**Program No. 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.**

**1.tcl**

set ns [ new Simulator ]

set tf [ open p1.tr w ]

$ns trace-all $tf

set nf [ open p1.nam w ]

$ns namtrace-all $nf

**# The below code is used to create the nodes.**

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

**#This is used to give color to the packets.**

$ns color 1 "red"

$ns color 2 "blue"

$n0 label "Source/udp0"

$n1 label "Source/udp1"

$n2 label "Router"

$n3 label "Destination/Null"

**#Vary the below Bandwidth and see the number of packets dropped.**

$ns duplex-link $n0 $n2 1Mb 300ms DropTail

$ns duplex-link $n1 $n2 20Mb 300ms DropTail

$ns duplex-link $n2 $n3 0.5Mb 300ms DropTail

**#The below code is used to set the queue size b/w the nodes**

$ns set queue-limit $n0 $n2 10

$ns set queue-limit $n1 $n2 10

$ns set queue-limit $n2 $n3 2

**#The below code is used to attach an UDP agent to n0, UDP agent to n1 and null agent to n3**

set udp0 [new Agent/UDP]

$ns attach-agent $n0 $udp0

set cbr0 [new Application/Traffic/CBR]

$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 null [new Agent/Null]

$ns attach-agent $n3 $null

**#The below code sets the udp0 packets to red and udp1 packets to blue color**

$udp0 set class\_ 1

$udp1 set class\_ 2

**#The below code is used to connect the agents**

$ns connect $udp0 $null

$ns connect $udp1 $null

**#The below code is used to set the packet size to 500**

$cbr1 set packetSize\_ 500Mb

**#The below code is used to set the interval of the packets,i.e., Data rate of the packets.**

**#if the data rate is high then packets drops are high.**

$cbr1 set interval\_ 0.005

**#finish procedure**

proc finish { } {

global ns nf tf

$ns flush-trace

close $tf

close $nf

exec nam p1.nam &

exit 0

}

**#ns scheduler**

$ns at 0.1 "$cbr0 start"

$ns at 0.1 "$cbr1 start"

$ns at 5.0 "finish"

**#start simulation**

$ns run

**1.awk**

BEGIN{

#include<stdio.h>

count=0;

}

{

if($1=="d") #d stands for the packets drops.

count++

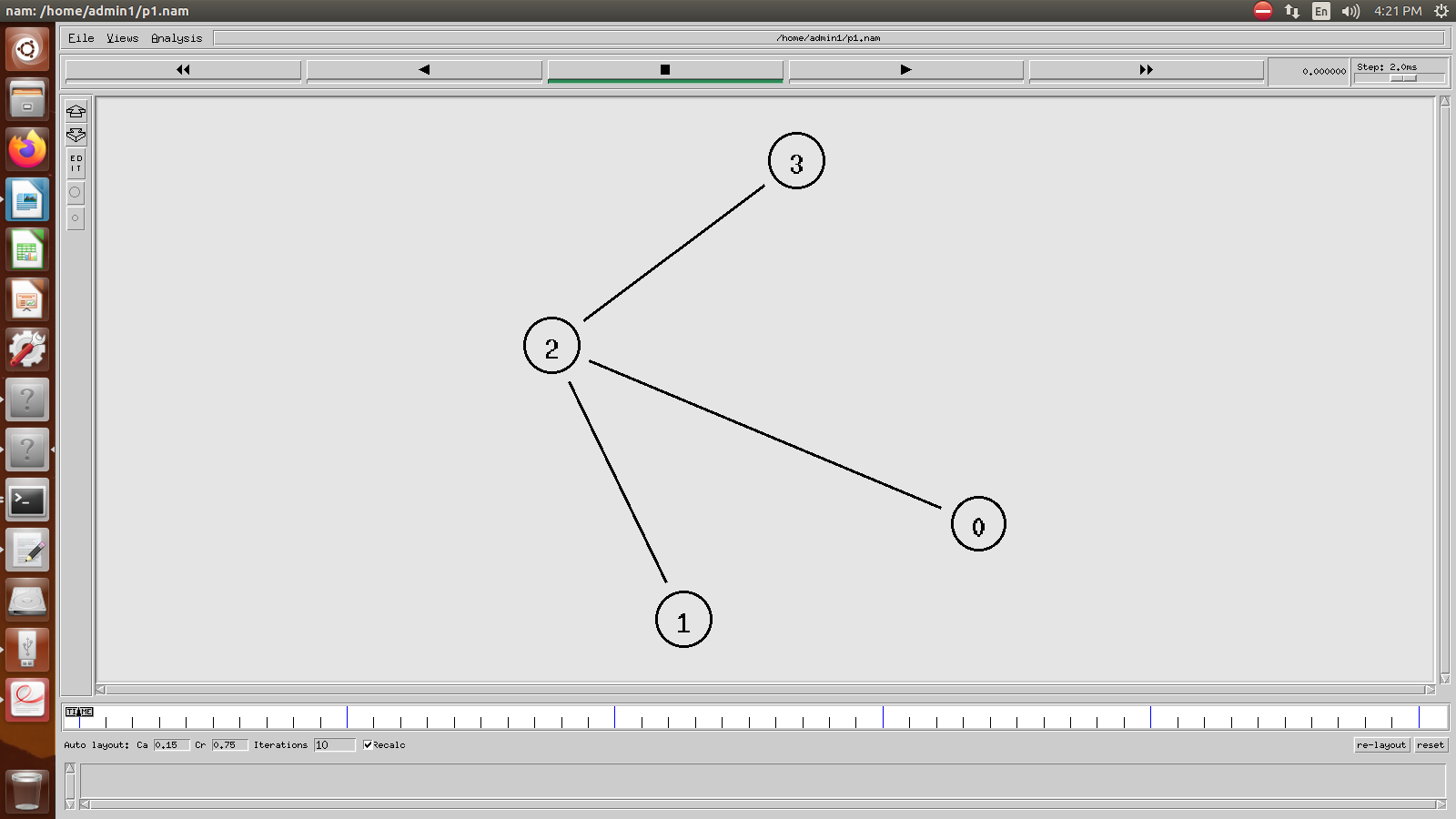
}

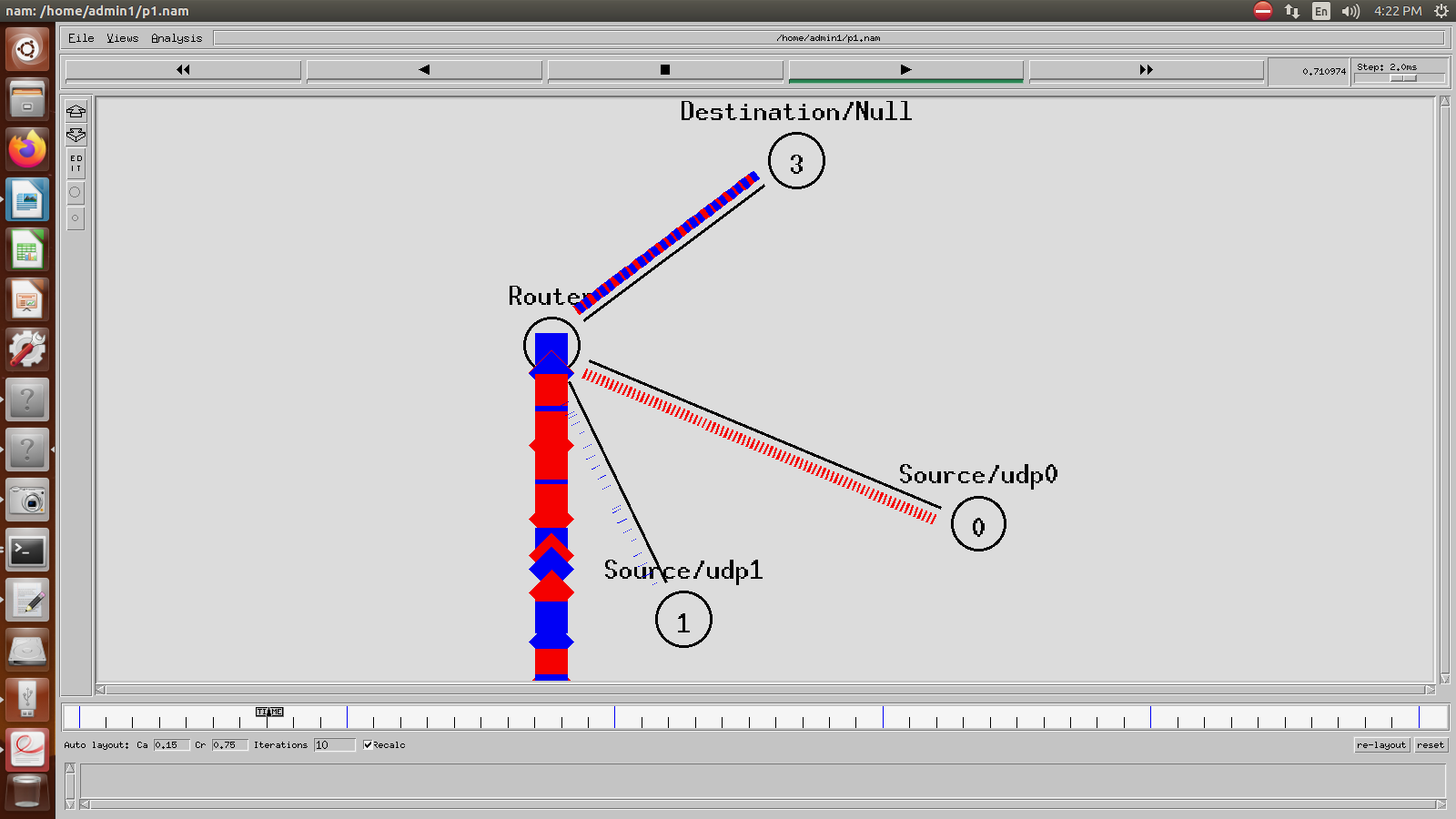
END{

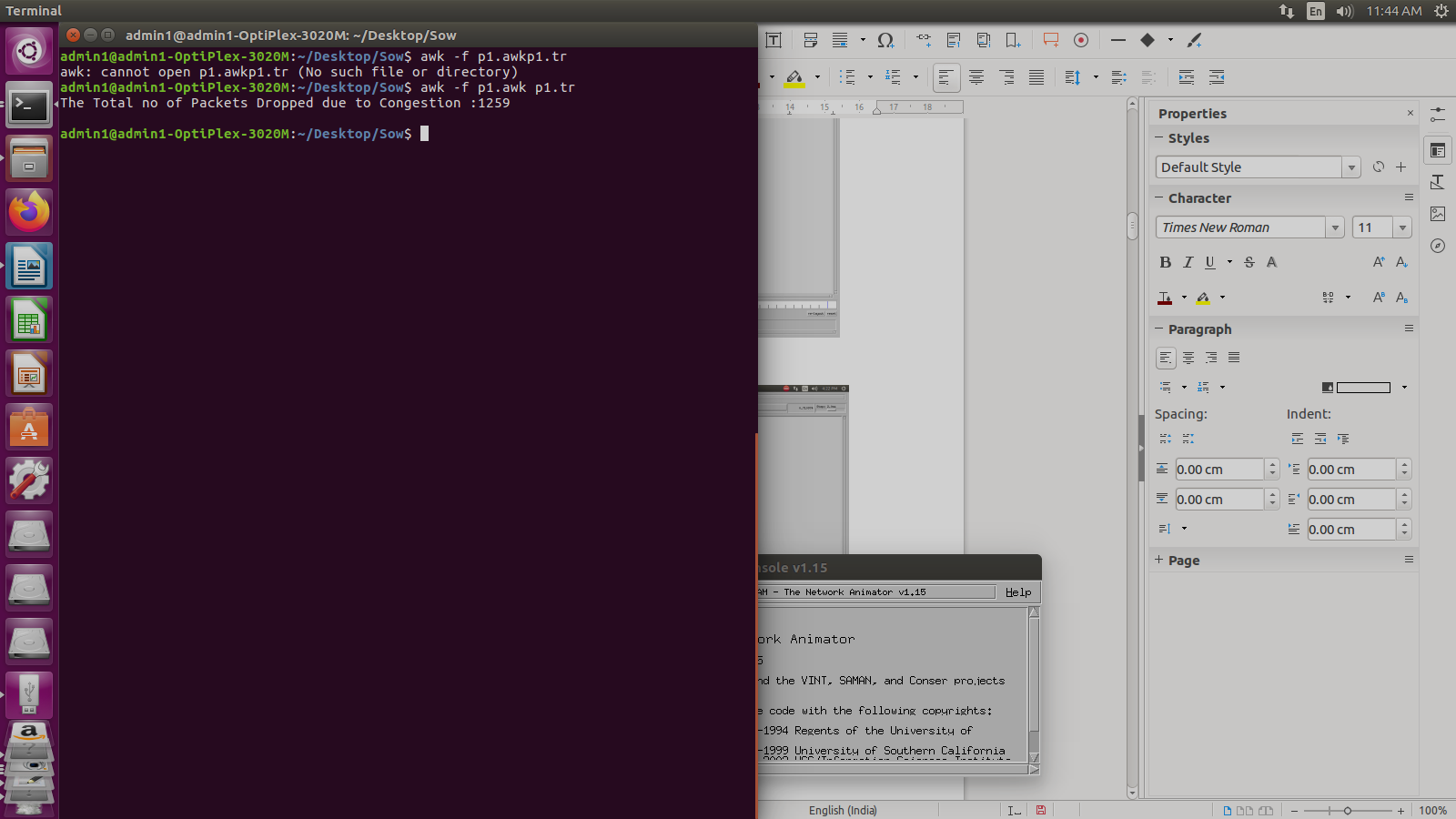
printf("The Total no of Packets Dropped due to Congestion :%d\n\n", count)

}

**Output:**







**Program No. 2: 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.**

**2.tcl**

set ns [new Simulator]

set nf [open lab2.nam w]

$ns namtrace-all $nf

set nd [open lab2.tr w]

$ns trace-all $nd

proc finish {} {

global ns nf nd

$ns flush-trace

close $nf

close $nd

exec nam lab2.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]

$ns duplex-link $n1 $n0 1Mb 10ms DropTail

$ns duplex-link $n2 $n0 1Mb 10ms DropTail

$ns duplex-link $n3 $n0 1Mb 10ms DropTail

$ns duplex-link $n4 $n0 1Mb 10ms DropTail

$ns duplex-link $n5 $n0 1Mb 10ms DropTail

$ns duplex-link $n6 $n0 1Mb 10ms DropTail

Agent/Ping instproc recv {from rtt} {

$self instvar node\_

puts "node [$node\_ id] recieved ping answer from $from with round-trip-time $rtt ms."

}

set p1 [new Agent/Ping]

set p2 [new Agent/Ping]

set p3 [new Agent/Ping]

set p4 [new Agent/Ping]

set p5 [new Agent/Ping]

set p6 [new Agent/Ping]

$ns attach-agent $n1 $p1

$ns attach-agent $n2 $p2

$ns attach-agent $n3 $p3

$ns attach-agent $n4 $p4

$ns attach-agent $n5 $p5

$ns attach-agent $n6 $p6

$ns queue-limit $n0 $n4 1

$ns queue-limit $n0 $n5 1

$ns queue-limit $n0 $n6 1

$ns connect $p1 $p4

$ns connect $p2 $p5

$ns connect $p3 $p6

$ns at 0.2 "$p1 send"

$ns at 0.4 "$p2 send"

$ns at 0.6 "$p3 send"

$ns at 1.0 "$p4 send"

$ns at 1.2 "$p5 send"

$ns at 1.4 "$p6 send"

$ns at 2.0 "finish"

$ns run

**2.awk**

BEGIN{

#include<stdio.h>

count=0;

}

{

if($1=="d") #d stands for the packets drops.

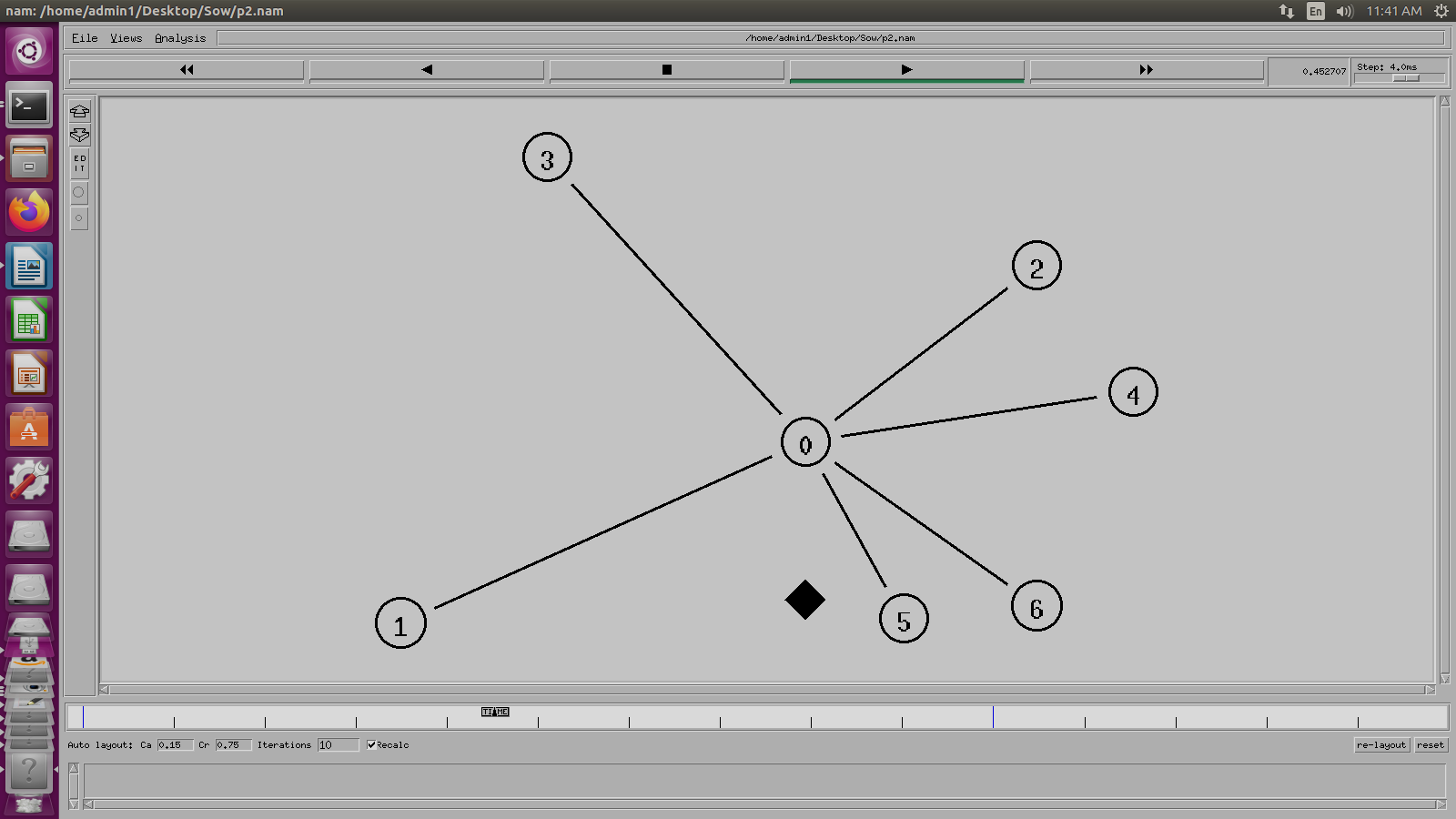
count++

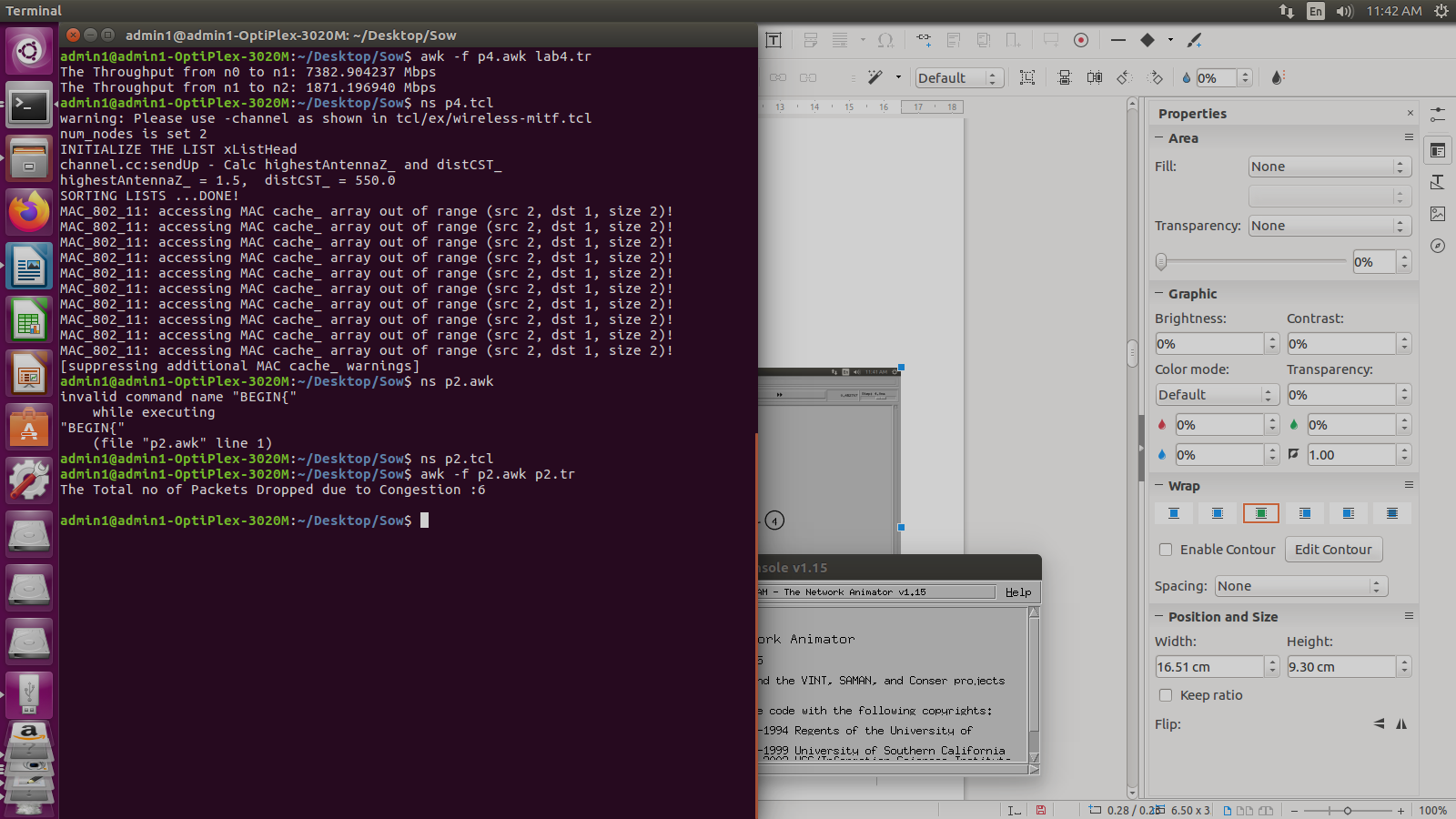
}

END{

printf("The Total no of Packets Dropped due to Congestion :%d\n\n", count)

}





**Program No. 3: Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source /destination**

**3. tcl**

set ns [new Simulator]

set nf [open lab3.nam w]

$ns namtrace-all $nf

set nd [open lab3.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 lab3.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 cwnd.tr w]

$tcp1 attach $tfile

$tcp1 trace cwnd\_

set tcp2 [new Agent/TCP/Reno]

$ns attach-agent $n2 $tcp2

set sink2 [new Agent/TCPSink]

$ns attach-agent $n8 $sink2

$ns connect $tcp2 $sink2

$tcp2 set class\_ 2

$tcp2 set packetSize\_ 55

set ftp2 [new Application/FTP]

$ftp2 attach-agent $tcp2

set tfile2 [open cwnd2.tr w]

$tcp2 attach $tfile2

$tcp2 trace cwnd\_

$ns at 0.5 "$ftp1 start"

$ns at 1.0 "$ftp2 start"

$ns at 5.0 "$ftp2 stop"

$ns at 5.0 "$ftp1 stop"

$ns at 5.5 "finish"

$ns run

**3.awk**

BEGIN {

}

{

if($6=="cwnd\_") {

printf("%f\t%f\n",$1,$7);

}

}

END {

}

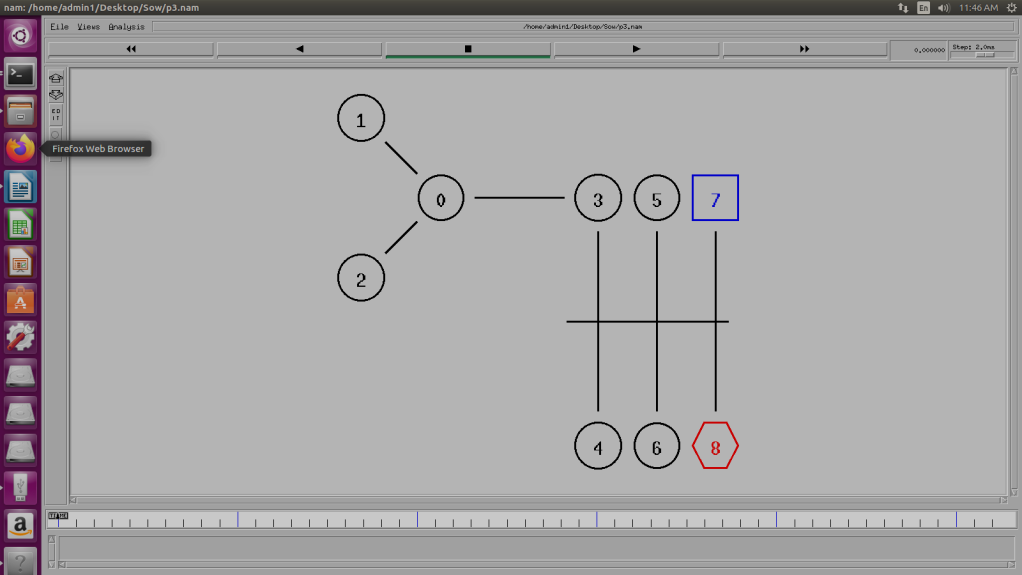
**Commands to run the program**

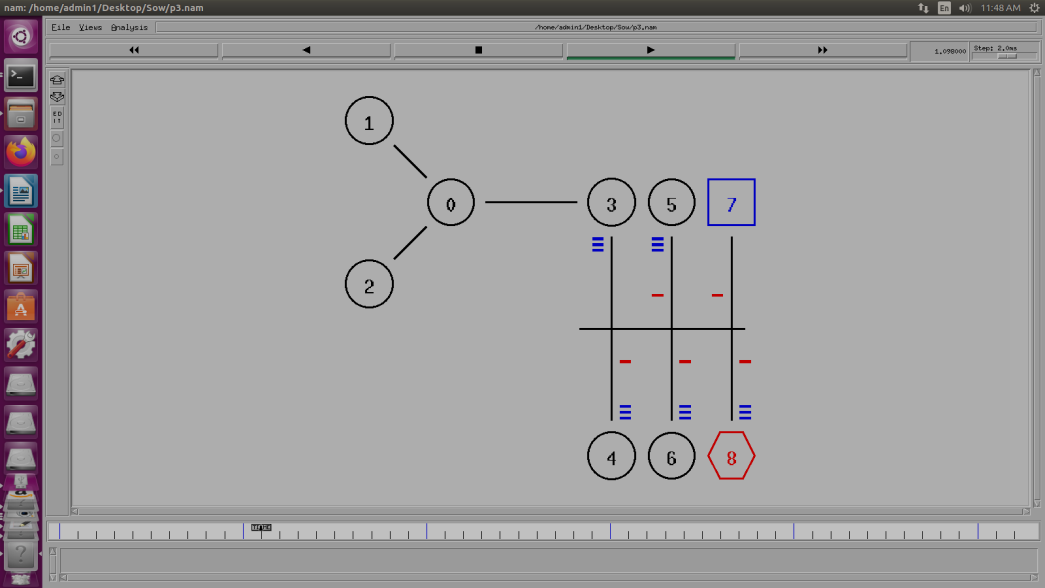
ns 3.tcl

awk -f 3.awk cwnd.tr > tcp1

awk -f 3.awk cwnd2.tr > tcp2

xgraph -x -y tcp1 tcp2







**Program No. 4: Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets**

**p4.tcl**

set ns [new Simulator]

set val(chan) Channel/WirelessChannel;

set val(prop) Propagation/TwoRayGround;

set val(netif) Phy/WirelessPhy;

set val(mac) Mac/802\_11;

set val(ifq) Queue/DropTail/PriQueue;

set val(ll) LL;

set val(ant) Antenna/OmniAntenna;

set val(ifqlen) 50;

set val(nn) 2;

set val(rp) DSDV;

set val(x) 1000.0;

set val(y) 1000.0;

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 $val(rp) \

-llType $val(ll) \

-macType $val(mac) \

-ifqType $val(ifq) \

-ifqLen $val(ifqlen) \

-antType $val(ant) \

-propType $val(prop) \

-phyType $val(netif) \

-channelType $val(chan) \

-topoInstance $topo \

-agentTrace ON \

-routerTrace ON \

-macTrace ON

create-god $val(nn)

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\_ 250

$n0 set Y\_ 250

$n0 set Z\_ 0

$n1 set X\_ 300

$n1 set Y\_ 300

$n1 set Z\_ 0

$n2 set X\_ 600

$n2 set Y\_ 600

$n2 set Z\_ 0

$ns at 0.1 "$n0 setdest 250 250 15"

$ns at 0.1 "$n1 setdest 300 300 25"

$ns at 0.1 "$n2 setdest 600 600 25"

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"

#The below code is used to provide the node movements.

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

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

proc finish {} {

global ns nf tf

$ns flush-trace

exec nam p4.nam &

close $tf

exit 0

}

$ns at 250 "finish"

$ns run

**p4.awk**

BEGIN{

#include<stdio.h>

count1=count2=pack1=pack2= time1=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

}

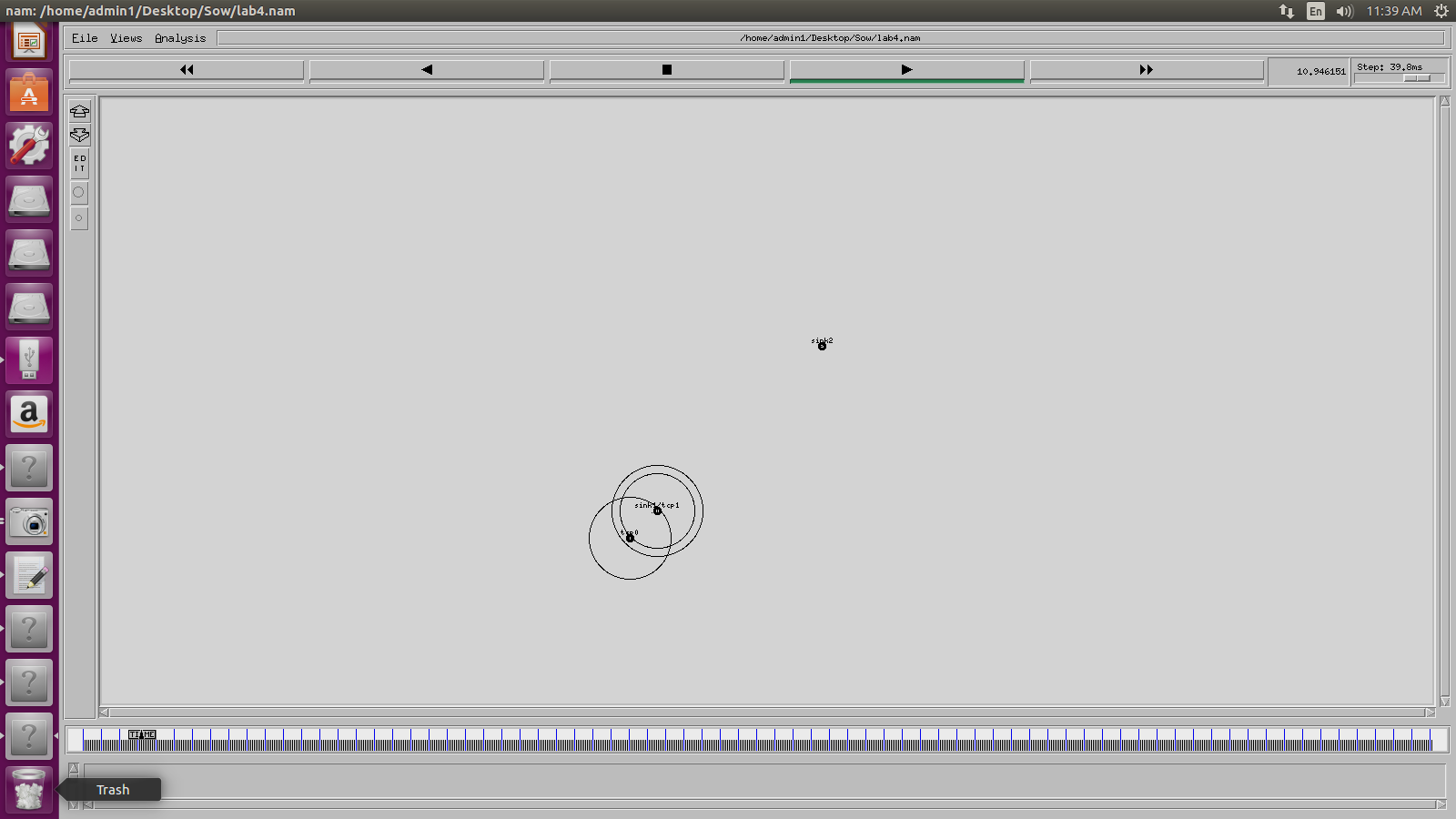
}

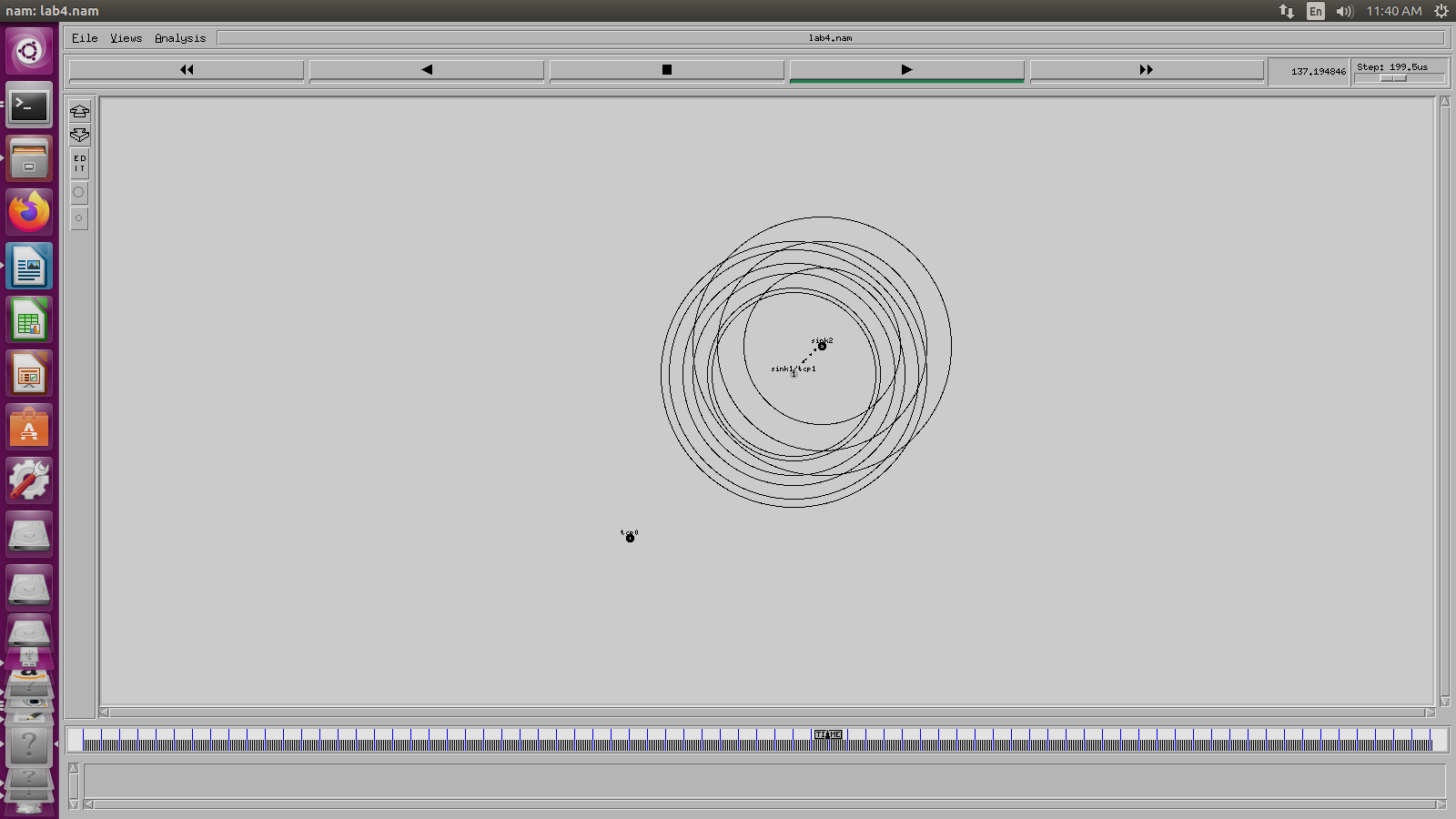
END{

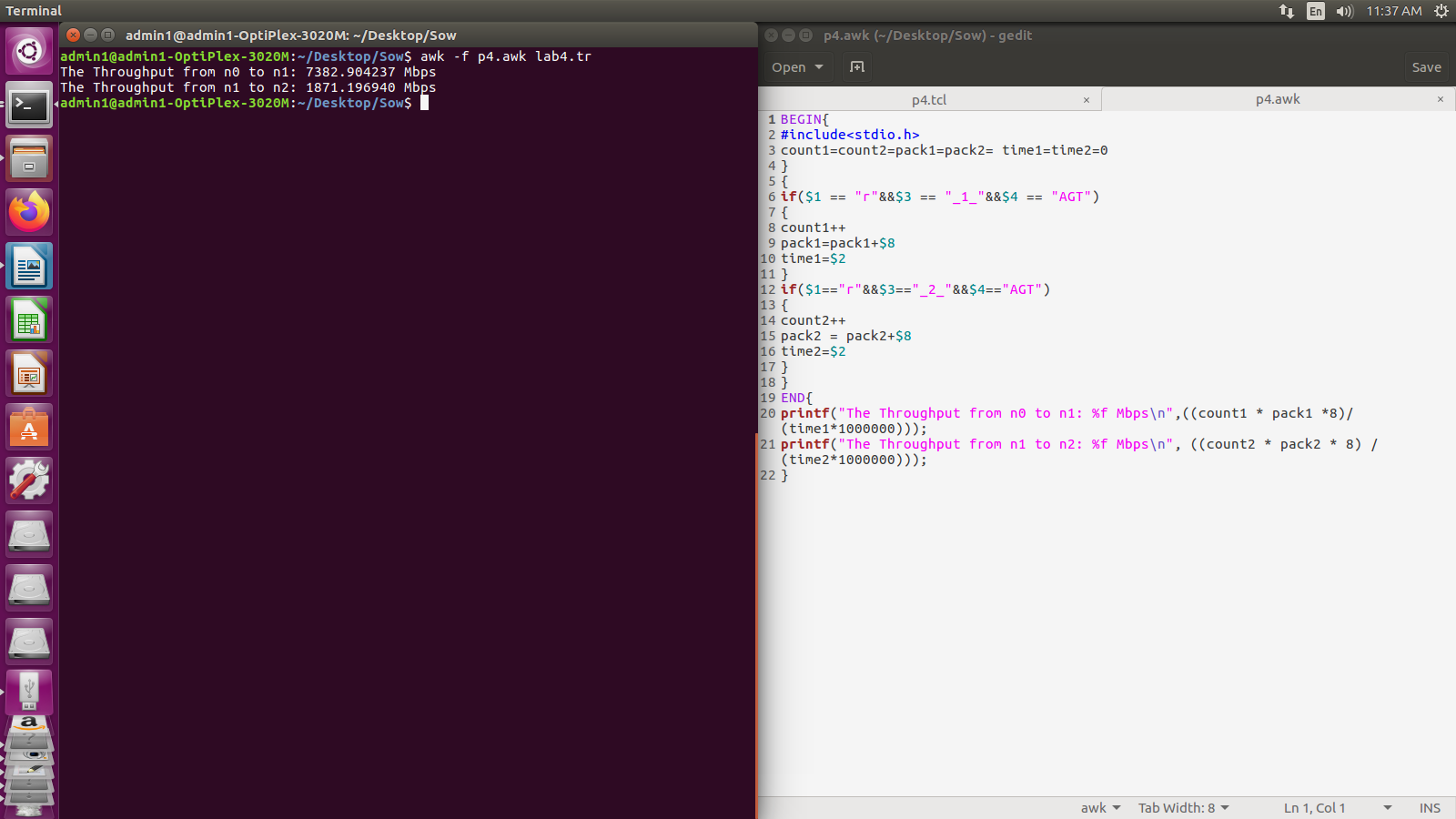
printf("The Throughput from n0 to n1: %f Mbps\n",((count1 \* pack1 \*8)/ time1\*1000000)));

printf("The Throughput from n1 to n2: %f Mbps", ((count2 \* pack2 \* 8) /(time2\*1000000)));

}







**Program No. 5: Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.**

**Note: Change directory using *cd ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts***

**p5.tcl**

set opt(title) zero;

set opt(stop) 100;

set opt(ecn) 0;

set opt(type) gsm;

set opt(secondDelay) 55;

set opt(minth) 30;

set opt(maxth) 0;

set opt(adaptive) 1 ;

set opt(flows) 0;

set opt(window) 30;

set opt(web) 2;

set opt(quiet) 0;

set opt(wrap) 100;

set opt(srcTrace) is;

set opt(dstTrace) bs2;

set opt(gsmbuf) 10;

set bwDL(gsm) 9600

set bwUL(gsm) 9600

set propDL(gsm) .500

set propUL(gsm) .500

set buf(gsm) 20

set ns [new Simulator]

set tf [open out.tr w]

$ns trace-all $tf

set nodes(is) [$ns node]

set nodes(ms) [$ns node]

set nodes(bs1) [$ns node]

set nodes(bs2) [$ns node]

set nodes(lp) [$ns node]

proc cell\_topo { } {

global ns nodes

$ns duplex-link $nodes(lp) $nodes(bs1) 3Mbps 10ms DropTail

$ns duplex-link $nodes(bs1) $nodes(ms) 1 1 RED

$ns duplex-link $nodes(ms) $nodes(bs2) 1 1 RED

$ns duplex-link $nodes(bs2) $nodes(is) 3Mbps 50ms DropTail

puts "Cell Topology"

}

proc set\_link\_params {t} {

global ns nodes bwUL bwDL propUL propDL buf

$ns bandwidth $nodes(bs1) $nodes(ms) $bwDL($t) duplex

$ns bandwidth $nodes(bs2) $nodes(ms) $bwDL($t) duplex

$ns delay $nodes(bs1) $nodes(ms) $propDL($t) duplex

$ns delay $nodes(bs2) $nodes(ms) $propDL($t) duplex

$ns queue-limit $nodes(bs1) $nodes(ms) $buf($t)

$ns queue-limit $nodes(ms) $nodes(bs1) $buf($t)

$ns queue-limit $nodes(bs2) $nodes(ms) $buf($t)

$ns queue-limit $nodes(ms) $nodes(bs2) $buf($t)

}

Queue/RED set summarystats\_ true

Queue/DropTail set summarystats\_ true

Queue/RED set adaptive\_ $opt(adaptive)

Queue/RED set q\_weight\_ 0.0

Queue/RED set thresh\_ $opt(minth)

Queue/RED set maxthresh\_ $opt(maxth)

Queue/DropTail set shrink\_drops\_ true

Agent/TCP set ecn\_ $opt(ecn)

Agent/TCP set window\_ $opt(window)

DelayLink set avoidReordering\_ true

source web.tcl

switch $opt(type) {

gsm -

gprs -

umts {cell\_topo}

}

set\_link\_params $opt(type)

$ns insert-delayer $nodes(ms) $nodes(bs1) [new Delayer]

$ns insert-delayer $nodes(bs1) $nodes(ms) [new Delayer]

$ns insert-delayer $nodes(ms) $nodes(bs2) [new Delayer]

$ns insert-delayer $nodes(bs2) $nodes(ms) [new Delayer]

if {$opt(flows) == 0} {

set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]

set ftp1 [[set tcp1] attach-app FTP]

$ns at 0.8 "[set ftp1] start"

}

proc stop {} {

global nodes opt

set wrap $opt(wrap)

set sid [$nodes($opt(srcTrace)) id]

set did [$nodes($opt(dstTrace)) id]

set a "out.tr"

set GETRC "../../../bin/getrc"

set RAW2XG "../../../bin/raw2xg"

exec $GETRC -s $sid -d $did -f 0 out.tr | $RAW2XG -s 0.01 -m $wrap -r > plot.xgr

exec $GETRC -s $did -d $sid -f 0 out.tr | $RAW2XG -a -s 0.01 -m $wrap >> plot.xgr

exec ./xg2gp.awk plot.xgr

exec xgraph -x time -y packets plot.xgr &

exit 0

}

$ns at $opt(stop) "stop"

$ns run

