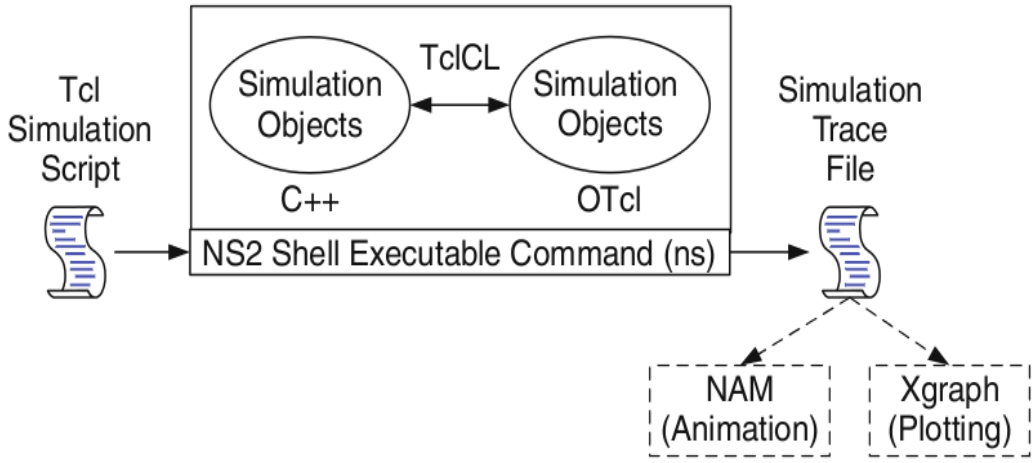
Introduction to NS-2:

- 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 behaviors.
- **Basic Architecture of NS2**



Tcl scripting

- Tcl 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.

Basics of TCL

Syntax: command arg1 arg2 arg3

- **Hello World!**

```
puts stdout{Hello, World!}

Hello, World!
```

- **Variables** Command Substitution

```
set a 5          set len [string length foobar]

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

- **Simple Arithmetic**

```
expr 7.2 / 4
```

- **Procedures**

```
proc Diag {a b} {
set c [expr sqrt($a * $a + $b * $b)]
return $c }
puts "Diagonal of a 3, 4 right triangle is [Diag 3 4]"
```

Output: Diagonal of a 3, 4 right triangle is 5.0

- **Loops**

```
while{$i < $n} {          for {set i 0} {$i < $n} {incr i} {
. . .                     . . .
}                       }
```

Wired TCL Script Components

- Create the event scheduler
- Open new files & turn on the tracing
- Create the nodes
- Setup the links
- Configure the traffic type (e.g., TCP, UDP, etc)
- Set the time of traffic generation (e.g., CBR, FTP)
- Terminate the simulation

NS Simulator Preliminaries.

1. Initialization and termination aspects of the ns simulator.
2. Definition of network nodes, links, queues and topology.
3. Definition of agents and of applications.
4. The nam visualization tool.
5. Tracing and random variables.

Initialization and Termination of TCL Script in NS-2

An ns simulation starts with the command

set ns [new Simulator]

Which is thus the first line in the tcl script? 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 dta 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 begins 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 ends 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,

$ns at 125.0 "finish"

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

$ns run

Definition of a network of links and nodes

The way to define a node is

set n0 [$ns node]

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:

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

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 "duplex-link" 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) 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:

#set Queue Size of link (n0-n2) to 20
$ns queue-limit $n0 $n2 20

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:

set tcp [new Agent/TCP]

The command **$ns attach-agent $n0 $tcp** defines the source node of the tcp connection.

The command

set sink [new Agent /TCPSink]

Defines the behavior of the destination node of TCP and assigns to it a pointer called sink.

#Setup a UDP connection

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

The below 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.

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set packetsize_ 100

$cbr set rate_ 0.01Mb

$cbr set random_ false

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 colors 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

$cbr set interval_ 0.005

The packet size can be set to some value using

$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:

$ns at <time> <event>

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

$ns at 0.1 "$cbr start"

$ns at 1.0 " $ftp start"

$ns at 124.0 "$ftp stop"

$ns at 124.5 "$cbr stop"

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 are:

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 (eg 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.
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.

XGRAPH

The xgraph program draws a graph on an x-display given data read from either data file or from standard input if no files are specified. It can display upto 64 independent

data sets using different colors and line styles for each set. It annotates the graph with a title, axis labels, grid lines or tick marks, grid labels and a legend.

Syntax:

Xgraph [options] file-name

Options are listed here

/-bd <color> (Border)

This specifies the border color of the xgraph window.

/-bg <color> (Background)

This specifies the background color of the xgraph window.

/-fg<color> (Foreground)

This specifies the foreground color of the xgraph window.

/-lf <fontname> (LabelFont)

All axis labels and grid labels are drawn using this font.

/-t<string> (Title Text)

This string is centered at the top of the graph.

/-x <unit name> (XunitText)

This is the unit name for the x-axis. Its default is "X".

/-y <unit name> (YunitText)

This is the unit name for the y-axis. Its default is "Y".

Awk- An Advanced

awk is a programmable, pattern-matching, and processing tool available in UNIX. It

works equally well with text and numbers.

awk is not just a command, but a programming language too. In other words, awk utility is a pattern scanning and processing language. It searches one or more files to see if they contain lines that match specified patterns and then perform associated actions, such as writing the line to the standard output or incrementing a counter each time it finds a match.

Syntax:

awk option 'selection_criteria {action}' file(s)

Here, selection_criteria filters input and select lines for the action component to act upon. The selection_criteria is enclosed within single quotes and the action within the curly braces. Both the selection_criteria and action forms an awk program.

Example: $ awk '/manager/ {print}' emp.lst

Variables

Awk allows the user to use variables of there choice. You can now print a serial number, using the variable kount, and apply it those directors drawing a salary exceeding 6700:

```
$ awk -F" |"   '$3 == "director" && $6 > 6700 {

kount =kount+1

printf " %3f %20s %-12s %d¥n", kount,$2,$3,$6 }'  empn.lst
```

THE -f OPTION: STORING awk PROGRAMS IN A FILE

You should holds large awk programs in separate file and provide them with the awk extension for easier identification. Let's first store the previous program in the file empawk.awk:

```
$ cat empawk.awk
```

Observe that this time we haven't used quotes to enclose the awk program. You can now use awk with the -f *filename* option to obtain the same output:

```
Awk –F"|" –f empawk.awk empn.lst
```

THE BEGIN AND END SECTIONS

Awk statements are usually applied to all lines selected by the address, and if there are no addresses, then they are applied to every line of input. But, if you have to print something before processing the first line, for example, a heading, then the BEGIN section can be used gainfully. Similarly, the end section useful in printing some totals after processing is over.

The BEGIN and END sections are optional and take the form

BEGIN {action}

END {action}

These two sections, when present, are delimited by the body of the awk program. You can use them to print a suitable heading at the beginning and the average salary at the end.

BUILT-IN VARIABLES

Awk has several built-in variables. They are all assigned automatically, though it is also possible for a user to reassign some of them. You have already used NR, which signifies the record number of the current line. We' ll now have a brief look at some of the other variable.

The FS Variable: as stated elsewhere, awk uses a contiguous string of spaces as the default field delimiter. FS redefines this field separator, which in the sample database happens to be the |. When used at all, it must occur in the BEGIN section so that the body of the program knows its value before it starts processing:

BEGIN {FS=" |" }

This is an alternative to the -F option which does the same thing.

The OFS Variable: when you used the print statement with comma-separated arguments, each argument was separated from the other by a space. This is awk' s default output field separator, and can reassigned using the variable OFS in the BEGIN section:

BEGIN { OFS=" ~" }

When you reassign this variable with a ~ (tilde), awk will use this character for delimiting the print arguments. This is a useful variable for creating lines with delimited fields.

The NF variable: NF comes in quite handy for cleaning up a database of lines that don't contain the right number of fields. By using it on a file, say emp.lst, you can locate those lines not having 6 fields, and which have crept in due to faulty data entry:

```
$awk  'BEGIN {FS = "|" }
NF! =6 {Print "Record No ", NR, "has", "fields" }'  empx.lst
```

Part-A

Experiment No: 1

Aim: Simulate a three node point to point network with duplex links between them. Set queue size and vary the bandwidth and find number of packets dropped.

```
set ns [new Simulator]
set nf [open lab1.nam w]
$ns namtrace-all $nf

set tf [open lab1.tr w]
$ns trace-all $tf
proc finish { } {
global ns nf tf
$ns flush-trace
close $nf
close $tf
exec nam lab1.nam &
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n2 200Mb 10ms DropTail
$ns duplex-link $n1 $n2 100Mb 5ms DropTail
$ns duplex-link $n2 $n3 1Mb 1000ms DropTail
$ns queue-limit $n0 $n2 10
$ns queue-limit $n1 $n2 10
set udp0 [new Agent/UDP]
$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 udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
set udp2 [new Agent/UDP]
$ns attach-agent $n2 $udp2
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
$ns connect $udp0 $null0
$ns connect $udp1 $null0
$ns at 0.1 "$cbr0 start"
$ns at 0.2 "$cbr1 start"
$ns at 1.0 "finish"
$ns run
```

AWK FILE.

```
BEGIN{ c=0;}
{
if($1== "d")
{
```

```
c++;
printf("%s¥t%s¥n",$5,$11);
}
}
END { printf("the number of packets dropped=%d¥n",c);}
```

Steps for execution:

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

  [root@localhost ~]# vi lab1.tcl
- Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.
- Open vi editor and type **awk** program. Program name should have the extension "**.awk** "

  [root@localhost ~]# vi lab1.awk
- Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.
- Run the simulation program

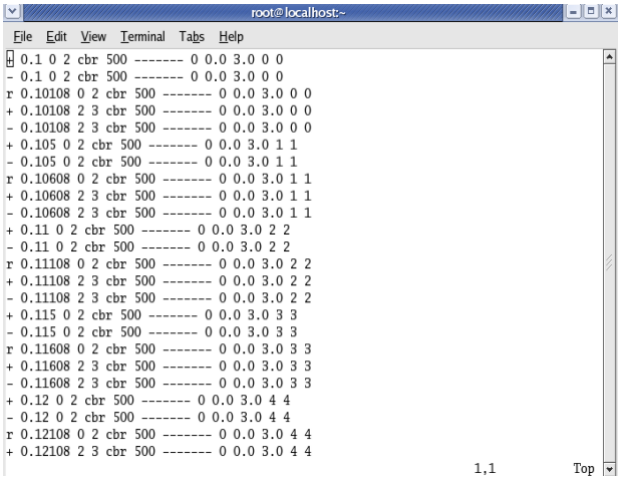
  [root@localhost~]# ns lab1.tcl
- Here **"ns"** indicates network simulator. We get the topology shown in the snapshot.
- Now press the play button in the simulation window and the simulation will begins.
- After simulation is completed run **awk file** to see the output ,

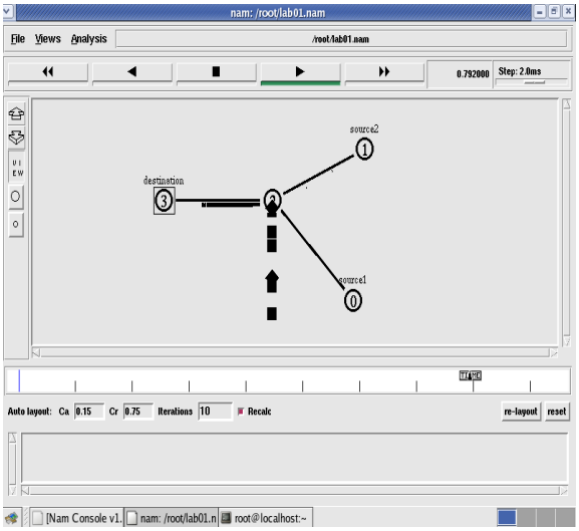
  [root@localhost~]# awk -f lab1.awk lab1.tr
- To see the trace file contents open the file as ,

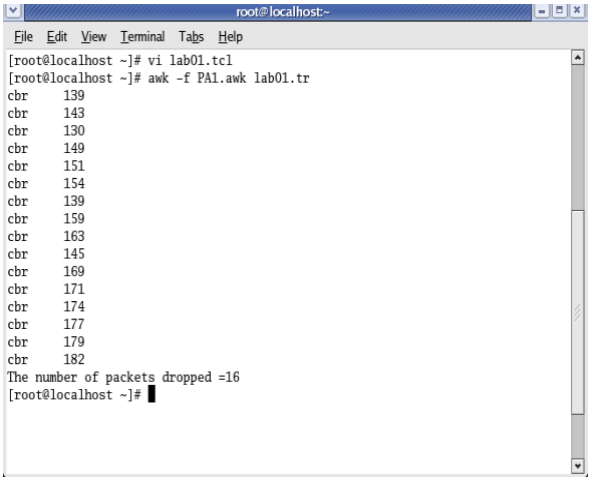
  [root@localhost~]# vi lab1.tr

Trace file contains 12 columns:

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


root@localhost:~
File Edit View Terminal Tabs Help
+ 0.1 0 2 cbr 500 ------- 0 0.0 3.0 0 0
- 0.1 0 2 cbr 500 ------- 0 0.0 3.0 0 0
r 0.10108 0 2 cbr 500 ------- 0 0.0 3.0 0 0
+ 0.10108 2 3 cbr 500 ------- 0 0.0 3.0 0 0
- 0.10108 2 3 cbr 500 ------- 0 0.0 3.0 0 0
+ 0.105 0 2 cbr 500 ------- 0 0.0 3.0 1 1
- 0.105 0 2 cbr 500 ------- 0 0.0 3.0 1 1
r 0.10608 0 2 cbr 500 ------- 0 0.0 3.0 1 1
+ 0.10608 2 3 cbr 500 ------- 0 0.0 3.0 1 1
- 0.10608 2 3 cbr 500 ------- 0 0.0 3.0 1 1
+ 0.11 0 2 cbr 500 ------- 0 0.0 3.0 2 2
- 0.11 0 2 cbr 500 ------- 0 0.0 3.0 2 2
r 0.11108 0 2 cbr 500 ------- 0 0.0 3.0 2 2
+ 0.11108 2 3 cbr 500 ------- 0 0.0 3.0 2 2
- 0.11108 2 3 cbr 500 ------- 0 0.0 3.0 2 2
+ 0.115 0 2 cbr 500 ------- 0 0.0 3.0 3 3
- 0.115 0 2 cbr 500 ------- 0 0.0 3.0 3 3
r 0.11608 0 2 cbr 500 ------- 0 0.0 3.0 3 3
+ 0.11608 2 3 cbr 500 ------- 0 0.0 3.0 3 3
- 0.11608 2 3 cbr 500 ------- 0 0.0 3.0 3 3
+ 0.12 0 2 cbr 500 ------- 0 0.0 3.0 4 4
- 0.12 0 2 cbr 500 ------- 0 0.0 3.0 4 4
r 0.12108 0 2 cbr 500 ------- 0 0.0 3.0 4 4
+ 0.12108 2 3 cbr 500 ------- 0 0.0 3.0 4 4
1,1 Top


nam: /root/lab01.nam
File Views Analysis
/root/lab01.nam
0.792000
Step: 2.0ms
source2
destination
source1
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout reset
[Nam Console v1.
nam: /root/lab01.n
root@localhost:~



Experiment No:2

Aim: 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.

```
set ns [ new Simulator ]
set nf [ open lab2.nam w ]
$ns namtrace-all $nf
set tf [ open lab2.tr w ]
$ns trace-all $tf
set n0 [$ns node]
set n1 [ $ns node ]
set n2 [ $ns node ]
set n3 [ $ns node ]
set n4 [ $ns node ]
set n5 [ $ns node ]
$ns duplex-link $n0 $n4 1005Mb 1ms DropTail
$ns duplex-link $n1 $n4 50Mb 1ms DropTail
```

```
$ns duplex-link $n2 $n4 2000Mb 1ms DropTail
$ns duplex-link $n3 $n4 200Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 1ms DropTail
set p1 [ new Agent/Ping ]
$ns attach-agent $n0 $p1
$p1 set packetSize_ 50000
$p1 set interval_ 0.0001
set p2 [ new Agent/Ping ]
$ns attach-agent $n1 $p2
set p3 [new Agent/Ping ]
$ns attach-agent $n2 $p3
$p3 set packetSize_ 30000
$p3 set interval_ 0.00001
set p4 [ new Agent/Ping ]
$ns attach-agent $n3 $p4
set p5 [ new Agent/Ping ]
$ns attach-agent $n5 $p5
$ns queue-limit $n0 $n4 5
$ns queue-limit $n2 $n4 3
$ns queue-limit $n4 $n5 2
Agent/Ping instproc recv { from rtt } {
$self instvar node_
puts "node [ $node_ id ] received answer from $from with round trip time $rtt msec  "
}
$ns connect $p1 $p5
$ns connect $p3 $p4
proc finish {} {
```

```
global ns nf tf
$ns flush-trace
close $nf
close $tf
exec nam lab2.nam &
exit 0
}
$ns at 0.1 "$p1 send "
$ns at 0.2 "$p1 send "
$ns at 0.3 "$p1 send "
$ns at 0.4 "$p1 send "
$ns at 0.5 "$p1 send "
$ns at 0.6 "$p1 send "
$ns at 0.7 "$p1 send "
$ns at 0.8 "$p1 send "
$ns at 0.9 "$p1 send "
$ns at 1.0 "$p1 send "

$ns at 1.1 "$p1 send "
$ns at 1.2 "$p1 send "
$ns at 1.3 "$p1 send "
$ns at 1.4 "$p1 send "
$ns at 1.5 "$p1 send "
$ns at 1.6 "$p1 send "
$ns at 1.7 "$p1 send "
$ns at 1.8 "$p1 send "
$ns at 1.9 "$p1 send "
```

```
$ns at 2.0 "$p1 send "

$ns at 2.1 "$p1 send "
$ns at 2.2 "$p1 send "
$ns at 2.3 "$p1 send "
$ns at 2.4 "$p1 send "
$ns at 2.5 "$p1 send "
$ns at 2.6 "$p1 send "
$ns at 2.7 "$p1 send "
$ns at 2.8 "$p1 send "
$ns at 2.9 "$p1 send "
$ns at 0.1 "$p3 send "

$ns at 1.1 "$p3 send "
$ns at 1.2 "$p3 send "
$ns at 1.3 "$p3 send "
$ns at 1.4 "$p3 send "
$ns at 1.5 "$p3 send "
$ns at 1.6 "$p3 send "
$ns at 1.7 "$p3 send "
$ns at 1.8 "$p3 send "
$ns at 1.9 "$p3 send "
$ns at 2.0 "$p3 send "

$ns at 2.1 "$p3 send "
$ns at 2.2 "$p3 send "
$ns at 2.3 "$p3 send "
```

```
$ns at 2.4 "$p3 send "
$ns at 2.5 "$p3 send "
$ns at 2.6 "$p3 send "
$ns at 2.7 "$p3 send "
$ns at 2.8 "$p3 send "
$ns at 2.9 "$p3 send "
$ns at 3.0 " finish "
$ns run
```

AWK FILE

```
BEGIN{ drop=0;}
{
if($1== "d")
{
drop++;

}
}
END { printf( "total number of %s packet dropped due to conestion =%d¥n",$5,drop);
```

Steps for execution

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

   [root@localhost ~]# vi lab2.tcl

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

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

[root@localhost ~]# vi lab2.awk

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

5) *Run the simulation program*

[root@localhost~]# ns lab2.tcl

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

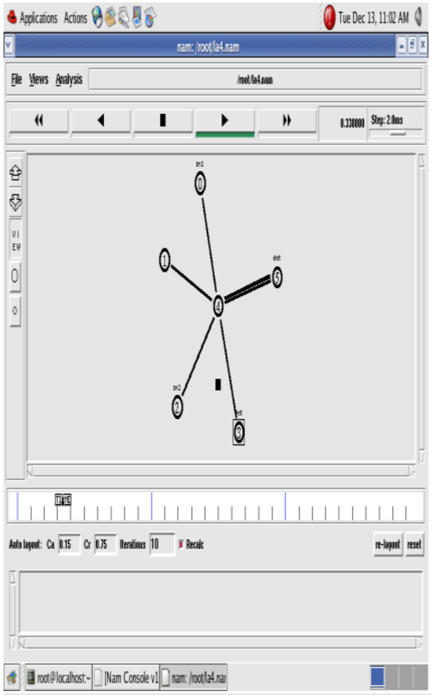
ii) *Now press the play button in the simulation window and the simulation will begins.*

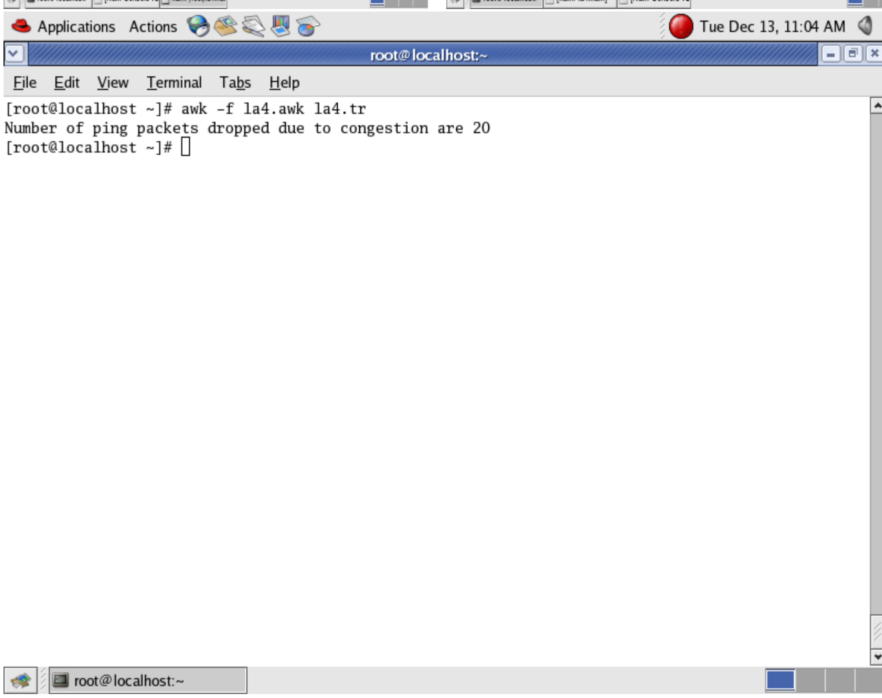
6) *After simulation is completed run **awk file** to see the output ,*

[root@localhost~]# awk -f lab2.awk lab2.tr

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

[root@localhost~]# vi lab2.tr





Topology **Output**

Experiment No:3

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

```
set ns [new Simulator]
set tf [open lab3.tr w]
$ns trace-all $tf
set nf [open lab3.nam w]
$ns namtrace-all $nf
set n0 [$ns node]
$n0 color "magenta"
$n0 label "src1"
set n1 [$ns node]
set n2 [$ns node]
$n2 color "magenta"
$n2 label "src2"
set n3 [$ns node]
$n3 color "blue"
$n3 label "dest2"
set n4 [$ns node]
set n5 [$ns node]
$n5 color "blue"
$n5 label "dest1"
$ns make-lan "$n0 $n1 $n2 $n3 $n4" 100Mb 100ms LL Queue/DropTail Mac/802_3
$ns duplex-link $n4 $n5 1Mb 1ms DropTail
```

```
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.0001
set sink5 [new Agent/TCPSink]
$ns attach-agent $n5 $sink5
$ns connect $tcp0 $sink5
set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ftp2 set packetSize_ 600
$ftp2 set interval_ 0.001
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3
$ns connect $tcp2 $sink3
set file1 [open file1.tr w]
$tcp0 attach $file1
set file2 [open file2.tr w]
$tcp2 attach $file2
$tcp0 trace cwnd_
$tcp2 trace cwnd_
proc finish { } {
global ns nf tf
$ns flush-trace
```

```
close $tf

close $nf

exec nam lab3.nam &

exit 0

}

$ns at 0.1 "$ftp0 start"

$ns at 5 "$ftp0 stop"

$ns at 7 "$ftp0 start"

$ns at 0.2 "$ftp2 start"

$ns at 8 "$ftp2 stop"

$ns at 14 "$ftp0 stop"

$ns at 10 "$ftp2 start"

$ns at 15 "$ftp2 stop"

$ns at 16 "finish"

$ns run
```

AWK FILE:

```
BEGIN{

}

{

if($6== "cwnd_")

 printf( "%f¥t%f¥t¥n",$1,$7);

}

END {

}
```

<u>Steps for execution:</u>

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

    [root@localhost ~]# vi lab3.tcl

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

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

    [root@localhost ~]# vi lab3.awk

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

5) *Run the simulation program*

    [root@localhost~]# ns lab3.tcl

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

    i. ***[root@localhost~]# awk -f lab3.awk file1.tr > a1***

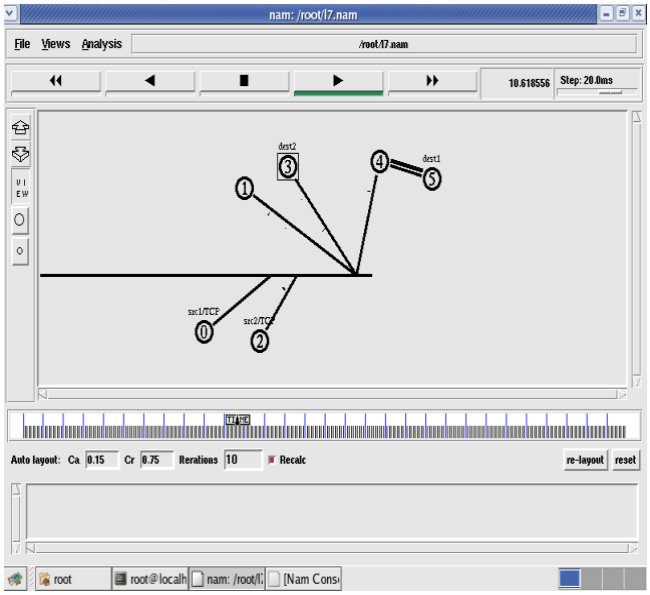
    ii. ***[root@localhost~]# awk -f lab3.awk file2.tr > a2***

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

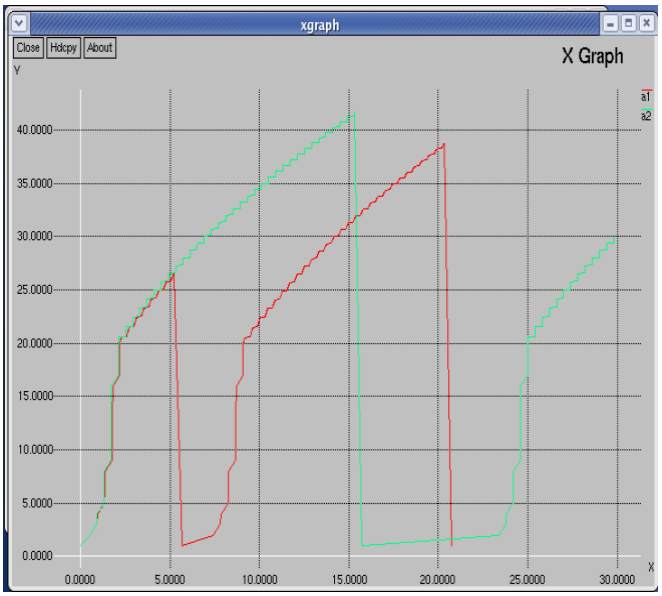
7) *Here we are using the congestion window trace files i.e.* ***file1.tr*** *and* ***file2.tr*** *and we are redirecting the contents of those files to new files say* ***a1*** *and* ***a2*** *using* ***output redirection operator (>)****.*

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

    [root@localhost~]# vi lab7.tr



Output



Graph

Experiment No:4

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

```
set ns [new Simulator]

set tf [open pg4.tr w]

$ns trace-all $tf

set topo [new Topography]

$topo load_flatgrid 1000 1000

set nf [open pg4.nam w]

$ns namtrace-all-wireless $nf 1000 1000

$ns node-config -adhocRouting DSDV ¥

-llType LL ¥
```

```
-macType Mac/802_11 ¥
-ifqType Queue/DropTail ¥
-ifqLen 50 ¥
-phyType Phy/WirelessPhy ¥
-channelType Channel/WirelessChannel ¥
-propType Propagation/TwoRayGround ¥
-antType Antenna/OmniAntenna ¥
-topoInstance $topo ¥
-agentTrace ON ¥
-routerTrace ON

create-god 3

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
$n0 label "tcp0"
$n1 label "sink1/tcp1"
$n2 label "sink2"
$n0 set X_ 50
$n0 set Y_ 50
$n0 set Z_ 0
$n1 set X_ 100
$n1 set Y_ 100
$n1 set Z_ 0
```

```
$n2 set X_ 600
$n2 set Y_ 600
$n2 set Z_ 0
$ns at 0.1 "$n0 setdest 50 50 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 tf nf
```

```
$ns flush-trace
close $tf
close $nf
exec nam pg4.nam &
exit 0
}
$ns at 250 "finish"
$ns run
```

AWK FILE:

```
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¥n",((count2*pack2*8)/(time2*1000000)));
}
```

Steps for execution:

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

   [root@localhost ~]# vi lab4.tcl

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

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

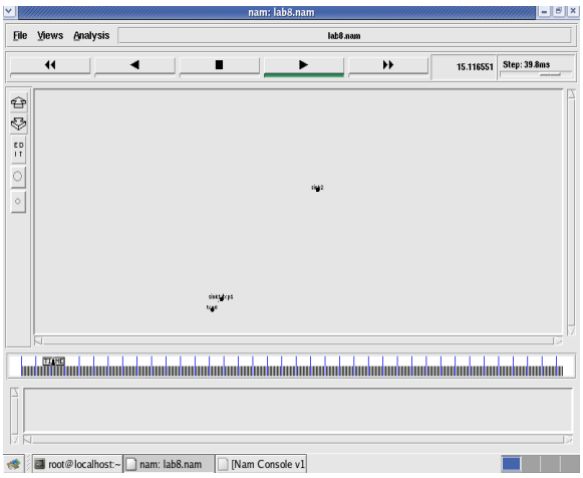
   [root@localhost ~]# vi lab4.awk

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

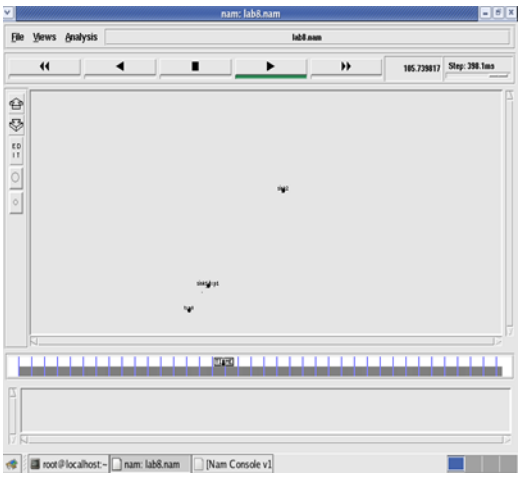
5) *Run the simulation program*

   [root@localhost~]# ns lab4.tcl

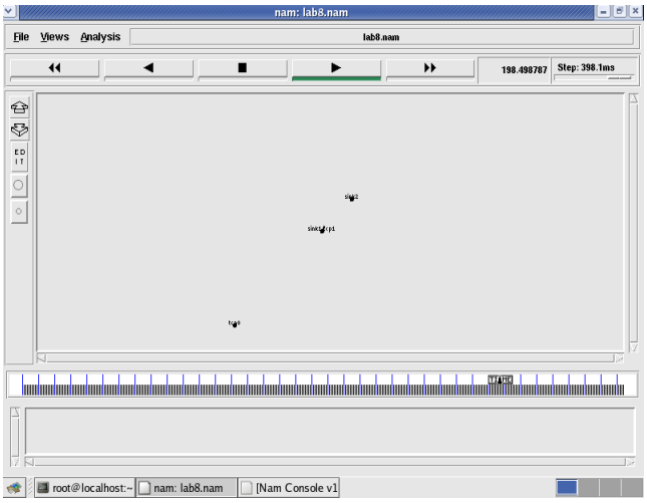
Topology:



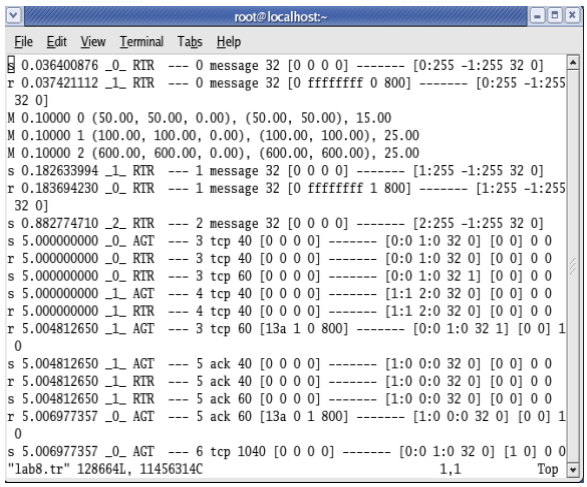
Node 1 and 2 are communicating



Node 2 is moving towards node 3



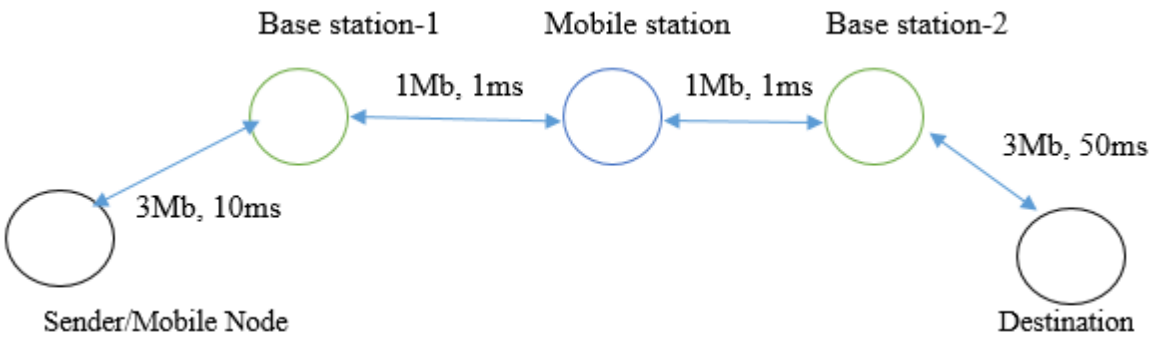
Node 2 is coming back from node 3 towards node1


root@localhost:~
File Edit View Terminal Tabs Help
s 0.036400876 _0_ RTR --- 0 message 32 [0 0 0 0] ------- [0:255 -1:255 32 0]
r 0.037421112 _1_ RTR --- 0 message 32 [0 ffffffff 0 800] ------- [0:255 -1:255
32 0]
M 0.10000 0 (50.00, 50.00, 0.00), (50.00, 50.00), 15.00
M 0.10000 1 (100.00, 100.00, 0.00), (100.00, 100.00), 25.00
M 0.10000 2 (600.00, 600.00, 0.00), (600.00, 600.00), 25.00
s 0.182633994 _1_ RTR --- 1 message 32 [0 0 0 0] ------- [1:255 -1:255 32 0]
r 0.183694230 _0_ RTR --- 1 message 32 [0 ffffffff 1 800] ------- [1:255 -1:255
32 0]
s 0.882774710 _2_ RTR --- 2 message 32 [0 0 0 0] ------- [2:255 -1:255 32 0]
s 5.000000000 _0_ AGT --- 3 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
r 5.000000000 _0_ RTR --- 3 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
s 5.000000000 _0_ RTR --- 3 tcp 60 [0 0 0 0] ------- [0:0 1:0 32 1] [0 0] 0 0
s 5.000000000 _1_ AGT --- 4 tcp 40 [0 0 0 0] ------- [1:1 2:0 32 0] [0 0] 0 0
r 5.000000000 _1_ RTR --- 4 tcp 40 [0 0 0 0] ------- [1:1 2:0 32 0] [0 0] 0 0
r 5.004812650 _1_ AGT --- 3 tcp 60 [13a 1 0 800] ------- [0:0 1:0 32 1] [0 0] 1
0
s 5.004812650 _1_ AGT --- 5 ack 40 [0 0 0 0] ------- [1:0 0:0 32 0] [0 0] 0 0
r 5.004812650 _1_ RTR --- 5 ack 40 [0 0 0 0] ------- [1:0 0:0 32 0] [0 0] 0 0
s 5.004812650 _1_ RTR --- 5 ack 60 [0 0 0 0] ------- [1:0 0:0 32 0] [0 0] 0 0
r 5.006977357 _0_ AGT --- 5 ack 60 [13a 0 1 800] ------- [1:0 0:0 32 0] [0 0] 1
0
s 5.006977357 _0_ AGT --- 6 tcp 1040 [0 0 0 0] ------- [0:0 1:0 32 0] [1 0] 0 0
"lab8.tr" 128664L, 11456314C 1,1 Top

Trace File

Experiment No:5

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



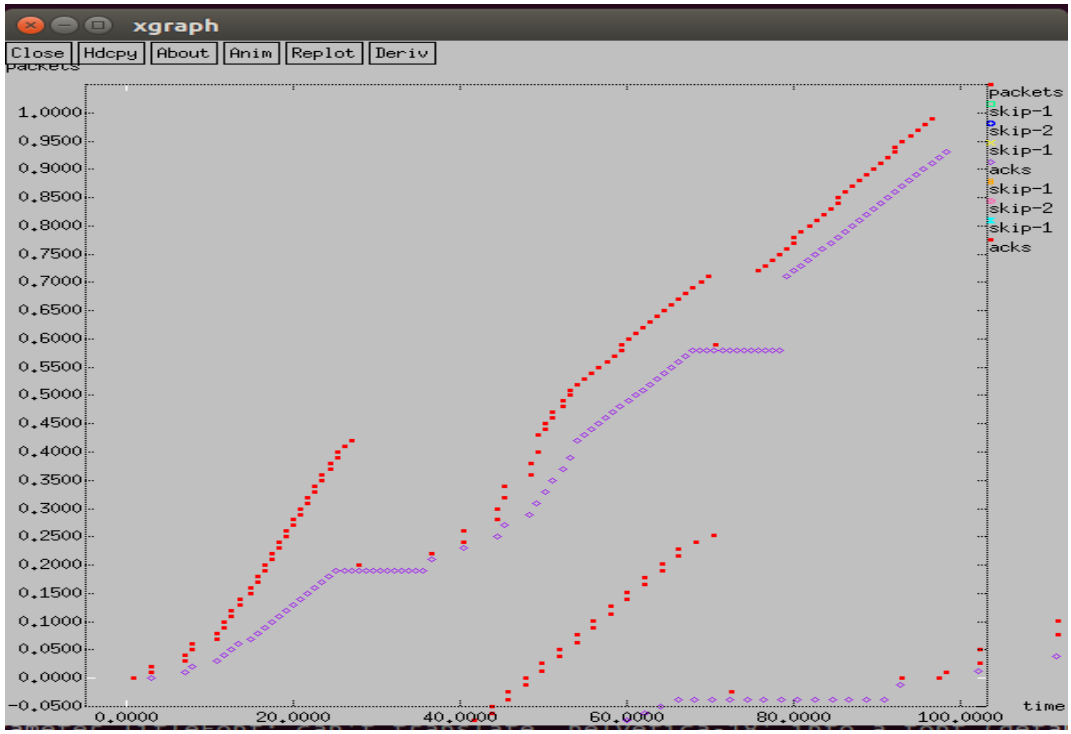

```
set opt(title) zero
set opt(stop) 100
set opt(ecn) 0
set opt(type) gsm
set opt(secondDelay) 55
set opt(minth) 30
set opt(maxth) 0
set opt(adaptive) 1
set opt(flows) 0
set opt(window) 30
set opt(web) 2
set opt(quiet) 0
```

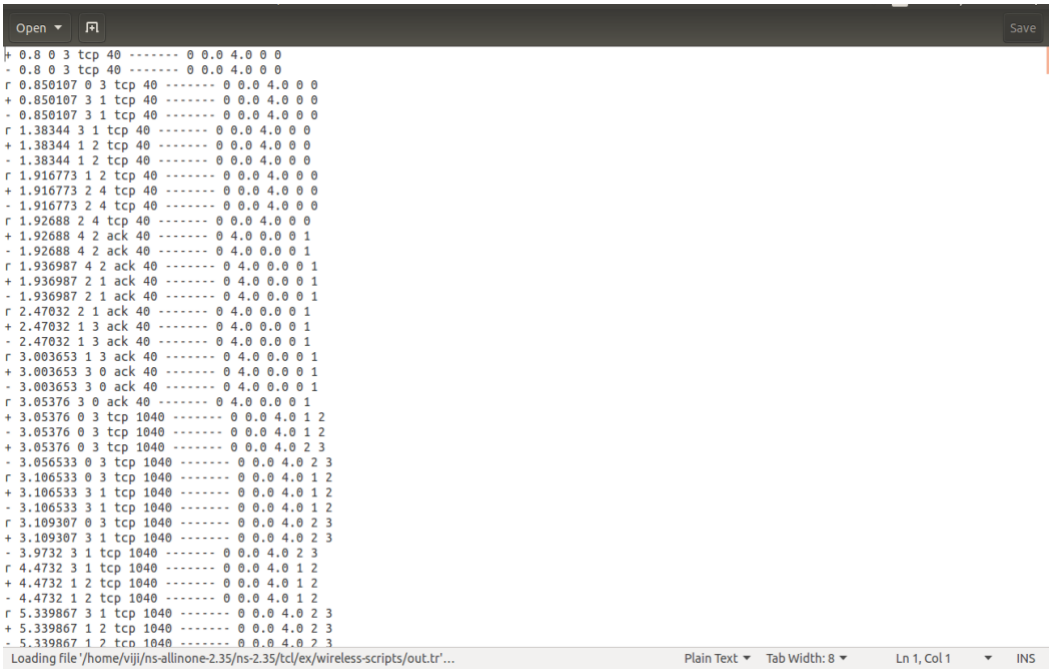
```
set opt(wrap) 100
set opt(srcTrace) is
set opt(dstTrace) bs2
set opt(gsmbuf) 10
set bwDL(gsm) 9600
set bwUL(gsm) 9600
set propDL(gsm) .500
set propUL(gsm) .500
set buf(gsm) 10
set ns [new Simulator]
set tf [open p5.tr w]
$ns trace-all $tf
set nf [open p5.nam w]
$ns namtrace-all $nf
set nodes(is) [$ns node]
set nodes(ms) [$ns node]
set nodes(bs1) [$ns node]
set nodes(bs2) [$ns node]
set nodes(lp) [$ns node]
proc cell_topo { } {
global ns nodes
$ns duplex-link $nodes(lp) $nodes(bs1) 3Mbps 10ms DropTail
$ns duplex-link $nodes(bs1) $nodes(ms) 1 1 RED
$ns duplex-link $nodes(ms) $nodes(bs2) 1 1 RED
$ns duplex-link $nodes(bs2) $nodes(is) 3Mbps 50ms DropTail
puts "Cell Topology"
}
```

```
proc set_link_params {t} {
global ns nodes bwUL bwDL propUL propDL buf
$ns bandwidth $nodes(bs1) $nodes(ms) $bwDL($t) simplex
$ns bandwidth $nodes(ms) $nodes(bs1) $bwUL($t) simplex
$ns bandwidth $nodes(bs2) $nodes(ms) $bwDL($t) simplex
$ns bandwidth $nodes(ms) $nodes(bs2) $bwUL($t) simplex
$ns delay $nodes(bs1) $nodes(ms) $propDL($t) simplex
$ns delay $nodes(ms) $nodes(bs1) $propDL($t) simplex
$ns delay $nodes(bs2) $nodes(ms) $propDL($t) simplex
$ns delay $nodes(ms) $nodes(bs2) $propDL($t) simplex
$ns queue-limit $nodes(bs1) $nodes(ms) $buf($t)
$ns queue-limit $nodes(ms) $nodes(bs1) $buf($t)
$ns queue-limit $nodes(bs2) $nodes(ms) $buf($t)
$ns queue-limit $nodes(ms) $nodes(bs1) $buf($t)
}
Queue/RED set summarystats_ true
Queue/DropTail set summarystats_ true
Queue/RED set adaptive_ $opt(adaptive)
Queue/RED set q_weight_ 0.0
Queue/RED set thresh_ $opt(minth)
Queue/RED set maxthresh_ $opt(maxth)
Queue/DropTail set shrink_drop_ true
Agent/TCP set ecn_ $opt(ecn)
Agent/TCP set window_ $opt(window)
DelayLink set avoidReordering_ true
switch $opt(type) {
gsm -
```

```
gprs -
umts {cell_topo}
}
set_link_params $opt(type)
$ns insert-delayer $nodes(ms) $nodes(bs1) [new Delayer]
$ns insert-delayer $nodes(bs1) $nodes(ms) [new Delayer]
$ns insert-delayer $nodes(ms) $nodes(bs2) [new Delayer]
$ns insert-delayer $nodes(bs2) $nodes(ms) [new Delayer]
if { $opt(flows) == 0 } {
set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]
set ftp1 [[set tcp1] attach-app FTP]
$ns at 0.8 "[set ftp1] start"
}
if {$opt(flows) > 0} {
set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]
set ftp1 [[set tcp1] attach-app FTP]
$tcp1 set window_ 100
$ns at 0.0 "[set ftp1] start"
$ns at 3.5 "[set ftp1] stop"
set tcp2 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]
set ftp2 [[set tcp2] attach-app FTP]
$tcp2 set window_ 3
$ns at 1.0 "[set ftp2] start"
$ns at 8.0 "[set ftp2] stop"
}
proc stop { } {
global nodes opt nf
```

```
set wrap $opt(wrap)
set sid [$nodes($opt(srcTrace)) id]
set did [$nodes($opt(dstTrace)) id]
if {$opt(srcTrace)=="is"} {
set a "-a p5.tr"
} else {
set a "p5.tr"
}
set GETRC "/home/sdit/ns-allinone-2.35/ns-2.35/bin/getrc"
set RAW2XG "/home/sdit/ns-allinone-2.35/ns-2.35/bin/raw2xg"
exec $GETRC -s $sid -d $did -f 0 p5.tr |¥
$RAW2XG -s 0.01 -m $wrap -r > plot.xgr
exec $GETRC -s $did -d $sid -f 0 p5.tr |¥
$RAW2XG -a -s 0.01 -m $wrap >> plot.xgr
exec $GETRC -s $sid -d $did -f 1 p5.tr |¥
$RAW2XG -s 0.01 -m $wrap -r >> plot.xgr
exec $GETRC -s $did -d $sid -f 1 p5.tr |¥
$RAW2XG -s 0.01 -m $wrap -a >> plot.xgr
exec nam p5.nam &
exec /home/sdit/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts/xg2gp.awk plot.xgr
if {!$opt(quiet)} {
exec xgraph -bb -tk -nl -m -x time -y packets plot.xgr &
}
exit 0
}
$ns at $opt(stop) "stop"
$ns run.
```


xgraph
Close
Hdcpy
About
Anim
Replot
Deriv
packets
1.0000
0.9500
0.9000
0.8500
0.8000
0.7500
0.7000
0.6500
0.6000
0.5500
0.5000
0.4500
0.4000
0.3500
0.3000
0.2500
0.2000
0.1500
0.1000
0.0500
0.0000
-0.0500
0.0000
20.0000
40.0000
60.0000
80.0000
100.0000
time
packets
skip-1
skip-2
skip-1
acks
skip-1
skip-2
skip-1
acks


Open
Save
+ 0.8 0 3 tcp 40 ------- 0 0.0 4.0 0 0
- 0.8 0 3 tcp 40 ------- 0 0.0 4.0 0 0
r 0.850107 0 3 tcp 40 ------- 0 0.0 4.0 0 0
+ 0.850107 3 1 tcp 40 ------- 0 0.0 4.0 0 0
- 0.850107 3 1 tcp 40 ------- 0 0.0 4.0 0 0
r 1.38344 3 1 tcp 40 ------- 0 0.0 4.0 0 0
+ 1.38344 1 2 tcp 40 ------- 0 0.0 4.0 0 0
- 1.38344 1 2 tcp 40 ------- 0 0.0 4.0 0 0
r 1.916773 1 2 tcp 40 ------- 0 0.0 4.0 0 0
+ 1.916773 2 4 tcp 40 ------- 0 0.0 4.0 0 0
- 1.916773 2 4 tcp 40 ------- 0 0.0 4.0 0 0
r 1.92688 2 4 tcp 40 ------- 0 0.0 4.0 0 0
+ 1.92688 4 2 ack 40 ------- 0 4.0 0.0 0 1
- 1.92688 4 2 ack 40 ------- 0 4.0 0.0 0 1
r 1.936987 4 2 ack 40 ------- 0 4.0 0.0 0 1
+ 1.936987 2 1 ack 40 ------- 0 4.0 0.0 0 1
- 1.936987 2 1 ack 40 ------- 0 4.0 0.0 0 1
r 2.47032 2 1 ack 40 ------- 0 4.0 0.0 0 1
+ 2.47032 1 3 ack 40 ------- 0 4.0 0.0 0 1
- 2.47032 1 3 ack 40 ------- 0 4.0 0.0 0 1
r 3.003653 1 3 ack 40 ------- 0 4.0 0.0 0 1
+ 3.003653 3 0 ack 40 ------- 0 4.0 0.0 0 1
- 3.003653 3 0 ack 40 ------- 0 4.0 0.0 0 1
r 3.05376 3 0 ack 40 ------- 0 4.0 0.0 0 1
+ 3.05376 0 3 tcp 1040 ------- 0 0.0 4.0 1 2
- 3.05376 0 3 tcp 1040 ------- 0 0.0 4.0 1 2
+ 3.05376 0 3 tcp 1040 ------- 0 0.0 4.0 2 3
- 3.056533 0 3 tcp 1040 ------- 0 0.0 4.0 2 3
r 3.106533 0 3 tcp 1040 ------- 0 0.0 4.0 1 2
+ 3.106533 3 1 tcp 1040 ------- 0 0.0 4.0 1 2
- 3.106533 3 1 tcp 1040 ------- 0 0.0 4.0 1 2
r 3.109307 0 3 tcp 1040 ------- 0 0.0 4.0 2 3
+ 3.109307 3 1 tcp 1040 ------- 0 0.0 4.0 2 3
- 3.9732 3 1 tcp 1040 ------- 0 0.0 4.0 2 3
r 4.4732 3 1 tcp 1040 ------- 0 0.0 4.0 1 2
+ 4.4732 1 2 tcp 1040 ------- 0 0.0 4.0 1 2
- 4.4732 1 2 tcp 1040 ------- 0 0.0 4.0 1 2
r 5.339867 3 1 tcp 1040 ------- 0 0.0 4.0 2 3
+ 5.339867 1 2 tcp 1040 ------- 0 0.0 4.0 2 3
Loading file '/home/viji/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts/out.tr'...
Plain Text
Tab Width: 8
Ln 1, Col 1
INS

Experiment No:6

Aim: Implement and study the performance of CDMA on NS2/NS3 (Using stack called Call net) or equivalent environment.

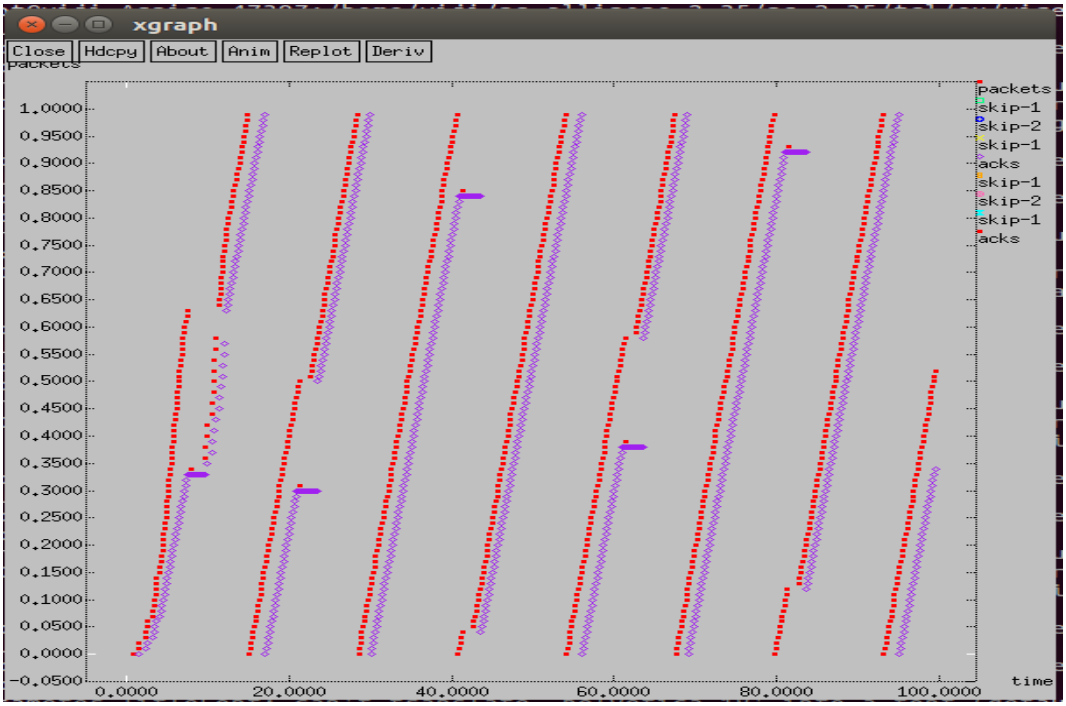
```
set opt(title) zero
;
set opt(stop) 100
;# Stop time.
set opt(ecn) 0
;

# Topology

set opt(type) umts ;#type of link:
#gsm gprs umts geo wlan_duplex wlan_ether wlan_complex
set opt(bwUL) 0
;# speed of congested link in kbps
set opt(bwDL) 0
;# speed of congested link in kbps
set opt(propUL) 0 ;# uplink delay of congested link in s
set opt(propDL) 0 ;# downlink delay of congested link in s
set opt(secondDelay) 55
;# average delay of access links in msace

# AQM parameters

set opt(maxth) 0
;
set opt(queue) DT
```

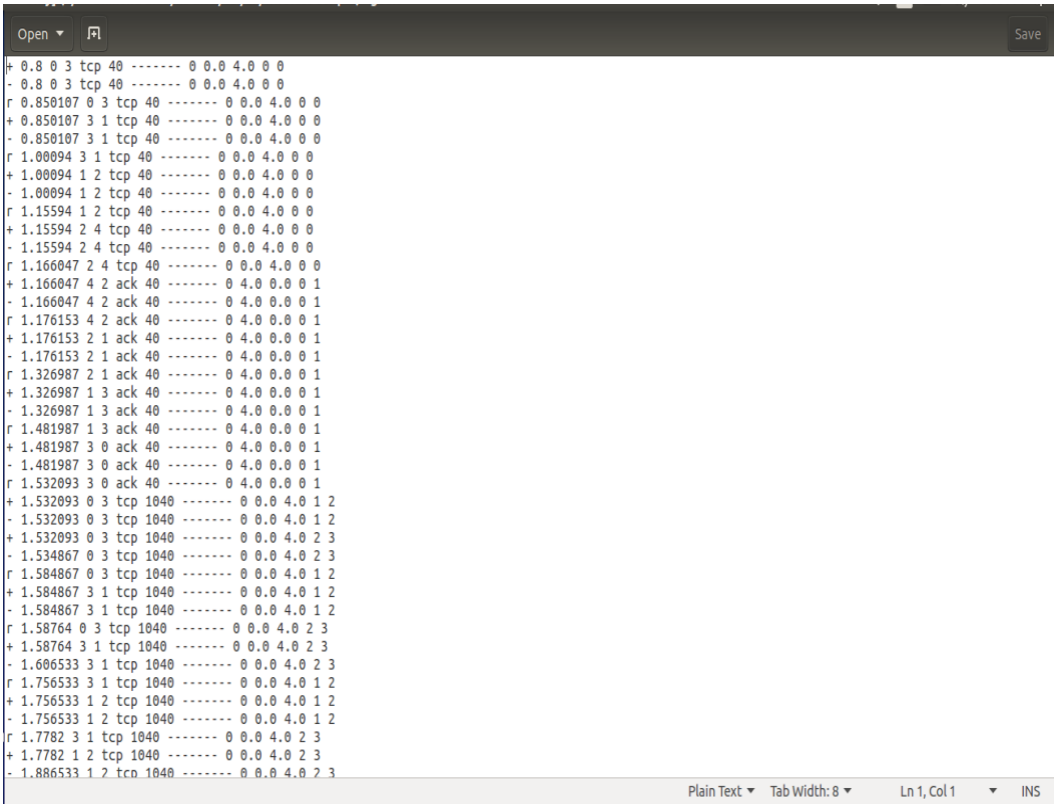
```
;# 0 for DropTail
;# 1 for RED
set opt(qsize) 0
;# Queue size in packets.

# Traffic generation.
set opt(flows) 1
;# number of long-lived TCP flows
set opt(shortflows) 0 ;# two short flows in the beginning
set opt(webers) 0
;# number of web users
set opt(window) 30 ;# window for long-lived traffic
set opt(smallpkt) 10 ;# inverse of fraction of TCP connections
set opt(web) 2
;# number of web sessions
set opt(pagesize) 10 ;# number of objects per page
set opt(objSize) 60 ;# average size of web object in pkts.
set opt(shape) 1.05 ;# shape parameter for Pareto distribution of web size
# a larger parameter means more small objects
set opt(interpage) 1 ;# interpage parameter for web traffic generator.

# Plotting statistics.

;# 1 to print link delays
set opt(quiet) 0
;# popup anything?
set opt(wrap) 90 ;# wrap plots?
set opt(srcTrace) is ;# where to plot traffic
set opt(dstTrace) bs ;# where to plot traffic
set opt(umtsbuf) 20
; # buffer size for umts


; #number of feedback reports in TFRC per RTT
#default downlink bandwidth in bps
set bwDL(umts) 384000
#default uplink bandwidth in bps
set bwUL(umts) 64000
#default downlink propagation delay in seconds
set propDL(umts) .150
#default uplink propagation delay in seconds
```

```
set propUL(umts) .150
#default buffer size in packets
set buf(umts) 20

proc cell_topo {} {
global ns nodes qm
$ns duplex-link $nodes(bs) $nodes(ms) 1 1 $qm
$ns duplex-link $nodes(lp) $nodes(ms) 3Mbps 10ms DropTail
$ns duplex-link $nodes(bs) $nodes(is) 3Mbps 50ms DropTail
puts "cell topology"
}
proc set_link_params {t} {
global ns nodes bwUL bwDL propUL propDL buf
$ns bandwidth $nodes(bs) $nodes(ms) $bwDL($t) simplex
$ns bandwidth $nodes(ms) $nodes(bs) $bwUL($t) simplex
$ns delay $nodes(bs) $nodes(ms) $propDL($t) simplex
$ns delay $nodes(ms) $nodes(bs) $propDL($t) simplex
$ns queue-limit $nodes(bs) $nodes(ms) $buf($t)
$ns queue-limit $nodes(ms) $nodes(bs) $buf($t)
}
if {$opt(queue) == "DT"} {
set qm DropTail
}
if {$opt(queue) == "RED"} {
set qm RED
}
set tcpTick_ 0.01
set pktsize 1460

Agent/TCP set tcpTick_ $tcpTick_
set out $opt(title)
proc stop {} {
global nodes opt
set wrap $opt(wrap)
set sid [$nodes($opt(srcTrace)) id]
set did [$nodes($opt(dstTrace)) id]
if {$opt(srcTrace) == "bs"} {
set a "-a umts.tr"
} else {
set a "umts.tr"
```

```
}
set GETRC "/home/sdit/ns-allinone-2.35/ns-2.35/bin/getrc"
set RAW2XG "/home/sdit/ns-allinone-2.35/ns-2.35/bin/raw2xg"
exec $GETRC -s $sid -d $did -f 0 umts.tr | ¥
$RAW2XG -s 0.01 -m $wrap -r > plot.xgr
exec $GETRC -s $did -d $sid -f 0 umts.tr | ¥
$RAW2XG -a -s 0.01 -m $wrap >> plot.xgr
exec $GETRC -s $sid -d $did -f 1 umts.tr | ¥
$RAW2XG -s 0.01 -m $wrap -r >> plot.xgr
exec $GETRC -s $did -d $sid -f 1 umts.tr | ¥
$RAW2XG -s 0.01 -m $wrap -a >> plot.xgr
exec /home/sdit/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts/xg2gp.awk plot.xgr
if {!$opt(quiet)} {
exec xgraph -bb -tk -nl -m -x time -y packets plot.xgr &
}
exit 0
}

set ns [new Simulator]
variable delayerUL
variable delayerDL
# RED and TCP parameters
Queue/RED set summarystats_ true
Queue/DropTail set summarystats_ true
Queue/RED set maxthresh_ $opt(maxth)
Queue/DropTail set shrink_drops_ true
Agent/TCP set ecn_ $opt(ecn)
Agent/TCP set packetSize_ $pktsize
Agent/TCP set window_ $opt(window)
DelayLink set avoidReordering_ true
# Create trace for latency measured with ping
set tf [open umts.tr w]
$ns trace-all $tf

# nodes with hierarhical routing and has no params
set nodes(lp) [$ns node]
set nodes(ms) [$ns node]
set nodes(bs) [$ns node]
set nodes(is) [$ns node]
# Create topology
```

```
switch $opt(type) {
umts {cell_topo}
}
set_link_params $opt(type)
set delayerDL [new Delayer]
set delayerUL [new Delayer]
$ns insert-delayer $nodes(ms) $nodes(bs) $delayerUL
$ns insert-delayer $nodes(bs) $nodes(ms) $delayerDL
# Set up forward TCP connection
if {$opt(flows) > 0} {
set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]
set ftp1 [[set tcp1] attach-app FTP]
$ns at 0.8 "[set ftp1] start"
}
if {$opt(shortflows) > 0} {
set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]
set ftp1 [[set tcp1] attach-app FTP]
$tcp1 set window_ 100
$ns at 0.0 "[set ftp1] start"
$ns at 3.5 "[set ftp1] stop"
set tcp2 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]
set ftp2 [[set tcp2] attach-app FTP]
$tcp2 set window_ 3
$ns at 1.0 "[set ftp2] start"
$ns at 8.0 "[set ftp2] stop"
}
# Add forward web traffic.
set req_trace_ 0
set count $opt(webers)
if ($count) {
add_web_traffic $opt(secondDelay) $opt(web) $opt(interpage) $opt(pagesize)
$opt(objSize) $opt(shape) 1
add_web_traffic $opt(secondDelay) [expr $opt(web)/2] $opt(interpage) $opt(pagesize)
$opt(objSize) $opt(shape) 0
}
$ns at $opt(stop) "stop"
$ns run
```


xgraph
Close
Hdcpy
About
Anim
Replot
Deriv
packets
1.0000
0.9500
0.9000
0.8500
0.8000
0.7500
0.7000
0.6500
0.6000
0.5500
0.5000
0.4500
0.4000
0.3500
0.3000
0.2500
0.2000
0.1500
0.1000
0.0500
0.0000
-0.0500
0.0000
20.0000
40.0000
60.0000
80.0000
100.0000
time
packets
skip-1
skip-2
skip-1
acks
skip-1
skip-2
skip-1
acks


Open
Save
+ 0.8 0 3 tcp 40 ------- 0 0.0 4.0 0 0
- 0.8 0 3 tcp 40 ------- 0 0.0 4.0 0 0
r 0.850107 0 3 tcp 40 ------- 0 0.0 4.0 0 0
+ 0.850107 3 1 tcp 40 ------- 0 0.0 4.0 0 0
- 0.850107 3 1 tcp 40 ------- 0 0.0 4.0 0 0
r 1.00094 3 1 tcp 40 ------- 0 0.0 4.0 0 0
+ 1.00094 1 2 tcp 40 ------- 0 0.0 4.0 0 0
- 1.00094 1 2 tcp 40 ------- 0 0.0 4.0 0 0
r 1.15594 1 2 tcp 40 ------- 0 0.0 4.0 0 0
+ 1.15594 2 4 tcp 40 ------- 0 0.0 4.0 0 0
- 1.15594 2 4 tcp 40 ------- 0 0.0 4.0 0 0
r 1.166047 2 4 tcp 40 ------- 0 0.0 4.0 0 0
+ 1.166047 4 2 ack 40 ------- 0 4.0 0.0 0 1
- 1.166047 4 2 ack 40 ------- 0 4.0 0.0 0 1
r 1.176153 4 2 ack 40 ------- 0 4.0 0.0 0 1
+ 1.176153 2 1 ack 40 ------- 0 4.0 0.0 0 1
- 1.176153 2 1 ack 40 ------- 0 4.0 0.0 0 1
r 1.326987 2 1 ack 40 ------- 0 4.0 0.0 0 1
+ 1.326987 1 3 ack 40 ------- 0 4.0 0.0 0 1
- 1.326987 1 3 ack 40 ------- 0 4.0 0.0 0 1
r 1.481987 1 3 ack 40 ------- 0 4.0 0.0 0 1
+ 1.481987 3 0 ack 40 ------- 0 4.0 0.0 0 1
- 1.481987 3 0 ack 40 ------- 0 4.0 0.0 0 1
r 1.532093 3 0 ack 40 ------- 0 4.0 0.0 0 1
+ 1.532093 0 3 tcp 1040 ------- 0 0.0 4.0 1 2
- 1.532093 0 3 tcp 1040 ------- 0 0.0 4.0 1 2
+ 1.532093 0 3 tcp 1040 ------- 0 0.0 4.0 2 3
- 1.534867 0 3 tcp 1040 ------- 0 0.0 4.0 2 3
r 1.584867 0 3 tcp 1040 ------- 0 0.0 4.0 1 2
+ 1.584867 3 1 tcp 1040 ------- 0 0.0 4.0 1 2
- 1.584867 3 1 tcp 1040 ------- 0 0.0 4.0 1 2
r 1.58764 0 3 tcp 1040 ------- 0 0.0 4.0 2 3
+ 1.58764 3 1 tcp 1040 ------- 0 0.0 4.0 2 3
- 1.606533 3 1 tcp 1040 ------- 0 0.0 4.0 2 3
r 1.756533 3 1 tcp 1040 ------- 0 0.0 4.0 1 2
+ 1.756533 1 2 tcp 1040 ------- 0 0.0 4.0 1 2
- 1.756533 1 2 tcp 1040 ------- 0 0.0 4.0 1 2
r 1.7782 3 1 tcp 1040 ------- 0 0.0 4.0 2 3
+ 1.7782 1 2 tcp 1040 ------- 0 0.0 4.0 2 3
- 1.886533 1 2 tcp 1040 ------- 0 0.0 4.0 2 3
Plain Text
Tab Width: 8
Ln 1, Col 1
INS

PART-B

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

```
import java.util.Scanner;

public class CRCDemoFinal {
	static String msg;
	static String genpoly="1011";
	static char t[]=new char[128];
	static char g[]=new char[128];
	static char cs[]=new char[128];
	static int mlen,glen,x,c,flag=0,test ;
	public static void main(String[] args)
	{
		Scanner in = new Scanner(System.in);

		System.out.println("Enter the message to be transmitted");

		msg=in.nextLine();

		mlen=msg.length();

			for(int i=0;i<mlen;i++)

				t[i]=msg.charAt(i);
```

```
System.out.println("Predefined generator polynomial is "+genpoly);

        g=genpoly.toCharArray();

        glen=genpoly.length();

                for (x=mlen;x<(mlen+glen-1);x++)
                        t[x]='0';
System.out.println("Zero extended message is"+ new String(t));

                crc();

System.out.println("Checksum is"+ new String(cs));

        for(x=mlen;x<mlen+glen-1;x++)
                {
                        t[x]=cs[x-mlen];
                }
System.out.println("Final codeword is"+ new String(t));

System.out.println("test error detection 1(yes) 0(no)");

                test=in.nextInt();

                if(test==1)
                {
        System.out.println("Enter the position error to be inserted:");
```

```
        x=in.nextInt();
        t[x]=(t[x]=='0')?'1':'0';
System.out.println("Errorneous data is"+new String(t));



}

crc();

for(x=0;x<(glen-1);x++)
{
  if(cs[x]== '1')

  {
        flag=1;
      break;
  }
}


if(flag == 1)

  System.out.print("  Error was detected during transfer");

else

  System.out.print("No Error was detected during transfer");
```

```
        }


public static void crc()
    {
    for(x=0;x<glen;x++)

        cs[x]=t[x];

        do
        {
            if (cs[0]=='1')

            xor();

            for(c=0;c<glen-1;c++)
            {
                cs[c]=cs[c+1];
            }
            cs[c]=t[x++];


        }while(x<=mlen+glen-1);

    }
```

```
        public static void xor()
        {
            for(c=1;c<glen;c++)
                cs[c]=((cs[c]==g[c])?'0':'1');
        }
    }
```

Output 1:

```
Enter the message to be transmitted
101101
Predefined generator polynomial is 1011
Zero extended message is101101000
Checksum is011
Final codeword is101101011
test error detection 1(yes) 0(no)
1
Enter the position error to be inserted:
3
Errorneous data is101001011
```

Output 2:

```
Enter the message to be transmitted
101101
Predefined generator polynomial is 1011
Zero extended message is101101000
Checksum is011
Final codeword is101101011
```

```
test error detection 1(yes) 0(no)
0
No Error was detected during transfer
```

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

```
import java.util.*;
public class BellmanDemo {
	static Scanner in =new Scanner(System.in);

	public static void main(String[] args)
	{
		int V,E=0,cn=0;

		int w[][]=new int[20][20];

		int edge[][]=new int[50][2];

		System.out.println("Enter the number of vertices");

		V= in.nextInt();

		System.out.println("Enter the weight matrix");

		for(int i=0;i<V;i++)
```

```
        for(int j=0;j<V;j++)
        {
                w[i][j]=in.nextInt();

                if(w[i][j]!=0)
                {
                        edge[E][0]=i;
                        edge[E++][1]=j;
                }
        }


        cn=bellmanford(w,V,E,edge);

        if(cn==1)

                System.out.println("No negative weight cycle¥n");

        else
                System.out.println("negative weight cycle exit¥n");

}

public static int bellmanford(int w[][],int V, int E,int edge[][])
{
        int u,v,s,flag=1;
        int distance[]=new int[20];
        int parent[]=new int[20];
```

```
for(int i=0;i<V;i++)
{
      distance[i]=999;
      parent[i]=-1;
}
System.out.println("Enter the source vertex");
s=in.nextInt();
distance[s-1]=0;
for(int i=0;i<V-1;i++)
{
      for(int k=0;k<E;k++)
      {
            u=edge[k][0];
            v=edge[k][1];
            if(distance[u]+w[u][v]< distance[v])
            {
                  distance[v]= distance[u]+w[u][v];
            parent[v]=u;
            }

      }
 }
for(int k=0;k<E;k++)
{
      u=edge[k][0];
      v=edge[k][1];
      if(distance[u]+w[u][v]< distance[v])
```

```
                        flag=0;
            }
            if (flag==1)
                  for(int i=0;i<V;i++)
            System.out.println("vertex"+(i+1)+ "-> cost" + distance[i]+ "parent=" +
(parent[i]+1));
                        return flag;
            }
}
```

Output:

```
Enter the number of vertices
3
Enter the weight matrix
0 5 1
999 0 999
999 2 0
Enter the source vertex
1
vertex1-> cost0parent=0
vertex2-> cost3parent=3
vertex3-> cost1parent=1
No negative weight cycle
```

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. Implement the above program using as message queues or FIFO as IPC channel.

ClientSide Program:

```
import java.util.* ;
import java.io.*;
import java.net.*;
public class SocketClient
{
	public static void main(String args[]) throws Exception
	{
	Scanner in =new Scanner(System.in);

	Socket sock =new Socket("127.0.0.1",4000);
```

```
System.out.println("Enter the filename to transfer");

String fname = in.nextLine();

OutputStream ostream = sock.getOutputStream();

PrintWriter pwrite= new PrintWriter(ostream,true);
pwrite.println(fname);

InputStream istream= sock.getInputStream();

BufferedReader socketRead = new BufferedReader(new InputStreamReader(istream));

String str;

while((str=socketRead.readLine())!=null)

{
System.out.println(str);

}
pwrite.close();

socketRead.close();

}
}
```

Server Side Program:

```
import java.io.*;

import java.net.*;

public class SocketServer

{
public static void main(String args[]) throws Exception
{

ServerSocket servsock = new ServerSocket(4000);

System.out.println("Server ready for connection");

Socket sock = servsock.accept();

System.out.println("connection is succesfull and ready for file trasfer");

InputStream istream=sock.getInputStream() ;

BufferedReader fileRead = new BufferedReader(new InputStreamReader(istream));

String fname = fileRead.readLine();

BufferedReader contentRead = new BufferedReader(new FileReader(fname));
```

```
OutputStream ostream = sock.getOutputStream();

PrintWriter pwrite= new PrintWriter(ostream,true);

String str;

while((str=contentRead.readLine())!=null)

{

pwrite.println(str);

}
sock.close();

servsock.close();

pwrite.close();

fileRead.close();

contentRead.close();
}
}
```

Output:

Open two Terminals.

First open server side program compile and run the program in terminal.

```
[sdit@localhost ~]$ cd /home/sdit/cnlab/

[sdit@localhost cnlab]$ cd cands /src

[sdit@localhost src]$ javac SocketServer.java

[sdit@localhost src]$ java  SocketServer

Server ready for connection
connection is succesfull and ready for file trasfer
```

Open Client Terminal next and compile and run progrm. Dont close server terminal

```
[sdit@localhost src]$ javac SocketClient.java

[sdit@localhost src]$ java SocketClient

Enter the filename to transfer

1.txt
[sdit@localhost src]$ java SocketClient

Enter the filename to transfer

new.txt

[sdit@localhost src]$ java SocketClient
```

```
Enter the filename to transfer

2.txt

hello,how are you?
```

4.Write a program on datagram socket for client/server to display the message on client side,typed at the server side.

```
import java.util.*;

//import java.io.*;
//import java.net.*;
import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

public class DatagramSocketServer
{

      public static void main(String args[]) throws Exception
```

```
{

        DatagramSocket  serverSocket = new DatagramSocket(9000);

        Scanner in = new Scanner(System.in);

        byte[] receiveData = new byte[1024];

              byte[] sendData = new byte[1024];

        System.out.println("******Server Display Terminal******");

        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);

        serverSocket.receive(receivePacket);

        System.out.println(new String(receivePacket.getData()));
        InetAddress IPAddress = receivePacket.getAddress();

              System.out.println(IPAddress);

              int port = receivePacket.getPort();

        System.out.println(port);

        while(true)
        {
```

```
                System.out.println("type some message to display at client side");

                        String message = in.nextLine();

                sendData = message.getBytes();

DatagramPacket sendPacket = new
DatagramPacket(sendData,sendData.length,IPAddress,port);

                serverSocket.send(sendPacket);

            }


        }

}
```

Client Side

```
import java.net.*;

public class DatagramClient {

        public static void main(String args[]) throws Exception
        {
                String line="Connected with the Client";
```

```
        DatagramSocket ClientSocket = new DatagramSocket();

        InetAddress IPAddress = InetAddress.getByName("localhost");
        byte[] SendData = new byte[1024];

        byte[] receiveData = new byte[1024];

        SendData=line.getBytes();

        DatagramPacket SendPacket = new DatagramPacket(SendData,SendData.length,IPAddress,9000);

        ClientSocket.send(SendPacket);

        System.out.println("CLIENT DISPLAY TERMINAL");

        while(true)
        {
            DatagramPacket receivePacket=new DatagramPacket(receiveData,receiveData.length);

            ClientSocket.receive(receivePacket);

            String messageReceived = new String(receivePacket.getData());
            System.out.println("MESSAGE TYPED AT SERVER SIDE IS:"+messageReceived);

        }

    }

}
```

Output:

Open two Terminals.

First open server side program compile and run the program in terminal.

```
[sdit@localhost cnlab]$ cd DatagramServerandClinet
[sdit@localhost DatagramServerandClinet]$ cd src
[sdit@localhost src]$ javac DatagramSocketServer.java
[sdit@localhost src]$ java DatagramSocketServer

******Server Display Terminal******
connected with the Client
/127.0.0.1
49453
type some message to display at client side
hello
type some message to display at client side
```

```
[sdit@localhost src]$ javac DatagramSocketClient.java
[sdit@localhost src]$ java DatagramSocketClient

******Client Display Terminal******
Server :hello
```

5.Write a program for simple RSA algorithm to encrypt and decrypt the data.

```
import java.util.*;
public class RsaDemo {

	public static void main(String[] args)
	{
		String msg;
		int pt[]=new int[100];
		int ct[]=new int[100];
		int z,d,n,e,p,q,mlen;
		Scanner in=new Scanner(System.in);

		do
		{
			System.out.println("enter the two prime number for p and q ");
			p=in.nextInt();
			q=in.nextInt();
		}while(prime(p)==0 || prime(q)==0);

			n=p*q;
			z=(p-1)*(q-1);
```

```
for(e=2;e<z;e++)
{
        if(gcd(e,z)==1)
                break;
}

System.out.println("Encryption key e is "+e);
System.out.println("Public key is (e,n) : "+e+","+n);

for(d=2;d<z;d++)
{
        if((e*d)%z == 1)
                break;
}

System.out.println("Decryption Key is "+d);
System.out.println("Private key is (d,n) => "+d+","+n);
in.nextLine();

System.out.println("enter the message for encryption ");
        msg=in.nextLine();
        mlen=msg.length();

        for(int i=0;i<mlen;i++)
                pt[i]=msg.charAt(i);

System.out.println("ASCII valus of PT array is ");
```

```
                for(int i=0;i<mlen;i++)
                        System.out.println(pt[i]);

            System.out.println("Encryption:Cipher text: ");
                for(int i=0;i<mlen;i++)
                        ct[i]=mu/ti(pt[i],e,n);
                for(int i=0;i<mlen;i++)
                        System.out.println(ct[i]+"¥t");
            System.out.println("¥n Decryption:Plain text: ");

                for(int i=0;i<mlen;i++)
                        pt[i]=mu/ti(ct[i],d,n);

                for(int i=0;i<mlen;i++)
                        System.out.println(pt[i]+"¥t");

                for(int i=0;i<mlen;i++)
                        System.out.println((char)pt[i]);
                        System.out.println("¥n");
    }

    public static int gcd(int x,int y)
    {
        if(y==0)
            return x;
        else
            return gcd(y,x%y);
```

```
	}

	public static int prime(int num)
	{
		int i;
		for(i=2;i<= num/2;i++)
		{
			if(num%i == 0)
				return 0;
		}
		return 1;
	}

	public static int multi(int base,int exp,int n)
	{
		int res=1,j;

		for(j=1;j<=exp;j++)
			res=((res*base)%n);
		return res;
	}

}
```

Output:

```
enter the two prime number for p and q
11 13
Encryption key e is 7
Public key is (e,n) : 7,143
Decryption Key is 103
Private key is (d,n) => 103,143
enter the message for encryption
hello
ASCII valus of PT array is
104
101
108
108
111
Encryption:Cipher text:
91
62
4
4
45

 Decryption:Plain text:
104
101
108
108
```

111

h

e

l

l

o

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

```
import java.util.Scanner;
import java.util.Random;
public class LeakyDemo
{
            public static void main(String args[])
            {
                  int bcktsize,j,iter,oprate,line,total=0,rem_bytes;

                  int pckt[]=new int[25];

                  Scanner in=new Scanner(System.in);

                  System.out.println("Enter the numbr of input lines");

                  line=in.nextInt();
```

```
System.out.println("Enter the numbr of iterations");

iter=in.nextInt();

System.out.println("Enter the bucketsize and output rate");

bcktsize=in.nextInt();

oprate=in.nextInt();

Random rand=new Random();

for(int i=0;i<iter;i++)
{
      System.out.println("Iteration"+(i+1));

      for(j=0;j<line;j++)
      {
            rand.setSeed(System.nanoTime());

            pckt[j]=rand.nextInt(25);

            total+=pckt[j];

            if(total<=bcktsize)
            {
```

```
                                    System.out.println("Input from line"+(j+1)+"with
packet size"+pckt[j]+"bytes are accepted for the biucket,no. of bytes in bucket
is"+total);
                              }

                              else

                              {
                                    total-=pckt[j];
                                    System.out.println("Input from line" +(j+1)+" with
packet size " +pckt[j]+" bytes are discarded for the bucket,no. of bytes in bucket
is"+total);
                              }
                        }

                        if(oprate>total)

                        {
                              rem_bytes=total;
                              total=0;
                              System.out.println("-------");
                              System.out.println("Output rate of bucket is" +oprate+"
bytes sent out of bucket is"+rem_bytes);
                              System.out.println("-------");
                        }

                        else

                        {
```

```
                                total-=oprate;
                                System.out.println("-------");
                                System.out.println("Output rate of bucket is" +oprate+"
no. of bytes remaining in bucket is"+total);
                                System.out.println("-------");
                            }
                        }
                    }
}
```

Output;Enter the numbr of input lines

2

Enter the numbr of iterations

2

Enter the bucketsize and output rate

15

8

Iteration1

Input from line1 with packet size10 bytes are accepted for the bucket,no. of bytes in bucket is10

Input from line2 with packet size17 bytes are discarded for the bucket,no. of bytes in bucket is10

Output rate of bucket is 8 no. of bytes remaining in bucket is2

Iteration2

Input from line1 with packet size 3 bytes are accepted for the bucket,no. of bytes in bucket is5

Input from line2 with packet size10bytes are accepted for the bucket,no. of bytes in bucket is15

Output rate of bucket is 8 no. of bytes remaining in bucket is7
