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.

```
set ns [ new Simulator ]          /* Letter S is capital */
set nf [ open p1.nam w ]          /* open a nam trace file in write mode */
$ns namtrace-all $nf              /* nf – nam file */

set tf [ open p1.tr w ]           /* tf- trace file */
$ns trace-all $tf

proc finish {} {                  /* provide space b/w proc and finish and all are in small case */
global ns nf tf
$ns flush-trace
close  $nf
close $tf
exec nam p1.nam &
exit 0
}

set n0 [$ns node]     /* creates 4 nodes */
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n0 $n2 200Mb 10ms DropTail        /*Letter M is capital Mb*/
$ns duplex-link $n1 $n2 100Mb 5ms DropTail         /*D and T are capital*/ $ns
$ns duplex-link $n2 $n3 1Mb 1000ms DropTail

$ns queue-limit $n0 $n2 10
$ns queue-limit $n1 $n2 10

set udp0 [new Agent/UDP]                      /* Letters A,U,D and P are capital */
$ns attach-agent $n0 $udp0

set cbr0 [new  Application/Traffic/CBR]       /* A,T,C,B and R are capital*/
$cbr0 set packetSize_ 500                     /*S is capital, space after underscore*/
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1

set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1

set udp2 [new Agent/UDP]
$ns attach-agent $n2 $udp2

set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
```

```
set null0 [new Agent/Null]                    /* A and N are capital */
$ns attach-agent $n3 $null0

$ns connect $udp0 $null0
$ns connect $udp1 $null0

$ns at 0.1 "$cbr0 start"
$ns at 0.2 "$cbr1 start"
$ns at 1.0 "finish"

$ns run
```

<u>AWK file</u> (Open a new editor using "vi command" and write awk file and save with ".awk" extension)

```
/*immediately after BEGIN should open braces '{'
BEGIN {
c=0;
}
{
   If ($1= ="d")
  {
     c++;
     printf("%s\t%s\n",$5,$11);
   }
 }
/*immediately after END should open braces '{'
END{
     printf("The number of packets dropped =%d\n",c);
}
```

<u>Steps for execution</u>

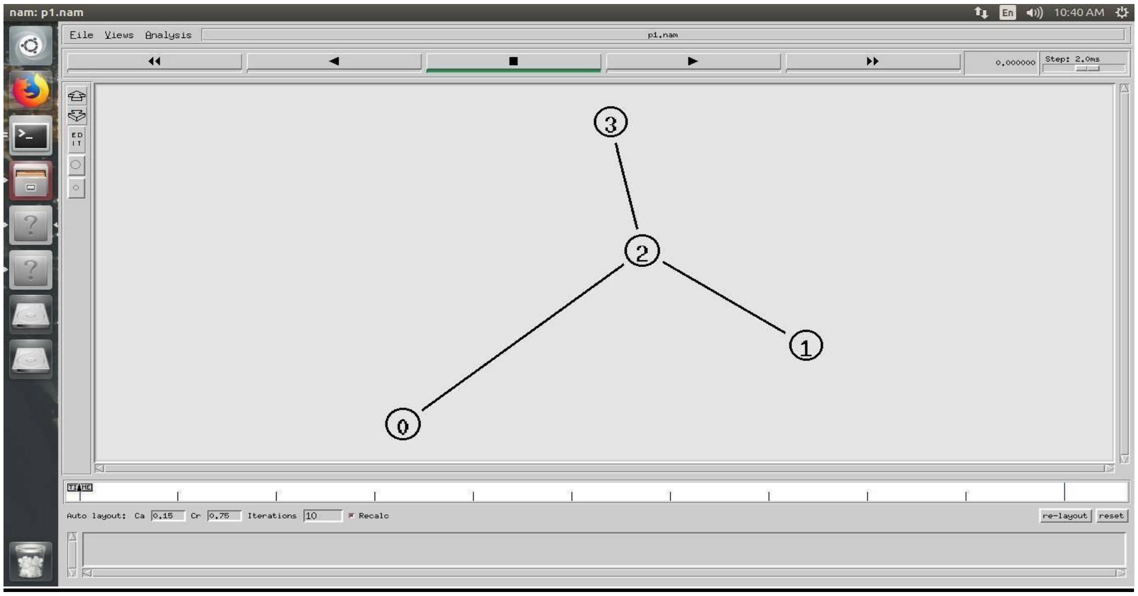
1) Open vi editor and type program. Program name should have the extension " **.tcl** "
   [root@localhost ~]# vi p1.tcl
2) Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.
3) Open vi editor and type **awk** program. Program name should have the extension "**.awk** "
   [root@localhost ~]# vi p1.awk
4) Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.

5) Run the simulation program
   [root@localhost~]# ns p1.tcl
   i) Here **"ns"** indicates network simulator. We get the topology shown in the snapshot.
   ii) Now press the play button in the simulation window and the simulation will begins.
6) After simulation is completed run **awk file** to see the output ,
   [root@localhost~]# awk –f p1.awk p1.tr
7) To see the trace file contents open the file as ,
   [root@localhost~]# vi p1.tr

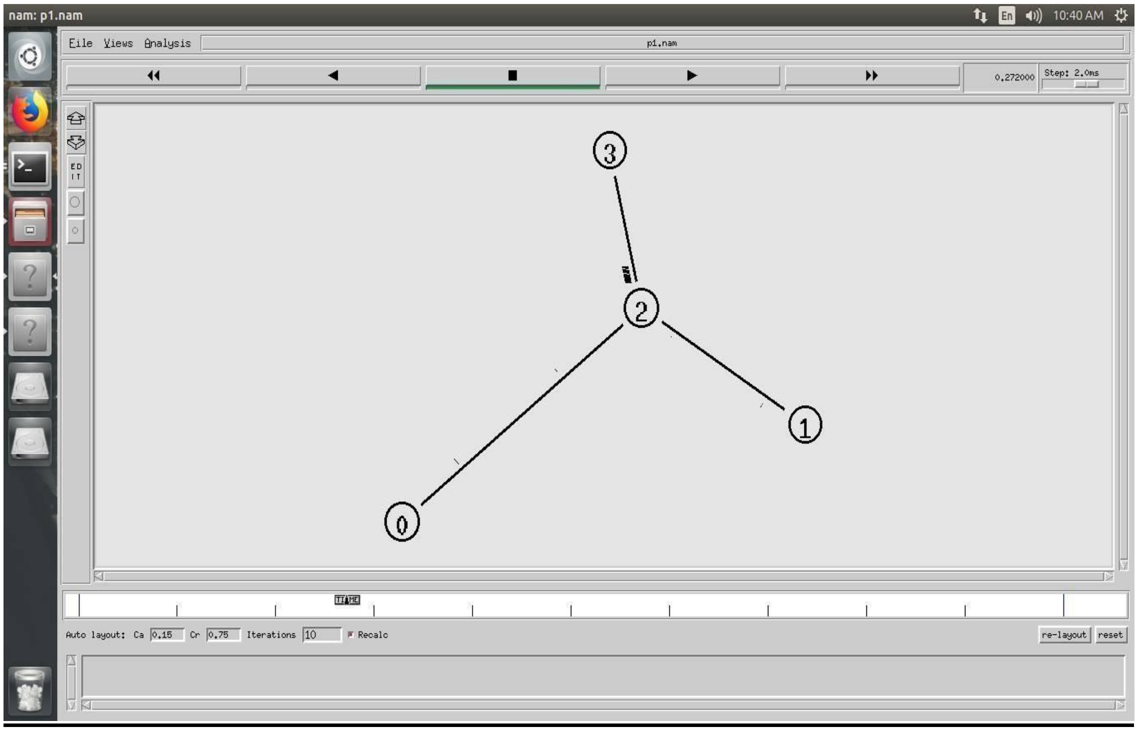
Trace file contains 12 columns:-

Event type, Event time, From Node, Source Node, Packet Type, Packet Size, Flags (indicated by --------), Flow ID, Source address, Destination address, Sequence ID, Packet ID

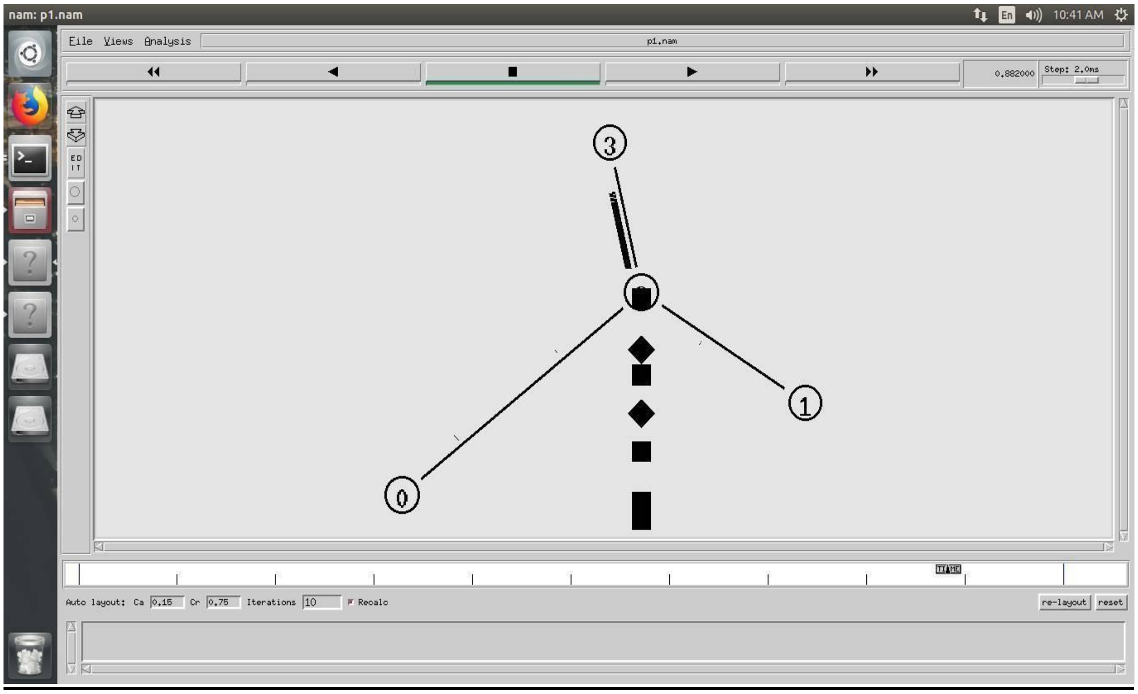
Topology



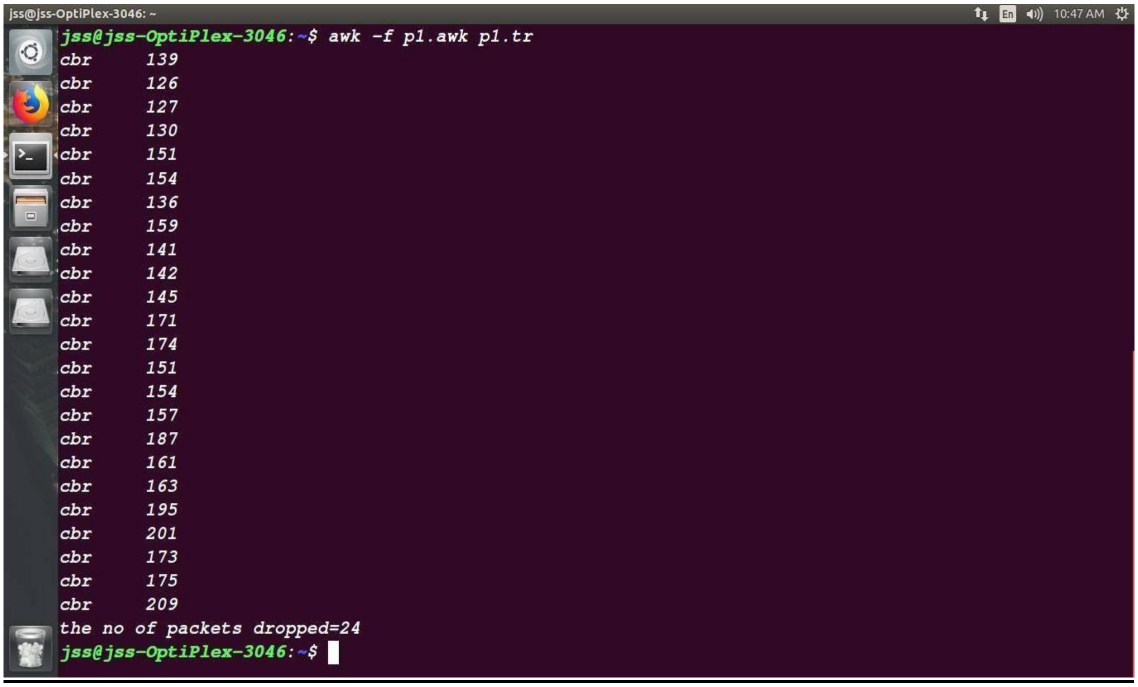
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc



Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc



Output



```
jss@jss-OptiPlex-3046:~$ awk -f p1.awk p1.tr
cbr     139
cbr     126
cbr     127
cbr     130
cbr     151
cbr     154
cbr     136
cbr     159
cbr     141
cbr     142
cbr     145
cbr     171
cbr     174
cbr     151
cbr     154
cbr     157
cbr     187
cbr     161
cbr     163
cbr     195
cbr     201
cbr     173
cbr     175
cbr     209
the no of packets dropped=24
jss@jss-OptiPlex-3046:~$
```

Note:

1. Set the queue size fixed from n0 to n2 as 10, n1-n2 to 10 and from n2-n3 as 5.
   Syntax: To set the queue size
   $ns set queue-limit <from> <to> <size>
   Eg: $ns set queue-limit $n0 $n2 10
2. Go on varying the bandwidth from 10, 20 30 . . and find the number of packets dropped at the node 2

2. Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the throughput with respect to transmission of packets.

```
set ns [new Simulator]
set tf [open p4.tr w]
$ns trace-all $tf
set topo [new Topography]
$topo load_flatgrid 1000 1000
set nf [open p4.nam w]
$ns namtrace-all-wireless $nf 1000 1000

$ns node-config -adhocRouting DSDV\
                -llType LL\
                -macType Mac/802_11\
                -ifqType Queue/DropTail\
                -ifqLen 50\
                -phyType Phy/WirelessPhy\
                -channelType Channel/WirelessChannel\
                -propType Propagation/TwoRayGround\
                -antType Antenna/OmniAntenna\
                -topoInstance $topo\
                -agentTrace ON\
                -routerTrace ON

create-god 3

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]

$n0 label "tcp0"
$n1 label "sink1/tcp1"
$n2 label "sink2"

$n0 set X_ 50
$n0 set Y_ 50
$n0 set Z_ 0
$n1 set X_ 100
$n1 set Y_ 100
$n1 set Z_ 0
$n2 set X_ 600
$n2 set Y_ 600
$n2 set Z_ 0

$ns at 0.1 "$n0 setdest 50 50 500"
$ns at 0.1 "$n1 setdest 100 100 500"
$ns at 0.1 "$n2 setdest 600 600 500"

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0

set ftp0 [new Application/FTP]
```

```
$ftp0 attach-agent $tcp0

set sink1 [new Agent/TCPSink]
$ns attach-agent $n1 $sink1
$ns connect $tcp0 $sink1

set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

set sink2 [new Agent/TCPSink]
$ns attach-agent $n2 $sink2
$ns connect $tcp1 $sink2

$ns at 5 "$ftp0 start"
$ns at 5 "$ftp1 start"

$ns at 100 "$n1 setdest 550 550 15"
$ns at 190 "$n1 setdest 70 70 15"

proc finish { } {
            global ns nf tf
            $ns flush-trace
            close  $nf
            close $tf
            exec nam p4.nam &
        exit 0
}


$ns at 250 "finish"
$ns run
```

<u>AWK file</u> (Open a new editor using “vi command” and write awk file and save with “.awk” extension)

```
BEGIN{
count1=0
count2=0
pack1=0
pack2=0
time1=0
time2=0
}
{

if($1=="r"&&$3=="_1_" &&$4=="AGT")
{
count1++
pack1=pack1+$8
time1=$2
}
if($1=="r"&&$3=="_2_" &&$4=="AGT")
{
count2++
pack2=pack2+$8
time2=$2
}
}

END{
printf("The Throughput from n0 to n1: %f Mbps\n",((count1*pack1*8)/(time1*1000000)));
printf("The Throughput from n1 to n2: %f Mbps",((count2*pack2*8)/(time2*1000000)));
}
```

Steps for execution

1) Open vi editor and type program. Program name should have the extension " **.tcl** "

   [root@localhost ~]# vi p4.tcl

2) Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.
3) Open vi editor and type **awk** program. Program name should have the extension "**.awk** "

   [root@localhost ~]# vi p4.awk

4) Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.
5) Run the simulation program

   [root@localhost~]# ns p4.tcl

   i) Here **"ns"** indicates network simulator. We get the topology shown in the snapshot.

   ii) Now press the play button in the simulation window and the simulation will begins.

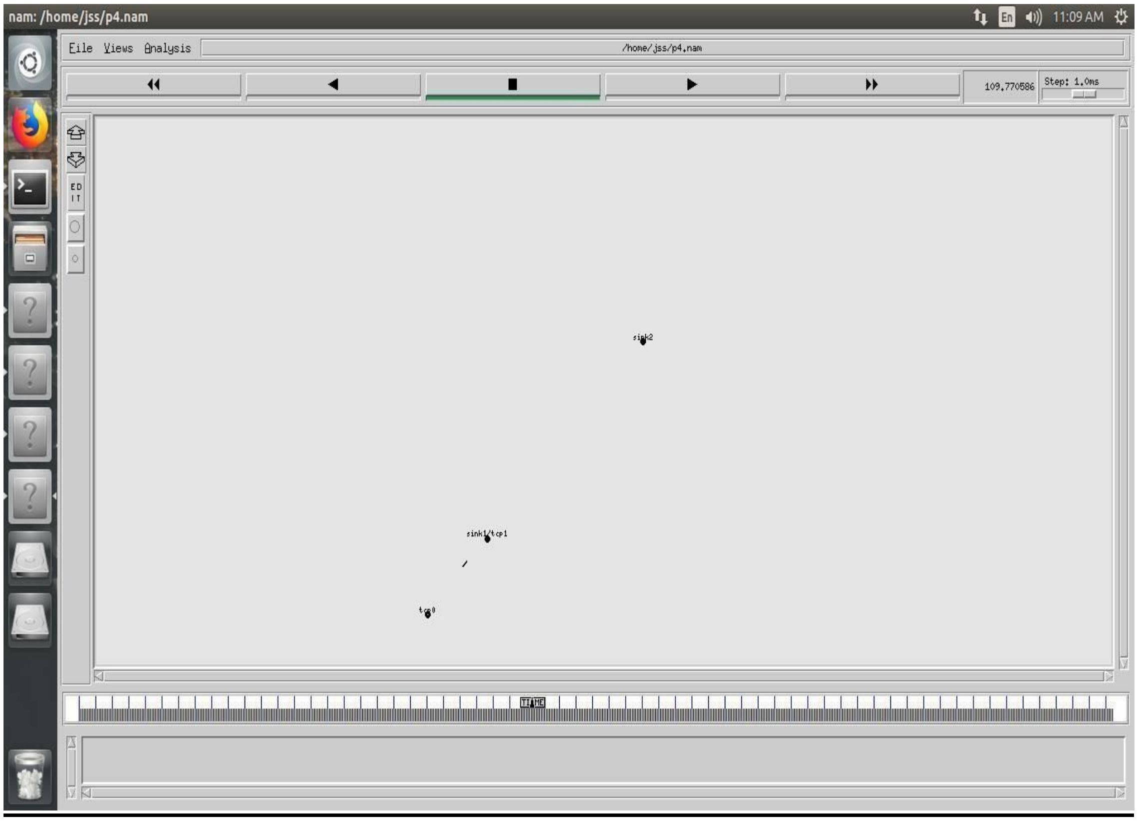
6) After simulation is completed run **awk file** to see the output ,

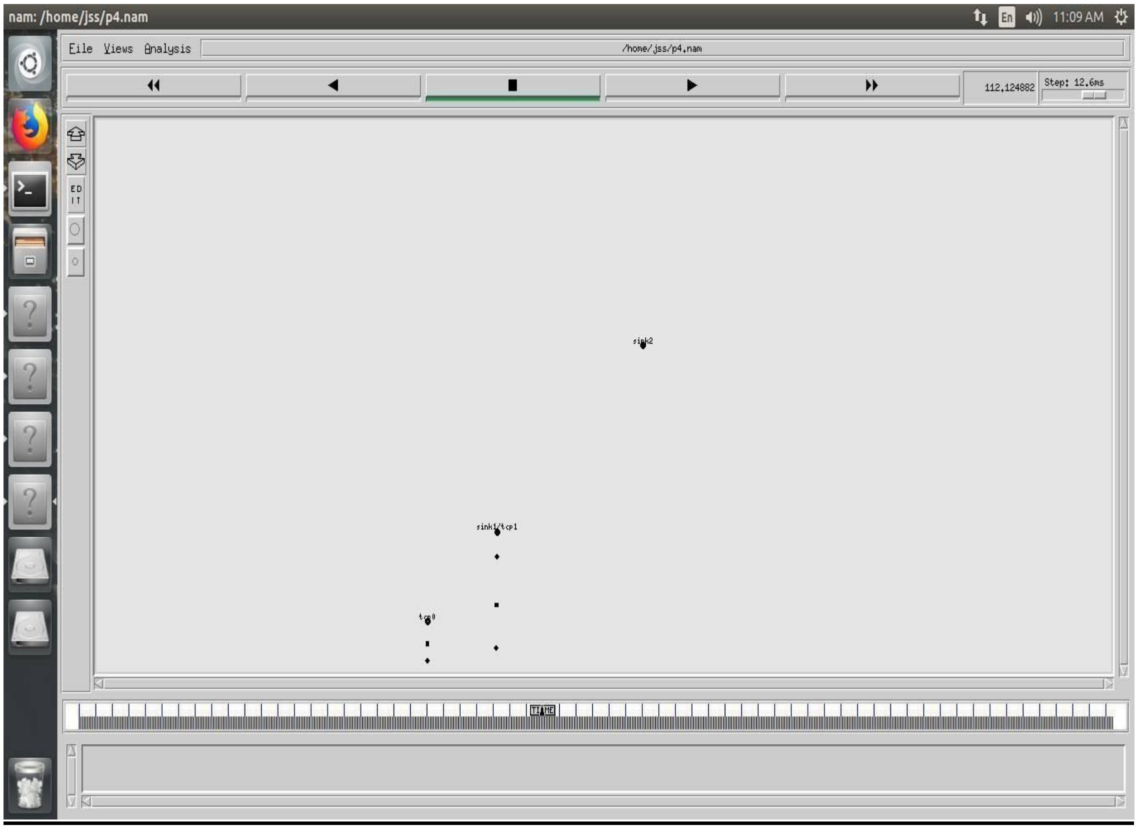
   [root@localhost~]# awk –f p4.awk p4.tr

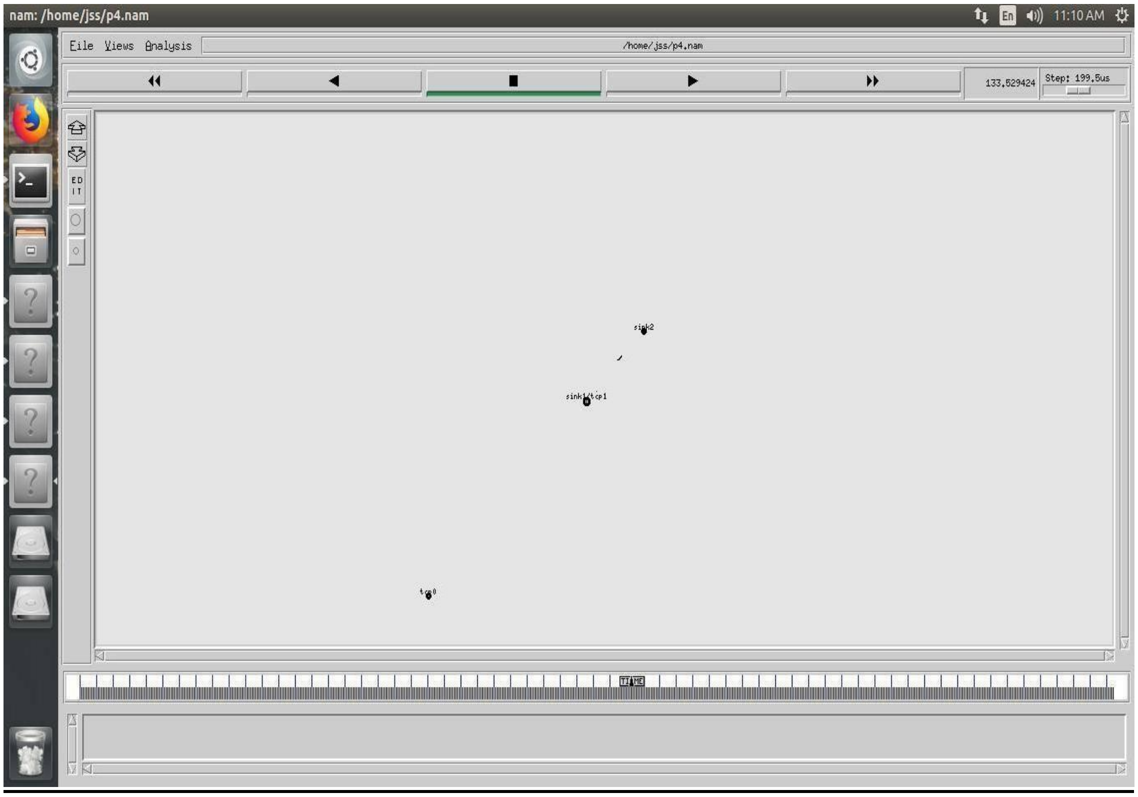
7) To see the trace file contents open the file as ,

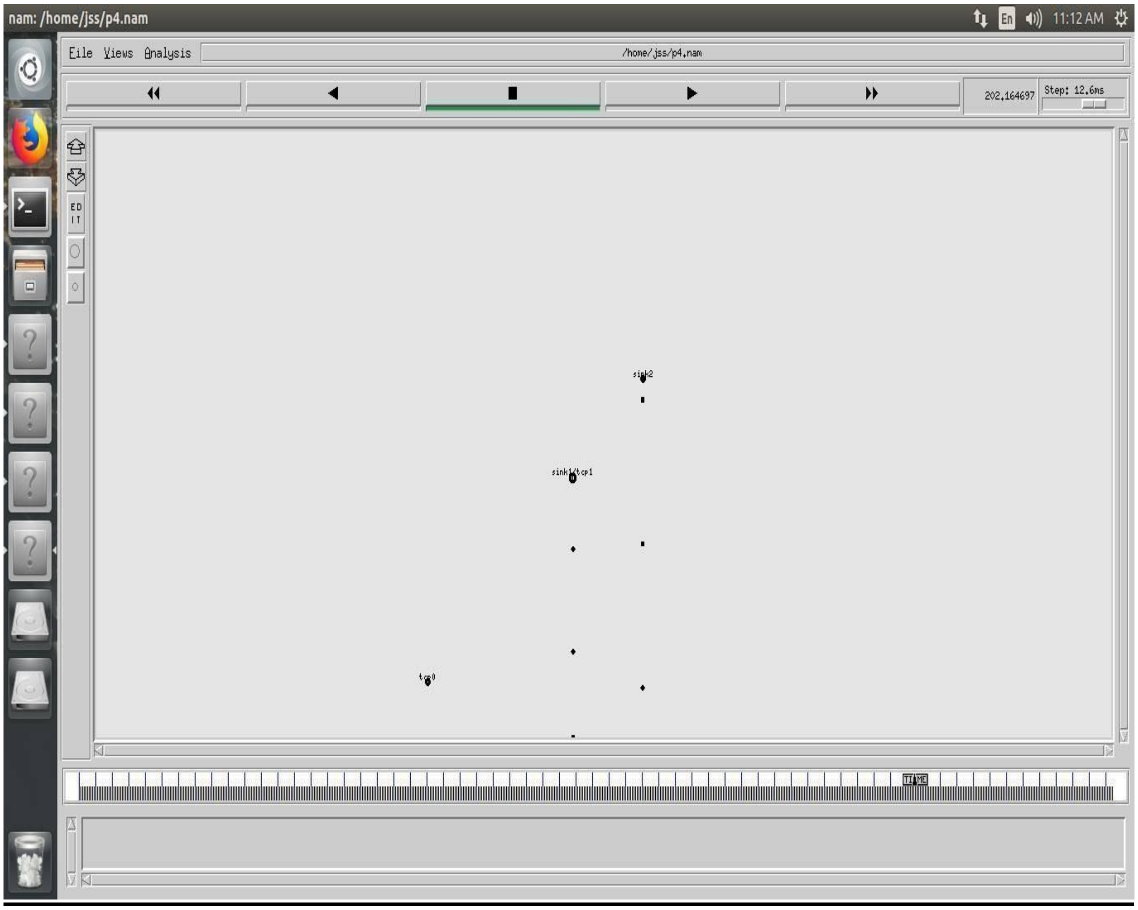
   [root@localhost~]# vi p4.tr

Topology

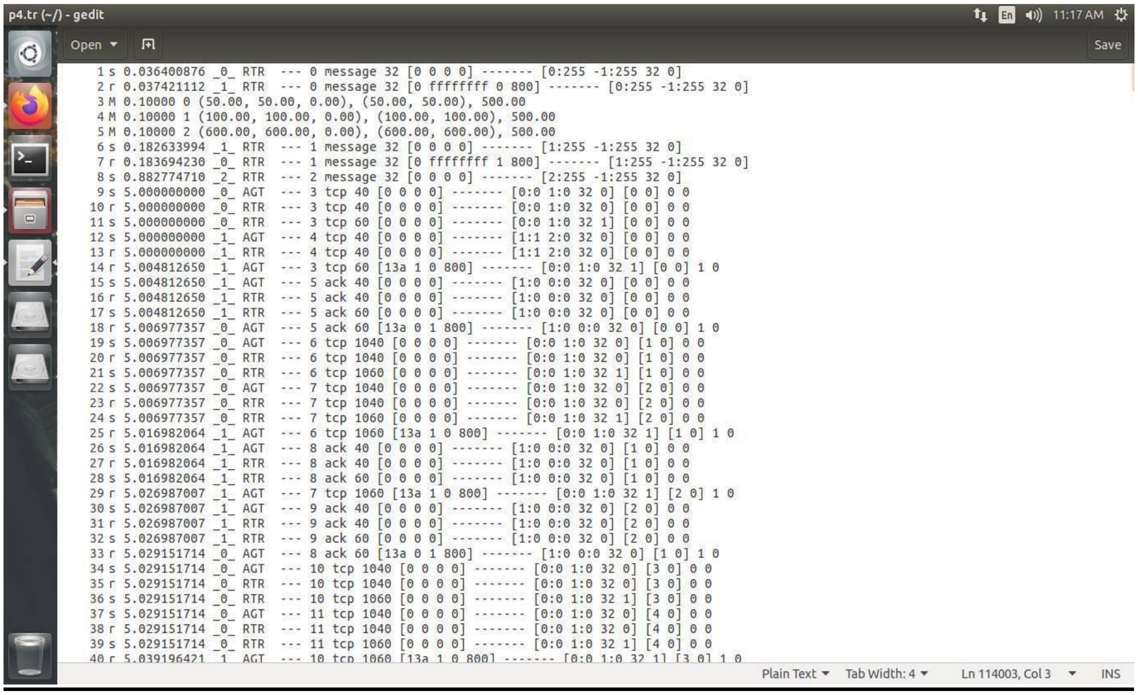

nam: /home/jss/p4.nam
11:09 AM
File Views Analysis
/home/jss/p4.nam
109.770586
Step: 1.0ms
EDIT
sink2
sink1/tcp1
tcp0
TIME


nam: /home/jss/p4.nam
11:09 AM
File Views Analysis
/home/jss/p4.nam
112.124882
Step: 12.6ms
EDIT
sink2
sink1/tcp1
tcp0
TIME


nam: /home/jss/p4.nam
11:10 AM
File Views Analysis
/home/jss/p4.nam
133.529424
Step: 199.5us
EDIT
sink2
sink1/tcp1
tcp0
TIME


nam: /home/jss/p4.nam
11:12 AM
File Views Analysis
/home/jss/p4.nam
202.164697
Step: 12.6ms
EDIT
sink2
sink1/tcp1
tcp0
TIME

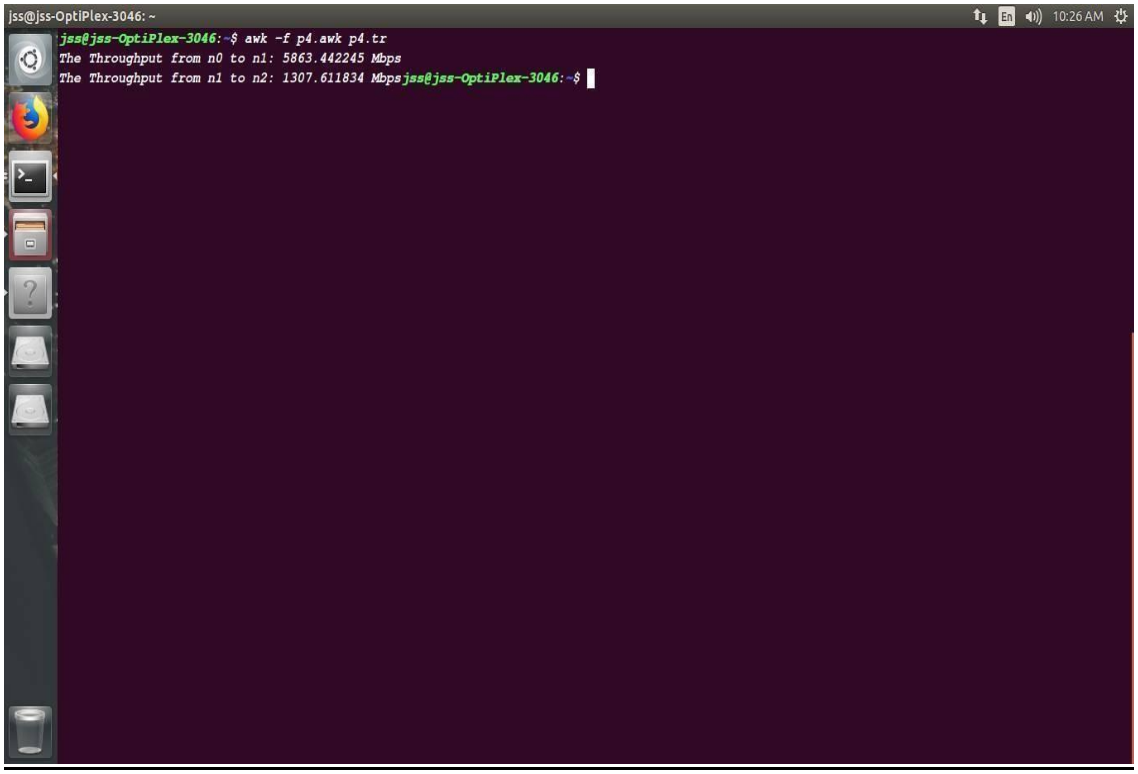
Trace file



```
s 0.036400876 _0_ RTR  --- 0 message 32 [0 0 0 0] ------- [0:255 -1:255 32 0]
r 0.037421112 _1_ RTR  --- 0 message 32 [0 ffffffff 0 800] ------- [0:255 -1:255 32 0]
M 0.10000 0 (50.00, 50.00, 0.00), (50.00, 50.00), 500.00
M 0.10000 1 (100.00, 100.00, 0.00), (100.00, 100.00), 500.00
M 0.10000 2 (600.00, 600.00, 0.00), (600.00, 600.00), 500.00
s 0.182633994 _1_ RTR  --- 1 message 32 [0 0 0 0] ------- [1:255 -1:255 32 0]
r 0.183694230 _0_ RTR  --- 1 message 32 [0 ffffffff 1 800] ------- [1:255 -1:255 32 0]
s 0.882774710 _2_ RTR  --- 2 message 32 [0 0 0 0] ------- [2:255 -1:255 32 0]
s 5.000000000 _0_ AGT  --- 3 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
r 5.000000000 _0_ RTR  --- 3 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
s 5.000000000 _0_ RTR  --- 3 tcp 60 [0 0 0 0] ------- [0:0 1:0 32 1] [0 0] 0 0
s 5.000000000 _1_ AGT  --- 4 tcp 40 [0 0 0 0] ------- [1:1 2:0 32 0] [0 0] 0 0
r 5.000000000 _1_ RTR  --- 4 tcp 40 [0 0 0 0] ------- [1:1 2:0 32 0] [0 0] 0 0
r 5.004812650 _1_ AGT  --- 3 tcp 60 [13a 1 0 800] ------- [0:0 1:0 32 1] [0 0] 1 0
s 5.004812650 _1_ AGT  --- 5 ack 40 [0 0 0 0] ------- [1:0 0:0 32 0] [0 0] 0 0
r 5.004812650 _1_ RTR  --- 5 ack 40 [0 0 0 0] ------- [1:0 0:0 32 0] [0 0] 0 0
s 5.004812650 _1_ RTR  --- 5 ack 60 [0 0 0 0] ------- [1:0 0:0 32 0] [0 0] 0 0
r 5.006977357 _0_ AGT  --- 5 ack 60 [13a 0 1 800] ------- [1:0 0:0 32 0] [0 0] 1 0
s 5.006977357 _0_ AGT  --- 6 tcp 1040 [0 0 0 0] ------- [0:0 1:0 32 0] [1 0] 0 0
r 5.006977357 _0_ RTR  --- 6 tcp 1040 [0 0 0 0] ------- [0:0 1:0 32 0] [1 0] 0 0
s 5.006977357 _0_ RTR  --- 6 tcp 1060 [0 0 0 0] ------- [0:0 1:0 32 1] [1 0] 0 0
s 5.006977357 _0_ AGT  --- 7 tcp 1040 [0 0 0 0] ------- [0:0 1:0 32 0] [2 0] 0 0
r 5.006977357 _0_ RTR  --- 7 tcp 1040 [0 0 0 0] ------- [0:0 1:0 32 0] [2 0] 0 0
s 5.006977357 _0_ RTR  --- 7 tcp 1060 [0 0 0 0] ------- [0:0 1:0 32 1] [2 0] 0 0
r 5.016982064 _1_ AGT  --- 6 tcp 1060 [13a 1 0 800] ------- [0:0 1:0 32 1] [1 0] 1 0
s 5.016982064 _1_ AGT  --- 8 ack 40 [0 0 0 0] ------- [1:0 0:0 32 0] [1 0] 0 0
r 5.016982064 _1_ RTR  --- 8 ack 40 [0 0 0 0] ------- [1:0 0:0 32 0] [1 0] 0 0
s 5.016982064 _1_ RTR  --- 8 ack 60 [0 0 0 0] ------- [1:0 0:0 32 0] [1 0] 0 0
r 5.026987007 _1_ AGT  --- 7 tcp 1060 [13a 1 0 800] ------- [0:0 1:0 32 1] [2 0] 1 0
s 5.026987007 _1_ AGT  --- 9 ack 40 [0 0 0 0] ------- [1:0 0:0 32 0] [2 0] 0 0
r 5.026987007 _1_ RTR  --- 9 ack 40 [0 0 0 0] ------- [1:0 0:0 32 0] [2 0] 0 0
s 5.026987007 _1_ RTR  --- 9 ack 60 [0 0 0 0] ------- [1:0 0:0 32 0] [2 0] 0 0
r 5.029151714 _0_ AGT  --- 8 ack 60 [13a 0 1 800] ------- [1:0 0:0 32 0] [1 0] 1 0
s 5.029151714 _0_ AGT  --- 10 tcp 1040 [0 0 0 0] ------- [0:0 1:0 32 0] [3 0] 0 0
r 5.029151714 _0_ RTR  --- 10 tcp 1040 [0 0 0 0] ------- [0:0 1:0 32 0] [3 0] 0 0
s 5.029151714 _0_ RTR  --- 10 tcp 1060 [0 0 0 0] ------- [0:0 1:0 32 1] [3 0] 0 0
s 5.029151714 _0_ AGT  --- 11 tcp 1040 [0 0 0 0] ------- [0:0 1:0 32 0] [4 0] 0 0
r 5.029151714 _0_ RTR  --- 11 tcp 1040 [0 0 0 0] ------- [0:0 1:0 32 0] [4 0] 0 0
s 5.029151714 _0_ RTR  --- 11 tcp 1060 [0 0 0 0] ------- [0:0 1:0 32 1] [4 0] 0 0
r 5.039196421 _1_ AGT  --- 10 tcp 1060 [13a 1 0 800] ------- [0:0 1:0 32 1] [3 0] 1 0
```

Here **"M"** indicates mobile nodes, **"AGT"** indicates Agent Trace, **"RTR"** indicates Router Trace

Output



```
jss@jss-OptiPlex-3046:~$ awk -f p4.awk p4.tr
The Throughput from n0 to n1: 5863.442245 Mbps
The Throughput from n1 to n2: 1307.611834 Mbpsjss@jss-OptiPlex-3046:~$
```

3. Write a program for error detecting code using CRC-CCITT (16- bits).

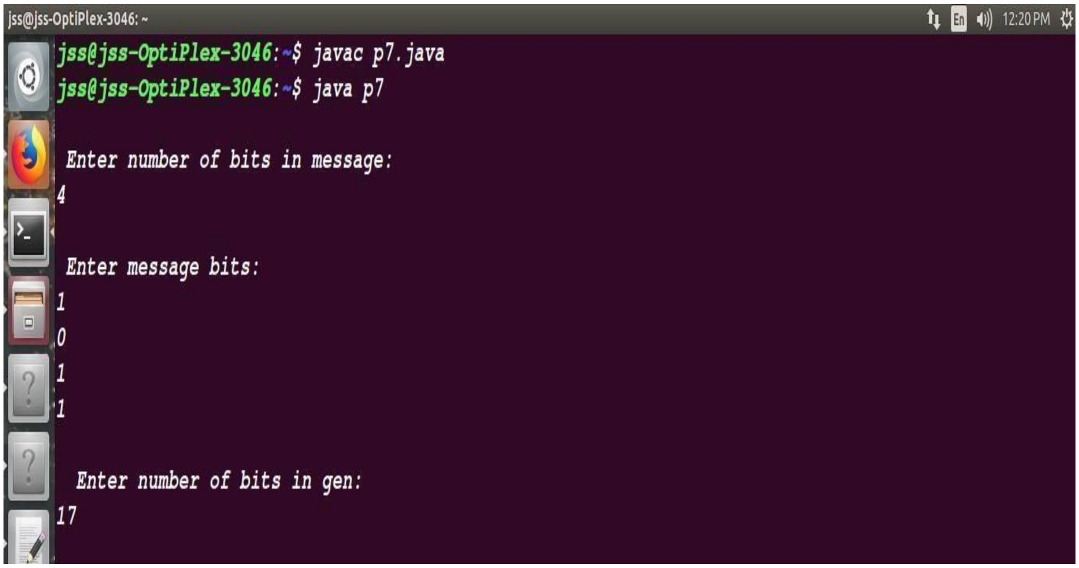
Source Code:

```
import java.io.*;
import java.*;

public class p7
{

public static void main(String a[]) throws IOException
{
InputStreamReader isr=new InputStreamReader(System.in);
BufferedReader br=new BufferedReader(isr);
int[] message;
int[] gen;
int[] app_message;
int[] rem;
int[] trans_message;
int message_bits,gen_bits, total_bits;

System.out.println("\n Enter number of bits in massege:");
message_bits=Integer.parseInt(br.readLine());
message=new int[message_bits];
System.out.println("\n Enter message bits:");
for(int i=0;i<message_bits;i++)
message[i]=Integer.parseInt(br.readLine());
System.out.println("\n  Enter number of bits in gen:");
gen_bits=Integer.parseInt(br.readLine());
gen = new int [gen_bits];
System.out.println("\n Enter gen bits :");
for(int i=0; i< gen_bits; i++)
{
gen[i]=Integer.parseInt(br.readLine());
}
total_bits=message_bits+gen_bits-1;
app_message=new int[total_bits];
rem=new int[total_bits];
trans_message=new int[total_bits];
for(int i=0;i< message.length;i++)
{
app_message[i]=message[i];
}
System.out.print("\n Message bits are:");
for(int i=0; i < message_bits; i++)
{
System.out.print(message[i]);
}
System.out.print("\n Generators bits are:");
for(int i=0;i < gen_bits;i++)
{
System.out.print(gen[i]);
}
System.out.print("\n Appended message is:");
for(int i=0; i<app_message.length; i++)
{
System.out.print(app_message[i]);
}
for(int j=0;j<app_message.length;j++)
{
rem[j]=app_message[j];
}
rem=computecrc(app_message,gen,rem);

for(int i=0;i<app_message.length;i++)
{
trans_message[i]=(app_message[i]^rem[i]);
}

System.out.println("*\n Transmitted message from the transmitteris :");
for(int i=0;i<trans_message.length;i++)
{
System.out.print(trans_message[i]);
}
System.out.println("\n Enter received message of +total_bits+ at receiver end:");
for(int i=0; i<trans_message.length;i++)
{
trans_message[i]=Integer.parseInt(br.readLine());
}
System.out.println("\n Received message is:");
for(int i=0; i< trans_message.length;i++)
{
System.out.print(trans_message[i]);
}
for(int j=0; j<trans_message.length; j++)
{
rem[j]=trans_message[j];
}
rem=computecrc(trans_message,gen,rem);
for(int i=0; i<rem.length; i++)
{
if(rem[i]!=0)
{
System.out.println("\n There is Error in the received message!!!");
break;
}
if(i==rem.length-1)
{
System.out.println("\n There is No Error in the received message!!!");
}
}
}

static int[] computecrc(int app_message[],int gen[],int rem[])
```

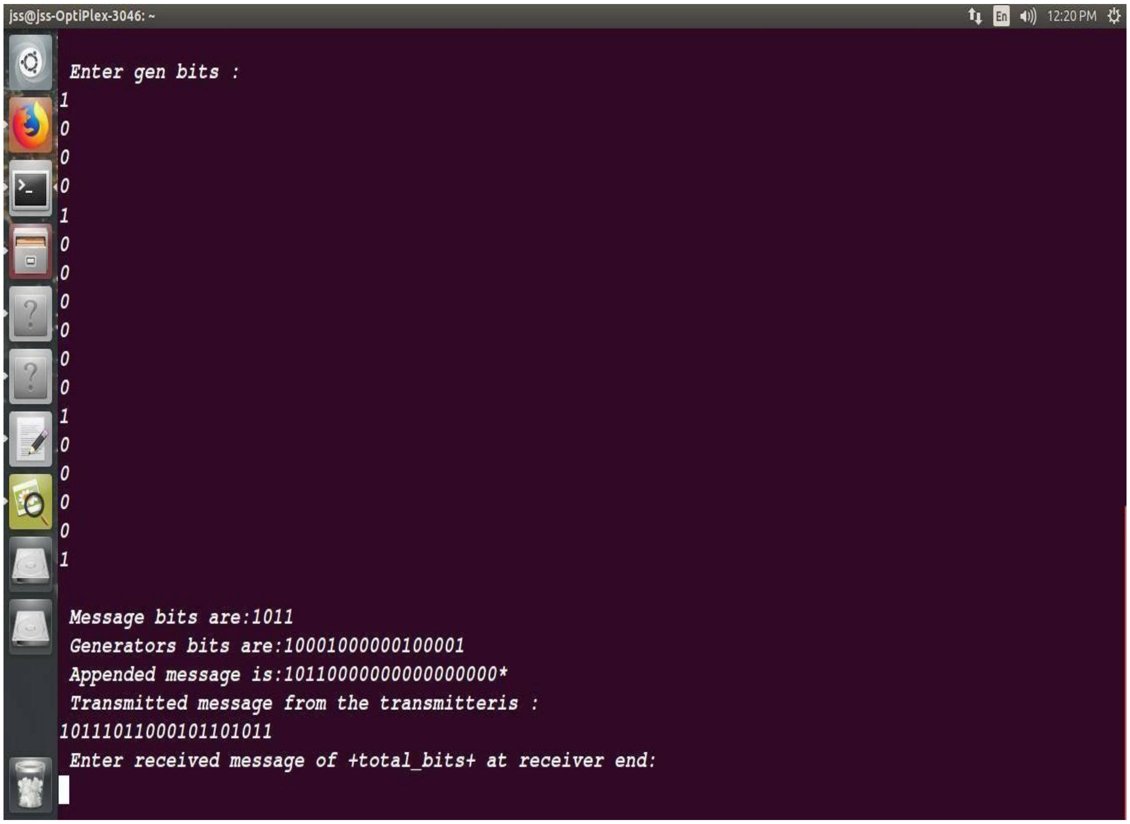
```
{
int current=0;
while(true)
{
for(int i=0;i<gen.length;i++)
{
rem[current+i]=(rem[current+i]^gen[i]);
}
while(rem[current]==0 && current!=rem.length-1)
{
current++;
}
if((rem.length-current)<gen.length)
{
break;
}
}
return rem;
}
}
```

Output:

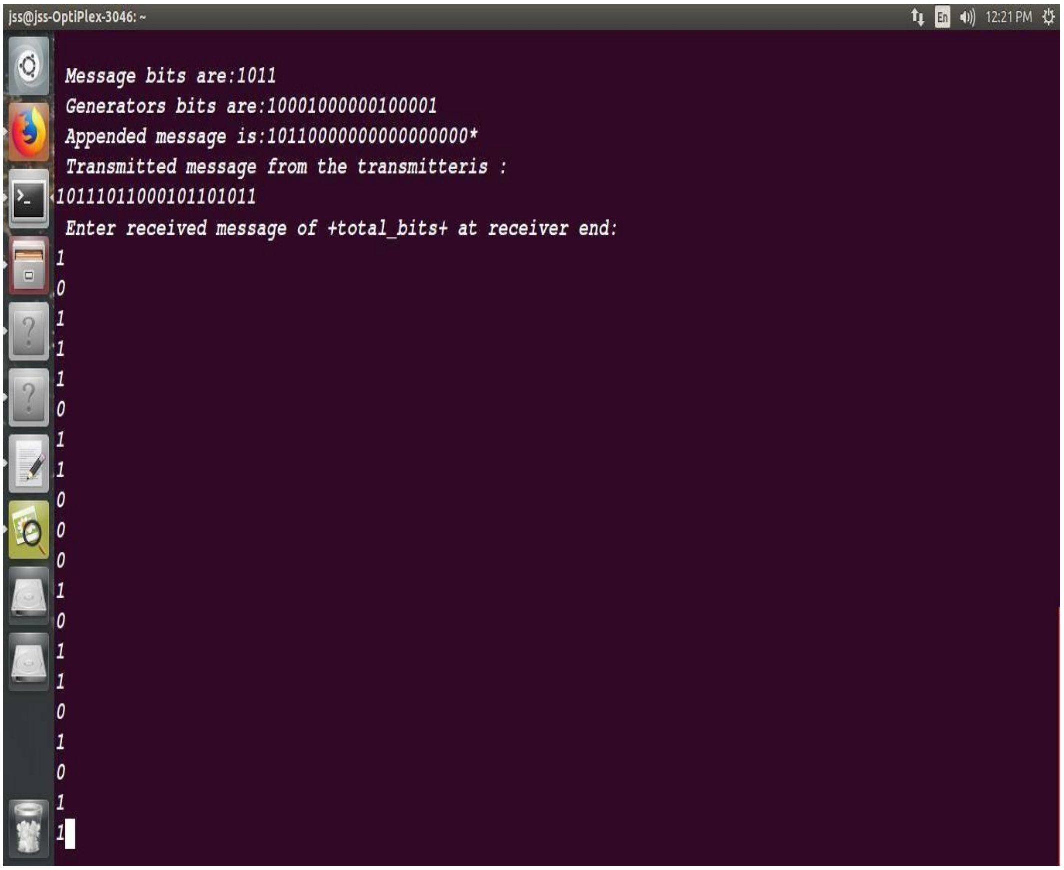
```
jss@jss-Optiplex-3046:~ vi p7.java

jss@jss-Optiplex-3046:~ javac p7.java

jss@jss-Optiplex-3046:~ java p7
```



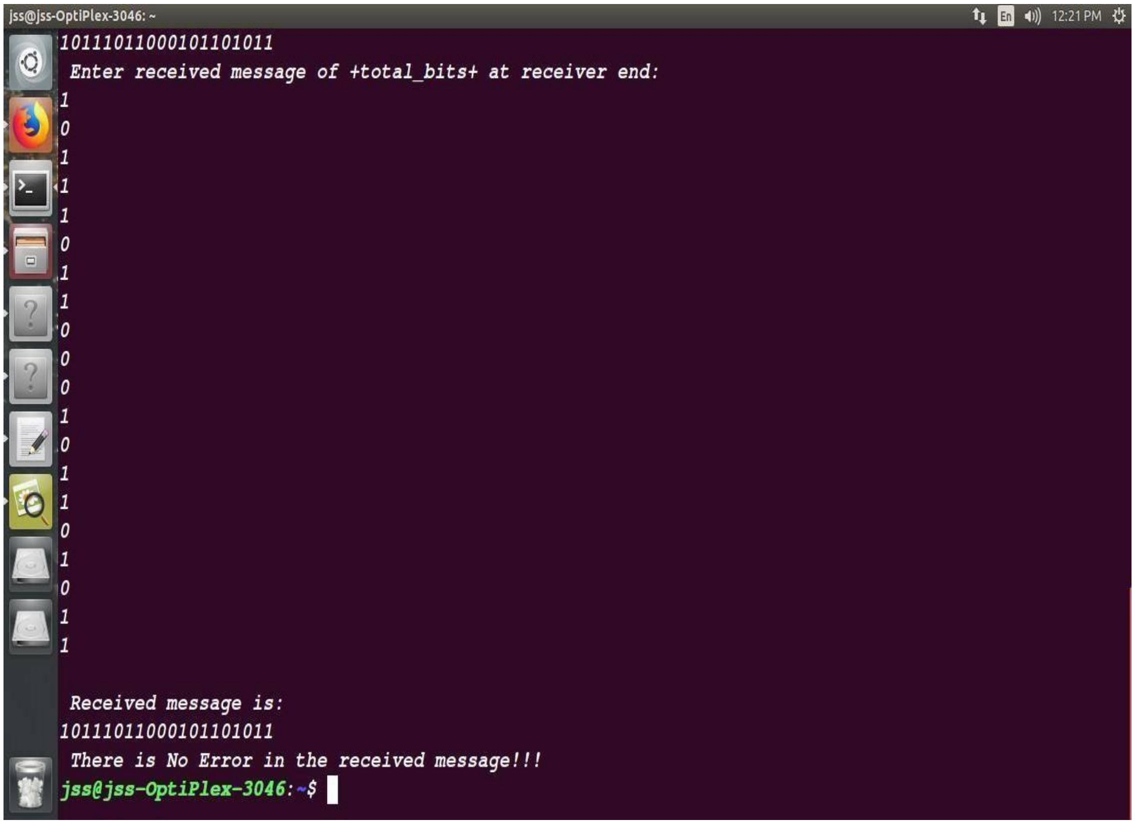
```
jss@jss-OptiPlex-3046:~$ javac p7.java
jss@jss-OptiPlex-3046:~$ java p7

 Enter number of bits in message:
4

 Enter message bits:
1
0
1
1

  Enter number of bits in gen:
17
```



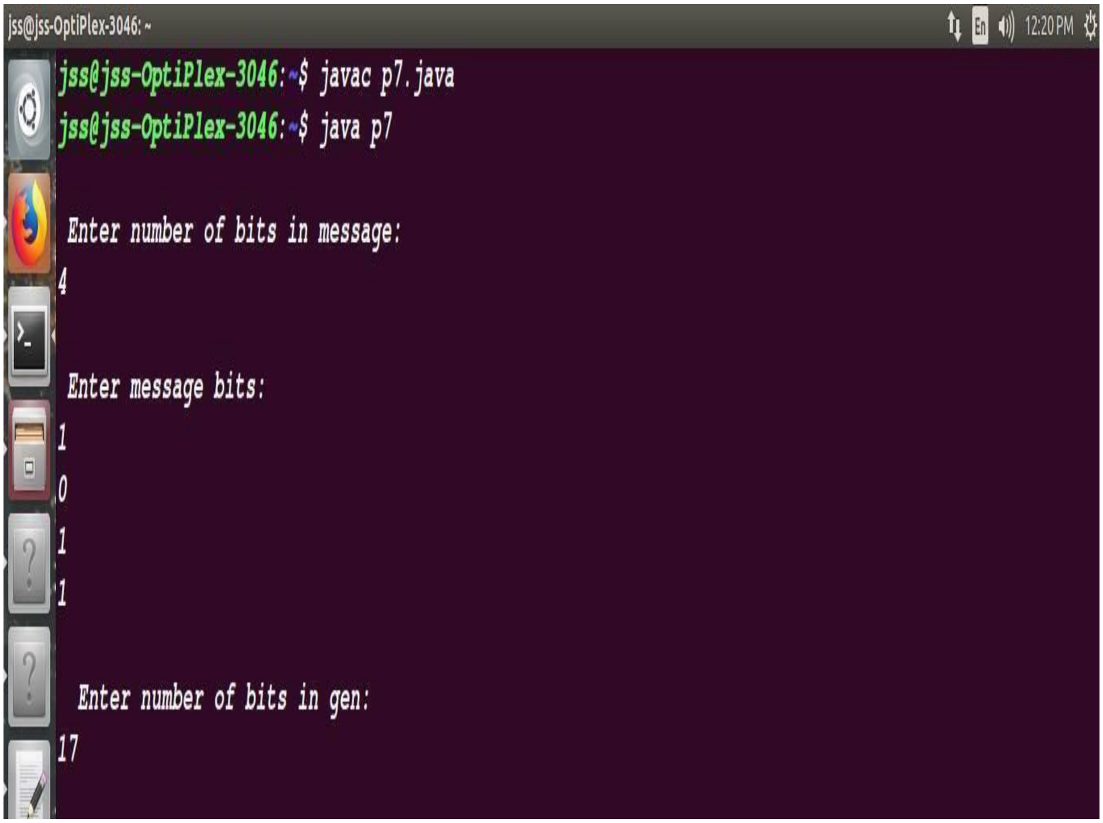
```
 Enter gen bits :
1
0
0
0
1
0
0
0
0
0
0
1
0
0
0
0
1


 Message bits are:1011
 Generators bits are:10001000000100001
 Appended message is:10110000000000000000*
 Transmitted message from the transmitteris :
10111011000101101011
 Enter received message of +total_bits+ at receiver end:
```



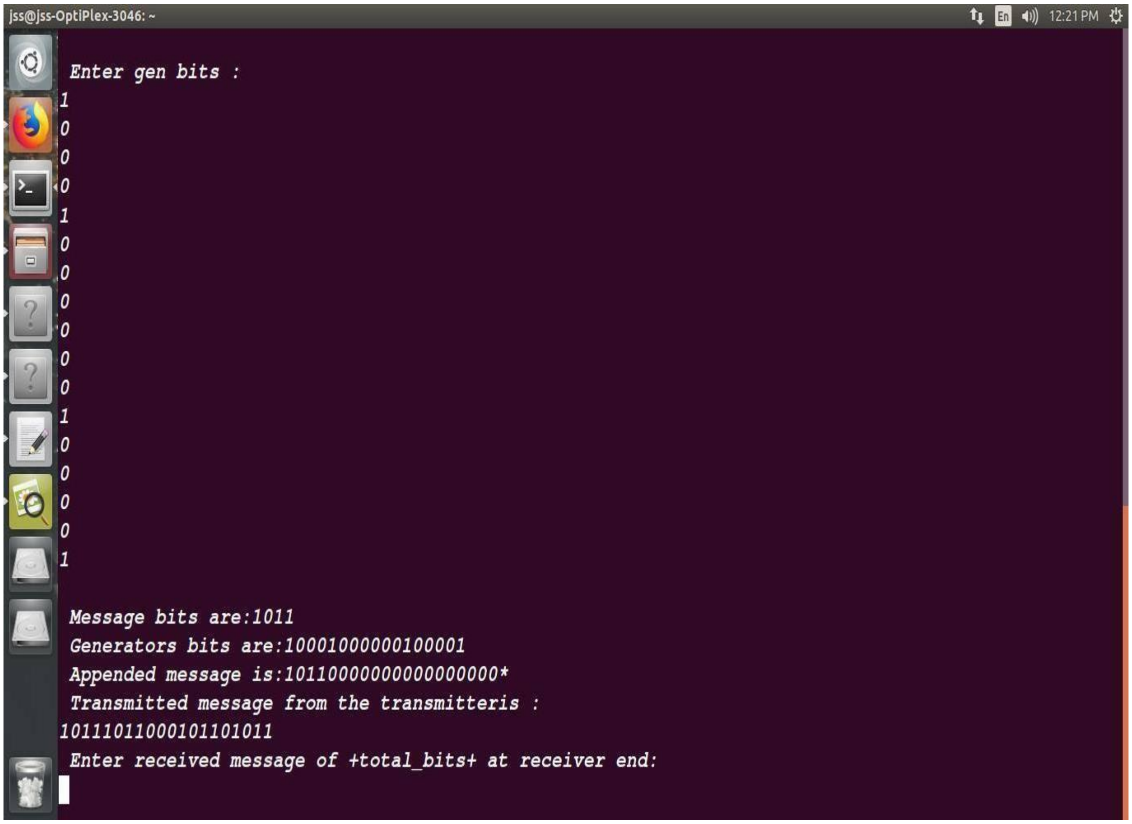
```
 Message bits are:1011
 Generators bits are:10001000000100001
 Appended message is:10110000000000000000*
 Transmitted message from the transmitteris :
10111011000101101011
 Enter received message of +total_bits+ at receiver end:
1
0
1
1
1
0
1
1
0
0
0
1
0
1
1
0
1
0
1
1
```



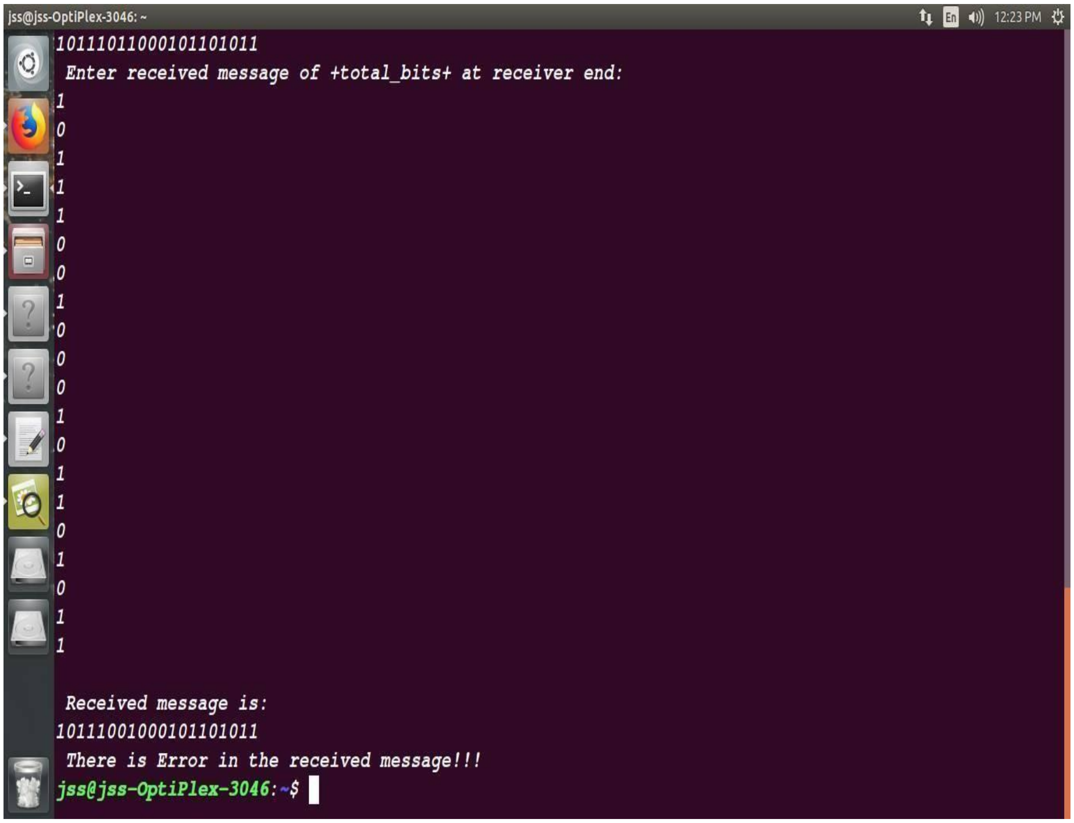
```
10111011000101101011
 Enter received message of +total_bits+ at receiver end:
1
0
1
1
1
0
1
1
0
0
0
1
0
1
1
0
1
0
1
1


 Received message is:
10111011000101101011
 There is No Error in the received message!!!
jss@jss-OptiPlex-3046:~$ 
```



```
jss@jss-OptiPlex-3046:~$ javac p7.java
jss@jss-OptiPlex-3046:~$ java p7

 Enter number of bits in message:
4

 Enter message bits:
1
0
1
1

  Enter number of bits in gen:
17
```



```
 Enter gen bits :
1
0
0
0
1
0
0
0
0
0
0
1
0
0
0
0
1


 Message bits are:1011
 Generators bits are:10001000000100001
 Appended message is:10110000000000000000*
 Transmitted message from the transmitteris :
10111011000101101011
 Enter received message of +total_bits+ at receiver end:
```



```
10111011000101101011
 Enter received message of +total_bits+ at receiver end:
1
0
1
1
1
0
0
1
0
0
0
1
0
1
1
0
1
0
1
1


 Received message is:
10111001000101101011
 There is Error in the received message!!!
jss@jss-OptiPlex-3046:~$
```

4. Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion in the network.

```
set ns [ new Simulator ]
set nf [ open p2.nam w ]
$ns namtrace-all $nf
set tf [ open p2.tr w ]
$ns trace-all $tf
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$ns duplex-link $n0 $n4 1005Mb 1ms DropTail
$ns duplex-link $n1 $n4 50Mb 1ms DropTail
$ns duplex-link $n2 $n4 2000Mb 1ms DropTail
$ns duplex-link $n3 $n4 200Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 1ms DropTail

set p1 [new Agent/Ping]
$ns attach-agent $n0 $p1
$p1 set packetSize_ 50000
$p1 set interval_ 0.0001

set p2 [new Agent/Ping]
$ns attach-agent $n1 $p2

set p3 [new Agent/Ping]
$ns attach-agent $n2 $p3
$p3 set packetSize_ 30000
$p3 set interval_ 0.00001


set p4 [new Agent/Ping]
$ns attach-agent $n3 $p4

set p5 [new Agent/Ping]
$ns attach-agent $n5 $p5

$ns queue-limit $n0 $n4 5
$ns queue-limit $n2 $n4 3
$ns queue-limit $n4 $n5 2
Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id]received answer from $from with round trip time $rtt msec"
}
$ns connect $p1 $p5
$ns connect $p3 $p4
```

```
proc finish { } {
global ns nf tf
$ns flush-trace
close  $nf
close $tf
exec nam p2.nam &
exit 0
}
$ns at 0.1 "$p1 send"
$ns at 0.2 "$p1 send"
$ns at 0.3 "$p1 send"
$ns at 0.4 "$p1 send"
$ns at 0.5 "$p1 send"
$ns at 0.6 "$p1 send"
$ns at 0.7 "$p1 send"
$ns at 0.8 "$p1 send"
$ns at 0.9 "$p1 send"
$ns at 1.0 "$p1 send"
$ns at 1.1 "$p1 send"
$ns at 1.2 "$p1 send"
$ns at 1.3 "$p1 send"
$ns at 1.4 "$p1 send"
$ns at 1.5 "$p1 send"
$ns at 1.6 "$p1 send"
$ns at 1.7 "$p1 send"
$ns at 1.8 "$p1 send"
$ns at 1.9 "$p1 send"
$ns at 2.0 "$p1 send"
$ns at 2.1 "$p1 send"
$ns at 2.2 "$p1 send"
$ns at 2.3 "$p1 send"
$ns at 2.4 "$p1 send"
$ns at 2.5 "$p1 send"
$ns at 2.6 "$p1 send"
$ns at 2.7 "$p1 send"
$ns at 2.8 "$p1 send"
$ns at 2.9 "$p1 send"


$ns at 0.1 "$p3 send"
$ns at 0.2 "$p3 send"
$ns at 0.3 "$p3 send"
$ns at 0.4 "$p3 send"
$ns at 0.5 "$p3 send"
$ns at 0.6 "$p3 send"
$ns at 0.7 "$p3 send"
$ns at 0.8 "$p3 send"
$ns at 0.9 "$p3 send"
$ns at 1.0 "$p3 send"
$ns at 1.1 "$p3 send"
$ns at 1.2 "$p3 send"
```

```
$ns at 1.3 "$p3 send"
$ns at 1.4 "$p3 send"
$ns at 1.5 "$p3 send"
$ns at 1.6 "$p3 send"
$ns at 1.7 "$p3 send"
$ns at 1.8 "$p3 send"
$ns at 1.9 "$p3 send"
$ns at 2.0 "$p3 send"
$ns at 2.1 "$p3 send"
$ns at 2.2 "$p3 send"
$ns at 2.3 "$p3 send"
$ns at 2.4 "$p3 send"
$ns at 2.5 "$p3 send"
$ns at 2.6 "$p3 send"
$ns at 2.7 "$p3 send"
$ns at 2.8 "$p3 send"
$ns at 2.9 "$p3 send"
$ns at 3.0 "finish"
$ns run
```

<u>AWK file</u> (Open a new editor using "vi command" and write awk file and save with ".awk" extension)

```
BEGIN{
drop=0;
}
{
 if($1=="d")
  {
   drop++;
   }
}
END{
printf("Total number of %s packets dropped due to congestion =%d\n",$5,drop);
}
```

<u>Steps for execution</u>

1) Open vi editor and type program. Program name should have the extension " **.tcl** "

   [root@localhost ~]# vi p2.tcl

2) Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.

3) Open vi editor and type **awk** program. Program name should have the extension "**.awk** "

   [root@localhost ~]# vi p2.awk

4) Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.

5) Run the simulation program

   [root@localhost~]# ns p2.tcl

i) Here **“ns”** indicates network simulator. We get the topology shown in the snapshot.

ii) Now press the play button in the simulation window and the simulation will begins.

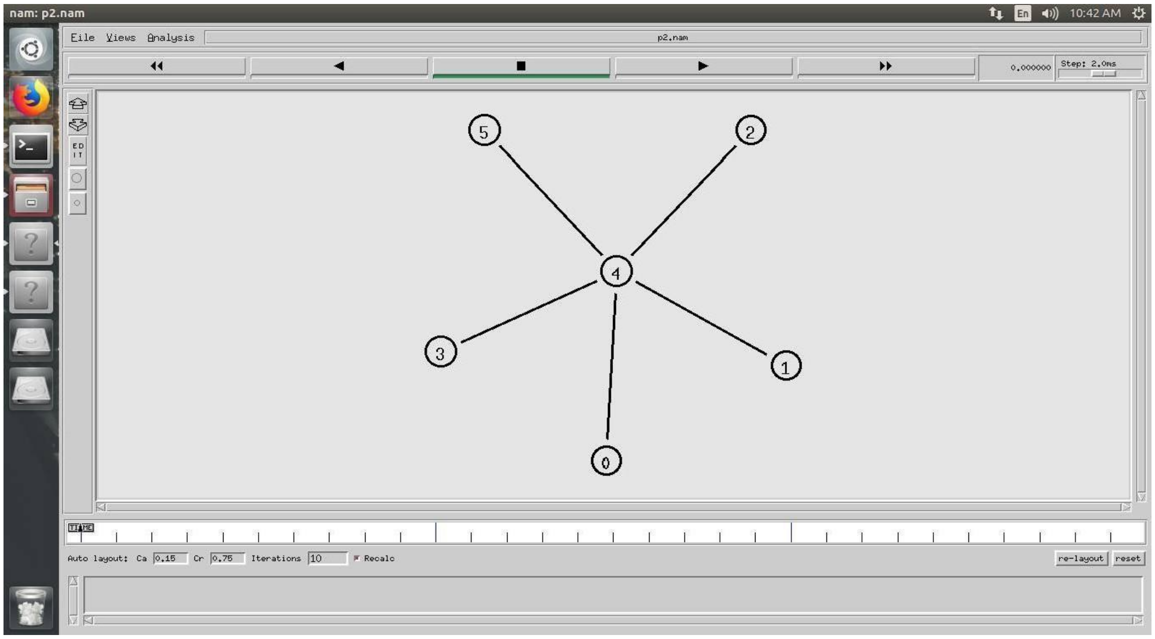
6) After simulation is completed run **awk file** to see the output ,

[root@localhost~]# awk –f p2.awk p2.tr

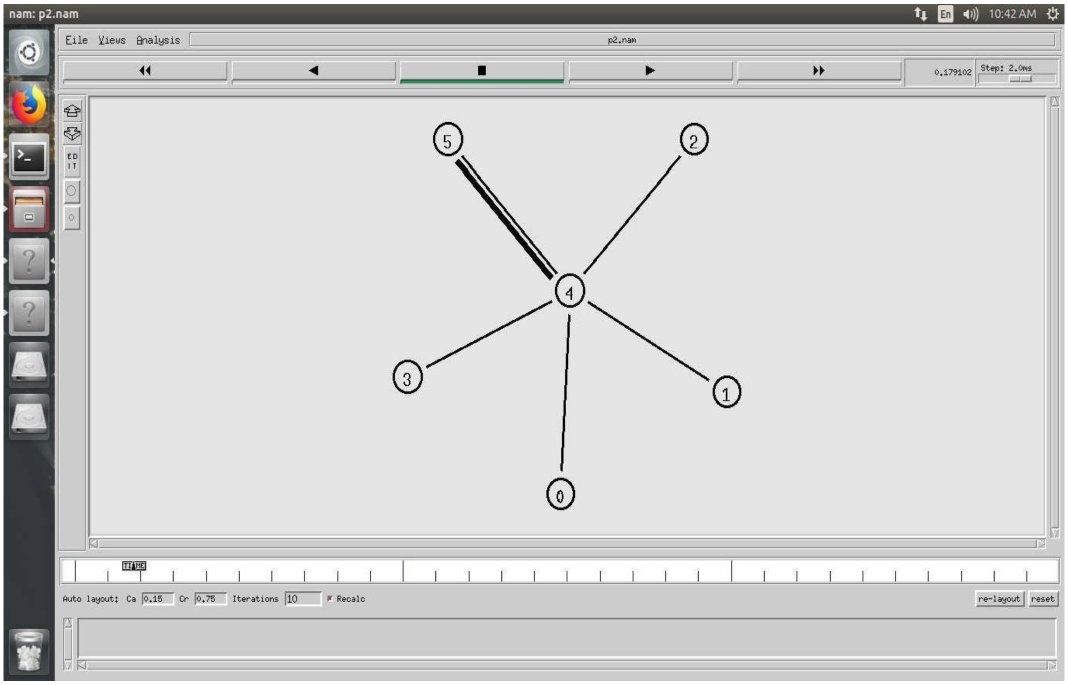
7) To see the trace file contents open the file as ,

[root@localhost~]# vi p2.tr

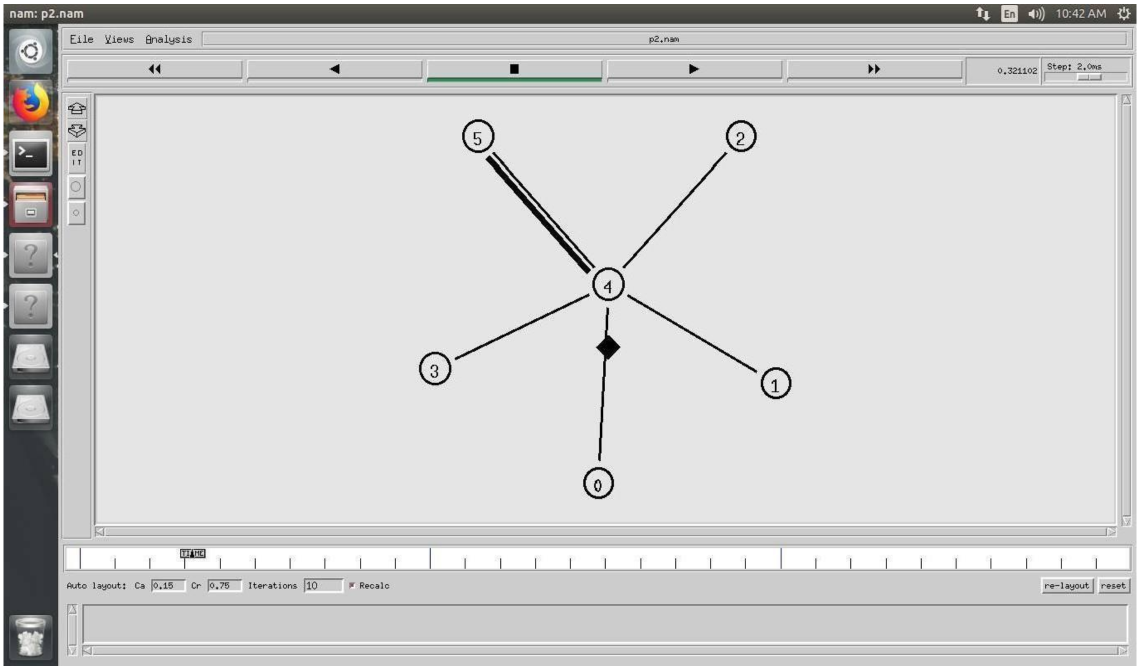
<u>Topology</u>

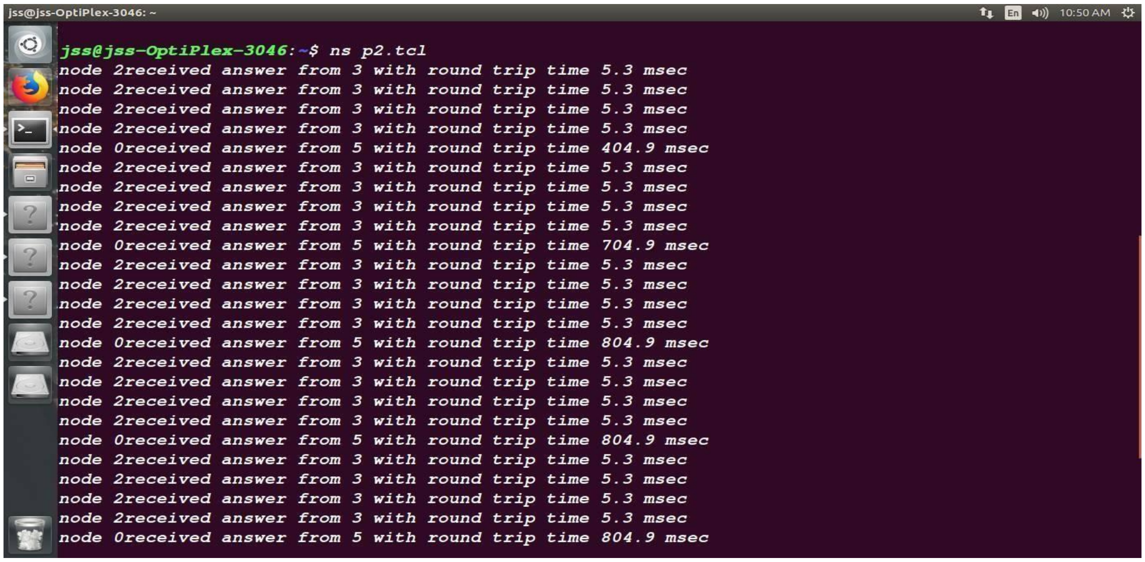


Auto layout: Ca 0.15 Cr 0.75 Iterations 10 ☒ Recalc

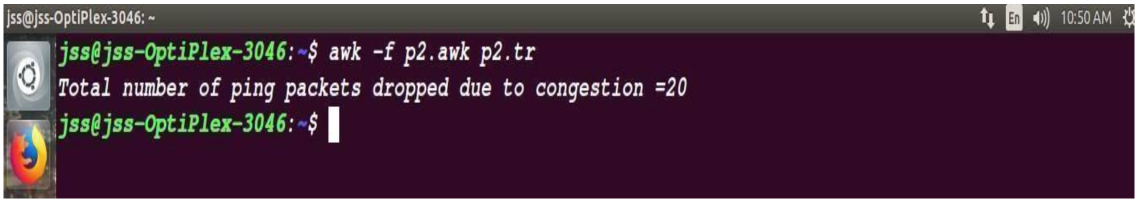


Auto layout: Ca 0.15 Cr 0.75 Iterations 10 ☒ Recalc




```
jss@jss-OptiPlex-3046:~$ ns p2.tcl
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 404.9 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 704.9 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 804.9 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 804.9 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 804.9 msec
```

Output


```
jss@jss-OptiPlex-3046:~$ awk -f p2.awk p2.tr
Total number of ping packets dropped due to congestion =20
jss@jss-OptiPlex-3046:~$
```

Note:

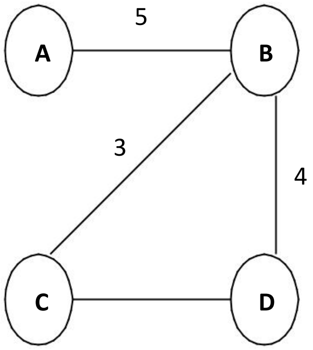
Vary the bandwidth and queue size between the nodes n0-n2 , n2-n4. n6-n2 and n2- n5 and see the number of packets dropped at the nodes.

5. Write a program to find the shortest path between vertices using bellman-ford algorithm.

Source code:

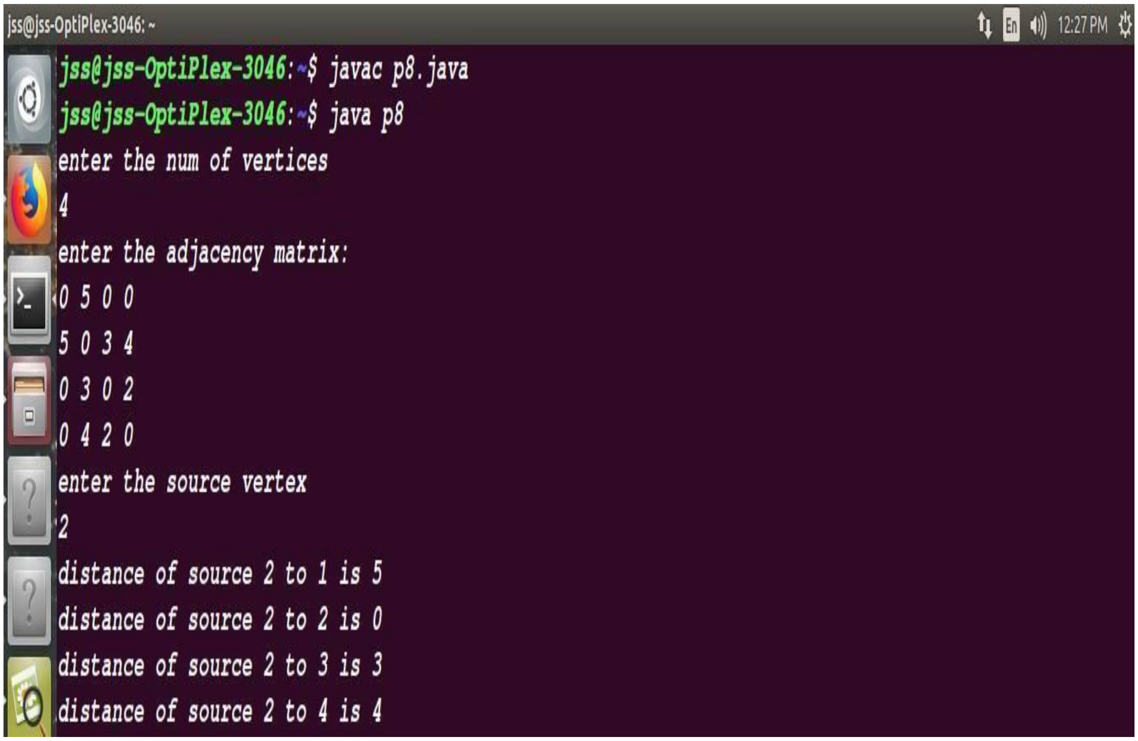
```
import java.util.Scanner;
public class p8
{
  private int d[];
  private int num_ver;
  public static final int max_value=999;
  public p8(int num_ver)
  {
    this.num_ver=num_ver;
    d=new int [num_ver+1];
  }
  public void bellmanfordevaluation(int source,int a[][])
  {
    for(int node=1; node<=num_ver; node++)
    {
      d[node]=max_value;
    }
     d[source]=0;
     for(int node=1; node<=num_ver-1; node++)
     {
      for(int sn=1;sn<=num_ver;sn++)
      {
       for(int dn=1;dn<=num_ver;dn++)
              {
          if(a[sn][dn]!=max_value)
           {
             if(d[dn]>d[sn]+a[sn][dn])
               d[dn]=d[sn]+a[sn][dn];
           }
         }
       }
     }
    for(int sn=1;sn<=num_ver;sn++)
    {
      for(int dn=1;dn<=num_ver;dn++)
        {
          if(a[sn][dn]!=max_value)
            { if(d[dn]>d[sn]+a[sn][dn])
                 System.out.println("the graph contains -ve edge cycle");
            }
        }
    }
    for(int vertex=1;vertex<=num_ver;vertex++)
     {
       System.out.println("disten of source"+source+"to"+vertex+"is"+d[vertex]);
     }
  }
  public static void main(String args[])
   {
     int num_ver=0;
     int source;
     Scanner scanner=new Scanner(System.in);
     System.out.println("enter the num of vertices");
     num_ver=scanner.nextInt();
     int a[][]=new int [num_ver+1] [num_ver+1];
     System.out.println("enter the adjacency matrix:");
     for(int sn=1;sn<=num_ver;sn++)
      {
        for(int dn=1;dn<=num_ver;dn++)
         {
             a[sn][dn]=scanner.nextInt();
             if(sn==dn)
             { a[sn][dn]=0;
              continue;
             }
            if(a[sn][dn]==0)
           {
             a[sn][dn]=max_value;
           }
         }
      }
   System.out.println("enter the source vertex");
   source=scanner.nextInt();
   p8 b=new p8(num_ver);
   b.bellmanfordevaluation(source,a);
   scanner.close();
  }
 }
```

Input graph:



Output:

```
jss@jss-Optiplex-3046:~ vi p8.java

jss@jss-Optiplex-3046:~ javac p8.java

jss@jss-Optiplex-3046:~ java p8
```



```
jss@jss-OptiPlex-3046:~$ javac p8.java
jss@jss-OptiPlex-3046:~$ java p8
enter the num of vertices
4
enter the adjacency matrix:
0 5 0 0
5 0 3 4
0 3 0 2
0 4 2 0
enter the source vertex
2
distance of source 2 to 1 is 5
distance of source 2 to 2 is 0
distance of source 2 to 3 is 3
distance of source 2 to 4 is 4
```

6. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.

```
set ns [new Simulator]
set tf [open p3.tr w]
$ns trace-all $tf
set nf [open p3.nam w]
$ns namtrace-all $nf

set n0 [$ns node]
$n0 color "magenta"
$n0 label "src1"
set n1 [$ns node]
set n2 [$ns node]
$n2 color "magenta"
$n2 label "src2"
set n3 [$ns node]
$n3 color "blue"
$n3 label "dest2"
set n4 [$ns node]
set n5 [$ns node]
$n5 color "blue"
$n5 label "dest1"

$ns make-lan "$n0 $n1 $n2 $n3 $n4" 100Mb 100ms LL Queue/DropTail Mac/802_3
$ns duplex-link $n4 $n5 1Mb 1ms DropTail

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.0001
set sink5 [new Agent/TCPSink]
$ns attach-agent $n5 $sink5

$ns connect $tcp0 $sink5

set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ftp2 set packetSize_ 600
$ftp2 set interval_ 0.001
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3

$ns connect $tcp2 $sink3

set file1 [open file1.tr w]
$tcp0 attach $file1
```

```
set file2 [open file2.tr w]
$tcp2 attach $file2

$tcp0 trace cwnd_
$tcp2 trace cwnd_

proc finish { } {
global ns nf tf
$ns flush-trace
close $tf
close $nf
exec nam p3.nam &
exit 0
}

$ns at 0.1 "$ftp0 start"
$ns at 5 "$ftp0 stop"
$ns at 7 "$ftp0 start"
$ns at 0.2 "$ftp2 start"
$ns at 8 "$ftp2 stop"
$ns at 14 "$ftp0 stop"
$ns at 10 "$ftp2 start"
$ns at 15 "$ftp2 stop"
$ns at 16 "finish"
$ns run
```

AWK file (Open a new editor using "vi command" and write awk file and save with ".awk" extension)

cwnd:- means congestion window

```
BEGIN {
}
{
if($6=="cwnd_") # don't leave space after writing cwnd_
printf("%f\t%f\t\n",$1,$7); # you must put \n in printf
}
END {
}
```

Steps for execution

1) Open vi editor and type program. Program name should have the extension " **.tcl** "
   [root@localhost ~]# vi p3.tcl
2) Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.
3) Open vi editor and type **awk** program. Program name should have the extension "**.awk** "
   [root@localhost ~]# vi p3.awk
4) Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.

5) Run the simulation program

   [root@localhost~]# ns p3.tcl

6) After simulation is completed run **awk file** to see the output ,

   i. **[root@localhost~]# awk –f p3.awk file1.tr > a1**

   ii. **[root@localhost~]# awk –f p3.awk file2.tr > a2**

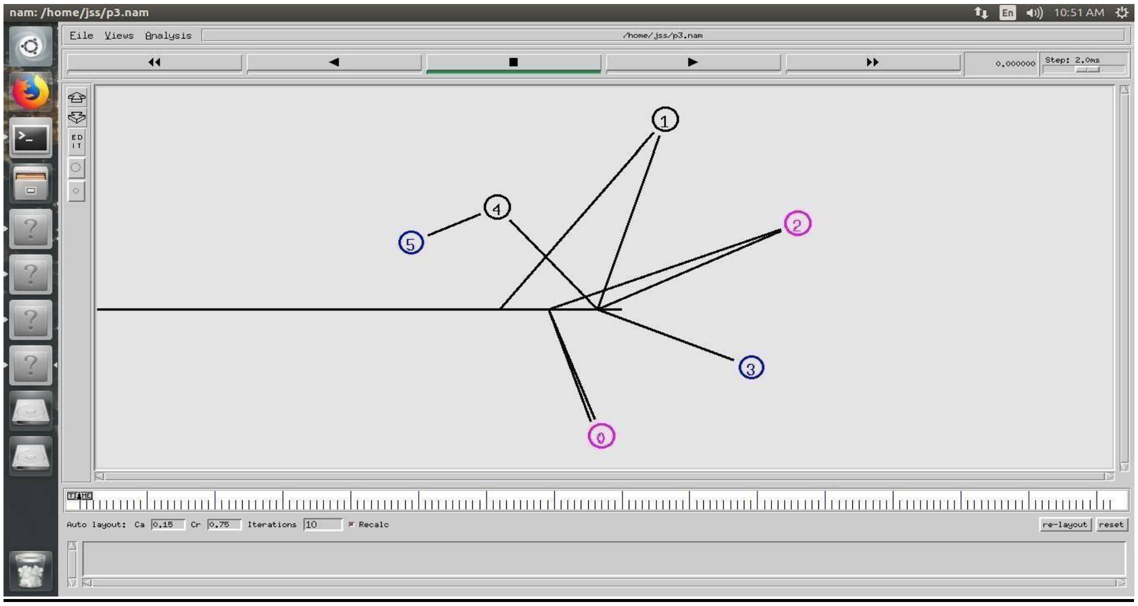
   iii. **[root@localhost~]# xgraph a1 a2**

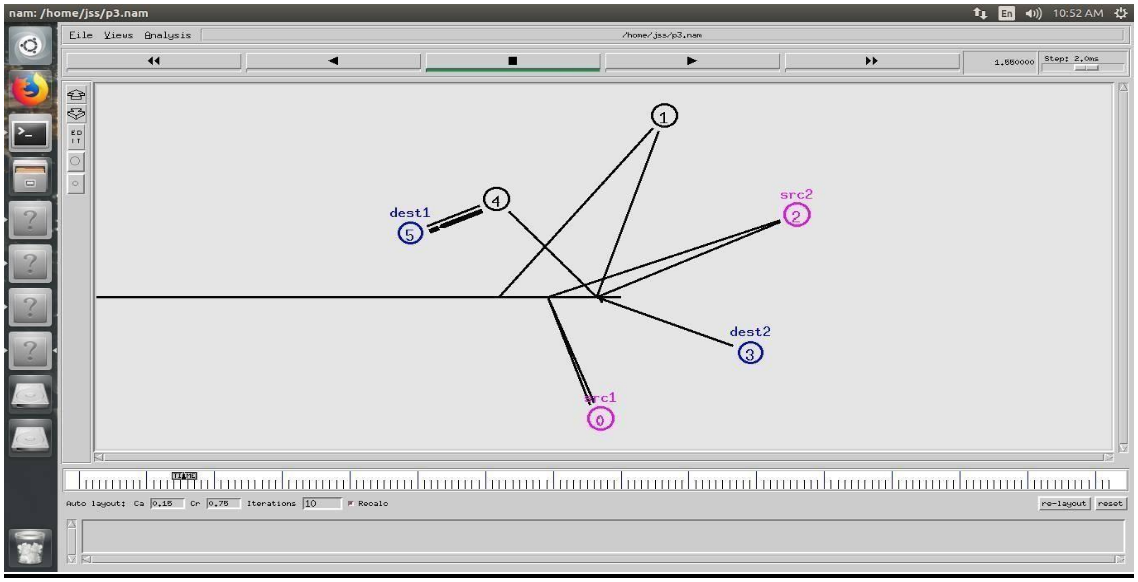
7) Here we are using the congestion window trace files i.e. **file1.tr** and **file2.tr** and we are redirecting the contents of those files to new files say **a1** and **a2** using **output redirection operator (>)**.

8) To see the trace file contents open the file as ,

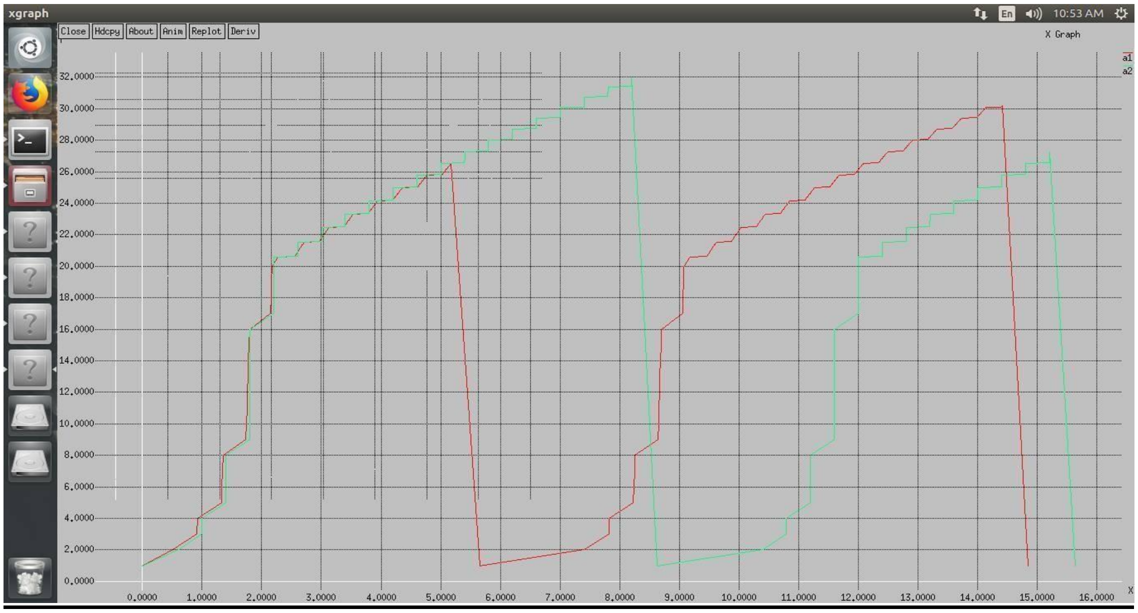
   [root@localhost~]# vi p3.tr

Topology





Output

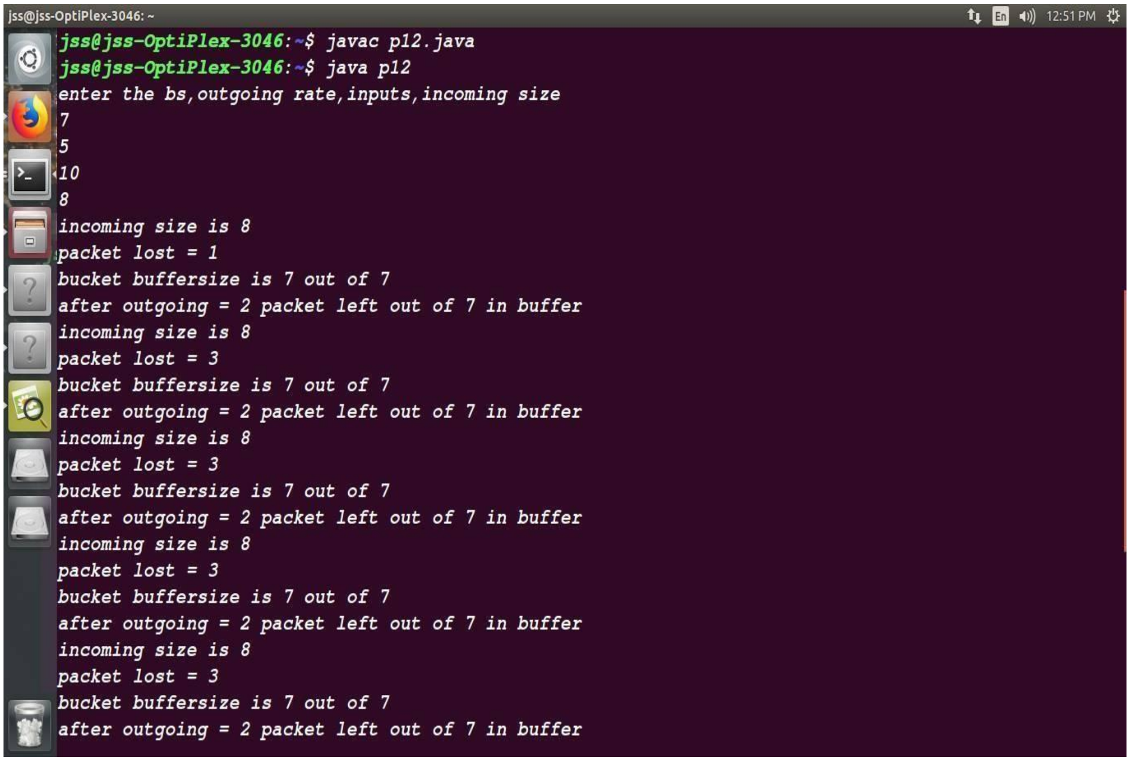

xgraph
10:53 AM
Close
Hdcpy
About
Anim
Replot
Deriv
X Graph
a1
a2
32.0000
30.0000
28.0000
26.0000
24.0000
22.0000
20.0000
18.0000
16.0000
14.0000
12.0000
10.0000
8.0000
6.0000
4.0000
2.0000
0.0000
0.0000
1.0000
2.0000
3.0000
4.0000
5.0000
6.0000
7.0000
8.0000
9.0000
10.0000
11.0000
12.0000
13.0000
14.0000
15.0000
16.0000
X

7. Write a program for congestion control using leaky bucket algorithm.

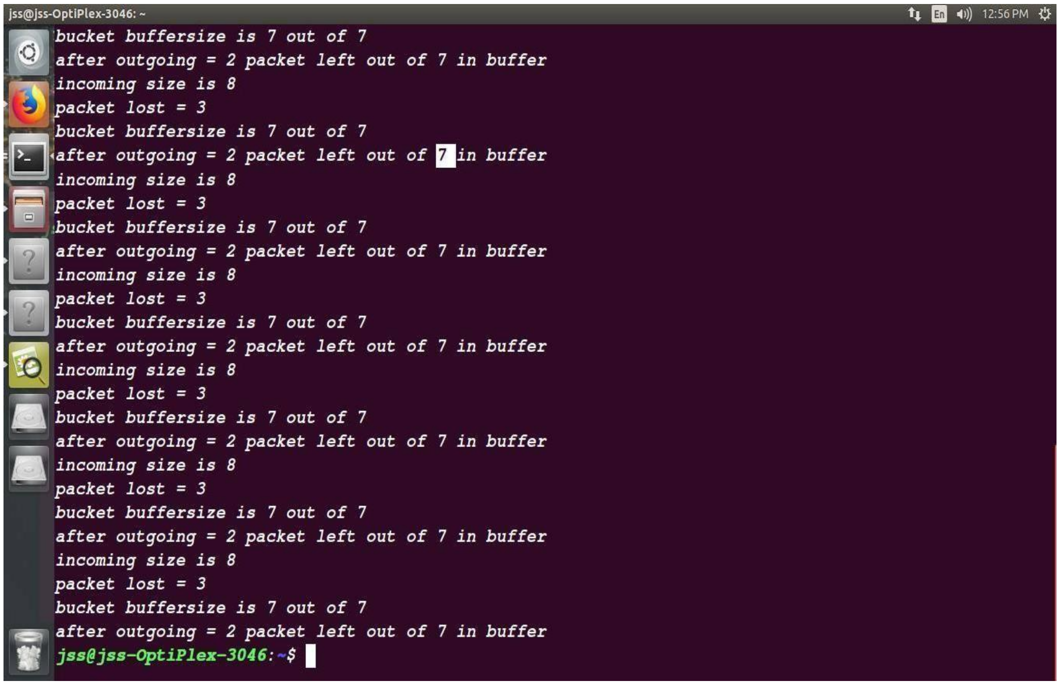
Source Code:

```
import java.util.Scanner;
public class p12
{
public static void main(String[] args) throws InterruptedException
{
Scanner in=new Scanner(System.in);
int n,incoming,outgoing,bs,s=0;
System.out.println("enter the bs,outgoing rate,inputs,incoming size");
bs=in.nextInt();
outgoing=in.nextInt();
n=in.nextInt();
incoming=in.nextInt();
while(n!=0)
{
System.out.println("incoming size is"+incoming);
if(incoming<=(bs-s))
{
s+=incoming;
System.out.println("bucket buffer size is"+s+"out of"+bs);
}
else
{
System.out.println("packet lost="+(incoming-(bs-s)));
s=bs;
System.out.println("bucket buffersize is"+s+"out of"+bs);
}
s-=outgoing;
System.out.println("after outgoing="+s+"packet left out of"+bs+"in buffer");
n--;
Thread.sleep(3000);
}
in.close();
}
}
```

Output:



```
jss@jss-OptiPlex-3046:~$ javac p12.java
jss@jss-OptiPlex-3046:~$ java p12
enter the bs,outgoing rate,inputs,incoming size
7
5
10
8
incoming size is 8
packet lost = 1
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
incoming size is 8
packet lost = 3
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
incoming size is 8
packet lost = 3
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
incoming size is 8
packet lost = 3
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
incoming size is 8
packet lost = 3
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
```



```
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
incoming size is 8
packet lost = 3
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
incoming size is 8
packet lost = 3
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
incoming size is 8
packet lost = 3
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
incoming size is 8
packet lost = 3
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
incoming size is 8
packet lost = 3
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
incoming size is 8
packet lost = 3
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
jss@jss-OptiPlex-3046:~$
```