**Networking Lab**

**Lab Assignment No 4**

**Name:** Iyer Sreekesh Subramanian **Roll No**. 24

**Aim**: To analyse the performance of a network for QoS (quality of service) parameters.

**Code and Output:**

1. **Wireless Simulation**

Code in wirelessQos.tcl:

#Wireless Network

#Setting values for variables in an associative array: val

#chan: Channel Type: Wireless

set val(chan) Channel/WirelessChannel ;

#prop: Radio Propagation Model: Two way propagation

set val(prop) Propagation/TwoRayGround ;

#netif: Network Interface Types: Wireless

set val(netif) Phy/WirelessPhy ;

#mac: MAC type: Cellular Communication

set val(mac) Mac/802\_11 ;

#ifq: Interface Queue type

set val(ifq) Queue/DropTail/PriQueue ;

#ll: Link Layer type

set val(ll) LL ;

#ant: Antenna Type

set val(ant) Antenna/OmniAntenna ;

#ifqlen: Interface queue length

set val(ifqlen) 50 ;

#nn: Number of nodes

set val(nn) 3 ;

#rp: ad-hoc routing protocol: Destination Sequenced Distance Vector (DSDV)

set val(rp) DSDV ;

#X, Y: Positions

set val(x) 500 ;

set val(y) 400 ;

#Stop: Stop time

set val(stop) 150 ;

#Create simulator object and link the trace files and nam trace

set ns [new Simulator]

set tracefd [open simple-dsdv.tr w]

set namtrace [open simwrls.nam w]

#Linking trace files to trace buffers

$ns trace-all $tracefd

$ns use-newtrace

$ns namtrace-all-wireless $namtrace $val(x) $val(y)

#Create topography flatgrid refers to movement in XY plane

set topo [new Topography]

$topo load\_flatgrid $val(x) $val(y)

#Create General Operations Director(GOD) object

#GOD object stores total number of mobile nodes & table of shortest hops required

create-god $val(nn)

#Configuring nodes

$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 \ # agentTrace: tracing at agent level turned ON/OFF

-routerTrace ON \ # routerTrace: tracing at router level turned ON/OFF

-macTrace OFF \ # macTrace: tracing at mac level turned ON/OFF

-movementTrace ON

#Creating three nodes

#Setting node positions, z=0 as topology is flatgrid

for {set i 0} {$i < $val(nn) } { incr i } {

set node\_($i) [$ns node]

}

#Setting node mobility

$node\_(0) set X\_ 5.0

$node\_(0) set Y\_ 5.0

$node\_(0) set Z\_ 0.0

$node\_(1) set X\_ 490.0

$node\_(1) set Y\_ 285.0

$node\_(1) set Z\_ 0.0

$node\_(2) set X\_ 150.0

$node\_(2) set Y\_ 240.0

$node\_(2) set Z\_ 0.0

# setdest params: X, Y, speed.

$ns at 10.0 "$node\_(0) setdest 250.0 250.0 3.0"

$ns at 15.0 "$node\_(1) setdest 45.0 285.0 5.0"

$ns at 110.0 "$node\_(0) setdest 480.0 300.0 5.0"

#Attaching transport layer protocol agents and application layer protocol agents

#to the nodes

#Setting source node, sink node

set tcp [new Agent/TCP/Newreno]

set sink [new Agent/TCPSink]

$ns attach-agent $node\_(0) $tcp

$ns attach-agent $node\_(1) $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ns at 10.0 "$ftp start"

#defines Node initial position. 30 is node size Must be called after mobility

for {set i 0} {$i < $val(nn)} { incr i } {

$ns initial\_node\_pos $node\_($i) 30

}

#Reset positions at stop

for {set i 0} {$i < $val(nn) } { incr i } {

$ns at $val(stop) "$node\_($i) reset";

}

#Start and stop wireless simulation

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "stop"

$ns at 150.01 "puts \"end simulation\" ; $ns halt"

#defining the stop procedure

proc stop {} {

#Flush trace buffers, close files and execute nam file

global ns tracefd namtrace

$ns flush-trace

close $tracefd

close $namtrace

exec nam simwrls.nam &

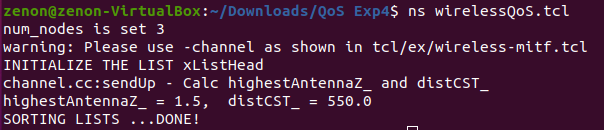
exit

}

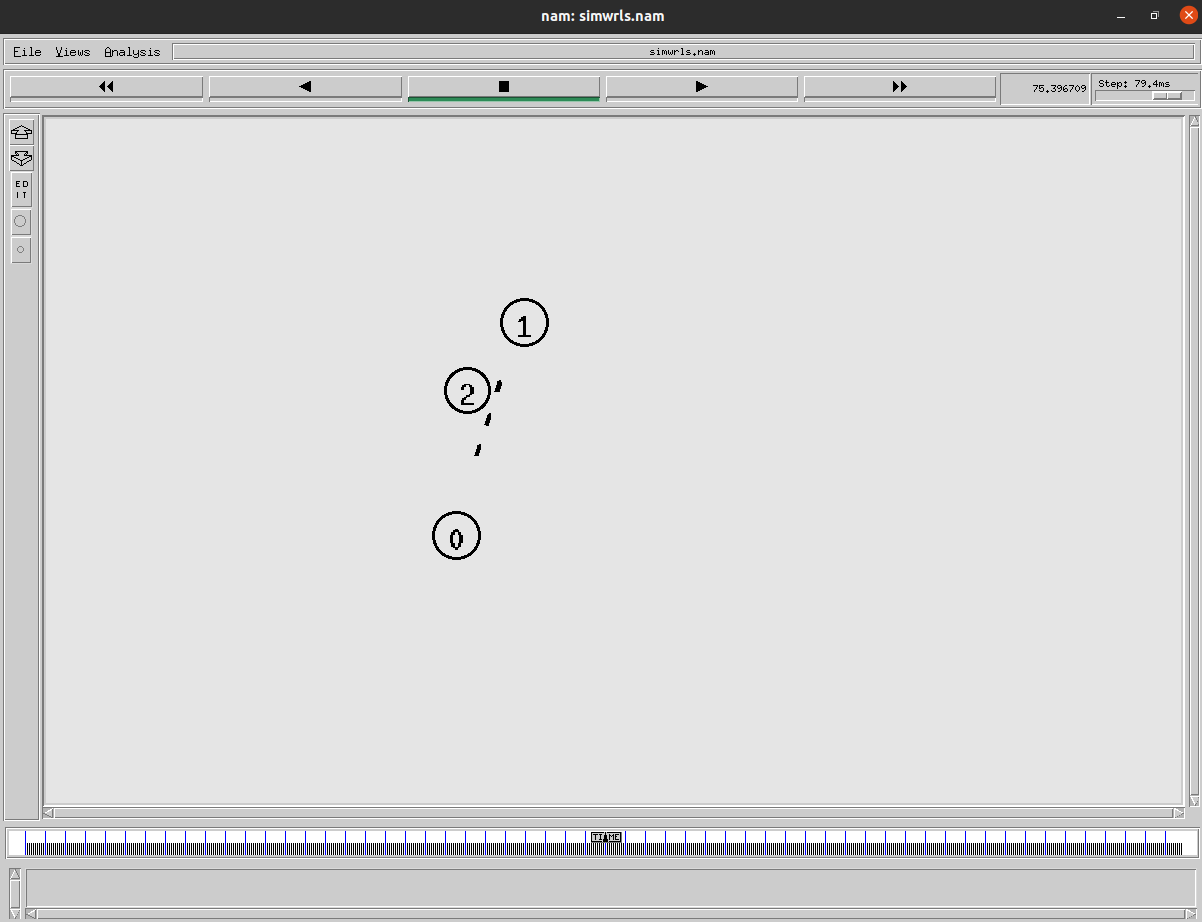
#running the simulation

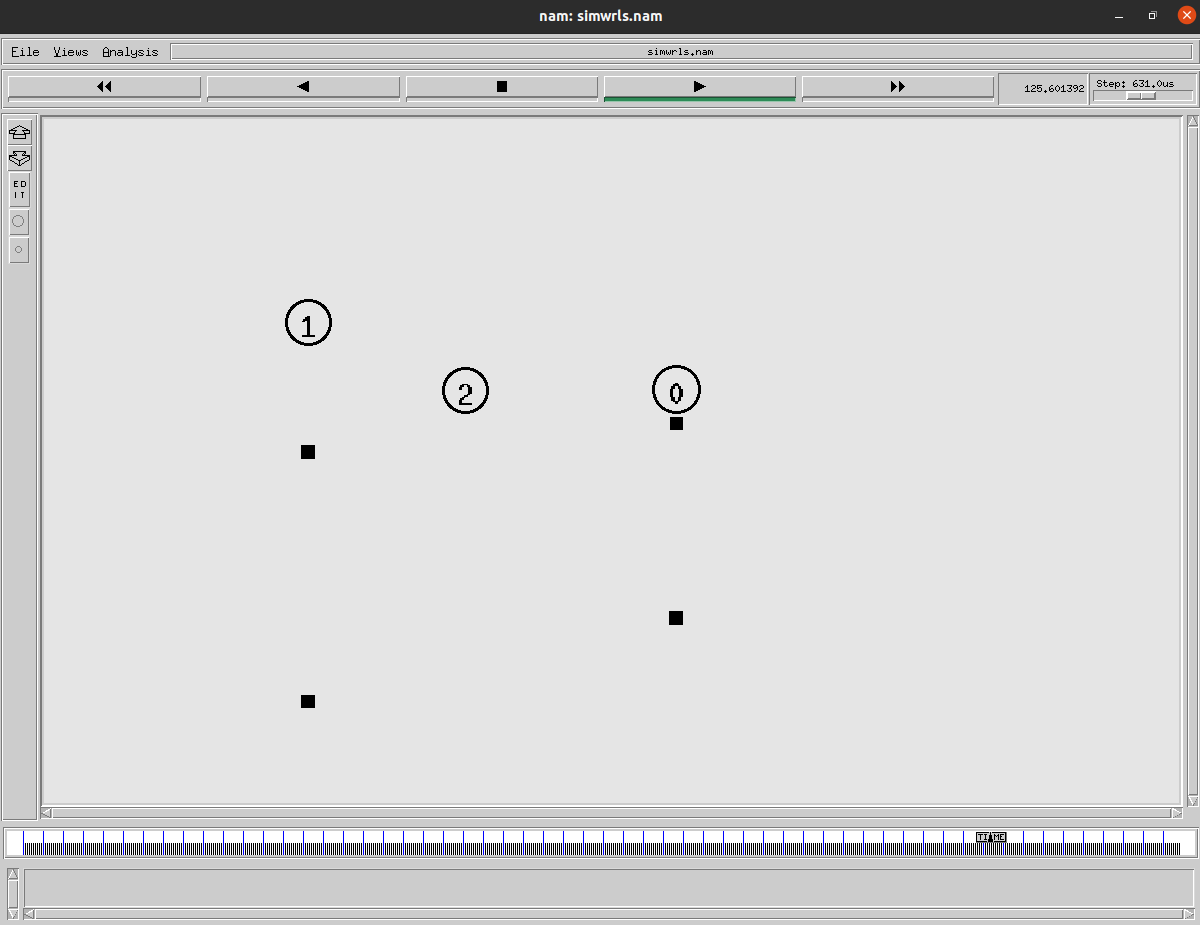
$ns run

**Output:**



**Screenshot of the NAM simulation**

****

****

1. **AWK Script to find Packet Delivery Ratio (PDR)**

Packet delivery ratio is defined as the ratio of data packets received by the destinations to those generated by the sources. Mathematically, it can be defined as:

PDR = S1 / S2

Where, S1 is the sum of data packets received by each destination and S2 is the sum of data packets generated by each source.

Code in pdr.awk:

BEGIN {

#initialising counter variables to count the

#number of sent and received packets

sendLine = 0;

recvLine = 0;

}

$0 ~/^s.\* AGT/ {

#incrementing sendLine counter if the line begins with s and has AGT in it

sendLine ++ ;

}

$0 ~/^r.\* AGT/ {

#incrementing recvLine counter if the line begins with r and has AGT in it

recvLine ++ ;

}

END {

#printing the counted values and the calculated PDR

printf "s:%d r:%d, Packet Delivery Ratio:%.4f \n", sendLine, recvLine, (recvLine/sendLine);

}

Output:



1. **AWK Script to find Throughput**

Network throughput is usually represented as an average and measured in bits per second (bps), or in some cases as data packets per second

Average throughput =

Code in throughput.awk:

BEGIN {

recvdSize = 0 #to store received packet size

startTime = 400 # high random start time

stopTime = 5 #low random stop time

}

{

#Analysing the trace file

event = $1 # send or received (s/r)

time = $3 # time of transaction (time of sending)

pkt\_size = $37 # packet size

level = $19 # application agent or routing protocol data (AGT/RTR)

# Find the starting time of simulation

if (level == "AGT" && event == "s" ) {

if (time < startTime){

startTime = time;

}

}

# Update total received packet size and store packets arrival time,

# to finally get the end time of the simulation

if (level == "AGT" && event == "r" ) {

if (time > stopTime) {

stopTime = time;

}

recvdSize += pkt\_size;

}

}

END {

# calculate the throughput and printing it

printf("Average Throughput[kbps] = %.2f\n",(recvdSize/(stopTime-startTime)))

}

Output:



1. **AWK Script to find total no of dropped packets**

Code in dropped.awk:

BEGIN {

#initialising counter variables to count the number of dropped packets

countDropped = 0;

}

$0 ~/^d/ {

#incrementing countDropped counter if the line begins with d

countDropped ++ ;

}

END {

#printing the counted number of dropped packets

printf "cbr Count of dropped packets: %d\n", countDropped;

}

Output:

