# COMPUTER NETWORKS LABORATORY

|  |  |
| --- | --- |
| Sub Code : 18CSL57 | IA Marks : 20 |
| Hrs / Week : 01I + 02P | Exam Hours : 03 |
| Total Hrs : 40 | Exam Marks : 80 |

**PART A - Simulation Exercises**

## For the experiments below modify the topology and parameters set for the experiment and take multiple rounds of reading and analyze the results available in log files. Plot necessary graphs and conclude. Use NS2/NS3.

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.
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.
3. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.
4. Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.
5. Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.
6. Implement and study the performance of CDMA on NS2/NS3 (Using stack called Call net) or equivalent environment.

## PART B

**Implement the following in Java:**

1. Write a program for error detecting code using CRC-CCITT (16- bits).
2. Write a program to find the shortest path between vertices using bellman-ford algorithm.
3. Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.
4. Write a program on datagram socket for client/server to display the messages on client side, typed at the server side.
5. Write a program for simple RSA algorithm to encrypt and decrypt the data.
6. Write a program for congestion control using leaky bucket algorithm.

## Conduction of Practical Examination:

1. All laboratory experiments are to be included for practical examination.
2. Students are allowed to pick one experiment from part A and part B with lot.
3. Strictly follow the instructions as printed on the cover page of answer script
4. Marks distribution: Procedure + Conduction + Viva: 80 Part A: 10+25+5 =40 Part B: 10+25+5 =40
5. Change of experiment is allowed only once and marks allotted to the procedure part to be made zero.

# PART - A

## Experiment 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.

**Step1:** Open text editor, type the below program and save with extention .tcl (**prog1.tcl**) set ns [new Simulator]

set nf [open prog1.nam w]

$ns namtrace-all $nf

set nd [open prog1.tr w]

$ns trace-all $nd

proc finish { } { global ns nf nd

$ns flush-trace close $nf

close $nd

exec nam prog1.nam & exit 0

}

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

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

$ns duplex-link $n1 $n2 512kb 10ms DropTail

$ns queue-limit $n1 $n2 10

set udp0 [new Agent/UDP]

$ns attach-agent $n0 $udp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.005

$cbr0 attach-agent $udp0 set sink [new Agent/Null]

$ns attach-agent $n2 $sink

$ns connect $udp0 $sink

$ns at 0.2 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

$ns at 5.0 "finish"

$ns run

**Step2:** Open text editor, type the below program and save with extention .awk (**prog1.awk**) BEGIN {

dcount = 0;

rcount = 0;

}

{

event = $1; if(event == "d")

{

dcount++;

}

if(event == "r")

{

rcount++;

}

} END {

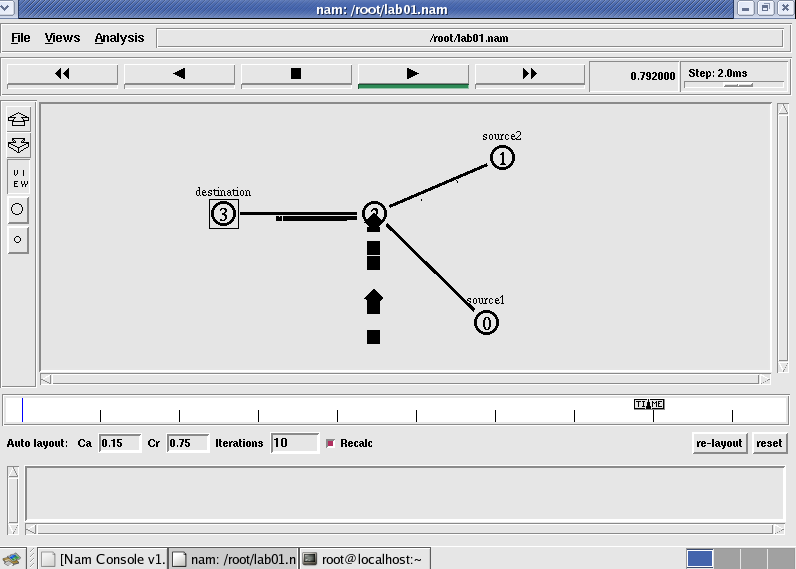
printf("The no.of packets dropped : %d\n ",dcount); printf("The no.of packets recieved : %d\n ",rcount);

}

**Step3**: Run the simulation program

## [root@localhost~]# ns prog1.tcl

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



**Step 4:**Now press the play button in the simulation window and the simulation will begins.

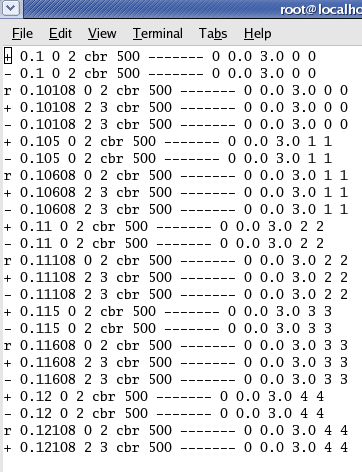
**Step 5:**After simulation is completed run **awk file** to see the output ,

## [root@localhost~]# awk –f prog1.awk prog1.tr

Number of packets droped = 16

**Step 6:**To see the trace file contents open the file as ,

## [root@localhost~]# vi prog1.tr



**Experiment 2:**

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

**Step1:** Open text editor, type the below program and save with extention .tcl (**prog3.tcl**) set ns [new Simulator]

set nf [open prog3.nam w]

$ns namtrace-all $nf

set nd [open prog3.tr w]

$ns trace-all $nd

proc finish {} { global ns nf nd

$ns flush-trace close $nf

close $nd

exec nam prog4.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 3

$ns queue-limit $n0 $n5 2

$ns queue-limit $n0 $n6 2

$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

**Step2:** Open text editor, type the below program and save with extention .awk (**prog3.awk**) BEGIN {

count=0;

}

{

event=$1; if(event=="d")

{

count++;

}

} END {

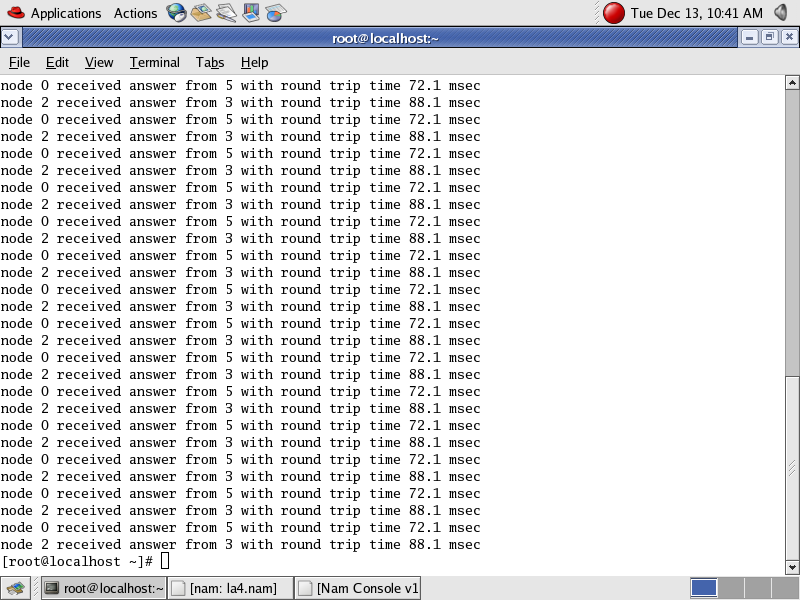
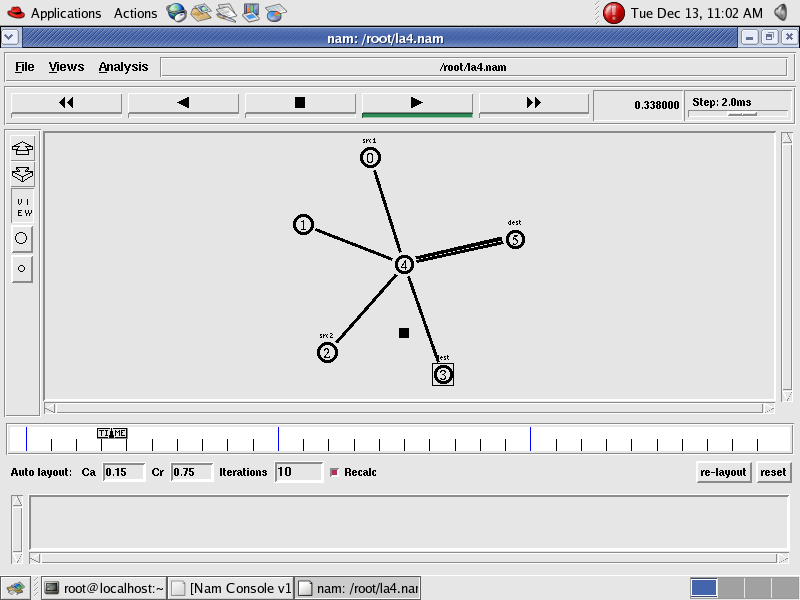
printf("No of packets dropped : %d\n",count);

}

**Step3**: Run the simulation program

## [root@localhost~]# ns prog3.tcl

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



**Step 4:**Now press the play button in the simulation window and the simulation will begins.

**Step 5:**After simulation is completed run **awk file** to see the output ,

## [root@localhost~]# awk –f prog3.awk prog3.tr

**Step 6:**To see the trace file contents open the file as ,

## [root@localhost~]# vi prog3.tr

**Experiment 3:**

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

**Step1:** Open text editor, type the below program and save with extention .tcl (**prog5.tcl**)

set ns [new Simulator]

set nf [open prog5.nam w]

$ns namtrace-all $nf

set nd [open prog5.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 prog5.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

**Step2:** Open text editor, type the below program and save with extention .awk (**prog5.awk**) BEGIN {

}

{

if($6=="cwnd\_") { printf("%f\t%f\n",$1,$7);

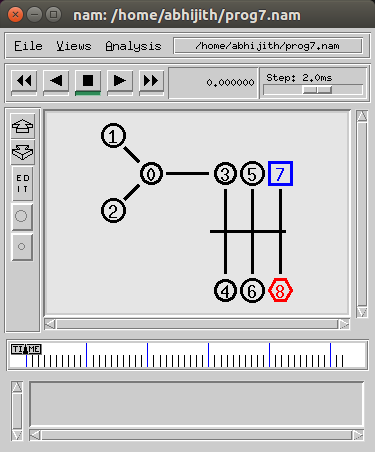
}

} END {

}

**Step3**: Run the simulation program

## [root@localhost~]# ns prog5.tcl

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

**Step 4:**Now press the play button in the simulation window and the simulation will begins.

**Step 5:**After simulation is completed run **awk file** and generate the graph , **[root@localhost~]# awk –f prog5.awk cwnd.tr> a1 [root@localhost~]# awk –f prog5.awk cwnd2.tr> a2**

## [root@localhost~]#xgraph a1 a2



**Step 6:**To see the trace file contents open the file as ,

## [root@localhost~]# vi prog5.tr

**Experiment 4:**

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

**Step1:** Open text editor, type the below program and save with extention .tcl (**prog6.tcl**)

set ns [new Simulator] set tf [open prog6.tr w]

$ns trace-all $tf

set topo [new Topography]

$topo load\_flatgrid 1000 1000 set nf [open prog6.nam w]

$ns namtrace-all-wireless $nf 1000 1000 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) AODV ; set val(x) 500 ;

set val(y) 400 ;

set val(stop) 10.0 ;

$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 OFF \

-movementTrace 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 15"

$ns at 0.1 "$n1 setdest 100 100 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"

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

$ns at 190 "$n1 setdest 70 70 15" proc finish { } {

global ns nf tf

$ns flush-trace

exec nam prog6.nam & close $tf

exit 0

}

$ns at 250 "finish"

$ns run

**Step2:** Open text editor, type the below program and save with extention .awk (**prog6.awk**) 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{

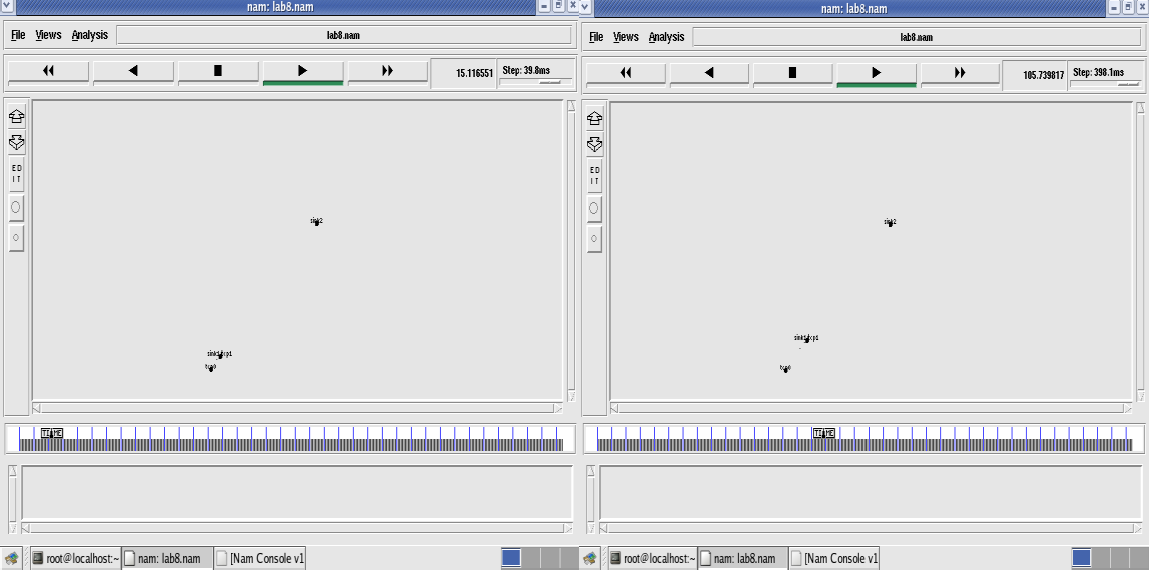
printf("The Throughput from n0 to n1: %f Mbps \n", ((count1\*pack1\*8)/(time1\*1000000))); printf("The Throughput from n1 to n2: %f Mbps", ((count2\*pack2\*8)/(time2\*1000000)));

}

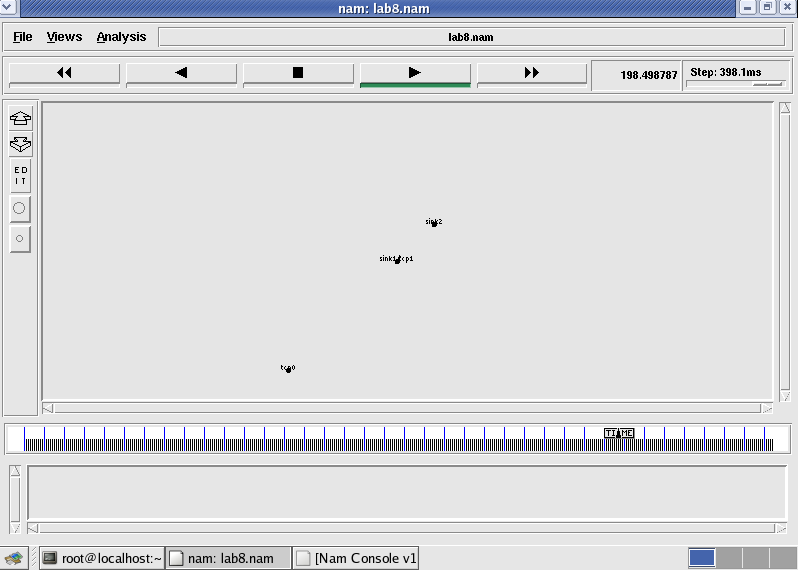
**Step3**: Run the simulation program

## [root@localhost~]# ns prog3.tcl

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



Node 1 and 2 are communicating Node 2 is moving towards node 3

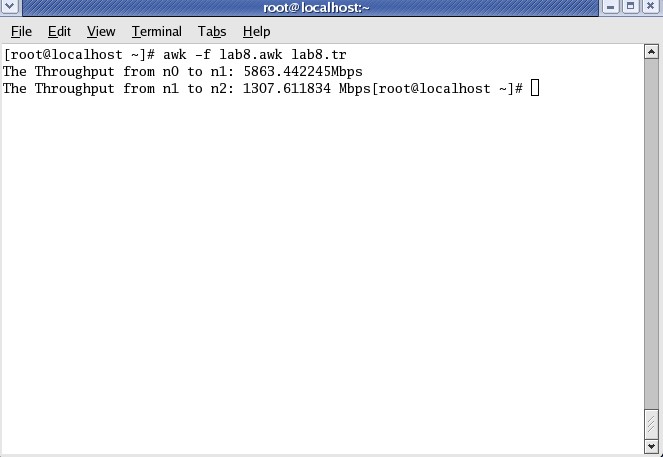


Node 2 is coming back from node 3 towards node1

**Step 4:**Now press the play button in the simulation window and the simulation will begins.

**Step 5:**After simulation is completed run **awk file** to see the output ,

## lb8[root@localhost~]# awk –f prog6.awk prog6.tr



**Step 6:**To see the trace file contents open the file as ,

## [root@localhost~]# vi prog2.tr

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

**Experiment 5:**

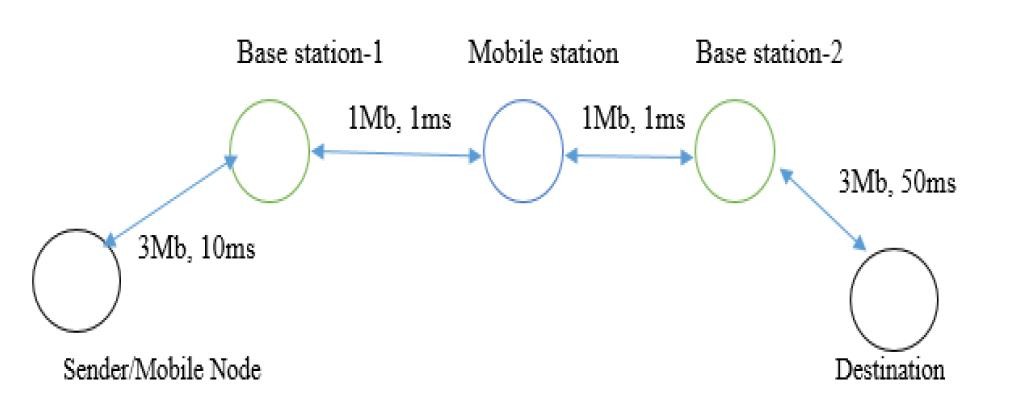
**Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.**

Second Generation (2G) technology is based on the technology known as global system for mobile communication (GSM). This technology enabled various networks to provide services like text messages, picture messages and MMS. The technologies used in 2G are either TDMA (Time Division Multiple Access) which divides signal into different time slots or CDMA (Code Division Multiple Access) which allocates a special code to each user so as to communicate over a multiplex physical channel.

GSM uses a variation of time division multiple access (TDMA). 2G networks developed as a replacement for first generation (1G) analog cellular networks, and the GSM standard originally described as a digital, circuit-switched network optimized for full duplex voice telephony. This expanded over time to include data communications, first by circuit-switched transport, then by packet data transport via GPRS (General Packet Radio Services).

GSM can be implemented on all the versions of NS2 (Since year 2004: ns-2.27, and later versions of NS2)

**Design:**



**Program:**

**Step1: Change the directory using**

**cd ~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts**

**Step 2: Type the following program using gedit and save as gsm.tcl**

set ns [new Simulator]

set tf [open out.tr w]

$ns trace-all $tf

set td [open out.nam w]

$ns namtrace-all $td

# General Parameters

set stop 100 ;# Stop time. # Topology

set type gsm ;#type of link:

# AQM parameters set minth 30

set maxth 0

set adaptive 1 ;# 1 for Adaptive RED, 0 for plain RED # Traffic generation.

set flows 0 ;# number of long-lived TCP flows set window 30 ;# window for long-lived traffic

# Plotting statistics.

set opt(wrap) 100 ;# wrap plots?

set opt(srcTrace) is ;# where to plot traffic set opt(dstTrace) bs2 ;# where to plot traffic #default downlink bandwidth in bps

set bwDL(gsm) 9600

#default downlink propagation delay in seconds set propDL(gsm) .500

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 "GSM Cell Topology"

}

proc set\_link\_params {t} { global ns nodes bwDL propDL

$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) 10

$ns queue-limit $nodes(bs2) $nodes(ms) 10

}

# RED and TCP parameters Queue/RED set adaptive\_ $adaptive Queue/RED set thresh\_ $minth Queue/RED set maxthresh\_ $maxth Agent/TCP set window\_ $window

source web.tcl

#Create topology switch $type { gsm -

cdma {cell\_topo}

}

set\_link\_params $type

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

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

# Set up forward TCP connection if {$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 tf td 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 nam out.nam &

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

exit 0

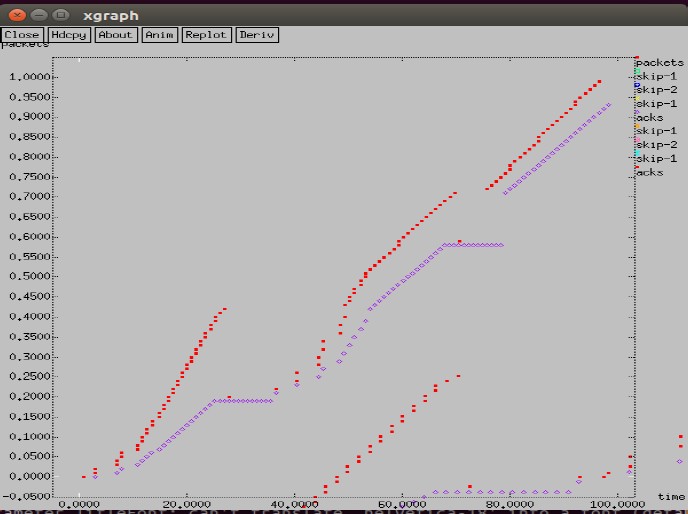
}

$ns at $stop "stop"

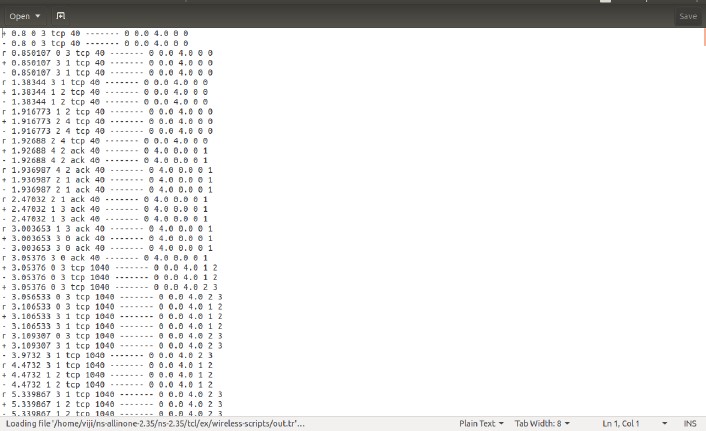
$ns run

**Output:**

**Step 3: Run the program using command ns gsm.tcl**



**GSM Trace File:**

**Step 4: open the trace file using command gedit out.tr**

**Experiment 6:**

**Implement and study the performance of CDMA on NS2/NS3 (Using stack calledCall net) or equivalent environment.**

3G networks developed as a replacement for second generation (2G) GSM standard network with full duplex voice telephony. CDMA is used as the access method in many mobile phone standards. IS-95, also called cdmaOne, and its 3G evolution CDMA2000, are often simply referred to as CDMA, but UMTS(The Universal Mobile Telecommunications System is a third generation mobile cellular system for networks based on the GSM standard.), the 3G standard used by GSM carriers, also uses wideband CDMA. Long-Term Evolution (LTE) is a standard for high-speed wireless communication which uses CDMA network technology.

3G technology generally refers to the standard of accessibility and speed of mobile devices. The standards of the technology were set by the International Telecommunication Union (ITU). This technology enables use of various services like GPS (Global Positioning System), mobile television and video conferencing. It not only enables them to be used worldwide, but also provides with better bandwidth and increased speed. The main aim of this technology is to allow much better coverage and growth with minimum investment.

CDMA can be implemented on all the versions of NS2 (Since year 2004: ns-2.27, and later versions of NS2)

## Design:

**Program:**

**Step1: Change the directory using**

**cd ~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts**

**Step 2: Type the following program using gedit and save as cdma.tcl**

set ns [new Simulator] set tf [open out.tr w]

$ns trace-all $tf

set td [open out.nam w]

$ns namtrace-all $td

# General Parameters

set stop 100 ;# Stop time. # Topology

set type cdma ;#type of link # AQM parameters

set minth 30

set maxth 0

set adaptive 1 ;# 1 for Adaptive RED, 0 for plain RED # Traffic generation.

set flows 0 ;# number of long-lived TCP flows set window 30 ;# window for long-lived traffic # Plotting statics.

set opt(wrap) 100 ;# wrap plots?

set opt(srcTrace) is ;# where to plot traffic set opt(dstTrace) bs2 ;# where to plot traffic #default downlink bandwidth in bps

set bwDL(cdma) 384000

#default downlink propagation delay in seconds set propDL(cdma) .150

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 " cdma Cell Topology"

}

proc set\_link\_para {t} { global ns nodes bwDL propDL

$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) 20

$ns queue-limit $nodes(bs2) $nodes(ms) 20

}

# RED and TCP parameters

Queue/RED set adaptive\_ $adaptive Queue/RED set thresh\_ $minth Queue/RED set maxthresh\_ $maxth Agent/TCP set window\_ $window

source web.tcl #Create topology switch $type { cdma {cell\_topo}

}

set\_link\_para $type

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

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

# Set up forward TCP connection if {$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 tf td 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 nam out.nam &

exec xgraph -x time -y packets plot.xgr & exit 0

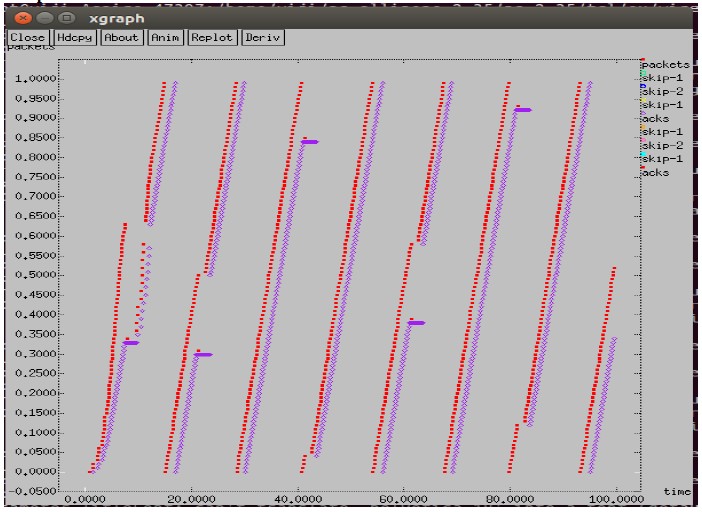
}

$ns at $stop "stop"

$ns run

## Output:

**Step 3: Run the program using command ns cdma.tcl**



## CDMA Trace File:

**Step 4: open the trace file using command gedit out.tr**

