

1. Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped.

FILENAME: pB_1.tcl

```
set ns [new Simulator]
```

```
$ns color 2 red
```

```
$ns rtproto Static
```

```
set traceFile [open pB_1.tr w]
```

```
$ns trace-all $traceFile
```

```
set namFile [open pB_1.nam w]
```

```
$ns namtrace-all $namFile
```

```
proc finish {} {
```

```
    global ns namFile traceFile
```

```
    $ns flush-trace
```

```
    close $traceFile
```

```
    close $namFile
```

```
    exec nam pB_1.nam &
```

```
    exit 0
```

```
}
```

```
set n(1) [$ns node]
```

```
set n(2) [$ns node]
```

```
set n(3) [$ns node]
```

```
$ns duplex-link $n(1) $n(2) 0.5Mb 20ms DropTail
```

```
$ns duplex-link $n(2) $n(3) 0.5Mb 20ms DropTail
```

```
$ns queue-limit $n(1) $n(2) 10
```

```
$ns queue-limit $n(2) $n(3) 10
```

```
$n(1) shape hexagon
```

```
$n(1) color red
```

```
$n(3) shape square
```

```
$n(3) color blue
```

```
set udp0 [new Agent/UDP]
```

```
$ns attach-agent $n(1) $udp0
```

```
set cbr0 [new Application/Traffic/CBR]
```

```
$cbr0 set packetSize_ 512
```

```
$cbr0 set interval_ 0.005
```

```
$cbr0 attach-agent $udp0
```

```
set null0 [new Agent/Null]
```

```
$ns attach-agent $n(3) $null0
```

```
$ns connect $udp0 $null0
```

```
$udp0 set fid_ 2
```

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

```
$ns at 2.0 "$cbr0 stop"
```

```
$ns at 2.0 "finish"
```

```
$ns run
```

```
FILENAME: pB_1.awk
```

```
BEGIN{ c=0;}
```

```
{
```

```
if($1=="d")
```

```
{
```

```
c++;
```

```
printf("%s\t%s\n", $5, $11);
```

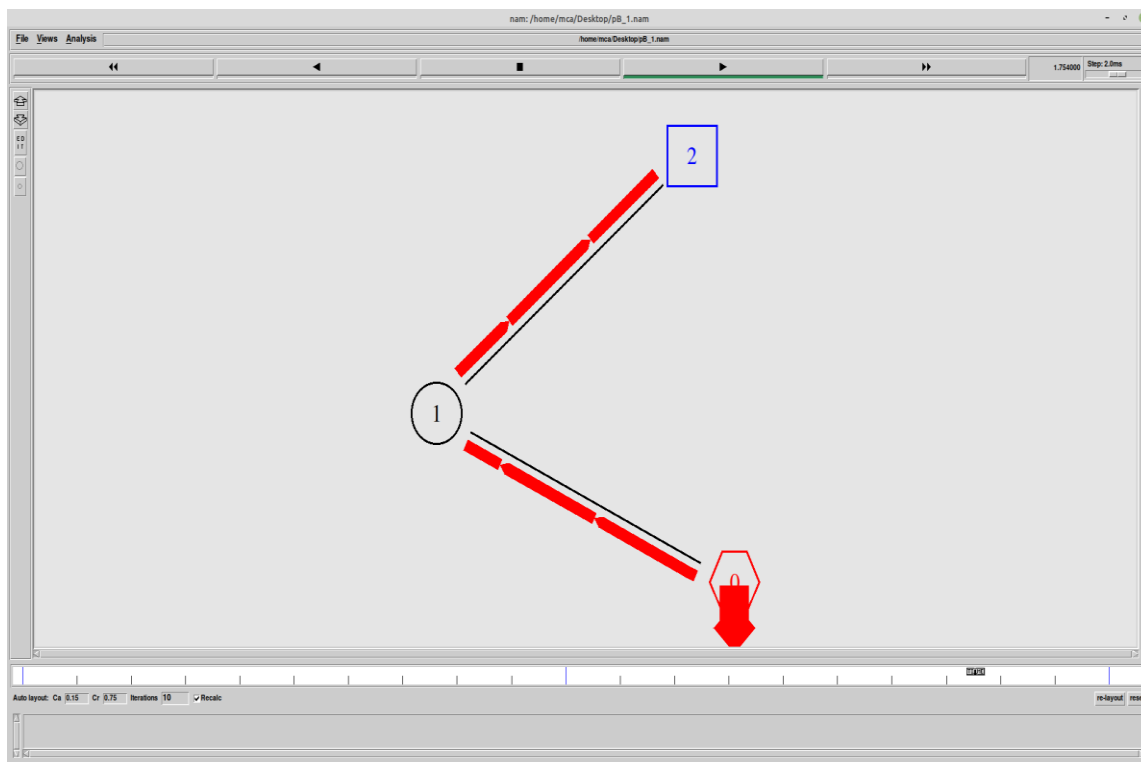
```
}
```

```
}
```

```
END{ printf("The number of packets dropped : %d\n",c);
```

```
}
```

output



2. Implement the data link layer framing methods such as character, character-stuffing and bit stuffing.

Program: - // BIT Stuffing program. `+#include<stdio.h>`

```

#include<stdio.h>
#include<string.h>
int main()
{
int a[20],b[30],i,j,k,count,n; printf("Enter Frame length :");
scanf("%d",&n);
    printf("Enter input frame (0's & 1's only): "); //enter the length of frame
for(i=0;i<n;i++)
scanf("%d",&a[i]);
i=0;
count=1; j=0;
while(i<n)
{    //count=0;
if(a[i]==1) //check if the data is '1'
{
b[j]=a[i];
for(k=i+1;a[k]==1 && k<n && count<5;k++)
{
j++;
b[j]=a[k];
count++; //increment and count the number of 1's if(count==5)
{
}
{
}
}
i=k;

j++; b[j]=0;

} i++; j++;

b[j]=a[i];

count=1;
}
printf("After stuffing the frame is:");
printf("01111110"); //append 01111110 at the beginning of the data for(i=0;i<j;i++)
printf("%d",b[i]);
printf("01111110"); //append 01111110 at the end of the data
return 0;
}}

```

Result: -

```

nandini@nandini-HP-G62-Notebook-PC:~/Desktop/Cpgm$ ./a.out
Enter Frame length:12
Enter input frame (0's & 1's only):1
1
0
0
1
0
1
1
1
1
1
1
0
After stuffing the frame is:0111111011001011111010111110

```

Program: //PROGRAM FOR CHARACTER STUFFING.

```
#include<stdio.h>
#include<string.h>
int main()
{
    int
    i=0,j=0,n,pos;
    char
    a[20],b[50],ch;
    printf("enter string\n"); //enter the character
    scanf("%s",&a);
    n=strlen(a); //length of string
    b[0]='d'; // append dle at the begin of
    string b[1]='l';
    b[2]='e';
    b[3]='s';
    b[4]='t';
    b[5]
    ]='x
    ';
    j=6;
    while(i<n)
    {
if( a[i]=='d' && a[i+1]=='l' && a[i+2]=='e') //if entered character is dle, then
output will be dledle//

        {
            b[j]='d';
            b[j+1]='l';
            b[j+2]='e';
            j=j+3;
        }

        b[j]
        =a[i
        ];
        i++;
    }
```

```

        j++;
    }
    b[j]='d';
    b[j+1]='l';
    b[j+2]='e';
    b[j+3]='e';
    b[j+4]='t';
    b[j+5]='x';
    b[j+6]='\0';
    printf("\nframe after
stuffing:\n");printf("%s",b);
    return 0;
}

```

Result:

```

nandini@nandini-HP-G62-Notebook-PC:~/Desktop/Cpgm$ ls
a.out bitstuff.c char1.c crc2.cpp dvr.c first.c leaky.c sliding.c stopwait.c
nandini@nandini-HP-G62-Notebook-PC:~/Desktop/Cpgm$ gcc char1.c
char1.c: In function 'main':
char1.c:11:9: warning: format '%s' expects argument of type 'char *', but argument 2 has type 'char (*)[20]' [-Wformat=]
scanf("%s",&a);
    ^
nandini@nandini-HP-G62-Notebook-PC:~/Desktop/Cpgm$ ./a.out
enter string
nadlegdledleetx

frame after stuffing:
dlestxnadlegdledledleetxdleetxnandini@nandini-HP-G62-Notebook-PC:~/Desktop/Cpgm$ ^C
nandini@nandini-HP-G62-Notebook-PC:~/Desktop/Cpgm$

```

- Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP
CRC 12

```

#include<stdio.h>

int genl[4],genl,frl,rem[4];

void main()
{
    int i,j,fr[8],dupfr[11],recfr[11],tlen,flag;
    frl=8; genl=4;
    printf("Enter frame:");
    for(i=0;i<frl;i++)
    {
        scanf("%d",&fr[i]);
        dupfr[i]=fr[i];
    }
}

```

```

printf("Enter generator:");
for(i=0;i<genl;i++)
scanf("%d",&gen[i]);
tlen=frl+genl-1;
for(i=frl;i<tlen;i++)
{
dupfr[i]=0;
}
remainder(dupfr);
for(i=0;i<frl;i++)
{
recfr[i]=fr[i];
}
for(i=frl,j=1;j<genl;i++,j++)
{
recfr[i]=rem[j];
}
remainder(recfr);
flag=0;
for(i=0;i<4;i++)
{
if(rem[i]!=0)
flag++;
}
if(flag==0)
{
printf("frame received correctly");
}
else
{

printf("the received frame is wrong");
}
}
remainder(int fr[])
{
int k,k1,i,j;

```

```

for(k=0;k<frl;k++)
{
if(fr[k]==1)
{
k1=k;
for(i=0,j=k;i<genl;i++,j++)
{
rem[i]=fr[j]^gen[i];
}
for(i=0;i<genl;i++)
{
fr[k1]=rem[i];
k1++;
}
}
}
}

```

4. Develop a simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism.

Experiment No: 4 SLIDING WINDOW PROTOCOL

AIM: Implementation of Sliding Window Protocol using C.

THEORY:

It allows multiple frames to be in transmit as compared to stop and wait protocol. In this the receiver has buffer of length W. Transmitter can send up to W frames without ACK. Each frame is numbered according to modular arithmetic. ACK includes number of next frame expected. Sequence number bounded by size of field (k).

C- LANGUAGE PROGRAM CODE

```

#include<stdio.h>

#include<conio.h>

void main()
{
char sender[50],receiver[50]; int i,winsize;

// clrscr();

printf("\n ENTER THE WINDOWS SIZE : ");

scanf("%d",&winsize);

```

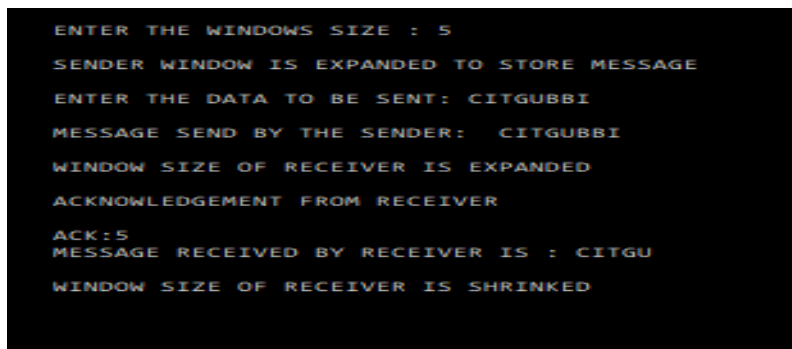
```

printf("\n SENDER WINDOW IS EXPANDED TO STORE MESSAGE \n"); printf("\n ENTER
THE DATA TO BE SENT: ");
fflush(stdin); gets(sender); for(i=0;i<winsize;i++) receiver[i]=sender[i]; receiver[i]=NULL;
printf("\n MESSAGE SEND BY THE SENDER:\n");
puts(sender);
printf("\n WINDOW SIZE OF RECEIVER IS EXPANDED\n"); printf("\n
ACKNOWLEDGEMENT FROM RECEIVER \n");
for(i=0;i<winsize;i++); printf("\n ACK:%d",i);
printf("\n MESSAGE RECEIVED BY RECEIVER IS : ");

puts(receiver);
printf("\n WINDOW SIZE OF RECEIVER IS SHRINKED \n");
getch();
}

```

OUTPUT



```

ENTER THE WINDOWS SIZE : 5
SENDER WINDOW IS EXPANDED TO STORE MESSAGE
ENTER THE DATA TO BE SENT: CITGUBBI
MESSAGE SEND BY THE SENDER: CITGUBBI
WINDOW SIZE OF RECEIVER IS EXPANDED
ACKNOWLEDGEMENT FROM RECEIVER
ACK:5
MESSAGE RECEIVED BY RECEIVER IS : CITGU
WINDOW SIZE OF RECEIVER IS SHRINKED

```

5. Implement Dijkstra's algorithm to compute the shortest path through a network

Experiment No: 5

SHORTEST PATH USING DIJKSTRA ALGORITHM

AIM: Write a C/C++ program to find the Shortest path algorithm using Dijkstra's algorithm.

THEORY:

Dijkstra's algorithm progressively identifies the closest no Dijkstra's algorithm progressively identifies the closest nodes from the source node in order of increasing path cost. The algorithm is iterative. The Dijkstra's algorithm calculates the shortest path between two points on a network using a graph made up of nodes and edges The algorithm divides the nodes into two sets : tentative and permanent . It chooses nodes, makes them tentative, examines them and if they pass the criteria makes them permanent.

ALGORITHM:

Step1: Declare array path [5] [5], min, a [5][5], index, t[5]; Step2: Declare and initialize st=1,ed=5

Step 3: Declare variables i, j, stp, p, edp Step 4: print —enter the cost —

Step 5: i=1

Step 6: Repeat step (7 to 11) until (i<=5)

Step 7: j=1

Step 8: repeat step (9 to 10) until (j<=5) Step 9: Read a[i] [j]

Step 10: increment j Step 11: increment i

Step 12: print —Enter the path|| Step 13: read p

Step 14: print —Enter possible paths|| Step 15: i=1

Step 16: repeat steps (17 to 21) until (i<=p)

Step 17: j=1

Step 18: repeat steps (19 to 20) until (i<=5) Step 19: read path[i][j]

Step 20: increment j Step 21: increment i Step 22: j=1

Step 23: repeat step(24 to 34) until(i<=p) Step 24: t[i]=0

Step 25: stp=st

Step 26: j=1

Step 27: repeat step(26 to 34) until(j<=5) Step 28: edp=path[i][j+1]

Step 29: t[i]= [ti]+a[stp][edp] Step 30: if (edp==ed) then Step 31: break;

Step 32: else

Step 33: stp=edp Step 34: end if Step 35: min=t[st]

Step 36: index=st

Step 37: repeat steps (38 to 41) until (i<=p) Step 38: min>t[i]

Step 39: min=t[i]

Step 40: index=i Step 41: end if

Step 42: print|| minimum cost|| min Step 43: print|| minimum cost pth||

Step 44: repeat steps (45 to 48) until (i<=5) Step 45: print path[index][i]

Step 46: if(path[idex][i]==ed) then Step 47: break

Step 48: end if End

C- LANGUAGE PROGRAM CODE

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

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

```
void main()
```

```
{
```

```
int path[5][5], i, j, min, a[5][5], p, st=1,ed=5,stp,edp,t[5],index;
```

```

// clrscr();
printf("enter the cost matrix\n"); for(i=1;i<=5;i++) for(j=1;j<=5;j++) scanf("%d",&a[i][j]);
printf("enter the paths\n"); scanf("%d",&p);
printf("enter possible paths\n"); for(i=1;i<=p;i++) for(j=1;j<=5;j++) scanf("%d",&path[i][j]);
for(i=1;i<=p;i++)
{
t[i]=0;
stp=st; for(j=1;j<=5;j++)
{
edp=path[i][j+1]; t[i]=t[i]+a[stp][edp]; if(edp==ed)
break; else stp=edp;
}
}
min=t[st];index=st; for(i=1;i<=p;i++)
{
if(min>t[i])
{
min=t[i]; index=i;
}
}
printf("minimum cost %d",min); printf("\n minimum cost path "); for(i=1;i<=5;i++)
{

printf("--> %d",path[index][i]); if(path[index][i]==ed)
break;
}
getch();
}

```

```

enter the cost matrix
0 1 4 2 0
1 0 3 7 0
4 3 0 5 0
2 7 5 0 6
0 0 0 6 0
enter the paths
4
enter possible paths
1 2 3 4 5
1 2 4 5 0
1 3 4 5 0
1 4 5 0 0
minimum cost 8
minimum cost path --> 1--> 4--> 5

```

6. Implement data encryption and data decryption

```

#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;

```

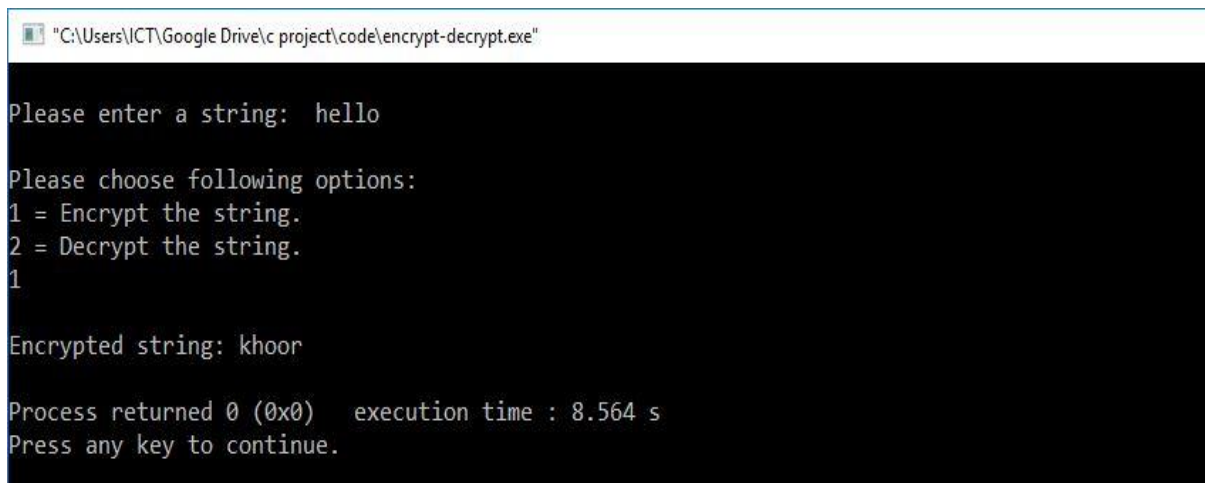
```

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;
}

```



```

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

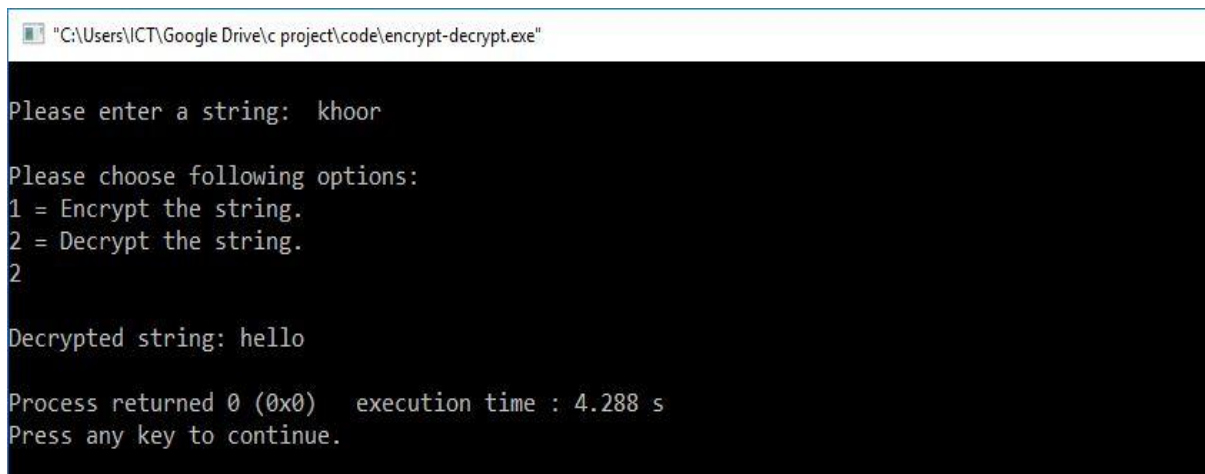
Please enter a string:  hello

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

Encrypted string: khour

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

```



```

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

Please enter a string:  khour

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

Decrypted string: hello

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

```

7. Simulate the network with five nodes n0, n1, n2, n3, n4, forming a star topology. The node n4 is at the centre. Node n0 is a TCP source, which transmits packets to node n3 (a TCP sink) through the node n4. Node n1 is another traffic source, and sends UDP packets to node n2 through n4. The duration of the simulation time is 10 seconds.

FILENAME: pB_2.tcl

set ns [new Simulator]

set nf [open pB_2.nam w]

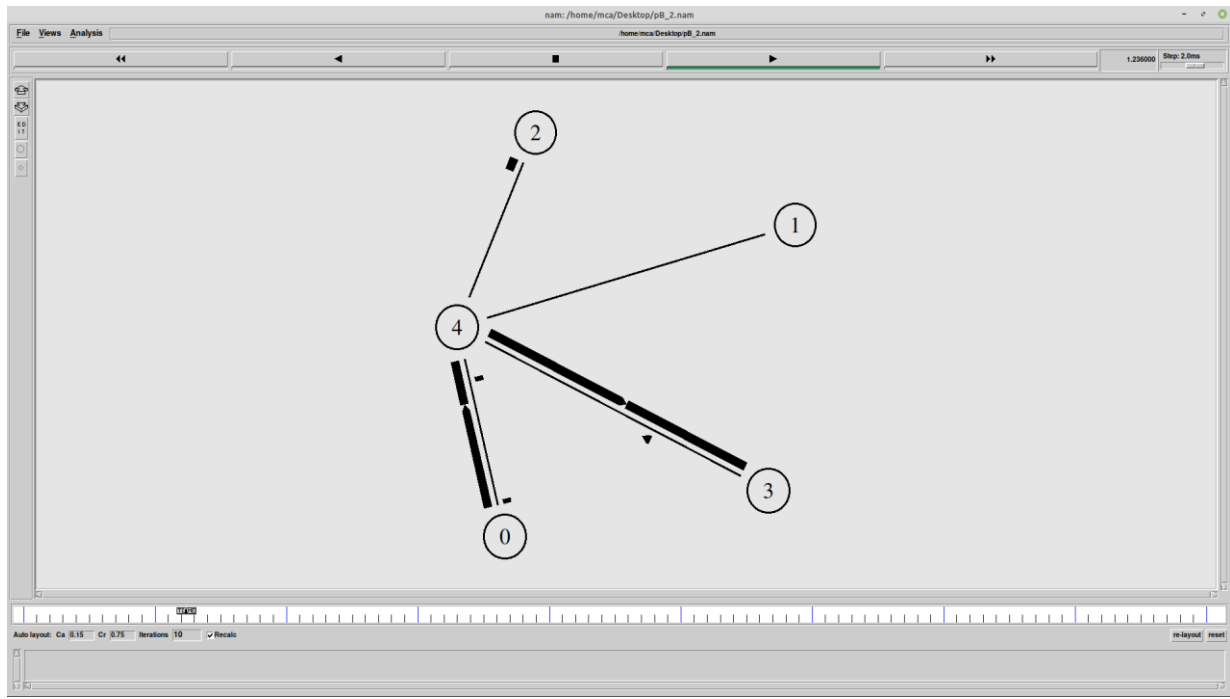
```
$ns namtrace-all $nf
set tf [open pB_2.tr w]
$ns trace-all $tf
proc finish { } {
    global ns tf nf
    $ns flush-trace
    close $nf
    close $tf
    exec nam pB_2.nam &
    exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
$ns duplex-link $n0 $n4 10Mb 1ms DropTail
$ns duplex-link $n1 $n4 10Mb 1ms DropTail
$ns duplex-link $n4 $n3 10Mb 1ms DropTail
$ns duplex-link $n4 $n2 10Mb 1ms DropTail
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
$ns connect $tcp0 $sink0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
set udp0 [new Agent/UDP]
$ns attach-agent $n1 $udp0
set null0 [new Agent/Null]
$ns attach-agent $n2 $null0
$ns connect $udp0 $null0
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp0
$ns at 0.0 "$cbr1 start"
$ns at 0.0 "$ftp0 start"
$ns at 9.0 "$cbr1 stop"
```

\$ns at 9.0 "\$ftp0 stop"

\$ns at 10.0 "finish"

\$ns run

output



8. Simulate to study transmission of packets over Ethernet LAN and determine the number of packets drop destination.

FILENAME: pB_3.tcl

```
set ns [new Simulator]
```

```
set nf [open pB_3.nam w]
```

```
$ns namtrace-all $nf
```

```
set nd [open pB_3.tr w]
```

```
$ns trace-all $nd
```

```
$ns color 1 Blue
```

```
$ns color 2 Red
```

```
proc finish { } {
```

```
global ns nf nd
```

```
$ns flush-trace
```

```
close $nf
```

```
close $nd
```

```
exec nam pB_3.nam &
```

```
exit 0
```

```
}
```

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
$n7 shape box
$n7 color Blue
$n8 shape hexagon
$n8 color Red
$ns duplex-link $n1 $n0 2Mb 10ms DropTail
$ns duplex-link $n2 $n0 2Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 20ms DropTail
$ns make-lan "$n3 $n4 $n5 $n6 $n7 $n8" 512Kb 40ms LL Queue/DropTail Mac/802_3
$ns duplex-link-op $n1 $n0 orient right-down
$ns duplex-link-op $n2 $n0 orient right-up
$ns duplex-link-op $n0 $n3 orient right
$ns queue-limit $n0 $n3 20
set tcp1 [new Agent/TCP/Vegas]
$ns attach-agent $n1 $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $n7 $sink1
$ns connect $tcp1 $sink1
$tcp1 set class_ 1
$tcp1 set packetsize_ 55
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
set tfile [open pB_3_tcp.tr w]
$tcp1 attach $tfile
$tcp1 trace pB_3_tcp_
set tcp2 [new Agent/TCP/Reno]
$ns attach-agent $n2 $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $n8 $sink2
```



```

set nt [open pB_5.tr w]
$ns trace-all $nt
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns queue-limit $n0 $n2 50
$ns queue-limit $n1 $n2 50
$ns queue-limit $n2 $n3 50
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
$ns connect $tcp0 $sink0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
Agent/TCP set packetSize_ 1000
set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
$ns connect $tcp1 $sink1
set telnet0 [new Application/Telnet]
$telnet0 set interval_ 0.005
$telnet0 attach-agent $tcp1
proc finish { } {
    global ns nf nt
    $ns flush-trace
    close $nf
    close $nt
    exec nam pB_5.nam &
    exec awk -f pB_5.awk pB_5.tr &
    exit 0
}
$ns at 0.5 "$telnet0 start"
$ns at 0.75 "$ftp0 start"
$ns at 4.5 "$telnet0 stop"
$ns at 4.75 "$ftp0 stop"
$ns at 5.0 "finish"
$ns run
FILENAME: pB_5.awk
BEGIN{ count=0; Rcount=0;}
{
    if($1=="r" && $5=="tcp")
    {

```

```
count=count+1;
if($6 >= 1000)
{
Rcount=Rcount+1;
}
}
}
END{
printf("Total Packets in Transmission: %d\n",count);
printf("Recieved: %d\n",Rcount);
printf("Loss: %d\n",count-Rcount);
}
```