              exit(1);
            }
          }
    }
}

```

## Output :

```
Server:  
cc fifo.c  
./a.out  
Server online!  
Client online!  
Waiting for request  
Sever: abc.txt found!  
Transferring the contents. .
```

```
Transfer complete  
Client:  
cc.fifo2.c  
./a.out  
Waiting for server  
Server online!  
Client: Enter the path  
abc.txt  
Waiting for reply  
File received  
Displaying the contents
```

Hello

File exists

## Program 7

**Write a program for simple RSA algorithm to encrypt and decrypt the data  
key generation algorithm RSA**

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

double min(double x, double y)
{
    return(x<y?x:y);
}

double max(double x,double y)
{
    return(x>y?x:y);
}

double gcd(double x,double y)
{
    if(x==y)
        return(x);
    else
        return(gcd(min(x,y),max(x,y)-min(x,y)));
}

long double modexp(long double a,long double x,long double n)
{
    long double r=1;

    while(x>0)
    {
        if ((int)(fmodl(x,2))==1)
        {
            r=fmodl((r*a),n);
        }

        a=fmodl((a*a),n);
        x/=2;
    }
    return(r);
}

int main()
{
    long double p,q,phi,n,e,d,cp,cq,dp,dq,mp,mq,sp,sq,rp,rq,qInv,h;
    long double ms,es,ds;

    do{
        printf("\n Enter prime numbers p and q:");
        scanf(" %Lf %Lf",&p,&q);
    }
    while(p==q);
```

```

n=p*q;
phi=(p-1)*(q-1);

do{
    printf("\n Enter prime value of e:");
    scanf(" %Lf",&e);
}
while((gcd(e,phi)!=1)&&e>phi); /*for e being relatively prime to
phi */

for(d=1;d<phi;++d)
{
    if(fmod((e*d),phi)==1)
        break;
}

printf("\n D within main = %Lf",d);

/* public key is {n,e} private key is d */

printf("\n Enter the message:");
scanf(" %Lf",&ms);

es=modexp(ms,e,n);
ds=modexp(es,d,n);

printf("\n Original Message : %Lf",ms);
printf("\n Encrypted Message : %Lf",es);
printf("\n Decrypted Message : %Lf\n",ds);

return(0);
}

```

### OUTPUT:

```

Enter prime numbers p and q:
223
101
Enter prime value of e:
2213
D within main = 18077.000000
Enter the message:123

Original Message : 123.000000
Encrypted Message : 18415.000000
Decrypted Message : 123.000000

```

## PROGRAM 8a

WRITE A PROGRAM TO GENERATE THE HAMMING CODE FOR THE GIVEN

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>

void main()
{
    /* variable declaration */
    int d[7];          /* Holds input data */
    int c[7];          /* Holds the 7-bit info generated */
    int i;

    printf("Enter the data bits(0's and 1's) with blank spaces
between them\n");
    for(i=0;i<4;i++)
        scanf("%d",&d[i]);

    /* Now the array d[] contains the 4 data bits in binary form */
    /* Next step is to find the output matrix c[] which has its
       first 4 bits as in d[] and the next 3 bits as per the
       equations,
       b5=b1+b3+b4
       b6=b1+b2+b4
       b7=b2+b3+b4 */

    for(i=0;i<4;i++)
        c[i]=d[i];

    c[4]=(d[0]+d[2]+d[3])%2;    /* Binary addition */
    c[5]=(d[0]+d[1]+d[3])%2;
    c[6]=(d[1]+d[2]+d[3])%2;

    /* Displaying the contents of the matrices c[] and d[] */
    printf("\nThe data bits are\n");
    for(i=0;i<4;i++)
        printf("%d",d[i]);

    printf("\nThe data after the addition of correction bits\n");
    for(i=0;i<7;i++)
        printf("%d",c[i]);
    getch();
}
```

## **OUTPUT:**

Enter the data bits(0's and 1's) with blank spaces between them  
1 0 1 0

The data bits are

1010

The data after the addition of correction bits

1010011

Enter the data bits(0's and 1's) with blank spaces between them  
1 1 0 0

The data bits are

1100

The data after the addition of correction bits

1100101

## PROGRAM 8b

WRITE A PROGRAM FOR DETECTION AND CORRECTION OF ERRORS  
IN HAMMING CODE

```
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>

void main()
{
    int h[3][7]={1,0,1,1,1,0,0,
                 1,1,0,1,0,1,0,
                 0,1,1,1,0,0,1};

    int r[7];          /* Holds input information */
    int s[3];          /* The syndrome matrix */
    int index;         /* Holds the error bit position */
    int i,j,sum;

    printf("Enter the 7-bit information(0's and 1's) with blank
spaces between each bit\n");
    for (i=0;i<7;i++)
        scanf("%d",&r[i]);
    for(j=0;j<3;j++)
    {
        sum=0;
        for(i=0;i<7;i++)
            sum+=h[j][i]*r[i];
        sum=sum%2;
        s[j]=sum;
    }

    /* Now we check if the syndrome matrix s[] is a null vector or
not. If it is, then the accepted info is error-free. Or else
we need to detect and correct the error */

    if(s[0]==0 && s[1]==0 && s[2]==0)
    {
        printf("\nError-free information\n");
        printf("The data bits are\n");
        for(i=0;i<4;i++)
            printf("%d",r[i]);
        printf("\n");
        exit(0);
    }

    /* Error bit position detection and correction
/* We compare the syndrome s[] matrix with each column of the
h[][] matrix until a match is found. The matching column
determines the index of the error bit and we can compliment
the bit at that position to get the corrected data */

    for(j=0;j<7;j++)
    {
        if(s[0]==h[0][j] && s[1]==h[1][j] && s[2]==h[2][j])
        {
```

```

        index=j;
        break;
    }
}

printf("\nThe error is in the bit no- %d\n",index+1);

/* compliment the error bit */
if(r[index]==0)
    r[index]=1;
else
    r[index]=0;

printf("The corrected information is\n");
for(i=0;i<7;i++)
    printf("%d",r[i]);
printf("\nThe data bits are\n");
for(i=0;i<4;i++)
    printf("%d",r[i]);
getch();
}

```

## OUTPUT

Enter the 7-bit information(0's and 1's) with blank spaces between each bit

1 0 1 0 0 1 1

Error-free information

The data bits are

1010

Enter the 7-bit information(0's and 1's) with blank spaces between each bit

1 0 1 0 0 0 1

The error is in the bit no- 6

The corrected information is

1010011

The data bits are

1010

## PROGRAM 9

Write a C/C++ program for congestion control using Leaky Bucket algorithm

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>

void main()
{ int
packets[8],i,j,clk,b_size,o_rate,i_rate,p_sz_rm=0,p_sz,p_time;
clrscr();
for(i=0;i<5;++i)
{ packets[i]=rand()%10;
if(packets[i]==0) --i;
}
printf("Enter output rate:");
scanf("%d",&o_rate);
printf("\nEnter bucket size:");
scanf("%d",&b_size);
for(i=0;i<5;++i)
{ if((packets[i]+p_sz_rm) > b_size)
{ if(packets[i]>b_size)
printf("\nIncoming packet size:%d greater than
bucket capacity\n",packets[i]);
else printf("Bucket size exceeded\n");
}
else
{
p_sz=packets[i];
p_sz_rm+=p_sz;
printf("\n-----
\n");
printf("Incoming packet:%d",p_sz);
printf("\nTransmission left:%d\n",p_sz_rm);
p_time=rand()%10;
printf("Next packet will come at %d",p_time);
for(clk=0;clk<p_time&&p_sz_rm>0;++clk)
{
printf("\nTime left %d---No packets to
transmit!!\n",p_time-clk);
sleep(1);
if(p_sz_rm)
{ printf("Transmitted\n");
if(p_sz_rm<o_rate)
p_sz_rm=0;
else p_sz_rm-=o_rate;
printf("Bytes remaining:%d\n",p_sz_rm);
}
else printf("No packets to transmit\n");
}
}
}
}
getch();
```

}  
**OUTPUT:**

Enter output rate:2

Enter bucket size:10

-----  
Incoming packet:6  
Transmission left:6  
Next packet will come at 7  
Time left 7-----No packets to transmit!!  
Transmitted  
Bytes remaining:4

Time left 6-----No packets to transmit!!  
Transmitted  
Bytes remaining:2

Time left 5-----No packets to transmit!!  
Transmitted  
Bytes remaining:0

-----  
Incoming packet:0  
Transmission left:0  
Next packet will come at 5

-----  
Incoming packet:2  
Transmission left:2  
Next packet will come at 5  
Time left 5-----No packets to transmit!!  
Transmitted  
Bytes remaining:0

-----  
Incoming packet:0  
Transmission left:0  
Next packet will come at 8

-----  
Incoming packet:6  
Transmission left:6  
Next packet will come at 6  
Time left 6-----No packets to transmit!!  
Transmitted  
Bytes remaining:4

Time left 5-----No packets to transmit!!  
Transmitted  
Bytes remaining:2

Time left 4-----No packets to transmit!!  
Transmitted  
Bytes remaining:0

## Here are some programs that we have tried,

### CRC-CCTIT Encoding (Only bit stream)

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

char gen[]="10001000000100001";
int k,t;

void cal_crc(char *a,int n)
{
    int i,j;
    for(i=0;i<n;i++)
    {
        if(a[i]=='1')
        {
            t=i;
            for(j=0;j<=k;j++)
            {
                if(a[t]==gen[j])
                    a[j]=0;
                else
                    a[j]=1;
            }
        }
    }
}

int main()
{
    char rem[25],a[25],*s,*msg,c;
    int i,j,genlen,n,flag=0;
    printf("\n generator polynomial : %s",gen);
    genlen=strlen(gen);
    k=genlen-1;
    printf("\n enter the message : ");
    scanf("%s",msg);
    n=strlen(msg);
    strcpy(a,msg);
    for(i=0;i<k;i++)
        a[n+i]='0';
    a[n+k]='\0';

    printf("\n message with appended zeroes : %s",a);
    cal_crc(a,n);

    for(i=0;i<k;i++)
        rem[i]=a[n+i];
    rem[k]='\0';

    printf("\n checksum appended : %s",rem);
    strcpy(a,msg);
    strcat(a,rem);
}
```

```

printf("\n encoded bit string : %s",a);

printf("\n enter the recieved polynomial :");
scanf("%s",s);
n=strlen(s);

cal_crc(s,n);

for(i=0;i<k;i++)
    rem[i]=s[n+i];
rem[k]='\0';

for(i=0;i<k;i++)
{
    if(rem[i]=='1')
        flag=1;
}
if(flag==0)
    printf("\n no error ");
else
    printf("\n error");
return 0;
}

```

## LEAKY BUCKET

```

#include<stdio.h>
#include<stdlib.h>

#define MIN(x,y) (x>y)?y:x

int main()
{
    int orate,drop=0,cap,x,count=0,inp[10]={0},i=0,nsec,ch;
    printf("\n enter bucket size : ");
    scanf("%d",&cap);
    printf("\n enter output rate :");
    scanf("%d",&orate);
    do{
        printf("\n enter number of packets coming at second %d :
",i+1);
        scanf("%d",&inp[i]);
        i++;
        printf("\n enter 1 to contiue or 0 to quit.....");
        scanf("%d",&ch);
    }while(ch);

    nsec=i;
    printf("\n second \t recieved \t sent \t dropped \t
remained \n");
    for(i=0;count || i<nsec;i++)

```

```

{
    printf("      %d",i+1);
    printf(" \t%d\t ",inp[i]);
    printf(" \t      %d\t ",MIN((inp[i]+count),orate));
    if((x=inp[i]+count-orate)>0)
    {
        if(x>cap)
        {
            count=cap;
            drop=x-cap;
        }
        else
        {
            count=x;
            drop=0;
        }
    }
    else
    {
        drop=0;
        count=0;
    }
    printf(" \t %d      \t %d \n",drop,count);
}
return 0;
}

```

## DISTANCE VECTOR ALGORITHM

```

#include<stdio.h>
#define MAX 10
struct dist_vect
{
    int dist[MAX];
    int from[MAX];
};
int main()
{
    int adj[MAX][MAX],n,i,j,hop[10][10]={0},k,count;
    struct dist_vect arr[10];
    printf("Enter the number of nodes\n");
    scanf("%d",&n);
    printf("Enter adjecency matrix\n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            scanf("%d",&adj[i][j]);
    }
    for(i=0;i<n;i++)
    {

```



```

long p,q,n,z,e,d=1;

long gcd(long x,long y)
{
    if(y==0)
        return x;
    if(y>x)
        return gcd(y,x);
    return gcd(y,x%y);
}

long rsa(long c,int flag)
{
    long t=1;
    int i;
    int val=flag?e:d;
    for(i=0;i<val;i++)
        t=(c*t)%n;
    return t;
}

int main()
{
    long int plain[100],encrypted[100],decrypted[100],i;
    char str[100];
    printf("\n enter 2 prime numbers p and q :\n");
    scanf("%d%d",&p,&q);
    n=p*q;
    z=(p-1)*(q-1);
    do{
        printf("\n enter the prime value of e :\n");
        scanf("%d",&e);
    }while(gcd(e,z)!=1 && e>n);
    while(((e*d)-1)%z)
        d++;

    printf("\n enter plain text :");
    scanf("%s",str);
    printf("\n encrypted text : \n");
    for(i=0;i<strlen(str);i++)
    {
        encrypted[i]=rsa(str[i],ENCRY);
        printf("%ld",encrypted[i]);
    }

    printf("\n decrypted text :\n");
    for(i=0;i<strlen(str);i++)
    {
        plain[i]=rsa(encrypted[i],DECRY);
        printf("%c",plain[i]);
    }
}

```

## CLIENT/SERVER PROGRAM

## Client

```
#include<stdio.h>
#include<unistd.h>
#include<sys/socket.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<stdlib.h>
#include<fcntl.h>

int main(int argc,char *argv[])
{
int cs,n,bufsize=1024;
char fname[256],*buf=malloc(bufsize);
struct sockaddr_in address;

address.sin_family=AF_INET;
address.sin_port=htons(15000);

inet_pton(AF_INET,argv[1],&address.sin_addr);

cs=socket(AF_INET,SOCK_STREAM,0);

connect(cs,(struct sockaddr *)&address,sizeof(address));

printf("\n enter the file name  :");
scanf("%s",fname);

send(cs,fname,255,0);

while(recv(cs,buf,bufsize,0))
printf("%s\n",buf);

printf("\n EOF");

return close(cs);
}
```

## Server

```
#include<stdio.h>
#include<unistd.h>
#include<sys/socket.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<stdlib.h>
#include<fcntl.h>

int main()
```

```

{
int cs,ns,fd,n;
int bufsize=1024;
char *buf=malloc(bufsize),fname[256];
struct sockaddr_in address;

address.sin_family=AF_INET;
address.sin_port=htons(15000);
address.sin_addr.s_addr=INADDR_ANY;
if(cs=socket(AF_INET,SOCK_STREAM,0))
    printf("\n socket created");

bind(cs,(struct sockaddr *)&address,sizeof(address));

listen(cs,3);

ns=accept(cs,(struct sockaddr *)NULL,NULL);

recv(ns,fname,255,0);

if((fd=open(fname,O_RDONLY))<0)
    printf("\n file not found");

while(n=read(fd,buf,bufsize))
    send(ns,buf,n,0);

close(ns);

return(close(cs));
}

```

## CLIENT/SERVER PROGRAM USING FIFO FILES

### Client

```

#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<fcntl.h>
#include<sys/stat.h>
int main()
{
    int fd,no;
    char buf[100];
    mkfifo("client",S_IFIFO|S_IRWXO|S_IRWXU|S_IRWXG);
    fd=open("server",O_WRONLY);
    if(fd==-1)
    {
        printf("Not able to open server fifo\n");
        return;
    }
}

```

```

    }
    printf("Enter filename\n");
    scanf("%s",buf);
    write(fd,buf,strlen(buf)+1);
    close(fd);
    fd=open("client",O_RDONLY);
    if(fd==-1)
    {
        printf("Can't open client fifo\n");
        return;
    }
    while((no=read(fd,buf,strlen(buf)))>0)
        printf("%s",buf);
}

```

## Server

```

#include<stdio.h>
#include<fcntl.h>
#include<unistd.h>
#include<string.h>
#include<sys/stat.h>
int main()
{
    int fd,fd1,no;
    char buf[100];
    mkfifo("server",S_IFIFO|S_IRWXU|S_IRWXO|S_IRWXG);
    fd=open("server",O_RDONLY);
    if(fd==-1)
    {
        printf("Cant open server fifo in server file\n");
        return ;
    }
    read(fd,buf,sizeof(buf));
    close(fd);
    fd=open("client",O_WRONLY);
    if(fd==-1)
    {
        printf("Can't open client fifo in server file\n");
        return;
    }
    fd1=open(buf,O_RDONLY);
    if(fd1==-1)
    {
        printf("Cant open filename specified in client program\n");
        return;
    }
    while((no=read(fd1,buf,sizeof(buf)))>0)
        write(fd,buf,sizeof(buf));
    close (fd);
    close(fd1);
}

```

## HAMMING ENCODING/DECODING

### Encoding

```
#include<stdio.h>

int main()
{
    int p[7],d[4],i;
    printf("\n enter the data bits : \n");
    for(i=0;i<4;i++)
        scanf("%d",&d[i]);

    p[0]=(d[0]+d[1]+d[3])%2;
    p[1]=(d[0]+d[2]+d[3])%2;
    p[2]=d[0];
    p[3]=(d[1]+d[2]+d[3])%2;
    p[4]=d[1];
    p[5]=d[2];
    p[6]=d[3];

    printf("\n encoded bits : ");
    for(i=0;i<7;i++)
        printf("%d",p[i]);
    return 0;
}
```

### Decoding

```
#include<stdio.h>
int main()
{
    int e[7],p[3],pos,i;
    printf("\n enter encoded bit sequence :\n");
    for(i=0;i<7;i++)
        scanf("%d",&e[i]);
    p[0]=(e[0]+e[2]+e[4]+e[6])%2;
    p[1]=(e[1]+e[2]+e[5]+e[6])%2;
    p[2]=(e[3]+e[4]+e[5]+e[6])%2;

    pos=p[2]*4 + p[1]*2 + p[0]*1;
    if(pos)
    {
        printf("\n bit stream recieved with error at position %d
",pos);
        if(e[pos-1] == 1)
            e[pos-1]=0;
        else
            e[pos-1]=1;
    }
}
```

```
        printf("\n corrected bit string : ");
        for(i=0;i<7;i++)
            printf("%d",e[i]);
    }
    else
        printf("\n bit string recieved without any error\n");
    return 0;
}
```