

SIMULATION-INTRODUCTION

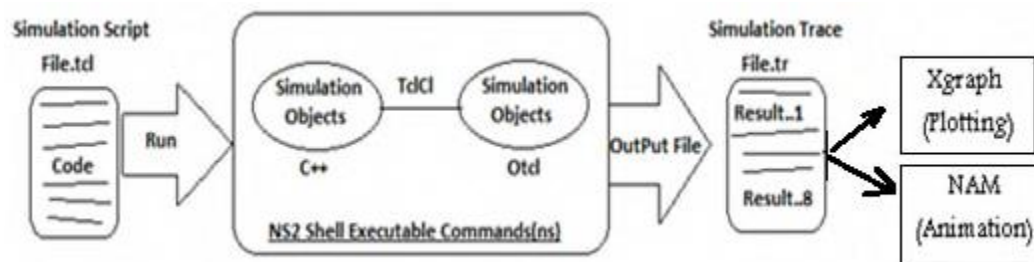
Network simulation is an important tool in developing, testing and evaluating network protocols. Simulation can be used without the target physical hardware, making it economical and practical for almost any scale of network topology and setup. It is possible to simulate a link of any bandwidth and delay, even if such a link is currently impossible in the real world. With simulation, it is possible to set each simulated node to use any desired software. This means that meaning deploying software is not an issue. Results are also easier to obtain and analyse, because extracting information from important points in the simulated network is as done by simply parsing the generated trace files.

Introduction to NS-2:

NS2 is an open-source simulation tool that runs on Linux. It is a discreet event simulator targeted at networking research and provides substantial support for simulation of routing, multicast protocols and IP protocols, such as UDP, TCP, RTP and SRM over wired and wireless (local and satellite) networks.

- Widely known as NS2, is simply an event driven simulation tool.
- Useful in studying the dynamic nature of communication networks.
- Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2.
- In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviours.

Basic Architecture of NS2



TCL – Tool Command Language

Tcl is a very simple programming language and it is a general purpose scripting language. [Interpreter]. Tcl runs on most of the platforms such as Unix, Windows, and Mac. The strength of Tcl is its simplicity. It is not necessary to declare a data type for variable prior to the usage.

Syntax of TCL

command arg1 arg2 arg3

Tcl scripts are made up of commands separated by newlines or semicolons. Commands all have the same basic form illustrated by the following example:

expr 20 + 10

This command computes the sum of 20 and 10 and returns the result, 30. You can try out this example and all the others in this page by typing them to a Tcl application such as tclsh; after a command completes, tclsh prints its result.

Each Tcl command consists of one or more words separated by spaces. In this example there are four words: expr, 20, +, and 10. The first word is the name of a command and the other words are arguments to that command. All Tcl commands consist of words,

but different commands treat their arguments differently.

The expr command treats all of its arguments together as an arithmetic expression, computes the result of that expression, and returns the result as a string. In the expr command the division into words isn't significant: you could just as easily have invoked the same command as

expr 20+10

However, for most commands the word structure is important, with each word used for a distinct purpose.

All Tcl commands return results. If a command has no meaningful result, then it returns an empty string as its result

Variables

Tcl allows you to store values in variables and use the values later in commands. The set command is used to write and read variables.

set a 5 set len [string length foobar]

set b \$a set len [expr [string length foobar] + 9]

The command returns the new value of the variable. You can read the value of a variable by invoking set with only a single argument:

set a(variable name)

Control structures

Tcl provides a complete set of control structures including commands for conditional execution, looping, and procedures. Tcl control structures are just commands that take Tcl scripts as arguments. The example below creates a Tcl procedure called `power`, which raises a base to an integer power:

```
proc power {base p} {  
    set result 1  
    while {$p > 0} {  
        set result [expr $result * $base]  
        set p [expr $p - 1]  
    }  
    return $result  
}
```

This script consists of a single command, `proc`. The `proc` command takes three arguments: the name of a procedure, a list of argument names, and the body of the procedure, which is a Tcl script. Note that everything between the curly brace at the end of the first line and the curly brace on the last line is passed verbatim to `proc` as a single argument. The `proc` command creates a new Tcl command named `power` that takes two arguments. You can then invoke `power` with commands like the following:

```
power 2 6  
power 1.15 5
```

When `power` is invoked, the procedure body is evaluated. While the body is executing it can access its arguments as variables: `base` will hold the first argument and `p` will hold the second.

The body of the `power` procedure contains three Tcl commands: `set`, `while`, and `return`. The `while` command does most of the work of the procedure. It takes two arguments, an expression (`$p > 0`) and a body, which is another Tcl script.

The `while` command evaluates its expression argument using rules similar to those of the C programming language and if the result is true (nonzero) then it evaluates the body as a Tcl script. It repeats this process over and over until eventually the expression evaluates to false (zero). In this case the body of the `while` command multiplied the result value by `base` and then decrements `p`.

When `p` reaches zero the result contains the desired power of `base`. The `return` command causes the procedure to exit with the value of variable `result` as the procedure's result.

Initialization and Termination of TCL Script in NS-2

An ns simulation starts with the command

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

This line declares a new variable as using the set command, you can call this variable as you wish, in general people declares it as ns because it is an instance of the Simulator class, so an object the code[new Simulator] is indeed the installation of the class Simulator using the reserved word new. In order to have output files with data on the simulation (**trace files**) or files used for visualization (**nam files**), we need to create the files using **open** command:

#Open the Trace file

```
set tracefile1 [open out.tr w]
$ns trace-all $tracefile1
```

#Open the NAM trace file

```
set namfile [open out.nam w]
$ns namtrace-all $namfile
```

The above creates a trace file called “**out.tr**” and a nam visualization trace file called “**out.nam**”. Within the tcl script, these files are not called explicitly by their names, but instead by pointers that are declared above and called “**tracefile1**” and “**namfile**” respectively. Remark that they begin with a # symbol. The second line open the file “**out.tr**” to be used for writing, declared with the letter “w”. The third line uses a simulator method called trace-all that have as parameter the name of the file where the traces will go. The last line tells the simulator to record all simulation traces in NAM input format. It also gives the file name that the trace will be written to later by the command **\$ns flush-trace**. In our case, this will be the file pointed at by the pointer “**\$namfile**”, i.e. the file “**out.tr**”. The termination of the program is done using a “**finish**” procedure.

#Define a “finish” procedure

```
proc finish { } {
    global ns tracefile1 namfile
    $ns flush-trace
    Close $tracefile1
    Close $namfile
    Exec nam out.nam &
    Exit 0
}
```

The word proc declares a procedure in this case called finish and without arguments. The word global is used to tell that we are using variables declared outside the procedure. The simulator method “flush-trace will dump the traces on the respective files. The tcl command “**close**” closes the trace files defined before and “**exec**” executes the nam program for visualization. The command exit will end the application and return the number 0 as status to the system. Zero is the default for a clean exit. Other values can be used to say that is a exit because something fails. At the end of ns program, we should call the procedure “**finish**” and specify at what time the termination should occur. For example, will be used to call “**finish**” at time 125sec. Indeed, the at method of the simulator allows us to schedule events explicitly. The simulation can then begin using the command

Definition of a network of links and nodes The way to define a node is

The node is created which is printed by the variable **n0**. When we shall refer to that node in the script we shall thus write **\$n0**. Once we define several nodes, we can define the links that connect them. An example of a definition of a link is:

Which means that **\$n0** and **\$n2** are connected using a bi-directional link that has 10ms of propagation delay and a capacity of 10Mb per sec for each direction. To define a directional link instead of a bi-directional one, we should replace “**duplexlink**” by “**simplex-link**”. In NS, an output queue of a node is implemented as a part of each link whose input is that node. The definition of the link then includes the way to handle overflow at that queue. In our case, if the buffer capacity of the output queue is exceeded then the last packet to arrive is dropped. Many alternative options exist, such as the RED (Random Early Discard)

```
$ns at 125.0 finish
```

```
$ns run
```

```
set n0 [$ns node]
```

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

Mechanism, the FQ (Fair Queuing), the DRR (Deficit Round Robin), the stochastic Fair Queuing (SFQ) and the CBQ (which including a priority and a round-robin scheduler). In ns, an output queue of a node is implemented as a part of each link whose input is that node. We should also define the buffer capacity of the queue related to each link. An example would be:

Agents and Applications We need to define routing (sources, destinations) the agents (protocols) the application that use them.

FTP over TCP TCP is a dynamic reliable congestion control protocol. It uses Acknowledgements created by the destination to know whether packets are well received. There are number variants of the TCP protocol, such as Tahoe, Reno, NewReno, Vegas. The type of agent appears in the first line: The command **\$ns attach-agent \$n0 \$tcp** defines the source node of the tcp connection. The command defines the behaviour of the destination node of TCP and assigns to it a pointer called sink.

```
#Setup a UDP connection
```

```
#set Queue Size of link (n0-n2) to 20
```

```
$ns queue-limit $n0 $n2 20
```

```
set tcp [new Agent/TCP]
```

```
set sink [new Agent /TCPSink]
```

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

```
$ns attach-agent $n1 $udp
```

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

```
$ns attach-agent $n5 $null
```

```
$ns connect $udp $null
```

\$udp set fid_2

```
#setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set packetSize_ 100
$cbr set rate_ 0.01Mb
$cbr set random_ false
$cbr set interval_ 0.005
```

Above shows the definition of a CBR application using a UDP agent. The command `$ns attach-agent $n4 $sink` defines the destination node. The command `$ns connect $tcp $sink` finally makes the TCP connection between the source and destination nodes. TCP has many parameters with initial fixed defaults values that can be changed if mentioned explicitly. For example, the default TCP packet size has a size of 1000bytes. This can be changed to another value, say 552bytes, using the command `$tcp set packetSize_ 552`. When we have several flows, we may wish to distinguish them so that we can identify them with different colours in the visualization part. This is done by the command `$tcp set fid_ 1` that assigns to the TCP connection a flow identification of “1”. We shall later give the flow identification of “2” to the UDP connection.

CBR over UDP A UDP source and destination is defined in a similar way as in the case of TCP. Instead of defining the rate in the command `$cbr set rate_ 0.01Mb`, one can define the time interval between transmission of packets using the command.

The packet size can be set to some value using

```
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set packetSize_ 100
$cbr set rate_ 0.01Mb
$cbr set random_ false
$cbr set interval_ 0.005

$cbr set packetSize_ <packet size>
```

Scheduling Events NS is a discrete event based simulation. The tcp script defines when event should occur. The initializing command `set ns [new Simulator]` creates an event scheduler, and events are then scheduled using the format:

The scheduler is started when running ns that is through the command `$ns run`. The beginning and end of the FTP and CBR application can be done through the following command Structure of Trace Files When tracing into an output ASCII file, the trace is organized in 12 fields as follows in fig shown below, the meaning of the fields is: Event Time from Node to Node PKT Type PKT Size Flags Fid Src Addr Dest Addr Seq Num Pkt id

1. The first field is the event type. It is given by one of four possible symbols r, +, -, d which correspond respectively to receive (at the output of the link), enqueued, dequeued and dropped.
2. The second field gives the time at which the event occurs.
3. Gives the input node of the link at which the event occurs.
4. Gives the output node of the link at which the event occurs.
5. Gives the packet type (e.g. CBR or TCP)
6. Gives the packet size
7. Some flags
8. This is the flow id (fid) of IPv6 that a user can set for each flow at the input
OTcl script one can further use this field for analysis purposes; it is also used when specifying stream color for the NAM display.

\$ns at <time><event>

\$ns at 0.1 \$cbr start

\$ns at 1.0 \$ftp start

\$ns at 124.0 \$ftp stop

\$ns at 124.5 \$cbr stop

9. This is the source address given in the form of “node. Port”.
10. This is the destination address, given in the same form.
11. This is the network layer protocol’s packet sequence number. Even though UDP implementations in a real network do not use sequence number, ns keeps track of UDP packet sequence number for analysis purposes
12. The last field shows the Unique id of the packet.

NS2 Scenarios Generator (NSG) Tool:

NS2 Scenarios Generator (NSG) is a tcl script generator tool used to generate TCL Scripts automatically!!!

NSG is a Java based tool that runs on any platform and can generate TCL Scripts for Wired as well as Wireless Scenarios for Network Simulator - 2. The procedure to execute these TCL Scripts on NS-2 is same as those of manually written TCL Scripts.

To start NSG2 open terminal

java -jar NSG2.jar

it opens a graphical user interface on top of the window click on scenario, select the type of connection (new wired scenario/wireless scenario).

To create *node* click on node tab and click on the screen, you can create any no of nodes as per the need.

PROGRAM 1

TITLE

Implement Three nodes point – to – point network with duplex links between them for different topologies. 1Set the queue size, vary the bandwidth, and find the number of packets dropped for various iterations.

AIM

Analyze the traffic between the nodes using different bandwidth, propagation delay and queue size of point to point duplex link and its effects on packet transmission.

DESCRIPTION

Ns2(network simulator2) is used for this experiment, in which point to point duplex link is created between the node with varying queuing capacity of node. Bandwidth, propagation delay of a point to point link and queuing capacity of a node is very important to minimize the affect on packet transmission. Ns2 simulated data traffic analyzed by setting the different bandwidth, propagation delay and queuing capacity of a link and correspondingly its effect on packet transmission is noted. Unix grep commands are used for analyzing the out.tr trace log file generated upon executing the ns2 simulation script for determining the packet drops.

INPUT

- (a) Different Bandwidth, propagation delay & queuing capacity of node of duplex link
- (b) Unix grep commands for analyzing the out.tr trace log file for determining the number of packet drop.

EXPECTED OUTPUT

- (a) generate the out.tr trace log file and out.nam network animation file.
- (b) number of packet drop

STEPS

- To start NSG2 open terminal
java -jar NSG2.jar

Create three nodes by clicking on the screen by selecting the node
click on the link tab =>select link type duplex link
=>select queue type Drop Tail

=>enter capacity(bandwidth), propagation delay, queue size
=>select two nodes to create a link between them

- Click on agent tab
 - for the source node=>select agent type “TCP”
=>enter packet size in bytes

=>click on the source node (N0) and drag

- for the destination node=>select agent type “TCP Sink”
=>enter packet size in bytes
=>click on the destination node (N2) and drag
- Connect source to destination node by dragging TCP to TCP Sink (Virtual connection)
- Click on application tab =>select application type(FTP)
=>enter start time and stop time
- Click on TCP and then drag(FTP0)
- Click on parameter tab =>enter simulation time, trace file and nam file
- Click on TCL tab to generate code
- Save the code with the file name with extension .tcl
- Run the code in terminal by typing **ns filename.tcl**
- It would generate an animated topology window where transmission of packets between nodes can be viewed
- Now press the play button in topology window and the simulation begins
- To find the no of packets dropped type the following code in terminal
grep -c “^d” filename.tr
- To see the trace file contents open the file as
gedit filename.tr
- Trace file contains 12 columns:
- Event type, Event time, From node, To node, Packet type, Packet size, Flax, Flow id, Source address, Destination address, Sequence id, Packet id

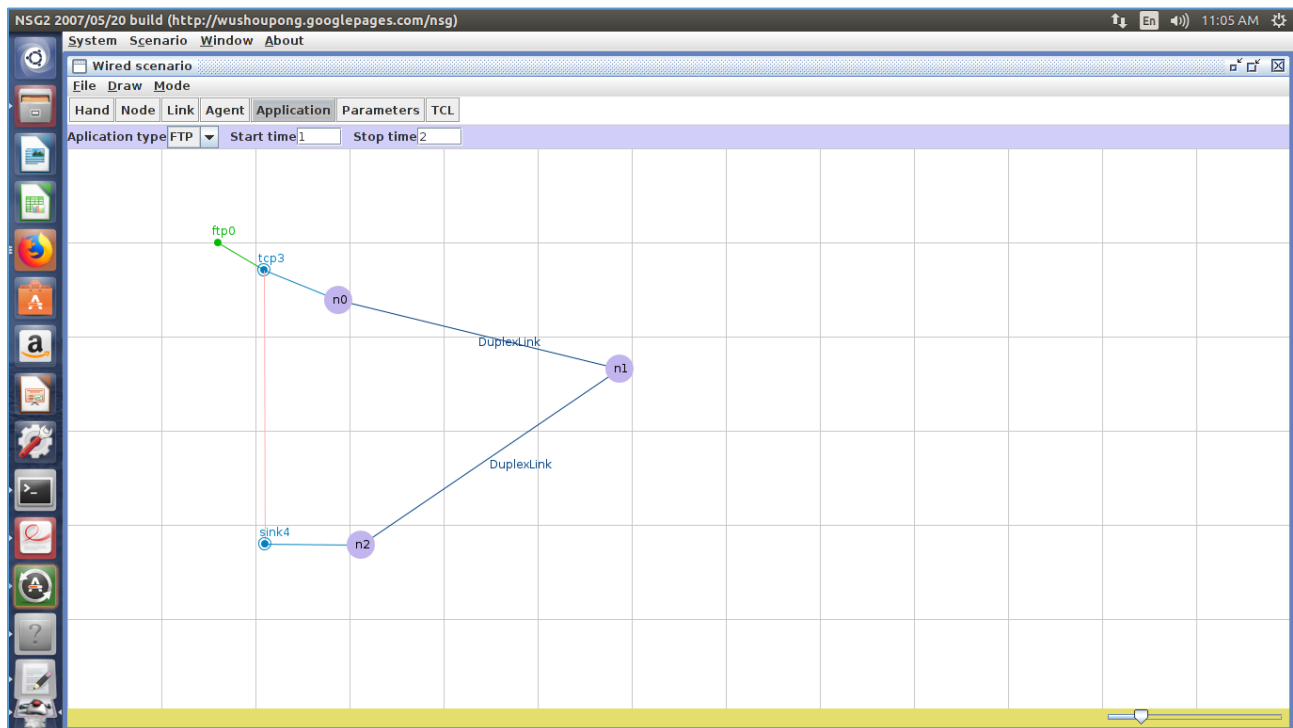


Figure 1: NSG2 code generator screen

PROGRAM

Prg1.tcl

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

```
set tracefile [open prog1.tr w]
$ns trace-all $tracefile
```

```
set namfile [open prog1.nam w]
$ns namtrace-all $namfile
```

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

```
$ns duplex-link $n0 $n1 100Mbps 10ms DropTail
$ns queue-limit $n0 $n1 5
$ns duplex-link $n1 $n2 100Mbps 10ms DropTail
$ns queue-limit $n1 $n2 3
```

```
$ns duplex-link-op $n0 $n1 orient right-down
$ns duplex-link-op $n1 $n2 orient left-down
```

```
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n2 $sink
$ns connect $tcp $sink
$tcp set PacketSize_ 2500
```

```
set ftp [new Application/FTP]
$ftp attach-agent $tcp
```

```
$ns at 0.5 "$ftp start"
$ns at 2.0 "$ftp stop"
$ns at 2.5 "Finish"
```

```
proc Finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam prog1.nam &
    exit 0
}
puts "simulation starts..."
```

```
$ns run
```

RESULT

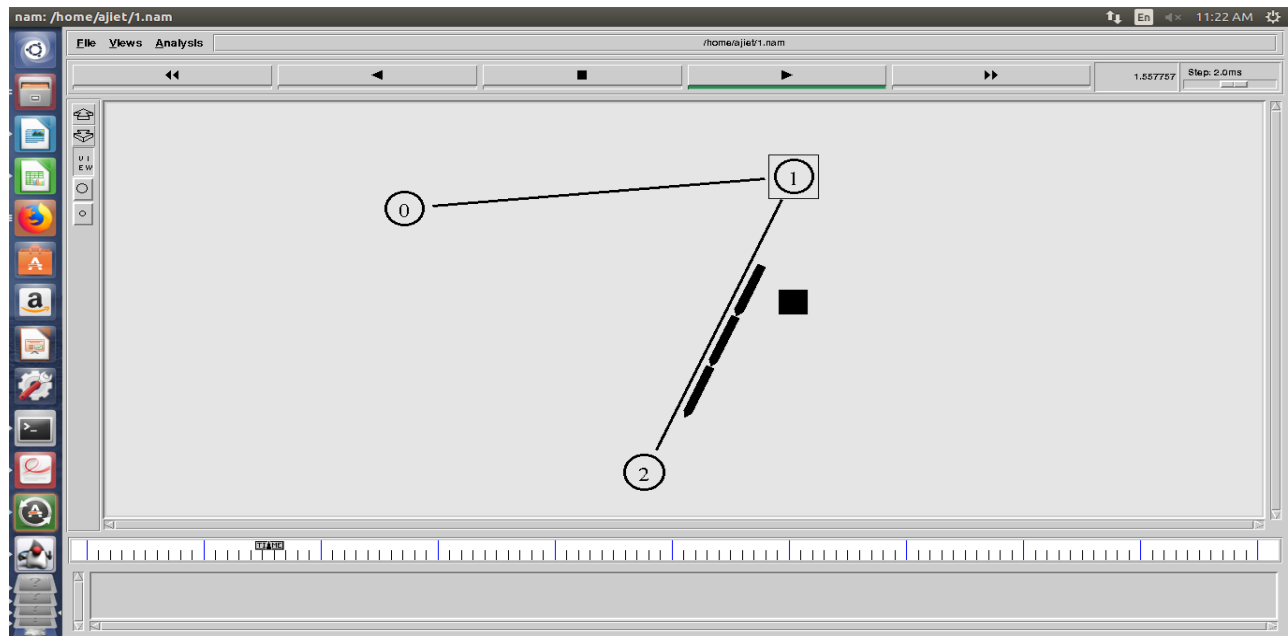


Figure 2: NAM output animation

VIVA QUESTIONS AND ANSWERS**1. Which are the different types of networks?**

Personal Area **Network** (PAN)
Local Area **Network** (LAN)
Wireless Local Area **Network** (WLAN)
Campus Area **Network** (CAN)
Metropolitan Area **Network** (MAN)
Wide Area **Network** (WAN)
Storage-Area **Network** (SAN)

2. What is network topology?

Network topology is the arrangement of the elements (links, nodes, etc.) of a communication network

3. Which are the different types of network topologies?

Point-to-point topology is the simplest of all the network topologies. There is a direct link between two computers that want to communicate

Bus topology all the nodes are connected to one main cable which acts as a backbone

Star topology each computer is connected to a central hub using a point-to-point connection.

Ring topology the computers in the network are connected in a circular fashion, and the data travels in one direction.

Mesh topology every node has a direct point to point connection to every other node

4. Which are the different modes of data transmission?

Simplex: In this type of transmission mode, data can be sent only in one direction

Half duplex: data can be transmitted in both directions, but not at the same time.

Full duplex: data can be sent in both directions simultaneously.

5. What is a packet?

A *packet* is the unit of data that is routed between an origin and a destination on the Internet or any other *packet-switched network*.

6. Define jitter?

Jitter is defined as a variation in the delay of received packets. At the sending side, packets are sent in a continuous stream with the packets spaced evenly apart. Due to network congestion, improper queuing, or configuration errors, this steady stream can become lumpy, or the delay between each packet can vary instead of remaining constant.

PROGRAM 2

TITLE

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

AIM

To understand how Extended Service Set is created and works by providing services to nodes in wireless LAN and analyzing the wireless traffic for determining packet drops.

DESCRIPTION

In this experiment, network simulator2 is used to create a IEEE 802.11 Wireless LAN consisting of mobile nodes and Extended Service set. An extended service set (ESS) is one or more interconnected basic service sets (BSSs) and their associated LANs. Each BSS consists of a single access point (AP) together with all wireless client devices (stations, also called STAs) creating a local or enterprise 802.11 wireless LAN (WLAN). Wireless mobile nodes

INPUT

- Two TCP traffic between pair of nodes
- Code to analyze the out.tr trace log file for determining the number of packets dropped.

EXPECTED OUTPUT

- Generate the out.tr trace log file, and
- Generate the out.nam network animation file depicting the IEEE 802.11 Wireless LAN.

STEPS

- Open NSG2 → select the type of connection -wireless scenario
- Create n nodes by clicking on the screen by selecting the node
- Click on agent tab
 - for the source node => select agent type “UDP”
=> enter packet size in bytes
=> click on the source node and drag
 - for the destination node => select agent type “NULL”
=> enter packet size in bytes
=> click on the destination node and drag
- Create TCP connections between source(n5) and destination(n3)
- Connect source to destination node by dragging UDP to NULL (Virtual connection)
- Click on application tab => select application type(CBR)
=> enter start time and stop time
- Click on parameter tab => enter simulation time, trace file and nam file
- Click on TCL tab to generate code
- Save the code with the file name with extension .tcl
- Edit the highlighted code

- Run the code in terminal by typing **ns filename.tcl**
- It would generate an animated topology window where transmission of packets between nodes can be viewed

Now press the play button in topology window and the simulation begins

PROGRAM

```
# This script is created by NSG2 beta1
# <http://wushoupong.googlepages.com/nsg>

#=====
#   Simulation parameters setup
#=====
set val(chan) Channel/WirelessChannel ;# channel type
set val(prop) Propagation/TwoRayGround ;# radio-propagation model
set val(netif) Phy/WirelessPhy ;# network interface type
set val(mac) Mac/802_11 ;# MAC type
set val(ifq) Queue/DropTail/PriQueue ;# interface queue type
set val(ll) LL ;# link layer type
set val(ant) Antenna/OmniAntenna ;# antenna model
set val(ifqlen) 50 ;# max packet in ifq
set val(nn) 6 ;# number of mobilenodes
set val(rp) DSDV ;# routing protocol
set val(x) 810 ;# X dimension of topography
set val(y) 600 ;# Y dimension of topography
set val(stop) 10.0 ;# time of simulation end

#=====
#   Initialization
#=====
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo [new Topography]
$topo load flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel

#=====
#   Mobile node parameter setup
```

```
#=====
$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) \
    -channel     $chan \
    -topoInstance $topo \
    -agentTrace  ON \
    -routerTrace ON \
    -macTrace    ON \
    -movementTrace ON
```

```
#=====
#   Nodes Definition
#=====
#Create 6 nodes
set n0 [$ns node]
$ns0 set X_ 270
$ns0 set Y_ 408
$ns0 set Z_ 0.0
$ns initial_node_pos $ns0 20
set n1 [$ns node]
$ns1 set X_ 503
$ns1 set Y_ 404
$ns1 set Z_ 0.0
$ns initial_node_pos $ns1 20
set n2 [$ns node]
$ns2 set X_ 710
$ns2 set Y_ 416
$ns2 set Z_ 0.0
$ns initial_node_pos $ns2 20
set n3 [$ns node]
$ns3 set X_ 621
$ns3 set Y_ 191
$ns3 set Z_ 0.0
$ns initial_node_pos $ns3 20
set n4 [$ns node]
$ns4 set X_ 402
$ns4 set Y_ 176
$ns4 set Z_ 0.0
$ns initial_node_pos $ns4 20
set n5 [$ns node]
$ns5 set X_ 249
$ns5 set Y_ 174
$ns5 set Z_ 0.0
$ns initial_node_pos $ns5 20
```

```
#=====
#   Configure mobile nodes           x, y speed
#=====
$ns at 1.5 "$n1 setdest 390.0 460.0 40.0"
$ns at 1.5 "$n4 setdest 472.0 510.0 50.0"
$ns at 1.5 "$n5 setdest 523.0 570.0 40.0"

#=====
#   Agents Definition
#=====
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n5 $sink1
$ns connect $tcp0 $sink1
$tcp0 set packetSize_ 1500

#Setup a UDP connection
set udp2 [new Agent/UDP]
$ns attach-agent $n2 $udp2
set null3 [new Agent/Null]
$ns attach-agent $n3 $null3
$ns connect $udp2 $null3
$udp2 set packetSize_ 1500

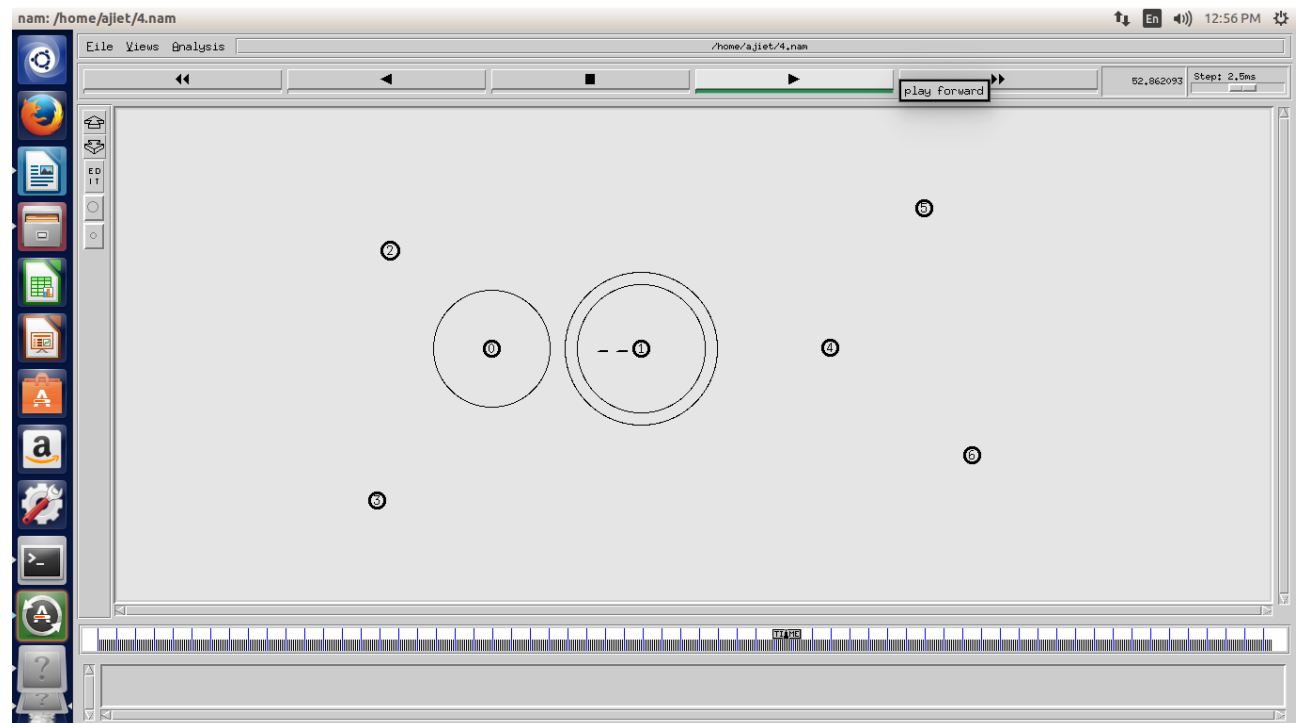
#=====
#   Applications Definition
#=====
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 2.0 "$ftp0 stop"

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp2
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
$ns at 1.0 "$cbr1 start"
$ns at 2.0 "$cbr1 stop"

#=====
#   Termination
#=====
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
```



```
$ns flush-trace
close $tracefile
close $namfile
exec nam out.nam &
exit 0
}
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns at $val(stop) "\n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```



To find the number of packets sent:
Grep “^r” out.tr | grep -c “_3_ AGT”

VIVA QUESTIONS AND ANSWERS

1. What is a wireless LAN?

A **wireless local area network (WLAN)** is a wireless computer network that links two or more devices using wireless communication within a limited area such as a home, school, computer laboratory, or office building.

2. Define Access Point in wireless network?

Access points (APs), normally wireless routers, are base stations for the wireless network. They transmit and receive radio frequencies for wireless enabled devices to communicate with.

3. Define BSS?

The basic service set (BSS) is a set of all stations that can communicate with each other at PHY layer. Every BSS has an identification (ID) called the BSSID, which is the MAC address of the access point servicing the BSS.

There are two types of BSS: Independent BSS (also referred to as IBSS), and infrastructure BSS. An independent BSS (IBSS) is an ad hoc network that contains no access points, which means they cannot connect to any other basic service set.

4. Define ESS?

An extended service set (ESS) is a set of connected BSSs. Access points in an ESS are connected by a distribution system. Each ESS has an ID called the SSID which is a 32-byte (maximum) character string.

5. Describe Distribution System

A distribution system (DS) connects access points in an extended service set.

The concept of a DS can be used to increase network coverage through roaming between cells. DS can be wired or wireless. Current wireless distribution systems are mostly based on WDS or MESH protocols, though other systems are in use.

6. Which are the two types of wireless networks?

Infrastructure and **ad hoc** mode. In *ad hoc* mode, mobile units transmit directly peer-to-peer. In infrastructure mode, mobile units communicate through an access point that serves as a bridge to other networks

PROGRAM 3

TITLE

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

DESCRIPTION

CRC generator using polynomials

If we consider the data unit 1001 and divisor or polynomial generator 1011 their polynomial representation is:

Data	1001	$x^3 + 1$	
Division (polynomial generator)	1011	$x^3 + x + 1$	

Divisor
 $x^3 + x + 1$

$$\begin{array}{r}
 x^3 + x \\
 \hline
 x^6 + + x^3 \quad \leftarrow \text{Dividend} \\
 x^6 + x^4 + x^3 \\
 \hline
 x^4 \\
 x^4 + x^2 + x \\
 \hline
 x^2 + x
 \end{array}$$

→ Remainder (degree is less than that of divisor)

$x^6 + x^3$	$x^2 + x$
Data	Remainder

CRC division using polynomial

- Now string of n 0s (one less than that of divisor) is appended to data. Now data is 1001000 and its corresponding polynomial representation is $x^6 + x^3$.
- The division of $x^6 + x^3$ by $x^3 + x + 1$ is shown in fig.
- The polynomial generator should have following properties:
 1. It should have at least two terms.
 2. The coefficient of the term x^0 should be 1.
 3. It should not be divisible by x .
 4. It should be divisible by $x + 1$.

Some Standard Generator Polynomials are shown below:

Name	Generator Polynomial
CRC – 8	$x^8 + x^2 + x + 1$
CRC – 10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$
CRC - 16	$x^{16} + x^{12} + x^5 + 1$

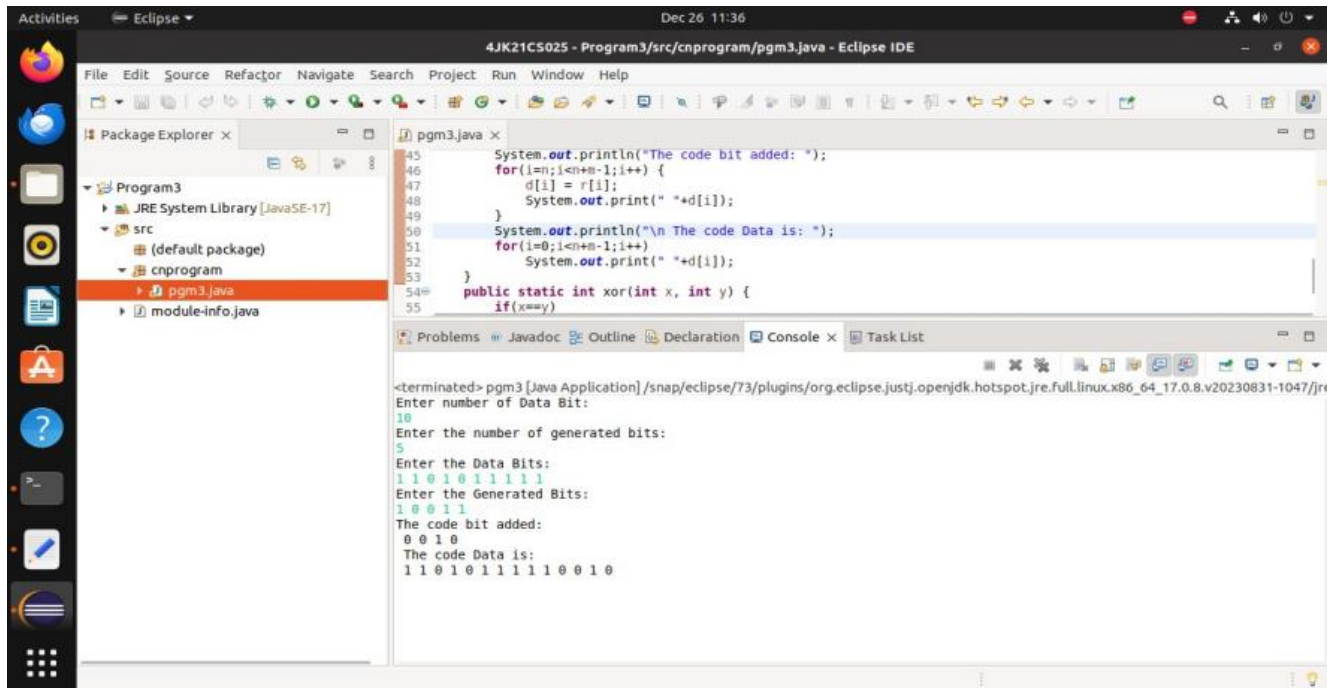
PROGRAM

```
import java.util.Scanner;
class crc{
public static void main(String args[])
{
    int i,j,k;
    int[] r;
    int[] z;
    Scanner sc=new Scanner(System.in);
    System.out.print("enter no.of Data Bit:");
    int n=sc.nextInt();
    System.out.println("enter the no.of generator Bits:");
    int m=sc.nextInt();
    int[] d=new int[n+m];
    int[] g=new int[m];
    System.out.println("Enter the Data Bits");
    for(i=0;i<n;i++)
        d[i]=sc.nextInt();
    System.out.println("Enter the Generator Bits");
    for(j=0;j<m;j++)
        g[j]=sc.nextInt();
    for(i=0;i<m-1;i++)
        d[n+i]=0;
    r=new int[m+n];
    z=new int[m];
    for(i=0;i<m;i++)
    {
        r[i]=d[i];
        z[i]=0;
    }
    for(i=0;i<n;i++)
    {
        k=0;
        int msb=r[i];
        for(j=i;j<m+i;j++)
        {
            if(msb==0)
                r[j]=xor(r[j],z[k]);
            else
                r[j]=xor(r[j],g[k]);
        }
    }
}
```

```
                k++;
            }
            r[m+i]=d[m+i];
        }
        System.out.print("the Code bit added : ");
        for(i=n;i<n+m-1;i++)
        {
            d[i]=r[i];
            System.out.print(d[i]);
        }
        System.out.println("the Code Data is : ");
        for(i=0;i<n+m-1;i++)
            System.out.print(d[i]);
    }

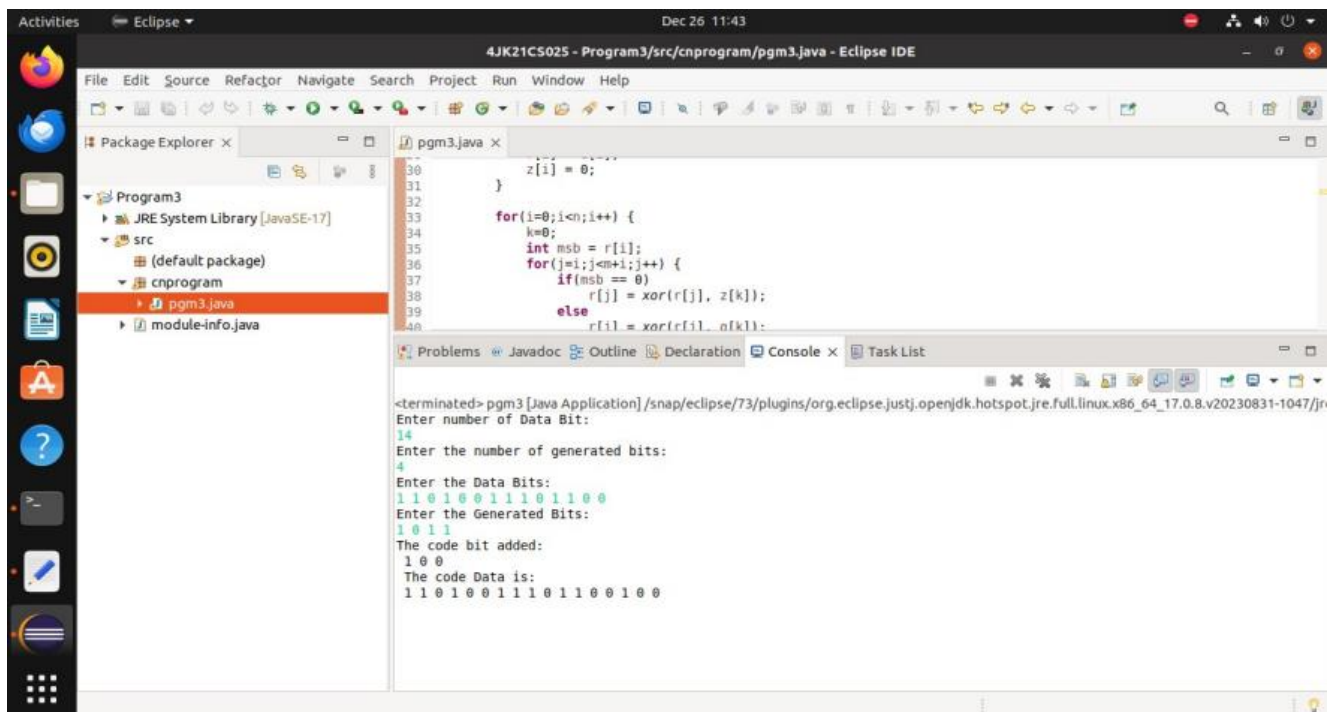
    public static int xor(int x,int y)
    {
        if(x==y)
            return(0);
        else
            return(1);
    }
}
```

RESULT



```
4JK21CS025 - Program3/src/cnprogram/pgm3.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer x
Program3
  JRE System Library [JavaSE-17]
  (default package)
  cnprogram
    pgm3.java
    module-info.java
pgm3.java x
45 System.out.println("The code bit added: ");
46 for(i=0;i<n+m-1;i++) {
47     d[i] = r[i];
48     System.out.print(" *d[i]);
49 }
50 System.out.println("\n The code Data is: ");
51 for(i=0;i<n+m-1;i++)
52     System.out.print(" *d[i]);
53 }
54 public static int xor(int x, int y) {
55     if(x==y)
```

```
<terminated> pgm3 [Java Application] /snap/eclipse/73/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.8.v20230831-1047/jr
Enter number of Data Bit:
10
Enter the number of generated bits:
5
Enter the Data Bits:
1 1 0 1 0 1 1 1 1 1
Enter the Generated Bits:
1 0 0 1 1
The code bit added:
0 0 1 0
The code Data is:
1 1 0 1 0 1 1 1 1 1 0 0 1 0
```



```
4JK21CS025 - Program3/src/cnprogram/pgm3.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer x
Program3
  JRE System Library [JavaSE-17]
  (default package)
  cnprogram
    pgm3.java
    module-info.java
pgm3.java x
30     z[i] = 0;
31 }
32
33 for(i=0;i<n;i++) {
34     k=0;
35     int msb = r[i];
36     for(j=i;j<m+1;j++) {
37         if(msb == 0)
38             r[j] = xor(r[j], z[k]);
39         else
40             r[j] = xor(r[j], n[k]);
```

```
<terminated> pgm3 [Java Application] /snap/eclipse/73/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.8.v20230831-1047/jr
Enter number of Data Bit:
14
Enter the number of generated bits:
4
Enter the Data Bits:
1 1 0 1 0 0 1 1 1 0 1 1 0 0
Enter the Generated Bits:
1 0 1 1
The code bit added:
1 0 0
The code Data is:
1 1 0 1 0 0 1 1 1 0 1 1 0 0 1 0 0
```

VIVA QUESTION AND ANSWERS**1. What are the types of errors?**

- a. Single-Bit error: In a single-bit error, only one bit in the data unit has changed
- b. Burst Error: A Burst error means that two or more bits in the data have changed.

2. What is Error Detection? What are its methods?

Data can be corrupted during transmission. For reliable communication errors must be deducted and corrected. Error Detection uses the concept of redundancy, which means adding extra bits for detecting errors at the destination. The common Error Detection methods are

- a. Vertical Redundancy Check (VRC)
- b. Longitudinal Redundancy Check (VRC)
- c. Cyclic Redundancy Check (VRC)
- d. Checksum

3. What is CRC?

CRC, is the most powerful of the redundancy checking techniques, is based on binary division.

4. Compare Error Detection and Error Correction:

The correction of errors is more difficult than the detection. In error detection, checks only any error has occurred. In error correction, the exact number of bits that are corrupted and location in the message are known. The number of the errors and the size of the message are important factors.

5. What is Forward Error Correction?

Forward error correction is the process in which the receiver tries to guess the message by using redundant bits.

6. Define Retransmission?

Retransmission is a technique in which the receiver detects the occurrence of an error and asks the sender to resend the message. Resending is repeated until a message arrives that the receiver believes is error-free.

7. What are Cyclic Codes?

Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword.

8. Define Encoder?

A device or program that uses predefined algorithms to encode, or compress audio or video data for storage or transmission use. A circuit that is used to convert between digital video and analog video.

9. Define Decoder?

A device or program that translates encoded data into its original format (e.g. it decodes the data). The term is often used in reference to MPEG-2 video and sound data, which must be decoded before it is output.

10. What is Bit Stuffing?

Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.

11. What is Error Control?

Error control is both error detection and error correction. It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. In the data link layer, the term error control refers primarily to methods of error detection and retransmission.

PROGRAM 4

TITLE

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.

AIM

To understand the working principle of ICMP Ping message and deeper insights into the congestion scenario caused by successive ping message among nodes.

DESCRIPTION

Ping is the one of popular mechanism for internet control messaging protocol. Ping message is used for determining the reachability and aliveness of the remote/ destination machine in a network. In this experiment, network simulator2 is used for creating network topology consisting of 6 nodes interconnected by point to point duplex link. Nodes on the created topology issues ping command to the other nodes in the network and generate traffic. Node upon receiving the ping message will respond by sending a ping reply message to the requesting node and generate return traffic in the network. Successive ping message by different nodes generates huge traffic on the network and causes packet drop.

INPUT

- (a) ping message from nodes.
- (b) Unix 'grep' command for analyzing the out.tr trace log file for determining the number of packet drop.

EXPECTED OUTPUT

- (a) Ping response from corresponding nodes.
- (b) generate the out.tr trace log file and out.nam network animation file.
- (c) number of packet drop

STEPS

open terminal → Open NSG2 using
java -jar NSG2.jar

Create n nodes by clicking on the screen by selecting the node
click on the link tab =>select link type duplex link
=>select queue type Drop Tail

=>enter capacity(bandwidth), propagation delay, queue size
=>select two nodes to create a link between them

- Click on parameter tab =>enter simulation time, trace file and nam file
- Click on TCL tab to generate code

- Save the code with the file name with extension .tcl
- The highlighted code need to be edited in the program
- Run the code in terminal by typing **ns filename.tcl**
- It would generate an animated topology window where transmission of packets between nodes can be viewed
- Now press the play button in topology window and the simulation begins
- To find the no of packets dropped type the following code in terminal
grep -c “^d” filename.tr

PROGRAM

```
set val(stop) 10.0;
set ns [new Simulator]

set tracefile [open prog2.tr w]
$ns trace-all $tracefile

set namfile [open prog2.nam w]
$ns namtrace-all $namfile

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 color 1 red
$ns color 2 blue

$ns duplex-link $n0 $n1 100.0Mb 10ms DropTail
$ns queue-limit $n0 $n1 4
$ns duplex-link $n1 $n2 50.0Mb 10ms DropTail
$ns queue-limit $n1 $n2 4
$ns duplex-link $n2 $n3 1.0Mb 10ms DropTail
$ns queue-limit $n2 $n3 5
$ns duplex-link $n3 $n4 1.0Mb 10ms DropTail
$ns duplex-link $n4 $n5 10.0Mb 10ms DropTail

$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right
$ns duplex-link-op $n2 $n3 orient right-down
$ns duplex-link-op $n3 $n4 orient left

$ns duplex-link-op $n4 $n5 orient left

Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id] recived ping answer from \
```

```
#$from with round-trip-time $rtt ms"
}

set p0 [new Agent/Ping]
$ns attach-agent $n0 $p0
$p0 set packetSize_ 50000
$p0 set fid_ 1

set p5 [new Agent/Ping]
$ns attach-agent $n5 $p5
$p5 set packetSize_ 50000
$p5 set fid_ 2

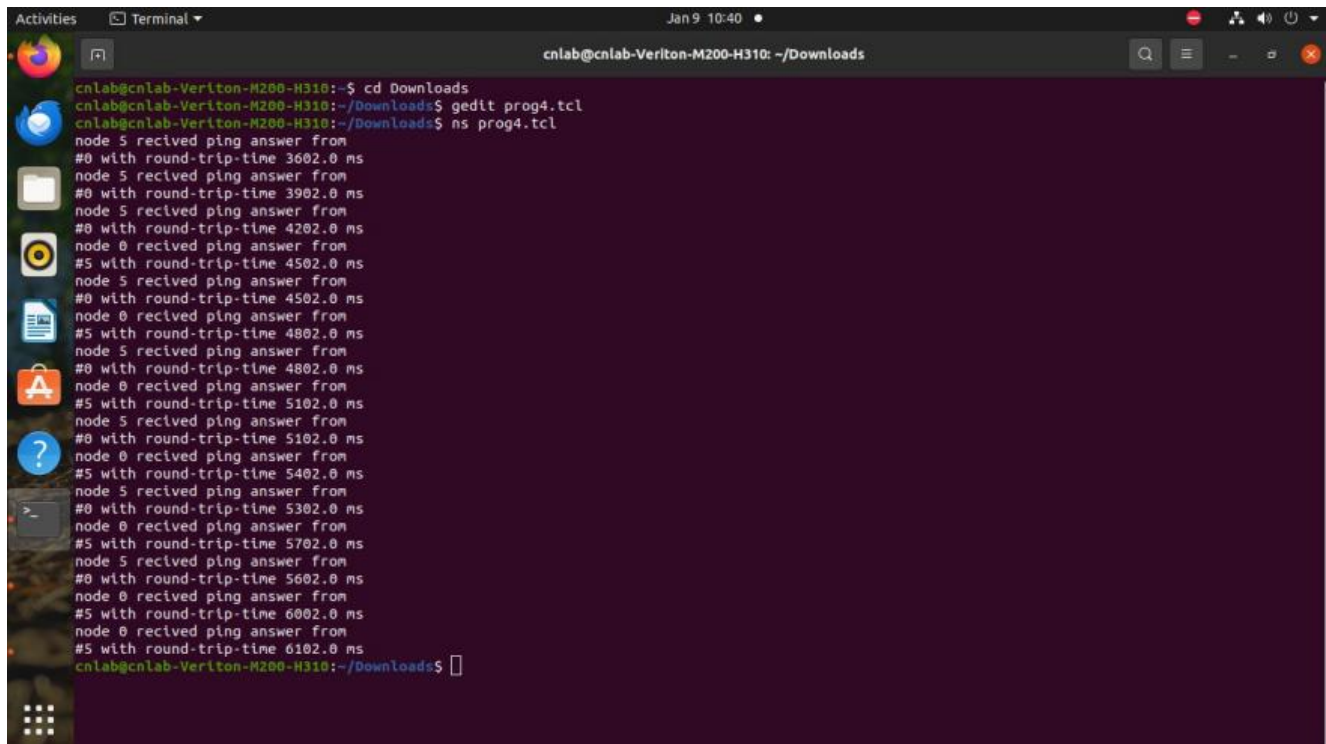
$ns connect $p0 $p5

$ns at 0.1 "$p0 send"
$ns at 0.2 "$p0 send"
$ns at 0.3 "$p0 send"
$ns at 0.4 "$p0 send"
$ns at 0.5 "$p0 send"
$ns at 0.6 "$p0 send"
$ns at 0.7 "$p0 send"
$ns at 0.9 "$p0 send"
$ns at 1.0 "$p0 send"

$ns at 0.1 "$p5 send"
$ns at 0.2 "$p5 send"
$ns at 0.3 "$p5 send"
$ns at 0.4 "$p5 send"
$ns at 0.5 "$p5 send"
$ns at 0.6 "$p5 send"
$ns at 0.7 "$p5 send"
$ns at 0.9 "$p5 send"
$ns at 1.0 "$p5 send"

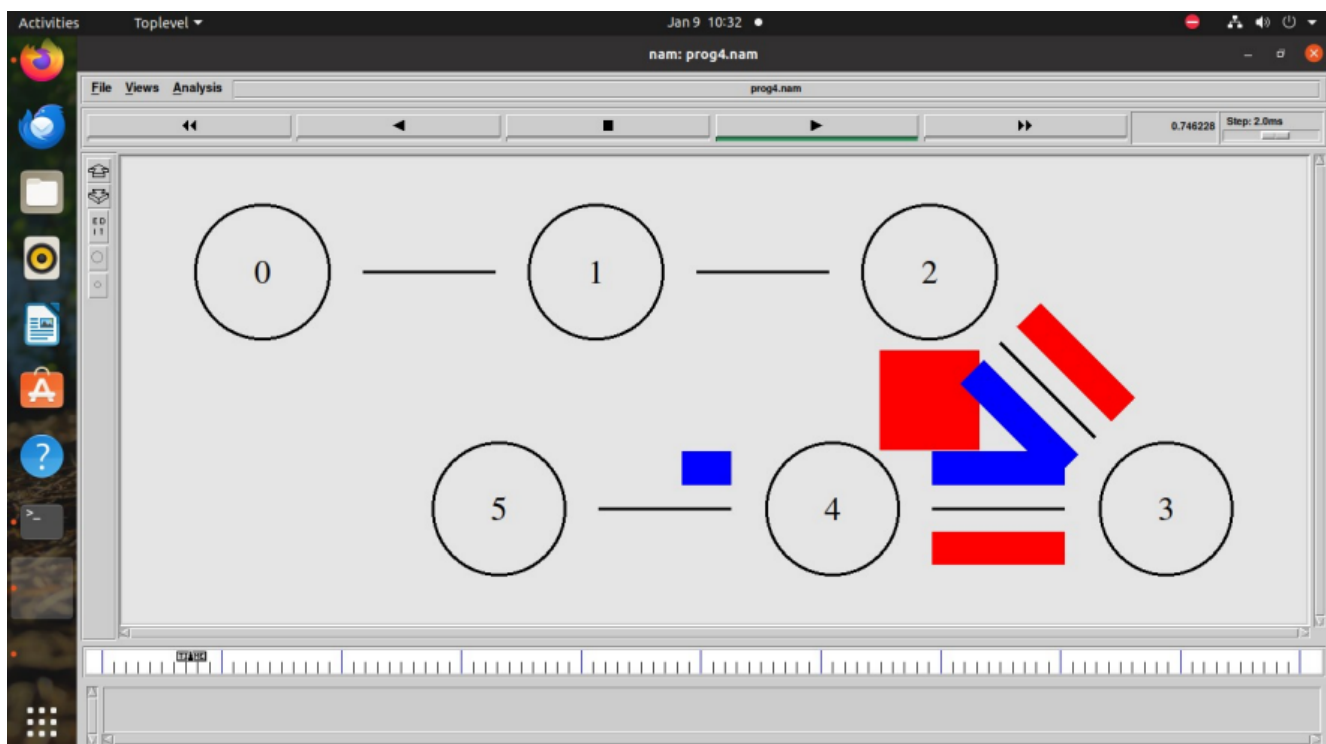
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam prog2.nam &
    exit 0
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

RESULT



A terminal window titled 'Terminal' showing the execution of a network simulation. The user is at the prompt 'cnlab@cnlab-Veriton-M200-H310: ~/Downloads'. They run 'cd Downloads', 'gedit prog4.tcl', and 'ns prog4.tcl'. The output shows a series of ping messages between nodes 0 and 5, with round-trip times increasing from 3602.0 ms to 6102.0 ms. The terminal window has a dark background and a light-colored text area.

```
cnlab@cnlab-Veriton-M200-H310:~/Downloads$ cd Downloads
cnlab@cnlab-Veriton-M200-H310:~/Downloads$ gedit prog4.tcl
cnlab@cnlab-Veriton-M200-H310:~/Downloads$ ns prog4.tcl
node 5 recived ping answer from
#0 with round-trip-time 3602.0 ms
node 5 recived ping answer from
#0 with round-trip-time 3902.0 ms
node 5 recived ping answer from
#0 with round-trip-time 4202.0 ms
node 0 recived ping answer from
#5 with round-trip-time 4502.0 ms
node 5 recived ping answer from
#0 with round-trip-time 4502.0 ms
node 0 recived ping answer from
#5 with round-trip-time 4802.0 ms
node 5 recived ping answer from
#0 with round-trip-time 4802.0 ms
node 0 recived ping answer from
#5 with round-trip-time 5102.0 ms
node 5 recived ping answer from
#0 with round-trip-time 5102.0 ms
node 0 recived ping answer from
#5 with round-trip-time 5402.0 ms
node 5 recived ping answer from
#0 with round-trip-time 5302.0 ms
node 0 recived ping answer from
#5 with round-trip-time 5702.0 ms
node 5 recived ping answer from
#0 with round-trip-time 5602.0 ms
node 0 recived ping answer from
#5 with round-trip-time 6002.0 ms
node 0 recived ping answer from
#5 with round-trip-time 6102.0 ms
cnlab@cnlab-Veriton-M200-H310:~/Downloads$
```



VIVA QUESTIONS AND ANSWERS

1. Define Ping?

Ping is a computer network administration software utility used to test the reachability of a host on an Internet Protocol (IP) network. It measures the round-trip time for *messages* sent from the originating host to a destination computer that are echoed back to the source.

2. What is network protocol?

Network protocols are formal standards and policies comprised of rules, procedures and formats that define communication between two or more devices over a *network*

3. Define Bandwidth?

Bandwidth is the amount of data that can be transmitted in a fixed amount of time

4. Define a router?

Routers perform the traffic directing functions on the **Internet**. Data sent through the **internet**, such as a web page or email, is in the form of **data packets**. A packet is typically forwarded from one router to another router through the networks until it reaches its destination node

5. Describe network congestion?

It is the reduced quality of service that occurs when a **network** node or link is carrying more data than it can handle. Typical effects include queuing delay, packet loss or the blocking of new connections

6. Which are the two congestion control mechanisms?

- prevents the congestion from happening
- removes congestion after it has taken place

PROGRAM 5

TITLE

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

DESCRIPTION

Bellman-Ford algorithm solves the single-source shortest-path problem in the general case in which edges of a given digraph can have negative weight as long as G contains no negative cycles.

This algorithm, like Dijkstra's algorithm uses the notion of edge relaxation but does not use with greedy method. Again, it uses $d[u]$ as an upper bound on the distance $d[u, v]$ from u to v .

The algorithm progressively decreases an estimate $d[v]$ on the weight of the shortest path from the source vertex s to each vertex v in V until it achieve the actual shortest-path. The algorithm returns Boolean TRUE if the given digraph contains no negative cycles that are reachable from source vertex s otherwise it returns Boolean FALSE.

BELLMAN-FORD (G, w, s)

1. INITIALIZE-SINGLE-SOURCE (G, s)
2. for each vertex $i = 1$ to $V[G] - 1$ do
3. for each edge (u, v) in $E[G]$ do
4. RELAX (u, v, w)
5. For each edge (u, v) in $E[G]$ do
6. if $d[u] + w(u, v) < d[v]$ then
7. return FALSE
8. return TRUE

Asymptotic complexity:

- Average case (random data): $O(|V| |E|)$
- Worst case: $O(|V| |E|)$

Conclusion:

Thus, the Bellman-Ford algorithm runs in $O(E)$ time.

PROGRAM

```
import java.util.Scanner;

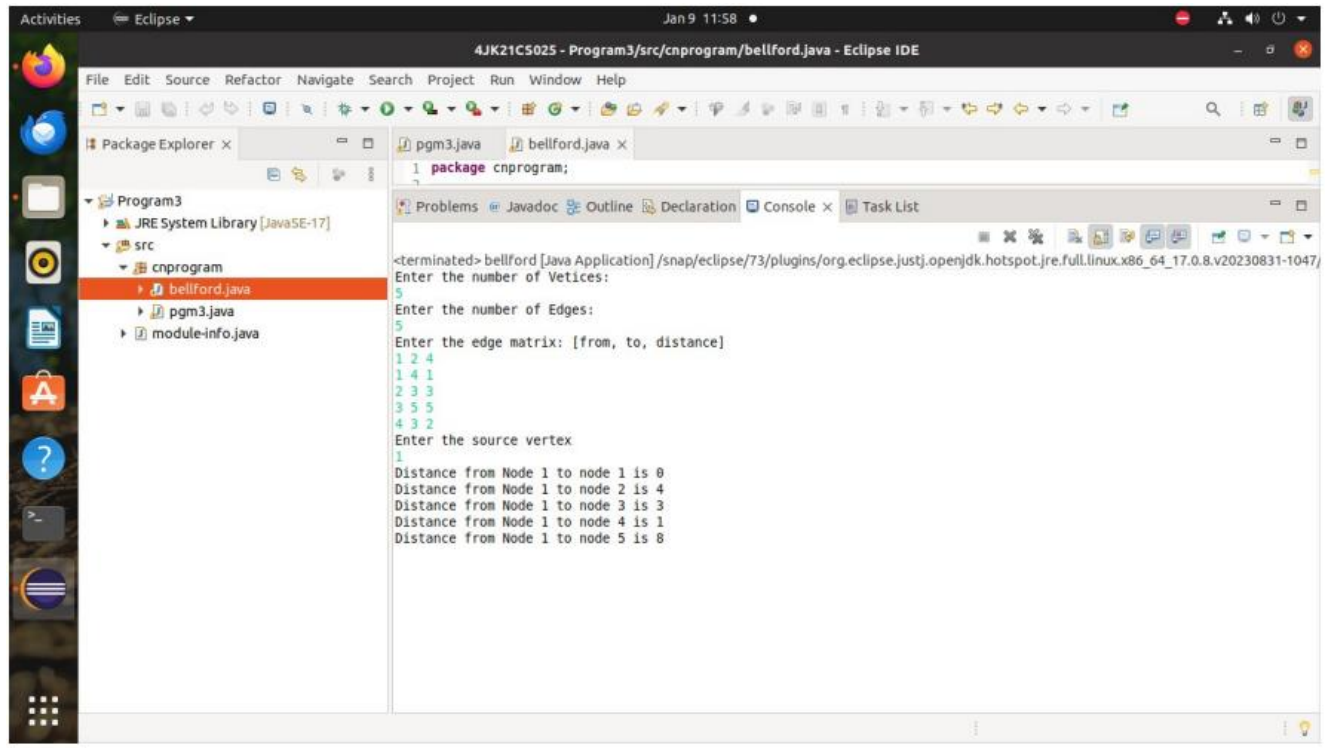
public class Bellman
{
    public static void bellmanFord(int numvertex,int source,int numedges,int edgemat[][])
    {
        int srcdist[]=new int[numvertex+1];
        for(int i=1;i<=numvertex;i++)
        {
            srcdist[i]=999;
        }
        srcdist[source]=0;
        for(int i=1;i<=numvertex;i++)
        {
            for(int j=1;j<=numedges;j++)
            {
                int u=edgemat[j][0];
                int v=edgemat[j][1];
                int w=edgemat[j][2];
                if(srcdist[v]>srcdist[u]+w)
                {
                    srcdist[v]=srcdist[u]+w;
                }
            }
        }
        for(int i=1;i<=numvertex;i++)
        {
            System.out.println("Distance from Node "+source+"to node "+i+"is" +srcdist[i]);
        }
    }

    public static void main(String args[])
    {
        int numvertex=0;
        int source;
        int numedges=0;
        Scanner scanner=new Scanner(System.in);
        System.out.print("enter the number of vertices:");
        numvertex=scanner.nextInt();
        System.out.print("enter the number of edges:");
        numedges=scanner.nextInt();
        int edgemat[][]=new int[numedges+1][3];
        System.out.println("enter the edge matrix:[from,to,distance]");
        for(int i=1;i<=numedges;i++)
        {
            edgemat[i][0]=scanner.nextInt();
            edgemat[i][1]=scanner.nextInt();
            edgemat[i][2]=scanner.nextInt();
        }
    }
}
```



```
}  
System.out.print("enter the source vertex:");  
source=scanner.nextInt();  
bellmanFord(numvertex,source,numedges,edgemat);  
}  
}
```

RESULT



VIVA QUESTIONS AND ANSWERS

1. What is Bellman Ford Algorithm?

The Bellman-Ford algorithm computes single-source shortest paths in a weighted digraph (where some of the edge weights may be negative).

2. Advantages of bellman-ford algorithm?

- (a) Cost is minimized when building a network using BF algo.
- (b) Maximizes the performance of the system. Also finds min path weight.
- (c) It allows splitting of traffic between several paths. It thus increases system performance.

3. What is the Difference between Routing Protocol and Routed Protocol?

Routing Protocol is responsible For Sending and Receiving a Route from One Router to another Router in the Network. When Ever We Will Enable a Routing protocol on the router, in That case Router Automatically creates A Route on the router. As for Example—RIP, IGRP, EIGRP, and OSPF

Routed Protocol is responsible for provides the communication From Source device To Destination Device in the Network. As For Example—TCP/IP, IPX/SPX, apple talk

4. What is RIP?

RIP, short for Routing Information Protocol is used by routers to send data from one network to another. It efficiently manages routing data by broadcasting its routing table to all other routers within the network. It determines the network distance in units of hops.

5. What is strongly connected and weekly connected graph?

A digraph G for which each vertex u has a path to each other vertex v is said to be strongly connected. A digraph G whose underlying graph is connected but for which a pair of vertices u, v exists such that there is no path from u to v is said to be weakly connected.

6. What is Complete Graph?

A complete graph is a simple undirected graph in which every pair of distinct vertices is connected by a unique edge.

7. List and define the different ways of representing a graph?

- i. Adjacency Matrix
- ii. Adjacency List

Adjacency Matrix is 2-Dimensional Array which has the size $V \times V$, where V are the number of vertices in the graph.

Adjacency List is the Array[] of Linked List, where array size is same as number of Vertices in the graph.

8. What is minimum spanning tree?

A minimum spanning tree (MST) or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted (un)directed graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight.

9. What are the applications of MST?

Network design: *telephone, electrical, hydraulic, TV cable, computer, road*

Approximation algorithms for NP-hard problems [traveling salesperson problem](#), [Steiner tree](#).

Cluster analysis: k clustering problem can be viewed as finding an MST and deleting the $k-1$ most expensive edges.

PROGRAM 6

TITLE

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

AIM

To understand working principle of Ethernet LAN and congestion scenario using multiple data traffic.

DESCRIPTION

- Ethernet LAN denoted by IEEE 802.3 is one of the popular computer networking technology. In this experiment, ns2 simulator is used for creating Ethernet LAN and set the two different data traffic between pair of nodes using TCP as transport layer agent. Simulated data traffic between pair of nodes is analysed for determining the packet drop due to congestion in the network. Congestion window for each TCP traffic is plotted on graph using xgraph tool.

INPUT

- Ethernet LAN Bandwidth, propagation delay, Queue Type and channel type
- two TCP traffic between pair of nodes
- initial Congestion window for both the traffic.
- Unix grep command for analyzing the out.tr trace log file for determining the number of packet drop.

EXPECTED OUTPUT

- generate the out.tr trace log file, winfile0, winfile1 and out.nam network animation file.
- Generate WindowSize_file0, WindowSize_file1 holding congestion window size of both the traffic at different instance of time.
- number of packet drop.
- Xgraph plotted graph of depicting the congestion window of both the traffic.

STEPS

Create n nodes by clicking on the screen by selecting the node

click on the link tab =>select link type duplex link

=>select queue type Drop Tail

=>enter capacity(bandwidth), propagation delay, queue size

=>select two nodes to create a link between them

- Click on agent tab
 - for the source node=>select agent type “TCP”
 - =>enter packet size in bytes
 - =>click on the source node and drag

- for the destination node=>select agent type “TCP Sink”
=>enter packet size in bytes
=>click on the destination node and drag
- Create TCP connections between n1 to n5 and n0 to n5
- Connect source to destination node by dragging TCP to TCP Sink (Virtual connection)
- Click on application tab =>select application type(FTP)
=>enter start time and stop time
- Click on TCP and then drag(FTP0)
- Click on parameter tab =>enter simulation time, trace file and nam file
- Click on TCL tab to generate code
- Save the code with the file name with extension .tcl
- Run the code in terminal by typing **ns filename.tcl**
- It would generate an animated topology window where transmission of packets between nodes can be viewed
- Now press the play button in topology window and the simulation begins
- To find the no of packets dropped type the following code in terminal
grep -c “^d” filename.tr
- To see the trace file contents open the file as
gedit filename.tr

PROGRAM

```

set val(stop) 10.0          ;

set ns [new Simulator]

set tracefile [open prog6.tr w]
$ns trace-all $tracefile

set namfile [open prog6.nam w]
$ns namtrace-all $namfile

set winfile0 [open WinFileReno w]
set winfile1 [open WinFileNewReno w]

$ns color 1 Red
$ns color 2 Blue

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 $n1 100.0Mb 10ms DropTail
$ns queue-limit $n0 $n1 50
$ns duplex-link-op $n0 $n1 color "blue"
$ns duplex-link $n2 $n1 100.0Mb 10ms DropTail
$ns queue-limit $n2 $n1 20
$ns duplex-link-op $n2 $n1 color "green"
$ns duplex-link $n1 $n3 100.0Mb 10ms DropTail
$ns queue-limit $n1 $n3 50

$ns duplex-link-op $n0 $n1 orient right-down
$ns duplex-link-op $n2 $n1 orient right-up
$ns duplex-link-op $n1 $n3 orient right

set lan [$ns newLan "$n3 $n4 $n5" 1Mb 40ms LLQueue/DropTail Mac/802_3 channel]

proc PlotWindow {tcpSource file} {
    global ns
    set time 0.1
    set now [$ns now]
    set cwnd [$tcpSource set cwnd_]
    puts $file "$now+$time"
    $ns at [expr $now+$time] "PlotWindow $tcpSource $file"
}

set tcp0 [new Agent/TCP/Reno]
$ns attach-agent $n0 $tcp0
$tcp0 set window_ 8000
$tcp0 set fid_ 1
set sink1 [new Agent/TCPSink]
$ns attach-agent $n4 $sink1
$ns connect $tcp0 $sink1
$tcp0 set packetSize_ 1500

set tcp3 [new Agent/TCP/Reno]
$ns attach-agent $n2 $tcp3
$tcp3 set window_ 8000
$tcp3 set fid_ 2
set sink4 [new Agent/TCPSink]
$ns attach-agent $n5 $sink4
$ns connect $tcp3 $sink4
$tcp3 set packetSize_ 1500

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 0.1 "$ftp0 start"
$ns at 0.1 "PlotWindow $tcp0 $winfile0"
$ns at 100.0 "$ftp0 stop"

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp3
```

```
$ns at 0.1 "$ftp1 start"
$ns at 0.1 "PlotWindow $tcp3 $winfile1"
$ns at 100.0 "$ftp1 stop"

proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam prog6.nam &
    exec xgraph WinFileReno WinFileNewReno &
    exit 0
}

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

RESULT

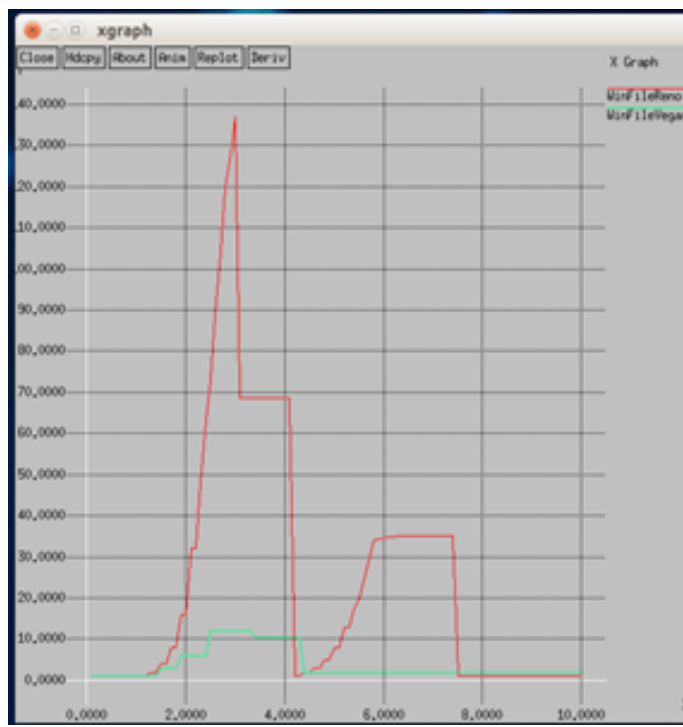


Figure: NAM output

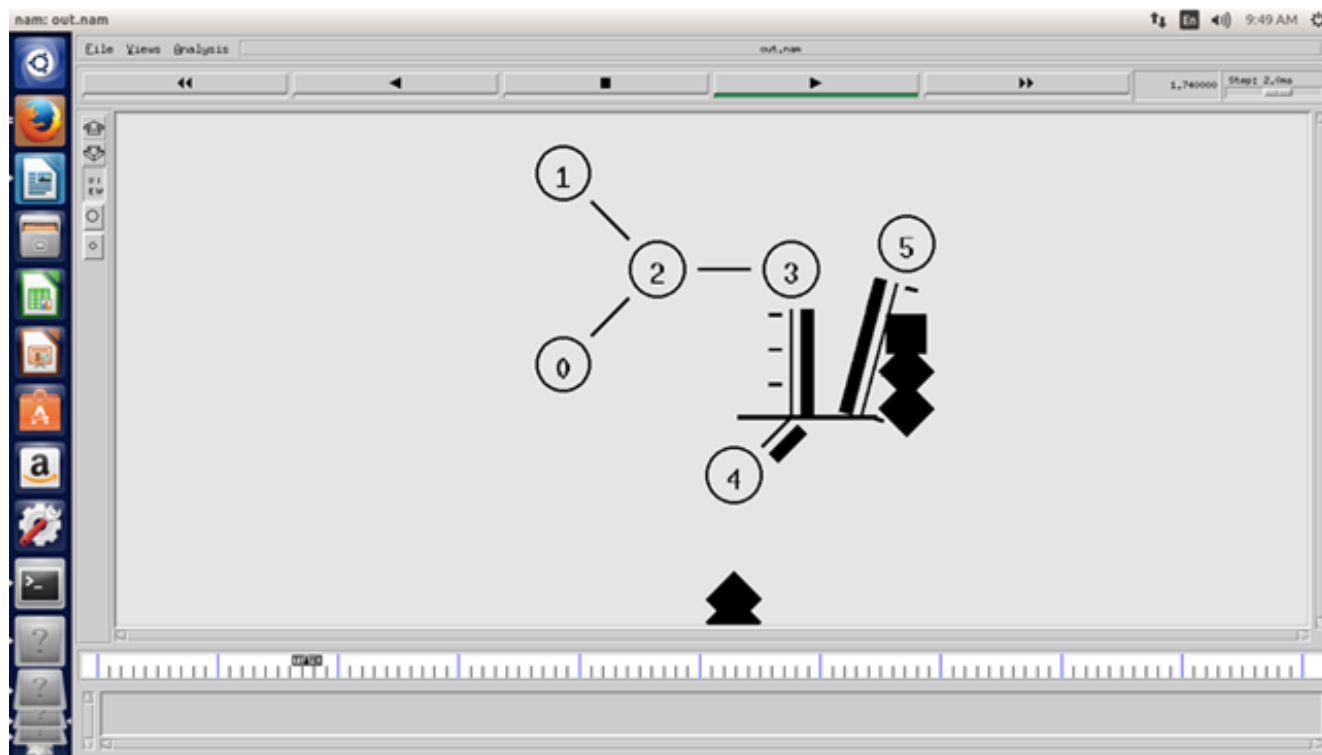


Figure: Congestion Window

VIVA QUESTIONS AND ANSWERS

1. What is Ethernet?

It is the standard way to connect computers on a network over a wired connection. It provides a simple interface for connecting multiple devices, such computers, routers, and switches. With a single router and a few Ethernet cables, you can create a LAN, which allows all connected devices to communicate with each other.

2. Which are the different layers of an OSI Model?

Layer 7: The application layer

Layer 6: The presentation layer

Layer 5: The session layer

Layer 4: The transport layer

Layer 3: The network layer

Layer 2: The data-link layer

Layer 1: The physical layer

3. Which are the two types of Internet Protocol (IP) traffic?

There are two types of Internet Protocol (IP) traffic. They are TCP or Transmission Control Protocol and UDP or User Datagram Protocol. TCP is connection oriented – once a connection is established, data can be sent bidirectional. UDP is a simpler, connectionless Internet protocol. Multiple messages are sent as packets in chunks using UDP.

4. Describe session?

A session is a semi-permanent interactive information interchange between two or more communicating devices or between a computer and user (see login session)

5. What is Demodulation?

It is the process of converting an analog signal to digital signal

PROGRAM 7

TITLE

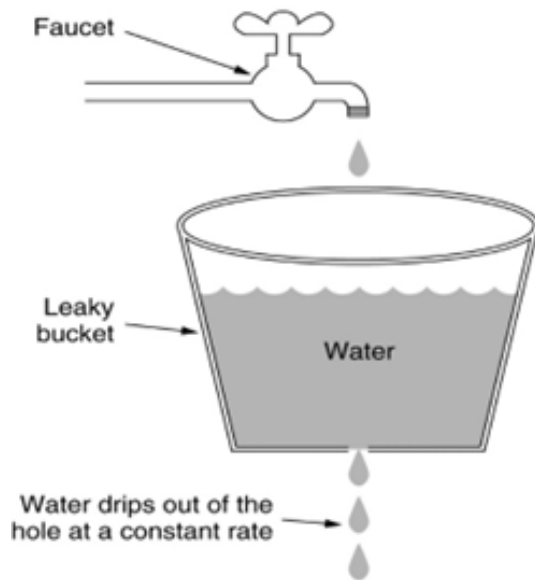
Write a program for congestion control using leaky bucket algorithm.

DESCRIPTION

The main concept of the leaky bucket algorithm is that the output data flow remains constant despite the variant input traffic, such as the water flow in a bucket with a small hole at the bottom. In case the bucket contains water (or packets) then the output flow follows a constant rate, while if the bucket is full any additional load will be lost because of spillover. In a similar way if the bucket is empty the output will be zero.

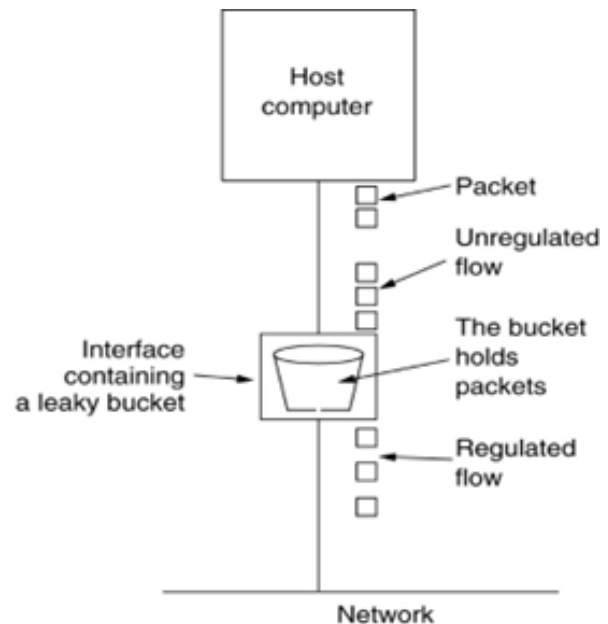
From network perspective, leaky bucket consists of a finite queue (bucket) where all the incoming packets are stored in case there is space in the queue, otherwise the packets are discarded. In order to regulate the output flow, leaky bucket transmits one packet from the queue in a fixed time (e.g. at every clock tick). In the following figure we can notice the main rationale of leaky bucket algorithm, for both the two approaches

(e.g. leaky bucket with water (a) and with packets (b)).



(a)

a) Leaky Bucket with water



(b)

b) Leaky Bucket with packets

PROGRAM

```
import java.util.*;
class prg7
{
    public static void main(String args[])
    {

        Scanner sc=new Scanner(System.in);
        int i,size,nop,opr,temp=0;
        int[] datarate=new int[100];
        System.out.println("enter the bucket size");
        size=sc.nextInt();
        System.out.println("enter the number of packets");
        nop=sc.nextInt();
        System.out.println("enter the dara rate");

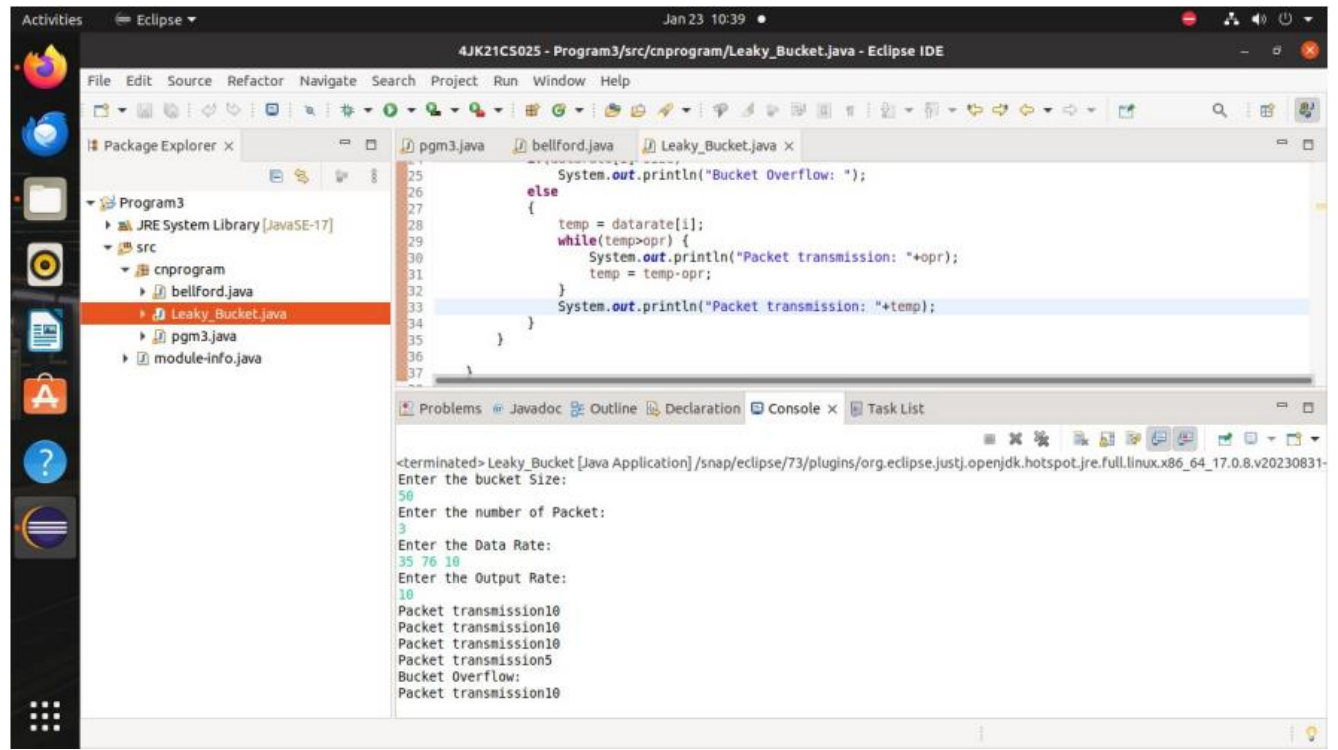
        for(i=0;i<nop;i++)
            datarate[i]=sc.nextInt();

        System.out.println("enter the output rate");
        opr=sc.nextInt();

        for(i=0;i<nop;i++)
        {
            if(datarate[i]>size)
                System.out.println("bucket overflow");
            else
            {
                temp=datarate[i];

                while(temp>opr)
                {
                    System.out.println("packet transmission"+opr);
                    temp=temp-opr;
                }
                System.out.println("packet transmission"+temp);
            }
        }
    }
}
```

RESULT



The screenshot displays the Eclipse IDE interface. The Package Explorer on the left shows the project structure: Program3, JRE System Library [JavaSE-17], src, cnprogram, bellford.java, Leaky_Bucket.java, pgm3.java, and module-info.java. The main editor shows the code for Leaky_Bucket.java, which includes a while loop for packet transmission and a System.out.println statement for bucket overflow. The Console window at the bottom shows the program's execution, including prompts for bucket size, number of packets, data rate, and output rate, followed by the program's output.

```
4JK21CS025 - Program3/src/cnprogram/Leaky_Bucket.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer x
Program3
  JRE System Library [JavaSE-17]
  src
    cnprogram
      bellford.java
      Leaky_Bucket.java
      pgm3.java
      module-info.java
  pgm3.java
  bellford.java
  Leaky_Bucket.java x
25      System.out.println("Bucket Overflow: ");
26      else
27      {
28          temp = datarate[i];
29          while(temp>opr) {
30              System.out.println("Packet transmission: "+opr);
31              temp = temp-opr;
32          }
33          System.out.println("Packet transmission: "+temp);
34      }
35  }
36
37
Problems Javadoc Outline Declaration Console x Task List
<terminated> Leaky_Bucket [Java Application] /snap/eclipse/73/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.8.v20230831-
Enter the bucket Size:
50
Enter the number of Packet:
3
Enter the Data Rate:
35 76 10
Enter the Output Rate:
10
Packet transmission10
Packet transmission10
Packet transmission10
Packet transmission5
Bucket Overflow:
Packet transmission10
```

VIVA QUESTIONS AND ANSWERS**1. What is leaky bucket algorithm?**

In this algorithm, an interface is connected between a host and the network and the interface has a finite buffer space. If a packet arrives at the interface when the buffer is full then it is discarded by the interface. It is called leaky bucket algorithm because the outgoing rate of packet from the buffer is constant no matter how much incoming traffic is there at the interface.

2. Define a network congestion.

When two or more nodes would simultaneously try to transmit packets to one node there is high probability that the number of packets would exceed the packet handling capacity of the network and lead to congestion.

3. List some ways to deal with congestion.

Several ways to handle congestion

Packet elimination

Flow Control

Buffer allocation

Choke packets

4. What is meant by choke packets?

A specialized packet that is used for flow control along a network. A router detects congestion by measuring the percentage of buffers in use, line utilization and average queue lengths. When it detects congestion, it sends choke packets across the network to all the data sources associated with the congestion.

5. What is packet dropping?

When a buffer becomes full a router can drop waiting packets- if not coupled with some other techniques, this can lead to greater congestion through retransmission.

6. What are the different techniques used to improve network congestion?

Buffering

Over-provisioning

Traffic shaping

Packet scheduling

7. What is traffic shaping?

Traffic shaping, also known as packet shaping, Quality of Service (QoS) or bandwidth management, is the manipulation and prioritization of network traffic to reduce the impact of heavy users or machines from effecting other users.

8. What is the advantage of leaky bucket algorithm?

Leaky-bucket algorithm is needed to regulate the data flow. It is used to improve the lifetime of network and prevent from traffic flows increases performance of our network.

9. What are the different applications of leaky bucket algorithm?

- Leaky bucket algorithm is used to regulate the traffic. It can be used in telecom network either as traffic shaping or traffic policing.
- In Telecom networks, the control of traffic (i.e. sending more packets into the network) is done using this algorithm
- Network monitors traffic flows continuously to ensure they meet their traffic contract. The process of monitoring and enforcing the traffic flow is called policing