SRI SAI RAM COLLEGE OF ENGINEERING

ANEKAL

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

A

MANUAL FOR COMPUTER NETWORKS LAB

(18CSL57) V SEMESTER

Prepared By:

Prof. Mala B A

Assistant Professor, Dept., of CSE, SSCE

COMPUTER NETWORK LABORATORY (Effective from the academic year 2018 -2019)

SEMESTER – V	Course Code 18CSL57
CIE Marks 40	Number of Contact Hours/Week 0:2:2
SEE Marks 60	Total Number of Lab Contact Hours 36
Exam Hours 03	Credits – 2

Course Learning Objectives:

This course (18CSL57) will enable students to:

- Demonstrate operation of network and its management commands
- Simulate and demonstrate the performance of GSM and CDMA
- Implement data link layer and transport layer protocols.

PART A

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

1. Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped.
2. Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
3. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.
4. Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.
5. Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.
6. Implement and study the performance of CDMA on NS2/NS3 (Using stack called Call net) or equivalent environment

PART B

Implement the following in Java:

7. Write a program for error detecting code using CRC-CCITT (16- bits).
8. Write a program to find the shortest path between vertices using bellman-ford

Algorithm.

9. 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.

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

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

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

Procedure For Conduction of Practical Examination

1. All laboratory experiments are to be included for practical examination.
2. Students are allowed to pick one experiment from part A and part B with lot.
3. Strictly follow the instructions as printed on the cover page of answer script
4. Marks distribution: Procedure + Conduction + Viva: 80

   Part A: 10+25+5 =40

   Part B: 10+25+5 =40

5.Change of experiment is allowed only once and marks allotted to the procedure part to be made zero.

COURSE OUTCOMES

The students should be :

CO1: Able to Analyze the working of networking protocols using modern tool NS2.

CO2: Able to Develop wired and wireless topology using XGraph, NAM in NS2.

CO3: Able to Simulate and demonstrate the performance of GSM and CDMA.

CO4: Able to Apply and develop the algorithms in data link layer, Network layer and application layer.

CO5: Able to Design client-server applications using TCP and UDP socket IPC.

TABLE OF CONTENTS

SL.NO	CONTENT	PAGE NO
1.	LAB SYLLABUS	5
2.	PROGRAM NO 1	19
3.	PROGRAM NO 2	22
4.	PROGRAM NO 3	26
5.	PROGRAM NO 4	29
6.	PROGRAM NO 5	33
7.	PROGRAM NO 6	38
8.	PROGRAM NO 7	43
9.	PROGRAM NO 8	47
10.	PROGRAM NO 9	50
11.	PROGRAM NO 10	52
12.	PROGRAM NO 11	54
13.	PROGRAM NO 12	56
14.	VIVA QUESTIONS	59
15.	CONTENT BEYOND SYLLABUS	62

Introduction to NS-2

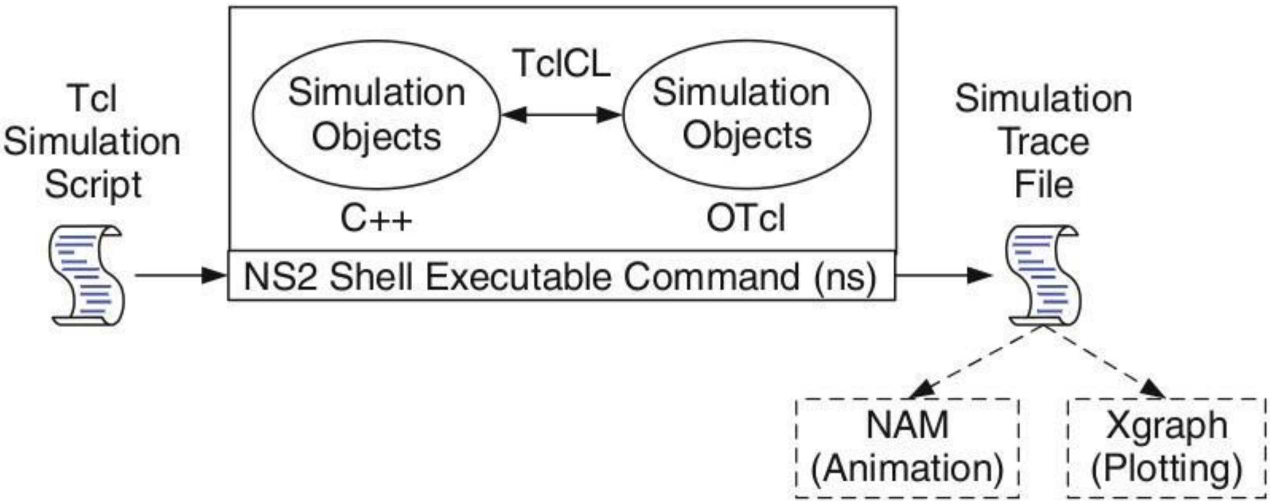
• Widely known as NS2, is simply an event driven simulation tool.

• 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 behaviours.

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 data 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]
```

We created a node that 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 behaviour 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

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

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 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

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.

```
$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"
```

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. Gives the packet type (eg CBR or TCP)
3. Gives the packet size
4. Some flags
5. 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.
6. This is the source address given in the form of "node.port".
7. This is the destination address, given in the same form.
8. 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

9. 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 {
count =count+1

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

THE –f OPTION: STORING awk PROGRAMS INA 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

The FILENAME Variable: FILENAME stores the name of the current file being processed. Like grep and sed, awk can also handle multiple filenames in the command line. By default, awk doesn’t print the filename, but you can instruct it to do so:

‘$6<4000 {print FILENAME, $0 }’

With FILENAME, you can device logic that does different things depending on the file that is processed.

NS2 Installation

- NS2 is a free simulation tool.
- It runs on various platforms including UNIX (or Linux), Windows, and Mac systems.
- NS2 source codes are distributed in two forms: the all-in-one suite and the component- wise.
- ‘all-in-one’ package provides an “install” script which configures the NS2 environment and creates NS2 executable file using the “make” utility.

NS-2 installation steps in Linux

- Go to **Computer File System** now paste the zip file **“ns-allinone-2.34.tar.gz”** into opt folder.
- Now **unzip** the file by typing the following **command**
  [root@localhost opt] # **tar -xzvf ns-allinone-2.34.tar.gz**
- After the files get extracted, we get ns-allinone-2.34 folder as well as zip file ns-allinone-2.34.tar.gz
  [root@localhost opt] # **ns-allinone-2.34 ns-allinone-2.34.tar.gz**

- Now go to ns-allinone-2.33 folder and install it
  [root@localhost opt] # **cd ns-allinone-2.34**
  [root@localhost ns-allinone-2.33] # **./install**
- Once the installation is completed successfully we get certain pathnames in that terminal which must be pasted in **".bash_profile"** file.
- First **minimize the terminal** where installation is done and **open a new terminal** and open the file **".bash_profile"**
  [root@localhost ~] # **vi .bash_profile**

- When we open this file, we get a line in that file which is shown below
  PATH=$PATH:$HOME/bin

To this line we must paste the path which is present in the previous terminal where **ns was installed**. First put **":"** then paste the path in-front of bin. That path is shown below. **":/opt/ns-allinone-2.33/bin:/opt/ns-allinone-2.33/tcl8.4.18/unix:/opt/ns-allinone- 2.33/tk8.4.18/unix".**

- In the next line type **"LD_LIBRARY_PATH=$LD_LIBRARY_PATH:"** and paste the **two paths** separated by **":"** which are present in the previous terminal i.e **Important notices section (1)**

"/opt/ns-allinone-2.33/otcl-1.13:/opt/ns-allinone-2.33/lib"

- In the next line type **"TCL_LIBRARY=$TCL_LIBRARY:"** and paste the path which is present in previous terminal i.e **Important Notices section (2)**

"/opt/ns-allinone-2.33/tcl8.4.18/library"

- In the next line type **"export LD_LIBRARY_PATH"**
- In the next line type **"export TCL_LIBRARY"**
- The next two lines are already present the file **"export PATH"** and **"unsetUSERNAME"**
- **Save the program (ESC + shift : wq and press enter)**
- Now in the terminal where we have opened **.bash_profile** file, type the following command to **check if path is updated correctly or not**

  [root@localhost ~] # **vi .bash_profile**

  [root@localhost ~] # **source .bash_profile**

- If **path is updated properly**, then we will **get the prompt** as shown below
  [root@localhost ~] #
- Now open the previous terminal where you have installed **ns**
  [root@localhost ns-allinone-2.33] #
- Here we need to configure three packages **"ns-2.33", "nam-1.13" and "xgraph-12.1"**
- **First**, configure **"ns-2.33"** package as shown below
  [root@localhost ns-allinone-2.33] # **cd ns-2.33**
  [root@localhost ns-2.33] # **./configure**
  [root@localhost ns-2.33] # **make clean**
  [root@localhost ns-2.33] # **make** [root@localhost ns-2.33] # **make install** [root@localhost ns-2.33] # **ns**

%

- If we get **"%"** symbol it indicates that **ns-2.33 configuration** was **successful**.
- **Second,** configure **"nam-1.13"** package as shown below
  [root@localhost ns-2.33] # **cd ..**
  [root@localhost ns-allinone-2.33] # **cd nam-1.13**
  [root@localhost nam-1.13] # **./configure**

[root@localhost nam-1.13] # **make clean**
[root@localhost nam-1.13] # **make** [root@localhost nam-1.13] # **make install** [root@localhost nam-1.13] # **ns**

%

- If we get **"%"** symbol it indicates that **nam-1.13 configuration** was **successful**.
- **Third,** configure **"xgraph-12.1"** package as shown below

[root@localhost nam-1.13] # **cd ..**
[root@localhost ns-allinone-2.33] # **cd xgraph-12.1**
[root@localhost xgraph-12.1] # **./configure**
[root@localhost xgraph-12.1] # **make clean**
[root@localhost xgraph-12.1] # **make** [root@localhost xgraph-12.1] # **make install** [root@localhost xgraph-12.1] # **ns**

%

This completes the installation process of "NS-2" simulator

PART-A

1. Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped.

```
set ns [new Simulator]          /* Letter S is capital */
set nf [open lab1.nam w] /* open a nam trace file in write mode */
$ns namtrace-all $nf          /* nf – nam file */

set tf [open lab1.tr w] /* tf- trace file */
$ns trace-all $tf

proc finish { } { /* provide space b/w proc and finish and all are in small case */
global ns nf tf
$ns flush-trace /* clears trace file contents */
close $nf
close $tf
exec nam lab1.nam &
exit 0
}

set n0 [$ns node] /* creates 4 nodes */
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n0 $n2 200Mb 10ms DropTail /*Letter M is capital Mb*/
$ns duplex-link $n1 $n2 100Mb 5ms DropTail /*D and T are capital*/

$ns duplex-link $n2 $n3 1Mb 1000ms DropTail

$ns queue-limit $n0 $n2 50
$ns queue-limit $n1 $n2 50
set udp0 [new Agent/UDP] /* Letters A,U,D and P are capital */

$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR] /* A,T,C,B and R are capital*/

$cbr0 set packetSize_ 500 /*S is capital, space after underscore*/
$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] /* A and N are capital */
$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 (Open a new editor using "gedit command" and write awk file and save with ".awk" extension)

/*immediately after BEGIN should open braces '{'

```
BEGIN{
drop=0;
}
{
if($1=="d" )
{
drop++;
printf("%s\t%s\n",$5,$11);
}
}
END{
printf("Total number of %s packets dropped due to congestion =%d\n",$5,drop);
}
```

Steps for execution:

amc@amc-p2-1274il:~/Desktop/NS2/day1$ gedit 1.tcl
amc@amc-p2-1274il:~/Desktop/NS2/day1$ ns 1.tcl

Note:

1. Set the queue size fixed from n0 to n2 as 10, n1-n2 to 10 and from n2-n3 as 5.
   Syntax: To set the queue size
   $ns set queue-limit <from> <to> <size> Eg:

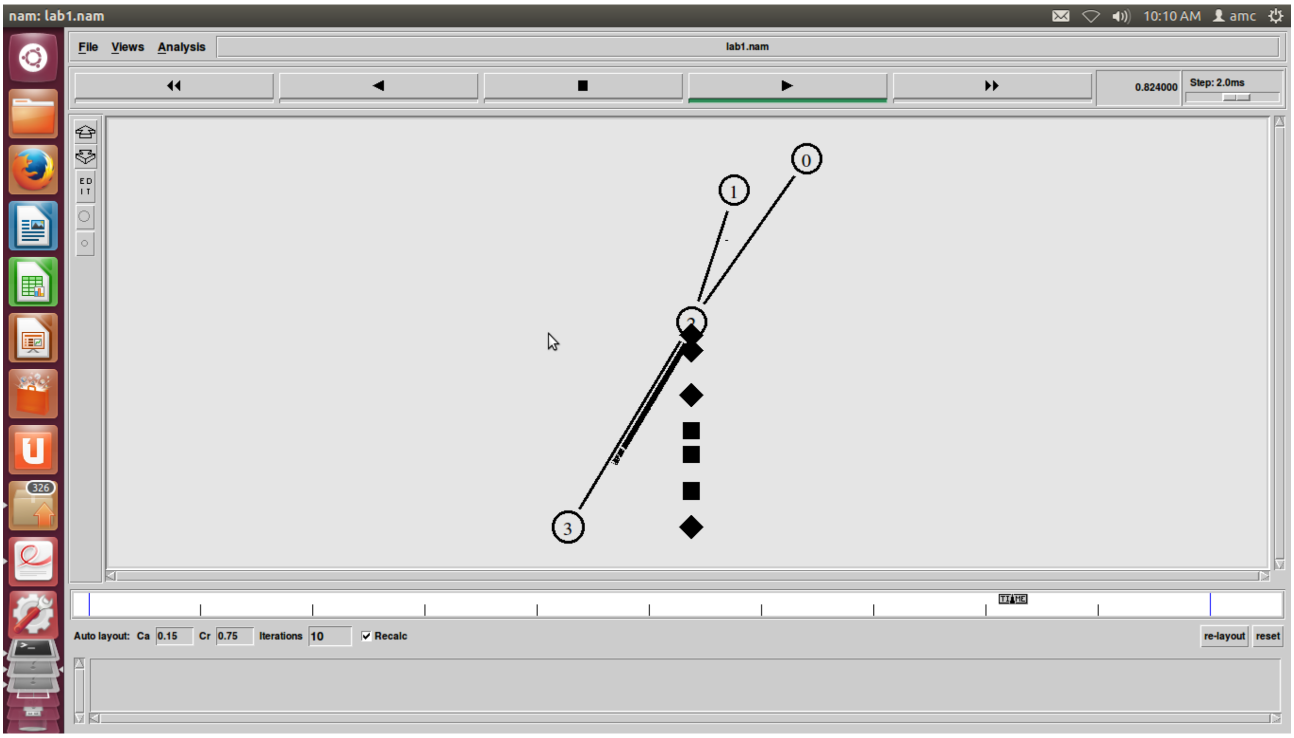
   $ns set queue-limit $n0 $n2 10

2. Go on varying the bandwidth from 10, 20 30 . . and find the number of packets dropped at the node 2

Trace file contains 12 columns:-

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

TOPOLOGY:



Output:

amc@amc-p2-1274il:~/Desktop/NS2/day1$ awk -f lab1.awk lab1.tr

cbr 139
cbr 126
cbr 127
cbr 130
cbr 151
cbr 154
cbr 136
cbr 159
cbr 141
cbr 142
cbr 145
cbr 171
cbr 174
cbr 151
cbr 154
cbr 157
cbr 187
cbr 161
cbr 163
cbr 195
cbr 201
cbr 173
cbr 175
cbr 209
Total number of cbr packets dropped due to congestion =24

amc@amc-p2-1274il:~/Desktop/NS2/day1$

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

```
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]
$n4 shape box

$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"
}

# please provide space between $node_ and id. No space between $ and from. No
```

```
#space between and $ and rtt */

$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"
$ns at 0.2 "$p3 send"
$ns at 0.3 "$p3 send"
$ns at 0.4 "$p3 send"
$ns at 0.5 "$p3 send"
$ns at 0.6 "$p3 send"
$ns at 0.7 "$p3 send"
$ns at 0.8 "$p3 send"
$ns at 0.9 "$p3 send"
```

```
$ns at 1.0 "$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 (Open a new editor using "gedit command" and write awk file and save with ".awk" extension)

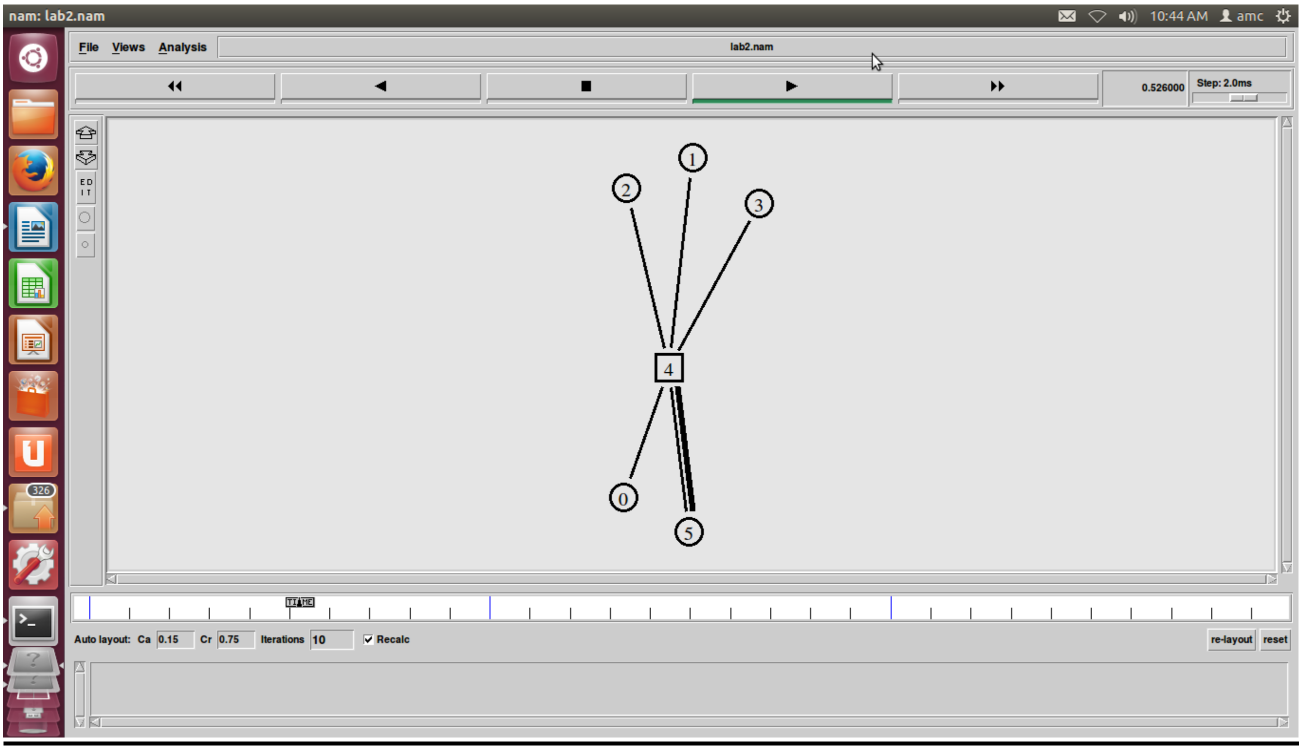
```
BEGIN{
drop=0;
}
{
if($1= ="d" )
{
drop++;
}
} END{
printf("Total number of %s packets dropped due to congestion =%d\n",$5,drop);
  }
```

Steps for execution:

amc@amc-p2-1274il:~/Desktop/NS2/day1$ gedit 2.tcl

amc@amc-p2-1274il:~/Desktop/NS2/day1$ ns 2.tcl

TOPOLOGY:



OUTPUT:

node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 0 received answer from 5 with round trip time 404.9 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 0 received answer from 5 with round trip time 704.9 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 0 received answer from 5 with round trip time 804.9 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 0 received answer from 5 with round trip time 804.9 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 0 received answer from 5 with round trip time 804.9 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec

```
node 2 received answer from 3 with round trip time 5.3 msec
node 0 received answer from 5 with round trip time 804.9 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 0 received answer from 5 with round trip time 804.9 msec
node 2 received answer from 3 with round trip time 5.3 msec

amc@amc-p2-1274il:~/Desktop/NS2/day1$ awk -f lab2.awk lab2.tr

The number of packets dropped =20

amc@amc-p2-1274il:~/Desktop/NS2/day1$
```

3. 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]
$n1 color "red"

set n2 [$ns node]
$n2 color "magenta"
$n2 label "src2"

set n3 [$ns node]
$n3 color "blue"
$n3 label "dest2"

set n4 [$ns node]
$n4 shape square

set n5 [$ns node]
$n5 color "blue"
$n5 label "dest1"

$ns make-lan "$n0 $n1 $n2 $n3 $n4" 50Mb 100ms LL Queue/DropTail Mac/802_3

$ns duplex-link $n4 $n5 1Mb 1ms DropTail
$ns duplex-link-op $n4 $n5 orient right
```

```
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 sink0 [new Agent/TCPSink]
$ns attach-agent $n5 $sink0
$ns connect $tcp0 $sink0
set tcp1 [new Agent/TCP]
$ns attach-agent $n2 $tcp1

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set packetSize_ 600
$ftp1 set interval_ 0.001

set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
$ns connect $tcp1 $sink1

set file1 [open file1.tr w]
$tcp0 attach $file1

set file2 [open file2.tr w]
$tcp1 attach $file2

$tcp0 trace cwnd_
$tcp1 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"
$ns at 0.2 "$ftp1 start"
$ns at 8 "$ftp1 stop"
$ns at 14 "$ftp0 stop"
$ns at 10 "$ftp1 start"
$ns at 15 "$ftp1 stop"
$ns at 16 "finish"

$ns run
```

AWK file (Open a new editor using “gedit command” and write awk file and save with “.awk” extension)

cwnd:- means congestion window

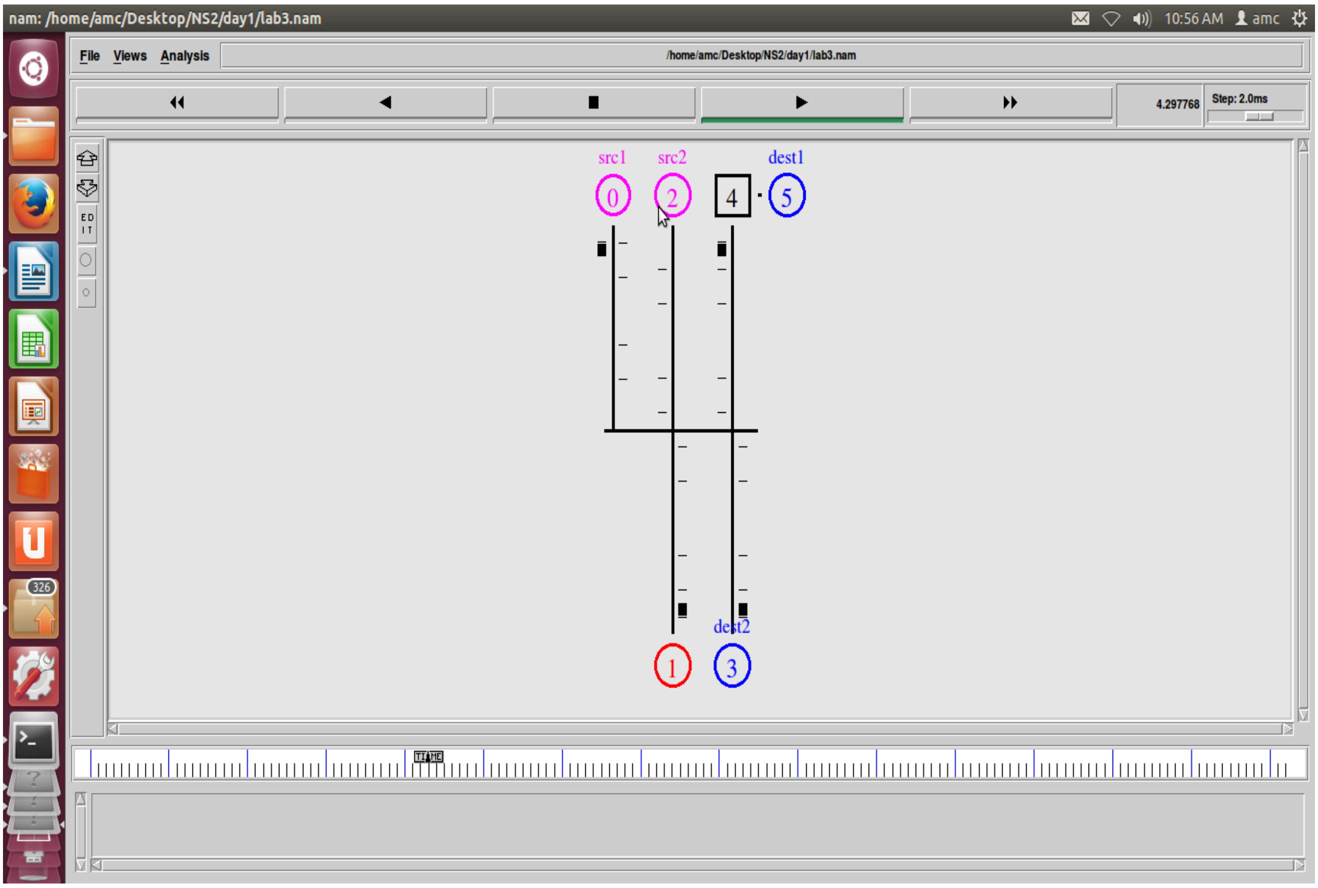
BEGIN {
}
{
if($6= ="cwnd_") /* don’t leave space after writing **cwnd_** */
printf("%f\t%f\t\n",$1,$7); /* you must put **\n** in printf */
}
END {
}

Steps for execution:

amc@amc-p2-1274il:~/Desktop/NS2/day1$ gedit 3.tcl

amc@amc-p2-1274il:~/Desktop/NS2/day1$ ns 3.tcl

TOPOLOGY:

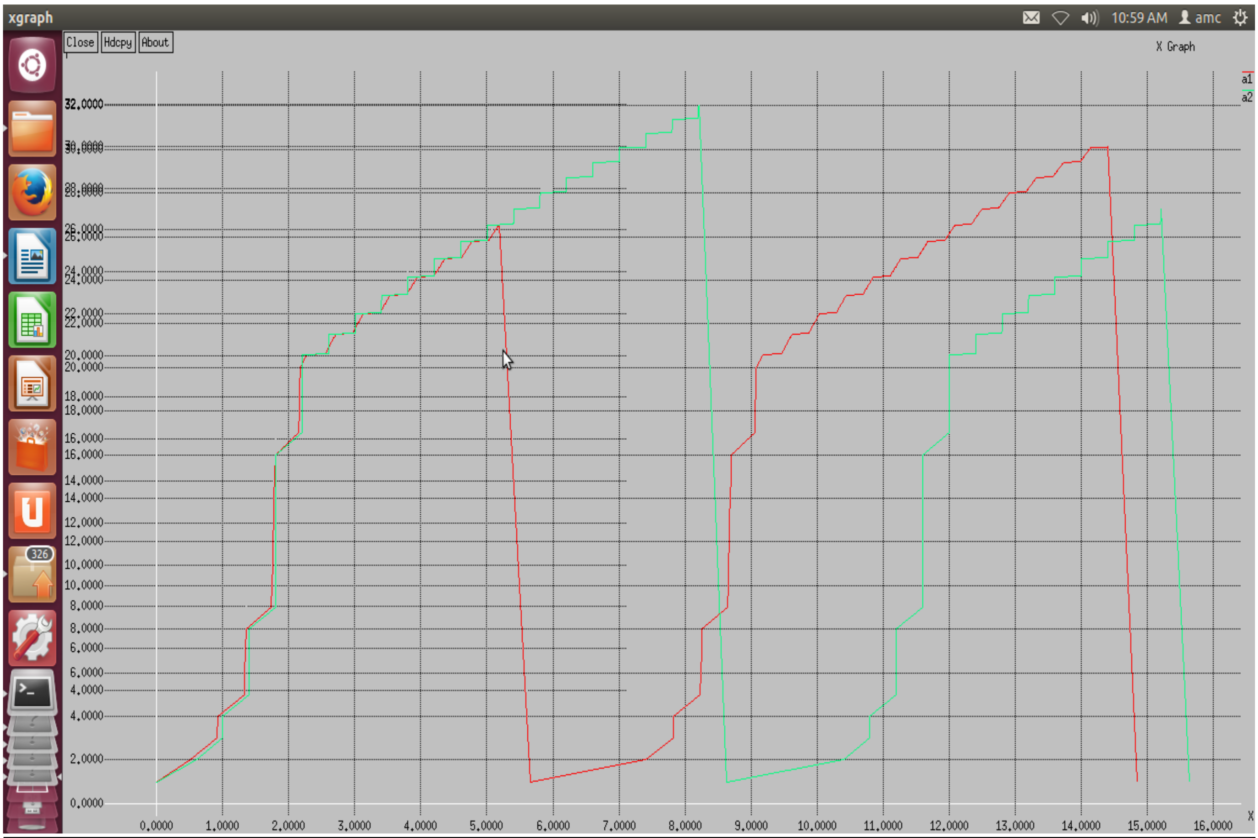


amc@amc-p2-1274il:~/Desktop/NS2/day1$ awk -f cwd.awk file1.tr >a1

amc@amc-p2-1274il:~/Desktop/NS2/day1$ awk -f cwd.awk file2.tr >a2

amc@amc-p2-1274il:~/Desktop/NS2/day1$ xgraph a1 a2

OUTPUT:



amc@amc-p2-1274il:~/Desktop/NS2/day1$

4. **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 lab4.tr w]
$ns trace-all $tf

set topo [new Topography]
$topo load_flatgrid 1000 1000

set nf [open lab4.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 nf tf
```

```
$ns flush-trace
exec nam lab4.nam &
close $tf
exit 0
}

$ns at 250 "finish"
$ns run
```

AWK file (Open a new editor using “gedit command” and write awk file and save with “.awk” extension)

```
BEGIN{
count1=0 count2=0 pack1=0 pack2=0 time1=0 time2=0
}
{
if($1= ="r"&& $3= ="_1_" && $4= ="AGT")
{
count1++ pack1=pack1+$8 time1=$2
}
if($1= ="r" && $3= ="_2_" && $4= ="AGT")
{
count2++ pack2=pack2+$8
time2=$2
}
}
END{
printf("The Throughput from n0 to n1: %f Mbps \n”, ((count1*pack1*8)/(time1*1000000)));
printf("The Throughput from n1 to n2: %f Mbps", ((count2*pack2*8)/(time2*1000000)));
}
```

Steps for execution:

amc@amc-p2-1274il:~/Desktop/NS2/day1$ gedit 4.tcl

amc@amc-p2-1274il:~/Desktop/NS2/day1$ ns 4.tcl

OUTPUT:

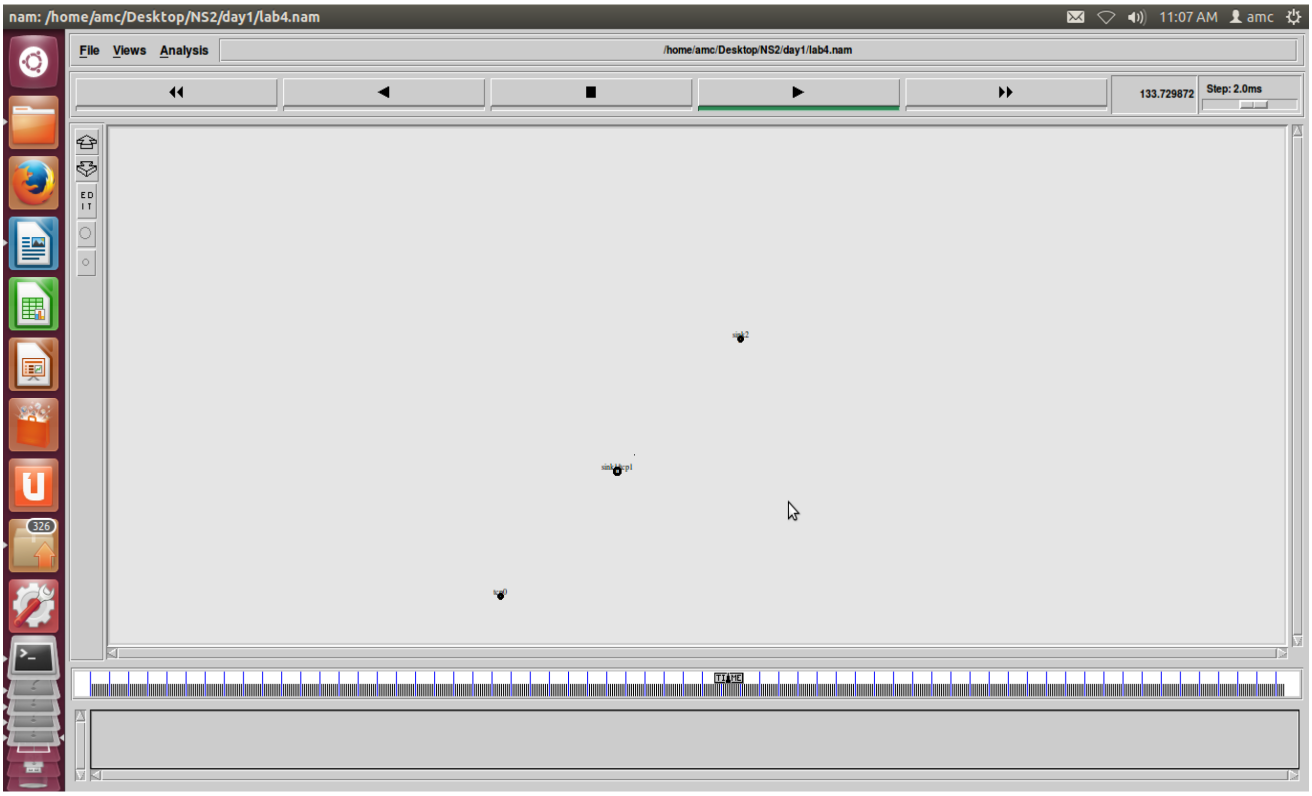
amc@amc-p2-1274il:~/Desktop/NS2/day1$ awk -f lab4.awk lab4.tr

The Throughput from n0 to n1: 5863.442245 Mbps

The Throughput from n1 to n2: 1307.611834 Mbps

amc@amc-p2-1274il:~/Desktop/NS2/day1$

TOPOLOGY:



TRACE FILE:

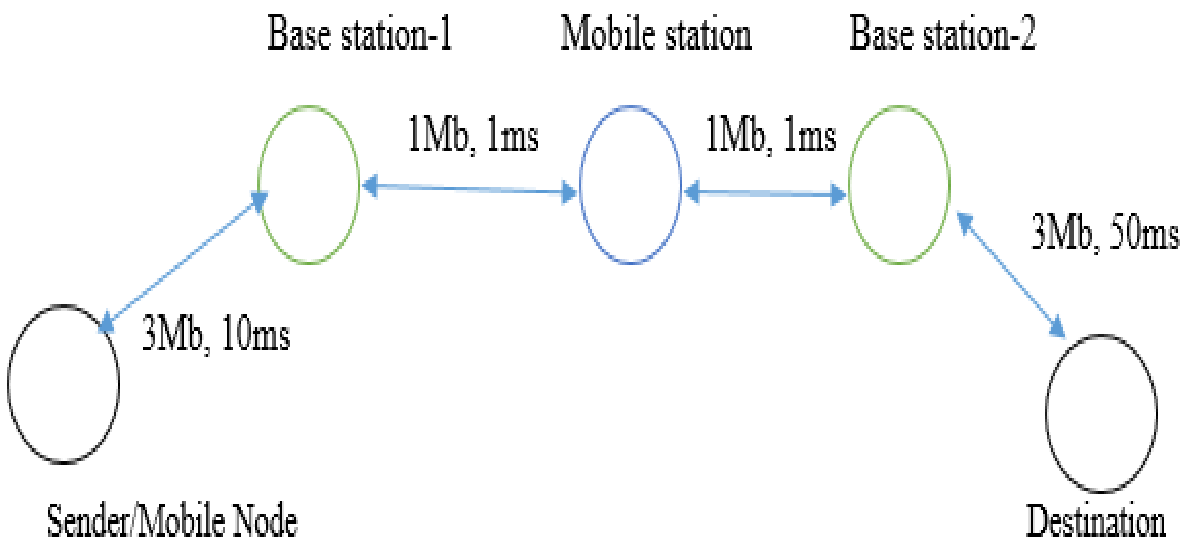
```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help
s 0.036400876 _0_ RTR  --- 0 message 32 [0 0 0 0] ------- [0:255 -1:255 32 0]
r 0.037421112 _1_ RTR  --- 0 message 32 [0 ffffffff 0 800] ------- [0:255 -1:255
 32 0]
M 0.10000 0 (50.00, 50.00, 0.00), (50.00, 50.00), 15.00
M 0.10000 1 (100.00, 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
```

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

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

GSM uses a variation of time division multiple access (TDMA). 2G networks developed as a replacement for first generation (1G) analog cellular networks, and the GSM standard originally described as a digital, circuit-switched network optimized for full duplex voice telephony. This expanded over time to include data communications, first by circuit-switched transport, then by packet data transport via GPRS (General Packet Radio Services).
GSM can be implemented on all the versions of NS2 (Since year 2004: ns-2.27, and later versions of NS2)

Design:



```
# General Parameters
set opt(title) zero        ;
set opt(stop) 100          ;# Stop time.
set opt(ecn) 0             ;
# Topology
set opt(type) gsm          ;#type of link:
set opt(secondDelay) 55         ;# average delay of access links in ms
# AQM parameters
set opt(minth) 30          ;
set opt(maxth) 0           ;
set opt(adaptive) 1        ;# 1 for Adaptive RED, 0 for plain RED
# Traffic generation.
set opt(flows) 0           ;# number of long-lived TCP flows
set opt(window) 30         ;# window for long-lived traffic
set opt(web) 2             ;# number of web sessions
# Plotting statistics.
set opt(quiet)   0         ;# popup anything?
set opt(wrap)    100       ;# wrap plots?
set opt(srcTrace) is       ;# where to plot traffic
set opt(dstTrace) bs2      ;# where to plot traffic
set opt(gsmbuf) 10              ; # buffer size for gsm

#default downlink bandwidth in bps
set bwDL(gsm)  9600
#default uplink bandwidth in bps
set bwUL(gsm)  9600
#default downlink propagation delay in seconds
set propDL(gsm)  .500
#default uplink propagation delay in seconds
set propUL(gsm)  .500
#default buffer size in packets
set buf(gsm)    10

set ns [new Simulator]
set tf [open out.tr w]
$ns trace-all $tf

set nodes(is) [$ns node]
set nodes(ms) [$ns node]
set nodes(bs1) [$ns node]
set nodes(bs2) [$ns node]
set nodes(lp) [$ns node]


proc cell_topo {} {
  global ns nodes
  $ns duplex-link $nodes(lp) $nodes(bs1) 3Mbps 10ms DropTail
  $ns duplex-link $nodes(bs1) $nodes(ms) 1 1 RED
  $ns duplex-link $nodes(ms) $nodes(bs2) 1 1 RED
  $ns duplex-link $nodes(bs2) $nodes(is) 3Mbps 50ms DropTail
  puts "Cell Topology"
}
proc set_link_params {t} {
```

```
  global ns nodes bwUL bwDL propUL propDL buf
  $ns bandwidth $nodes(bs1) $nodes(ms) $bwDL($t) 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(bs2) $buf($t)
}
# RED and TCP parameters
Queue/RED set summarystats_ true
Queue/DropTail set summarystats_ true
Queue/RED set adaptive_ $opt(adaptive)
Queue/RED set q_weight_ 0.0
Queue/RED set thresh_ $opt(minth)
Queue/RED set maxthresh_ $opt(maxth)
Queue/DropTail set shrink_drops_ true
Agent/TCP set ecn_ $opt(ecn)
Agent/TCP set window_ $opt(window)
DelayLink set avoidReordering_ true

source web.tcl

#Create topology
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]


# 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(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 out.tr"
        } else {
                set a "out.tr"
        }
        set GETRC "../../../bin/getrc"
      set RAW2XG "../../../bin/raw2xg"

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

      exec $GETRC -s $sid -d $did -f 1 out.tr | \
       $RAW2XG -s 0.01 -m $wrap -r >> plot.xgr
      exec $GETRC -s $did -d $sid -f 1 out.tr  | \
       $RAW2XG -s 0.01 -m $wrap -a  >> plot.xgr

        exec ./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
```

Steps for execution:

amc@amc-p2-1274il:~$ cd ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts/

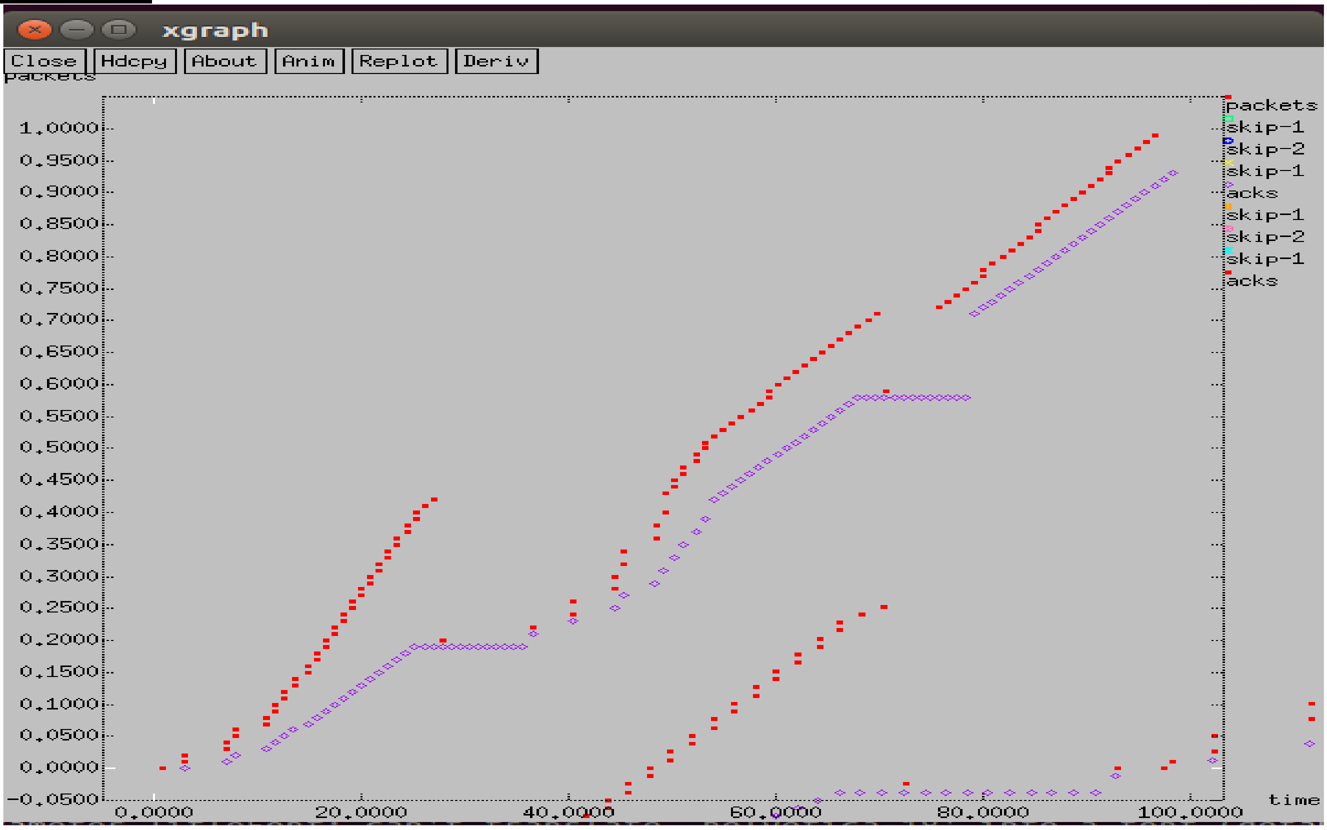
amc@amc-p2-1274il:~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts$ gedit mtp-gsm.tcl

amc@amc-p2-1274il:~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts$ ns mtp-gsm.tcl

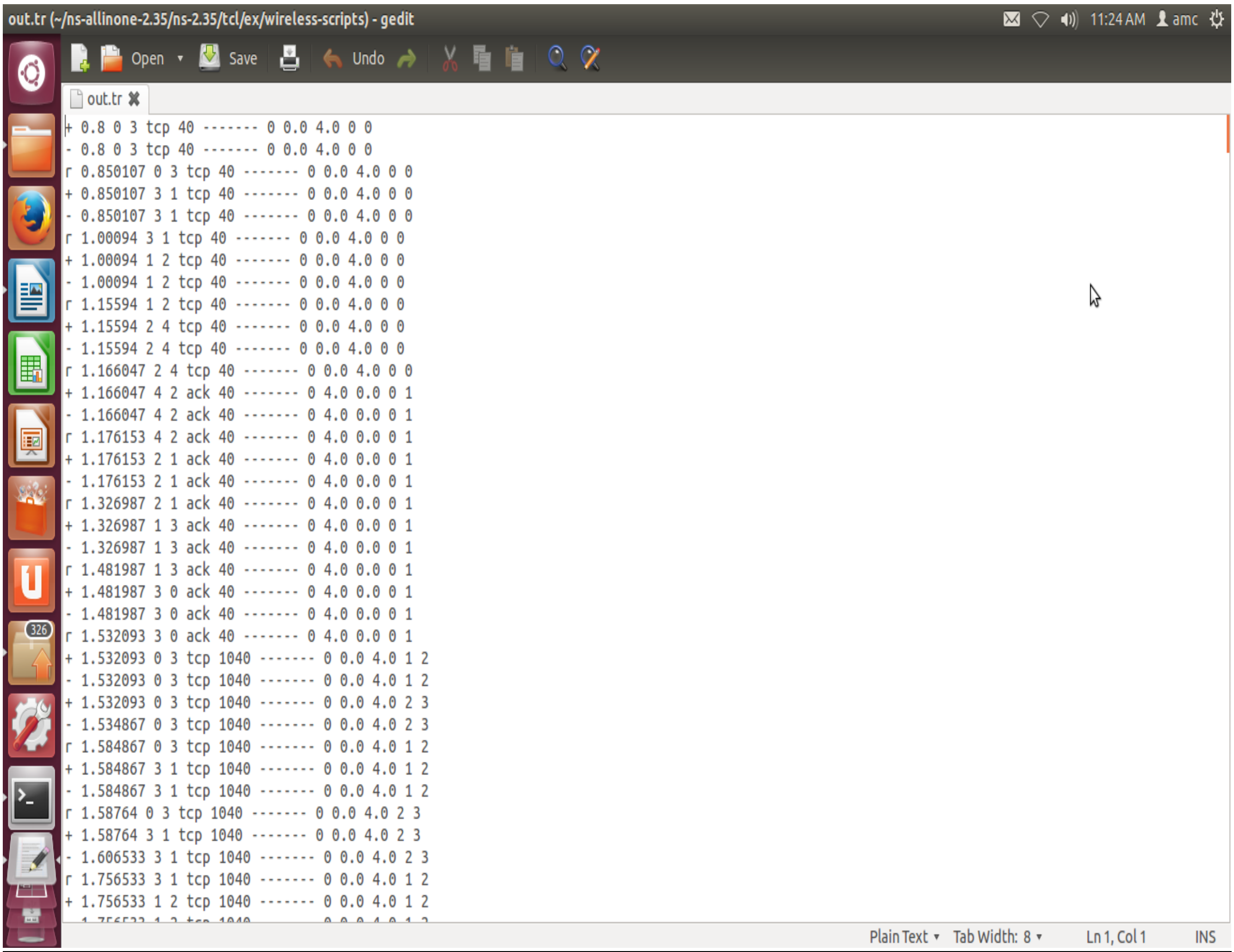
Cell Topology

amc@amc-p2-1274il:~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts$

OUTPUT:



GSM Trace file:




```
+ 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
```

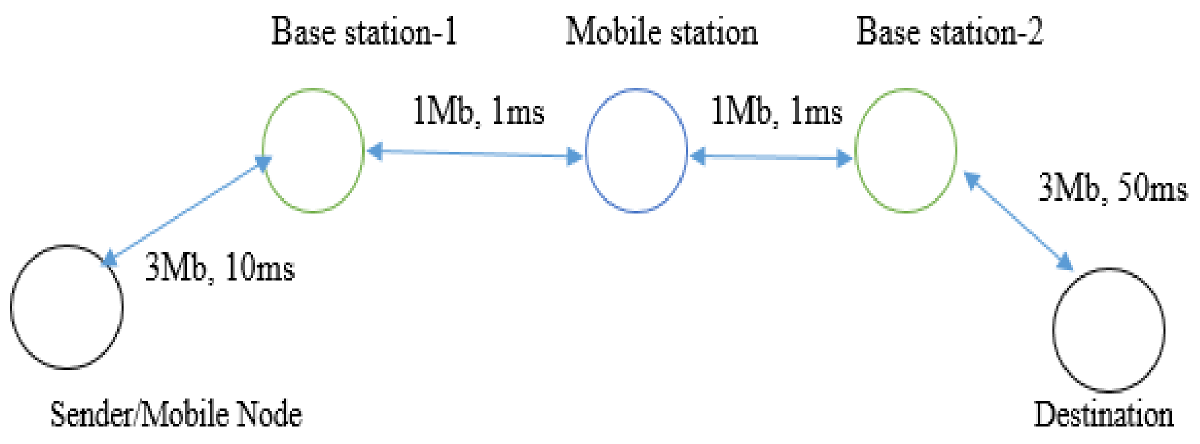
6. Implement and study the performance of CDMA on NS2/NS3 (Using stack called Call net) or equivalent environment.

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

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

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

Design:



```
# General Parameters
set opt(title) zero       ;
set opt(stop) 100         ;# Stop time.
set opt(ecn) 0            ;
# Topology
set opt(type) umts        ;#type of link:
set opt(secondDelay) 55         ;# average delay of access links in ms
# AQM parameters
set opt(minth) 30         ;
set opt(maxth) 0          ;
set opt(adaptive) 1       ;# 1 for Adaptive RED, 0 for plain RED
# Traffic generation.
set opt(flows) 0          ;# number of long-lived TCP flows
set opt(window) 30        ;# window for long-lived traffic
set opt(web) 2            ;# number of web sessions
# Plotting statistics.
set opt(quiet)   0        ;# popup anything?
set opt(wrap)    100      ;# wrap plots?
set opt(srcTrace) is      ;# where to plot traffic
set opt(dstTrace) bs2     ;# where to plot traffic
set opt(umtsbuf) 10             ; # buffer size for umts

#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

set ns [new Simulator]
set tf [open out.tr w]
$ns trace-all $tf

set nodes(is) [$ns node]
set nodes(ms) [$ns node]
set nodes(bs1) [$ns node]
set nodes(bs2) [$ns node]
set nodes(lp) [$ns node]

proc cell_topo {} {
  global ns nodes
  $ns duplex-link $nodes(lp) $nodes(bs1) 3Mbps 10ms DropTail
  $ns duplex-link $nodes(bs1) $nodes(ms) 1 1 RED
  $ns duplex-link $nodes(ms) $nodes(bs2) 1 1 RED
  $ns duplex-link $nodes(bs2) $nodes(is) 3Mbps 50ms DropTail
  puts "Cell Topology"
}
proc set_link_params {t} {
  global ns nodes bwUL bwDL propUL propDL buf
```

```
  $ns bandwidth $nodes(bs1) $nodes(ms) $bwDL($t) simplex
  $ns bandwidth $nodes(ms) $nodes(bs1) $bwUL($t) simplex
  $ns delay $nodes(bs1) $nodes(ms) $propDL($t) simplex
  $ns delay $nodes(ms) $nodes(bs1) $propDL($t) simplex
  $ns queue-limit $nodes(bs1) $nodes(ms) $buf($t)
  $ns queue-limit $nodes(ms) $nodes(bs1) $buf($t)
  $ns bandwidth $nodes(bs2) $nodes(ms) $bwDL($t) simplex
  $ns bandwidth $nodes(ms) $nodes(bs2) $bwUL($t) simplex
  $ns delay $nodes(bs2) $nodes(ms) $propDL($t) simplex
  $ns delay $nodes(ms) $nodes(bs2) $propDL($t) simplex
  $ns queue-limit $nodes(bs2) $nodes(ms) $buf($t)
  $ns queue-limit $nodes(ms) $nodes(bs2) $buf($t)
}

# RED and TCP parameters
Queue/RED set summarystats_ true
Queue/DropTail set summarystats_ true
Queue/RED set adaptive_ $opt(adaptive)
Queue/RED set q_weight_ 0.0
Queue/RED set thresh_ $opt(minth)
Queue/RED set maxthresh_ $opt(maxth)
Queue/DropTail set shrink_drops_ true
Agent/TCP set ecn_ $opt(ecn)
Agent/TCP set window_ $opt(window)
DelayLink set avoidReordering_ true

source web.tcl

#Create topology
switch $opt(type) {
 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]

# 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(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
```

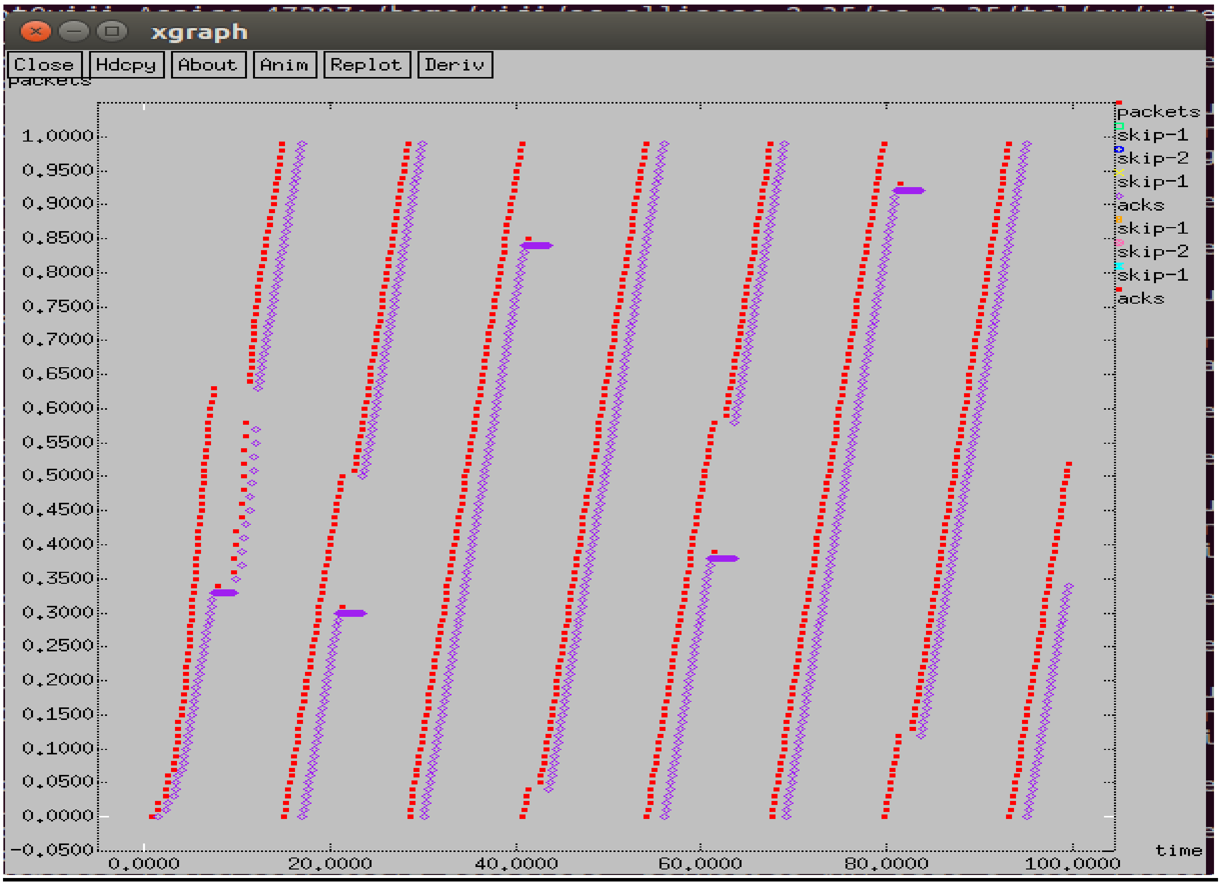
```
   $ns at 1.0  "[set ftp2] start"
   $ns at 8.0  "[set ftp2] stop"
}

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 out.tr"
        } else {
                set a "out.tr"
        }
        set GETRC "../../../bin/getrc"
      set RAW2XG "../../../bin/raw2xg"

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

      exec $GETRC -s $sid -d $did -f 1 out.tr | \
        $RAW2XG -s 0.01 -m $wrap -r >> plot.xgr
      exec $GETRC -s $did -d $sid -f 1 out.tr  | \
        $RAW2XG -s 0.01 -m $wrap -a  >> plot.xgr

        exec ./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
```

amc@amc-p2-1274il:~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts$ gedit mtp-umts.tcl

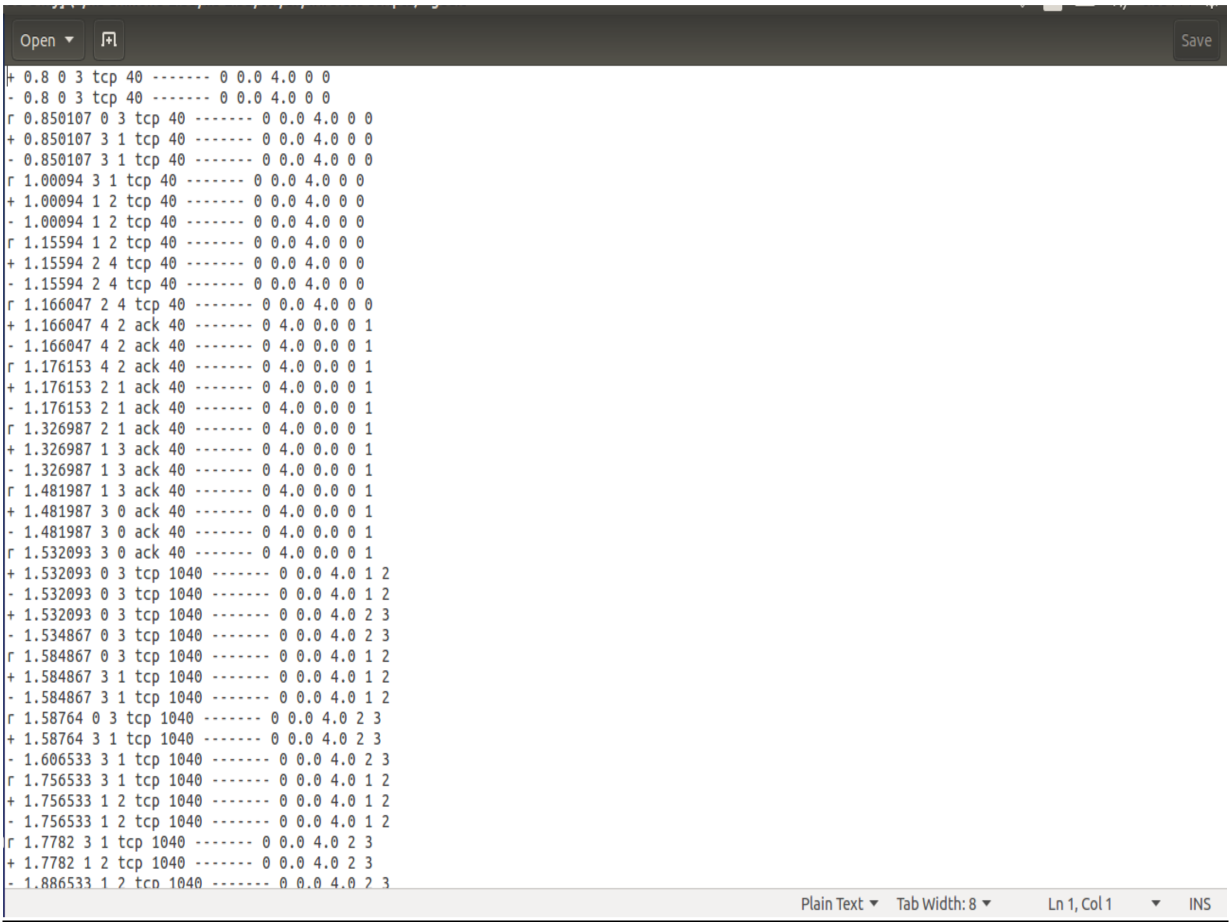
amc@amc-p2-1274il:~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts$ ns mtp-umts.tcl

Cell Topology

Output:



CDMA Trace File



```
+ 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
```

PART-B

Java is a general-purpose computer programming language that is simple, concurrent, class-based, object-oriented language. The compiled Java code can run on all platforms that support Java without the need for recompilation hence Java is called as "write once, run anywhere" (WORA).The Java compiled intermediate output called “byte-code” that can run on any Java virtual machine (JVM) regardless of computer architecture. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

In Linux operating system Java libraries are preinstalled. It’s very easy and convenient to compile and run Java programs in Linux environment. To compile and run Java Program is a two-step process:

1. Compile Java Program from Command Prompt

   [root@host ~]# javac Filename.java

   The Java compiler (Javac) compiles java program and generates a byte-code with the same file name and .class extension.

2. Run Java program from Command Prompt

   [root@host ~]# java Filename

   The java interpreter (Java) runs the byte-code and gives the respective output. It is important to note that in above command we have omitted the .class suffix of the byte- code (Filename.class).

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

Whenever digital data is stored or interfaced, data corruption might occur. Since the beginning of computer science, developers have been thinking of ways to deal with this type of problem. For serial data they came up with the solution to attach a parity bit to each sent byte. This simple detection mechanism works if an odd number of bits in a byte changes, but an even number of false bits in one byte will not be detected by the parity check. To overcome this problem developers have searched for mathematical sound mechanisms to detect multiple false bits. The **CRC** calculation or ***cyclic redundancy check*** was the result of this. Nowadays CRC calculations are used in all types of communications. All packets sent over a network connection are checked with a CRC. Also each data block on your hard disk has a CRC value attached to it. Modern computer world cannot do without these CRC calculations. So let's see why they are so widely used. The answer is simple; they are powerful, detect many types of errors and are extremely fast to calculate especially when dedicated hardware chips are used.

The idea behind CRC calculation is to look at the data as one large binary number. This number is divided by a certain value and the remainder of the calculation is called the CRC. Dividing in the CRC calculation at first looks to cost a lot of computing power, but it can be performed very quickly if we use a method similar to the one learned at school. We will as an example calculate the remainder for the character 'm'—which is 1101101 in binary notation— by dividing it by 19 or 10011. Please note that 19 is an odd number. This is necessary as we will see further on. Please refer to your schoolbooks as the binary calculation method here is not very different from the decimal method you learned when you were young. It might only look a little bit strange. Also notations differ between countries, but the method is similar.

```
                    1 0 1 = 5
                -------------
    1 0 0 1 1 / 1 1 0 1 1 0 1
                1 0 0 1 1 | |
                --------- | |
                  1 0 0 0 0 |
                  0 0 0 0 0 |
                  --------- |
                  1 0 0 0 0 1
                    1 0 0 1 1
                    ---------
                      1 1 1 0 = 14 = remainder
```

With decimal calculations you can quickly check that 109 divided by 19 gives a quotient of 5 with 14 as the remainder. But what we also see in the scheme is that every bit extra to check only costs one binary comparison and in 50% of the cases one binary subtraction. You can easilyincrease the number of bits of the test data string—for example to 56 bits if we use our example value "*Lammert*"—and the result can be calculated with 56 binary comparisons and an average of 28 binary subtractions. This can be implemented in hardware directly with only very few transistors involved. Also software algorithms can be very efficient.

All of the CRC formulas you will encounter are simply checksum algorithms based on modulo-2 binary division where we ignore carry bits and in effect the subtraction will be equal to an *exclusive or* operation. Though some differences exist in the specifics across different CRC formulas, the basic mathematical process is always the same:

- The message bits are appended with *c* zero bits; this *augmented message* is the dividend
- A predetermined *c+1*-bit binary sequence, called the *generator polynomial*, is the divisor
- The checksum is the *c*-bit remainder that results from the division operation

Table 1 lists some of the most commonly used generator polynomials for 16- and 32-bit CRCs. Remember that the width of the divisor is always one bit wider than the remainder. So, for example, you'd use a 17-bit generator polynomial whenever a 16-bit checksum is required.

	CRC-CCITT	**CRC-16**	**CRC-32**
Checksum Width	16 bits	16 bits	32 bits
Generator Polynomial	10001000000100001	11000000000000101	100000100110000010001110110110111

International Standard CRC Polynomials

SOURCE CODE:

```
import java.util.*;
import java.io.*;
public class CRC
{
   char t[]=new char[200];
   char cs[]=new char[200];
   char g[]=new char[200];
   int a,e,c;

   void xor()
   {
      for(int i=1;i<17;i++)
         cs[i]=((cs[i]==g[i])?'0':'1');
   }
   void crc()
   {
      for(e=0;e<17;e++)
         cs[e]=t[e];
      do
      {
         if(cs[0]=='1')
            xor();
         for(c=0;c<16;c++)
            cs[c]=cs[c+1];
            cs[c]=t[e++];
      }while(e<=a+16);
   }
   void operation()
   {

      Scanner read=new Scanner(System.in);
      String msg;
      String gs="10001000000100001";
      for(int i=0;i<gs.length();i++)
      g[i]=gs.charAt(i);

      System.out.println("enter the polynomial");
      msg=read.next();
      for(int i=0;i<msg.length();i++)
         t[i]=msg.charAt(i);

      System.out.print("\n generating polynomial is=");
      for(int i=0;i<gs.length();i++)
      System.out.print(g[i]);

      a=msg.length();
      for(e=a;e<a+16;e++)
         t[e]='0';
      System.out.print(" \n Modified message is=");
      for(int i=0;i<msg.length()+16;i++)
      System.out.print(t[i]);
```

```
        crc();

        System.out.println("\n Checksum is:");
        for(int i=0;i<16;i++)
            System.out.print(cs[i]);
        for(e=a;e<a+16;e++)
            t[e]=cs[e-a];
       System.out.println("\n final codeword is:");

       for(int i=0;i<a+16;i++)
        System.out.print(t[i]);
       System.out.println("\nTest error detection 0(yes)/1(no):");
       e=read.nextInt();
       if(e==0)
       {
           System.out.println("\nenter the position where error is to be inserted:");
            e=read.nextInt();
            t[e]=(t[e]=='0')?'1':'0';

             System.out.println("errornous data:")   ;
             for(int i=0;i<a+16;i++)
                 System.out.print(t[i]);
        }
         crc();
        for(e=0;(e<16)&&(cs[e]!='1');e++);
        if(e<16)
        System.out.println("error detected");
        else
        System.out.println("no error detected");
    }
    public static void main(String[] args)
    {
      CRC ob=new CRC();
      ob.operation();
    }

}
```

Output:

Run1:

enter the polynomial
1011101

 generating polynomial is=10001000000100001
 Modified message is=10111010000000000000000
 Checksum is:
1000101101011000
 final codeword is:
10111011000101101011000
Test error detection 0(yes)/1(no):
0

enter the position where error is to be inserted:

1
errornous data:
11111011000101101011000error detected

Run2:
enter the polynomial
1011101

generating polynomial is=10001000000100001
Modified message is=10111010000000000000000
Checksum is:
1000101101011000
final codeword is:
10111011000101101011000
Test error detection 0(yes)/1(no):
1
no error detected

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

Distance Vector Algorithm is a decentralized routing algorithm that requires that each router simply inform its neighbors of its routing table. For each network path, the receiving routers pick the neighbor advertising the lowest cost, then add this entry into its routing table for re-advertisement. To find the shortest path, Distance Vector Algorithm is based on one of two basic algorithms: the Bellman-Ford and the Dijkstra algorithms.

Routers that use this algorithm have to maintain the distance tables (which is a one-dimension array -- "a vector"), which tell the distances and shortest path to sending packets to each node in the network. The information in the distance table is always up date by exchanging information with the neighboring nodes. The number of data in the table equals to that of all nodes in networks (excluded itself). The columns of table represent the directly attached neighbors whereas the rows represent all destinations in the network. Each data contains the path for sending packets to each destination in the network and distance/or time to transmit on that path (we call this as "cost"). The measurements in this algorithm are the number of hops, latency, the number of outgoing packets, etc.

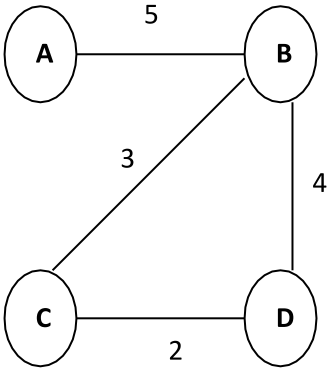
The Bellman–Ford algorithm is an algorithm that computes shortest paths from a single source vertex to all of the other vertices in a weighted digraph. It is slower than Dijkstra's algorithm for the same problem, but more versatile, as it is capable of handling graphs in which some of the edge weights are negative numbers. Negative edge weights are found in various applications of graphs, hence the usefulness of this algorithm. If a graph contains a "negative cycle" (i.e. a cycle whose edges sum to a negative value) that is reachable from the source, then there is no cheapest path: any path that has a point on the negative cycle can be made cheaper by one more walk around the negative cycle. In such a case, the Bellman–Ford algorithm can detect negative cycles and report their existence.

Source code:

```
import java.util.Scanner;
public class BellmanFord
{
private final int D[];
private final int num_ver;
public static final int MAX_VALUE = 999;
	public BellmanFord(int num_ver)
	{
		this.num_ver = num_ver;
	   D = new int[num_ver + 1];
	}

	public void BellmanFordEvaluation(int source, int A[][])
	{
		for (int node = 1; node <= num_ver; node++)
		{
			D[node] = MAX_VALUE;
		}
		D[source] = 0;
		for (int node = 1; node <= num_ver - 1; node++)
		{
			for (int sn = 1; sn <= num_ver; sn++)
			{
				for (int dn = 1; dn <= num_ver; dn++)
				{
					if (A[sn][dn] != MAX_VALUE)
					{
						if (D[dn] > D[sn]+ A[sn][dn])
							D[dn] = D[sn] + A[sn][dn];
					}
				}
			}
		}
		for (int sn = 1; sn <= num_ver; sn++)
		{
			for (int dn = 1; dn <= num_ver; dn++)
			{
				if (A[sn][dn] != MAX_VALUE)
				{
					if (D[dn] > D[sn]+ A[sn][dn])
					System.out.println("The Graph contains negative egde cycle");
				}
			}
		}
		for (int vertex = 1; vertex <= num_ver; vertex++)
		{
		System.out.println("distance of source " + source + " to "+ vertex + " is " + D[vertex]);
		}
	}
	public static void main(String[ ] args)
	{
		int num_ver = 0;
```

```
            int source;
                  Scanner scanner = new Scanner(System.in);
            System.out.println("Enter the number of vertices");
                  num_ver = scanner.nextInt();
                  int A[][] = new int[num_ver + 1][num_ver + 1];
            System.out.println("Enter the adjacency matrix");
                  for (int sn = 1; sn <=num_ver; sn++)
                  {
                           for (int dn = 1; dn <= num_ver; dn++)
                           {
                                    A[sn][dn] = scanner.nextInt();
                        if (sn == dn)
                                    {
                                             A[sn][dn] = 0;
                              continue;
                                    }
                                    if (A[sn][dn] == 0)
                                    {
                                             A[sn][dn] = MAX_VALUE;
                                    }
                           }
                  }
                  System.out.println("Enter the source vertex");
            source = scanner.nextInt();
                  BellmanFord b = new BellmanFord (num_ver);
            b.BellmanFordEvaluation(source, A);
            scanner.close();
         }
}
```

Output:
Run1:



Enter the number of vertices
4
Enter the adjacency matrix
0 5 0 0
5 0 3 4
0 3 0 2
0 4 2 0
Enter the source vertex
2
distance of source 2 to 1 is 5
distance of source 2 to 2 is 0
distance of source 2 to 3 is 3

distance of source 2 to 4 is 4

Run2:
Enter the number of vertices
4
Enter the adjacency matrix
0 5 0 0
5 0 -2 4
0 -3 0 -5
0 4 2 0
Enter the source vertex
2
The Graph contains negative egde cycle
The Graph contains negative egde cycle
The Graph contains negative egde cycle
The Graph contains negative egde cycle
distance of source 2 to 1 is -5
distance of source 2 to 2 is -15
distance of source 2 to 3 is -15
distance of source 2 to 4 is -17

9. 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.

Socket is an interface which enables the client and the server to communicate and pass on information from one another. Sockets provide the communication mechanism between two computers using TCP. A client program creates a socket on its end of the communication and attempts to connect that socket to a server. When the connection is made, the server creates a socket object on its end of the communication. The client and the server can now communicate by writing to and reading from the socket.

Source Code: Server

```
import java.net.*;
import java.io.*;
public class tcps
{
        public static void main(String args[]) throws IOException
        {
                ServerSocket sersock=new ServerSocket(5000);
                System.out.println("server ready for connection");
                Socket sock=sersock.accept();
                System.out.println("connection is successful");
                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();
                sersock.close();
                pwrite.close();
                fileRead.close();
                contentRead.close();
        }
}
```

Source Code: Client

```
import java.net.*;
import java.io.*;
public class tcpc
{
        public static void main(String args[]) throws IOException
        {
                  Socket sock=new Socket("127.0.0.1",5000);
                System.out.println("Enter the File Name");
                BufferedReader keyRead=new BufferedReader(new InputStreamReader(System.in));
                String fname=keyRead.readLine();
                OutputStream ostream=sock.getOutputStream();
                PrintWriter pwrite=new PrintWriter(ostream,true);
                System.out.println();
                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);
                }
                sock.close();
                pwrite.close();
                keyRead.close();
                socketRead.close();
        }
}
```

output:

serverside

```
amc@amc-p2-1274il:~$ gedit abc.txt
amc@amc-p2-1274il:~$ gedit tcps.java
amc@amc-p2-1274il:~$ javac tcps.java
amc@amc-p2-1274il:~$ java tcps
server ready for connection
connection is successful
amc@amc-p2-1274il:~$
```

clientside

```
amc@amc-p2-1274il:~$ javac tcpc.java
amc@amc-p2-1274il:~$ java tcpc
Enter the File Name
abc.txt
computer network lab
Information Science and Engg
amc@amc-p2-1274il:~$
```

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

A datagram socket is the one for sending or receiving point for a packet delivery service. Each packet sent or received on a datagram socket is individually addressed and routed. Multiple packets sent from one machine to another may be routed differently, and may arrive in any order.

<u>Source Code: Server</u>

```
import java.io.*;
 import java.net.*;
public class UDPS
{
        public static void main(String[] args)
        {
                DatagramSocket skt=null;
                try
                {
                skt=new DatagramSocket(6789);
                byte[] buffer = new byte[1000];
                while(true)
                {
                DatagramPacket request = new DatagramPacket(buffer,buffer.length);
                skt.receive(request);
                String[] message = (new String(request.getData())).split(" ");
                byte[] sendMsg= (message[1]+ " server processed").getBytes();
                DatagramPacket reply=       new
DatagramPacket(sendMsg,sendMsg.length,request.getAddress (),request.getPort());
                skt.send(reply);
                }
                }
                catch(Exception ex)
                {
                }
        }
}
```

<u>Source Code: Client</u>

```
import java.io.*;
```

```
import java.net.*;
public class UDPC
{
	public static void main(String[] args)
	{
		DatagramSocket skt; try
		{
			skt=new DatagramSocket();
			String msg= "network lab ";
			byte[] b = msg.getBytes();
			InetAddress host=InetAddress.getByName("127.0.0.1");
			int serverSocket=6789;
			DatagramPacket request =new DatagramPacket (b,b.length,host,serverSocket);
			skt.send(request);
			byte[] buffer =new byte[1000];
			DatagramPacket reply= new DatagramPacket(buffer,buffer.length);
			skt.receive(reply);
			System.out.println("client received:" +new String(reply.getData()));
			skt.close();
		}
		catch(Exception ex)
		{
		}
	}
}
```

Output:

serverside

```
amc@amc-p2-1274il:~/Desktop/NS2/day2/java$ gedit UDPS.java
amc@amc-p2-1274il:~/Desktop/NS2/day2/java$ javac UDPS.java
amc@amc-p2-1274il:~/Desktop/NS2/day2/java$ java UDPS
```

clientside

```
amc@amc-p2-1274il:~/Desktop/NS2/day2/java$ gedit UDPC.java
amc@amc-p2-1274il:~/Desktop/NS2/day2/java$ javac UDPC.java
amc@amc-p2-1274il:~/Desktop/NS2/day2/java$ java UDPC
client received:lab server processed
```

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

RSA is an example of public key cryptography. It was developed by Rivest, Shamir and Adelman. The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers.

The RSA algorithm's efficiency requires a fast method for performing the modular exponentiation operation. A less efficient, conventional method includes raising a number (the input) to a power (the secret or public key of the algorithm, denoted e and d, respectively) and taking the remainder of the division with N. A straight-forward implementation performs these two steps of the operation sequentially: first, raise it to the power and second, apply modulo. The RSA algorithm comprises of three steps, which are depicted below:

Key Generation Algorithm

1. Generate two large random primes, p and q, of approximately equal size such that their product n = p*q
2. Compute n = p*q and Euler's totient function (φ) phi(n) = (p-1)(q-1).
3. Choose an integer e, 1 < e < phi, such that gcd(e, phi) = 1.
4. Compute the secret exponent d, 1 < d < phi, such that e*d ≡ 1 (mod phi).
5. The public key is (e, n) and the private key is (d, n). The values of p, q, and phi should also be kept secret.

Encryption

Sender A does the following:-

1. Using the public key (e,n)
2. Represents the plaintext message as a positive integer M
3. Computes the cipher text C = M^e mod n.
4. Sends the cipher text C to B (Receiver).

Decryption

Recipient B does the following:-

1. Uses his private key (d, n) to compute M = C^d mod n.
2. Extracts the plaintext from the integer representative m.

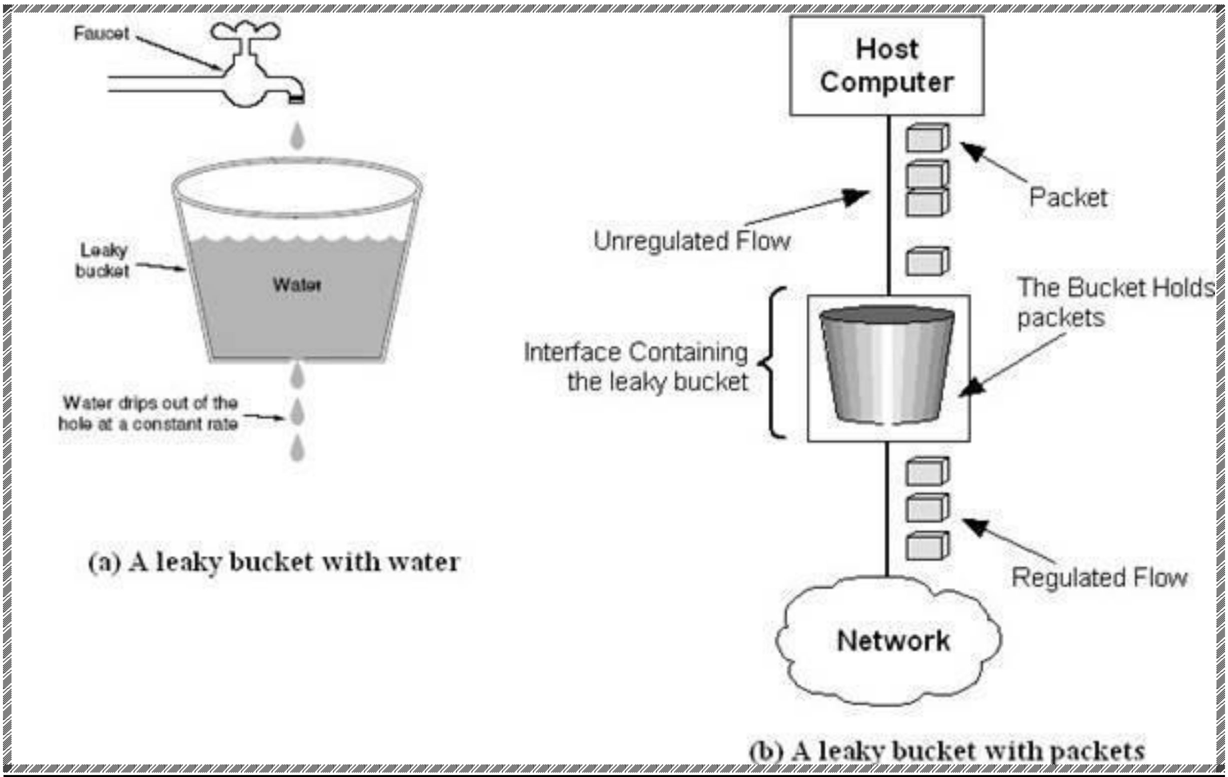
```
import java.util.*;
import java.io.*;

public class RSA
{

   public static int mul(int x, int y, int n)
   {
      int k=1,j;
      for(j=1;j<=y;j++)
         k=(k*x)%n;
      return k;
   }

   public static void main(String[] args)
   {
      String msg;
      int pt[]=new int[100];
      int ct[]=new int[100];
      int n, d, e, p, q,i;
      Scanner read=new Scanner(System.in);
      System.out.println("enter the message to encrypt:");
      msg=read.next();
      for(i=0;i<msg.length();i++)
        pt[i]=msg.charAt(i);
          n=253;d=17;e=13;
      System.out.println("\n cipher text is=");
      for(i=0;i<msg.length();i++)
         ct[i]=mul(pt[i],e,n);
      for(i=0;i<msg.length();i++)
          System.out.print(" "+ ct[i]);
      for(i=0;i<msg.length();i++)
         pt[i]=mul(ct[i],d,n);
      System.out.println("\ndecrypted message is= ");
      for(i=0;i<msg.length();i++)
         System.out.print("    "+(char)pt[i]);

   }

}
```

<u>Output:</u>

```
enter the message to encrypt:
AMCEC

 cipher text is=
 76 110 111 115 111
decrypted message is=
    A    M    C    E    C
```

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

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) A leaky bucket with packets

While leaky bucket eliminates completely bursty traffic by regulating the incoming data flow its main drawback is that it drops packets if the bucket is full. Also, it doesn't take into account the idle process of the sender which means that if the host doesn't transmit data for some time the bucket becomes empty without permitting the transmission of any packet.

```
import java.util.*;
 public class Bucket
 {
   static void solution(int pktsize, int output)
   {
      int buketsize=512;
      if(pktsize>buketsize)
      {
         System.out.println("Bucket overflow");
      }
      else
      {
         while(pktsize>output)
         {
          System.out.println(output+"bytes outputed");
          pktsize=pktsize-output;
         }
         if(pktsize>0)
         {
          System.out.println( pktsize+"bytes outputed");
         }
      }
   }

   public static void main(String[] args)
   {
     int output,pktsize,n;
     Scanner read=new Scanner(System.in);
     Random rand=new Random();
     System.out.println( "Enter output rate");
     output=read.nextInt();
      System.out.println( "Enter the number of packets");
      n=read.nextInt();
      for(int i=1;i<=n;i++)
      {
         pktsize=rand.nextInt(1000);
         System.out.println( "packetno:"+i+"packetsize="+pktsize);
         solution(pktsize,output);
      }
   }

}
```

Output:

```
Enter output rate
50
Enter the number of packets
5
packetno:1packetsize=417
50bytes outputed
```

50bytes outputed
50bytes outputed
50bytes outputed
50bytes outputed
50bytes outputed
50bytes outputed
50bytes outputed
17bytes outputed
packetno:2packetsize=917
Bucket overflow
packetno:3packetsize=866
Bucket overflow
packetno:4packetsize=721
Bucket overflow
packetno:5packetsize=20
20bytes outputed

VIVA QUESTIONS

1) What is a Link?
A link refers to the connectivity between two devices. It includes the type of cables and protocols used in order for one device to be able to communicate with the other.

2) What are the layers of the OSI reference model?
There are 7 OSI layers: Physical Layer, Data Link Layer, Network Layer, Transport Layer, Session Layer, Presentation Layer and Application Layer.

3) What is backbone network?
A backbone network is a centralized infrastructure that is designed to distribute different routes and data to various networks. It also handles management of bandwidth and various channels.

4) What is a LAN?
LAN is short for Local Area Network. It refers to the connection between computers and other network devices that are located within a small physical location.

5) What is a node?
A node refers to a point or joint where a connection takes place. It can be computer or device that is part of a network. Two or more nodes are needed in order to form a network connection.

6) What are routers?
Routers can connect two or more network segments. These are intelligent network devices that store information in its routing table such as paths, hops and bottlenecks. With this info, they are able to determine the best path for data transfer. Routers operate at the OSI Network Layer.

7) What is point to point link?
It refers to a direct connection between two computers on a network. A point to point connection does not need any other network devices other than connecting a cable to the NIC cards of both computers.

8) What is data encapsulation?
Data encapsulation is the process of breaking down information into smaller manageable chunks before it is transmitted across the network. It is also in this process that the source and destination addresses are attached into the headers, along with parity checks.

9) Describe Network Topology
Network Topology refers to the layout of a computer network. It shows how devices and cables are physically laid out, as well as how they connect to one another.

10) What is VPN?

VPN means Virtual Private Network, a technology that allows a secure tunnel to be created across a network such as the Internet. For example, VPNs allow you to establish a secure dialup connection to a remote server.

11) How does a network topology affect your decision in setting up a network?
Network topology dictates what media you must use to interconnect devices. It also serves as basis on what materials, connector and terminations that is applicable for the setup.

12) What are different ways of securing a computer network?
There are several ways to do this. Install reliable and updated anti-virus program on all computers. Make sure firewalls are setup and configured properly. User authentication will also help a lot. All of these combined would make a highly secured network.

13) What is NIC?
NIC is short for Network Interface Card. This is a peripheral card that is attached to a PC in order to connect to a network. Every NIC has its own MAC address that identifies the PC on the network.

14) What is WAN?
WAN stands for Wide Area Network. It is an interconnection of computers and devices that are geographically dispersed. It connects networks that are located in different regions and countries.

15) How many layers are there under TCP/IP?
There are four layers: the Network Layer, Internet Layer, Transport Layer and Application Layer.

16) What are gateways?
Gateways provide connectivity between two or more network segments. It is usually a computer that runs the gateway software and provides translation services. This translation is a key in allowing different systems to communicate on the network.

17) What is Hybrid Network?
A hybrid network is a network setup that makes use of both client-server and peer-to-peer architecture.

18) What is TCP/IP?
TCP/IP is short for Transmission Control Protocol / Internet Protocol. This is a set of protocol layers that is designed to make data exchange possible on different types of computer networks, also known as heterogeneous network.

20) What is the difference between a hub and a switch?
A hub acts as a multiport repeater. However, as more and more devices connect to it, it would not be able to efficiently manage the volume of traffic that passes through it.
A switch provides a better alternative that can improve the performance especially when high traffic volume is expected across all ports.

21) What is client/server?
Client/server is a type of network wherein one or more computers act as servers. Servers provide a centralized repository of resources such as printers and files. Clients refers to workstation that access the server.

22) Describe networking.
Networking refers to the inter connection between computers and peripherals for data communication. Networking can be done using wired cabling or through wireless link.

23)Describe Ethernet.
Ethernet is one of the popular networking technologies used these days. It was developed during the early 1970s and is based on specifications as stated in the IEEE. Ethernet is used in local area networks.

24) What is the difference between CSMA/CD and CSMA/CA?
CSMA/CD, or Collision Detect, retransmits data frames whenever a collision occurred. CSMA/CA, or Collision Avoidance, will first broadcast intent to send prior to data transmission.

25) What is the importance of Encryption on a network?

Encryption is the process of translating information into a code that is unreadable by the user. It is then translated back or decrypted back to its normal readable format using a secret key or password. Encryption help ensure that information that is intercepted halfway would remain unreadable because the user has to have the correct password or key for it.

26) What is RSA algorithm?

RSA is short for Rivest-Shamir-Adleman algorithm. It is the most commonly used public key encryption algorithm in use today.

27) What are the various types of key used in cryptography ?

Ans: here are two main types of cryptography:

- Secret key cryptography
- Public key cryptography

Secret key cryptography is also known as symmetric key cryptography. With this type of cryptography, both the sender and the receiver know the same secret code, called the key. Messages are encrypted by the sender using the key and decrypted by the receiver using the same key.
Public key cryptography also called asymmetric encryption, uses a pair of keys for encryption and decryption. With public key cryptography, keys work in pairs of matched public and private keys.

28) What is OSI and what role does it play in computer networks?

OSI (Open Systems Interconnect) serves as a reference model for data communication. It is made up of 7 layers, with each layer defining a particular aspect on how network devices connect and communicate with one another. One layer may deal with the physical media used, while another layer dictates how data is actually transmitted across the network.

29) What are MAC addresses?

MAC, or Media Access Control, uniquely identifies a device on the network. It is also known as physical address or Ethernet address. A MAC address is made up of 6-byte parts.

30) What is the equivalent layer or layers of the TCP/IP Application layer in terms of OSI reference model?

The TCP/IP Application layer actually has three counterparts on the OSI model: the Session layer, Presentation Layer and Application Layer.

CONTENT BEYOND SYLLABUS

1) Hamming code for Error Detection and Correction

Hamming Code. **Hamming code** is a set of **error-correction codes** that can be used to **detect and correct** the **errors** that can occur when the data is moved or stored from the sender to the receiver. It is technique developed by R.W. **Hamming**for **error correction**

Calculating the Hamming Code

The key to the Hamming Code is the use of extra parity bits to allow the identification of a single error. Create the code word as follows:

1. Mark all bit positions that are powers of two as parity bits. (positions 1, 2, 4, 8, 16, 32, 64, etc.)
2. All other bit positions are for the data to be encoded. (positions 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc.)
3. Each parity bit calculates the parity for some of the bits in the code word. The position of the parity bit determines the sequence of bits that it alternately checks and skips.
   Position 1: check 1 bit, skip 1 bit, check 1 bit, skip 1 bit, etc. (1,3,5,7,9,11,13,15,...)
   Position 2: check 2 bits, skip 2 bits, check 2 bits, skip 2 bits, etc. (2,3,6,7,10,11,14,15,...)
   Position 4: check 4 bits, skip 4 bits, check 4 bits, skip 4 bits, etc. (4,5,6,7,12,13,14,15,20,21,22,23,...)
   Position 8: check 8 bits, skip 8 bits, check 8 bits, skip 8 bits, etc. (8-15,24-31,40-47,...)
   Position 16: check 16 bits, skip 16 bits, check 16 bits, skip 16 bits, etc. (16-31,48-63,80-95,...)
   Position 32: check 32 bits, skip 32 bits, check 32 bits, skip 32 bits, etc. (32-63,96-127,160-191,...)
   etc.
4. Set a parity bit to 1 if the total number of ones in the positions it checks is odd. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

Here is an example:

A byte of data: 10011010
Create the data word, leaving spaces for the parity bits: _ _ 1 _ 0 0 1 _ 1 0 1 0
Calculate the parity for each parity bit (a ? represents the bit position being set):

- Position 1 checks bits 1,3,5,7,9,11:
  ? _ **1** _ **0** 0 **1** _ **1** 0 **1** 0. Even parity so set position 1 to a 0: **0** _ **1** _ **0** 0 **1** _ **1** 0 **1** 0
- Position 2 checks bits 2,3,6,7,10,11:
  0 **? 1** _ 0 **0 1** _ 1 **0 1** 0. Odd parity so set position 2 to a 1: 0 **1 1** _ 0 **0 1** _ 1 **0 1** 0
- Position 4 checks bits 4,5,6,7,12:
  0 1 1 **? 0 0 1** _ 1 0 1 **0**. Odd parity so set position 4 to a 1: 0 1 1 **1 0 0 1** _ 1 0 1 **0**
- Position 8 checks bits 8,9,10,11,12:
  0 1 1 1 0 0 1 **? 1 0 1 0**. Even parity so set position 8 to a 0: 0 1 1 1 0 0 1 **0 1 0 1 0**
- Code word: 011100101010.

```
import java.util.*;
class Hamming {
	public static void main(String args[]) {
		Scanner scan = new Scanner(System.in);
		System.out.println("Enter the number of bits for the Hamming data:");
		int n = scan.nextInt();
		int a[] = new int[n];

		for(int i=0 ; i < n ; i++) {
			System.out.println("Enter bit no. " + (n-i) + ":");
			a[n-i-1] = scan.nextInt();
		}

		System.out.println("You entered:");
		for(int i=0 ; i < n ; i++) {
			System.out.print(a[n-i-1]);
		}
		System.out.println();

		int b[] = generateCode(a);

		System.out.println("Generated code is:");
		for(int i=0 ; i < b.length ; i++) {
			System.out.print(b[b.length-i-1]);
		}
		System.out.println();
		System.out.println("Enter position of a bit to alter to check for error detection at the
receiver end (0 for no error):");
		int error = scan.nextInt();
		if(error != 0) {
			b[error-1] = (b[error-1]+1)%2;
		}
		System.out.println("Sent code is:");
		for(int i=0 ; i < b.length ; i++) {
			System.out.print(b[b.length-i-1]);
		}
		System.out.println();
		receive(b, b.length - a.length);
	}

	static int[] generateCode(int a[]) {
			int b[];

		int i=0, parity_count=0 ,j=0, k=0;
		while(i<a.length) {
				if(Math.pow(2,parity_count) == i+parity_count + 1) {
				parity_count++;
			}
			else {
				i++;
			}
		}
```

```
        b = new int[a.length + parity_count];
                for(i=1 ; i<=b.length ; i++) {
                if(Math.pow(2, j) == i) {
                        b[i-1] = 2;
                        j++;
                }
                else {
                        b[k+j] = a[k++];
                }
        }
        for(i=0 ; i<parity_count ; i++) {
                b[((int) Math.pow(2, i))-1] = getParity(b, i);
        }
        return b;
}
static int getParity(int b[], int power) {
        int parity = 0;
        for(int i=0 ; i < b.length ; i++) {
                if(b[i] != 2) {
                        int k = i+1;
                        String s = Integer.toBinaryString(k);
                        int x = ((Integer.parseInt(s))/((int) Math.pow(10, power)))%10;
                        if(x == 1) {
                                if(b[i] == 1) {
                                        parity = (parity+1)%2;
                                }
                        }
                }
        }
        return parity;
}
        static void receive(int a[], int parity_count) {

        int power;

        int parity[] = new int[parity_count];
        String syndrome = new String();
        for(power=0 ; power < parity_count ; power++) {
                        for(int i=0 ; i < a.length ; i++) {
                        int k = i+1;
                        String s = Integer.toBinaryString(k);
                        int bit = ((Integer.parseInt(s))/((int) Math.pow(10, power)))%10;
                        if(bit == 1) {
                                if(a[i] == 1) {
                                        parity[power] = (parity[power]+1)%2;
                                }
                        }
                }
                syndrome = parity[power] + syndrome;
        }
        int error_location = Integer.parseInt(syndrome, 2);
        if(error_location != 0) {
```

```
                    System.out.println("Error is at location " + error_location + ".");
                    a[error_location-1] = (a[error_location-1]+1)%2;
                    System.out.println("Corrected code is:");
                    for(int i=0 ; i < a.length ; i++) {
                        System.out.print(a[a.length-i-1]);
                    }
                    System.out.println();
                }
                else {
                    System.out.println("There is no error in the received data.");
                }
                System.out.println("Original data sent was:");
                power = parity_count-1;
                for(int i=a.length ; i > 0 ; i--) {
                    if(Math.pow(2, power) != i) {
                        System.out.print(a[i-1]);
                    }
                    else {
                        power--;
                    }
                }
                System.out.println();
        }
}
```

Output:

```
Enter the number of bits for the Hamming data:
7
Enter bit no. 7:
1
Enter bit no. 6:
0
Enter bit no. 5:
1
Enter bit no. 4:
0
Enter bit no. 3:
1
Enter bit no. 2:
0
Enter bit no. 1:
1
You entered:
1010101
Generated code is:
10100101111
Enter position of a bit to alter to check for error detection at the receiver end (0 for no error):
5
Sent code is:
10100111111
Error is at location 5.
Corrected code is:
10100101111
Original data sent was:
```

1010101

2) Diffie Hellman Algorithm

Diffie-Hellman is a way of generating a shared secret between two people in such a way that the secret can't be seen by observing the communication.That's an important distinction: You're not sharing information during the key exchange, you're creating a key together.

Algorithm

Step 1 : Choose two prime numbers **g(primitive root of p)** and **p.**

Step 2 : Alice selects a secret no(**a**) and computes $\mathbf{g^a}$ *mod* **p** , let's call it **A**. Alice sends **A** to Bob.

Step 3 : Bob selects a secret no(**b**) and computes $\mathbf{g^b}$ *mod* **p,** let's call it **B.** Bob sends **B** to Alice.

Step 4 : Alice computes ***S_A*** = B^a mod p

Step 5 : Bob computes ***S_B*** = A^b mod p

Step 6 : If **S_A=S_B** then Alice and Bob can agree for future communication.

Program:

```
import java.util.*;
 class Diffie_Hellman
{
 public static void main(String args[])
 {
 Scanner sc=new Scanner(System.in);
 System.out.println("Enter modulo(p)");
 int p=sc.nextInt();
 System.out.println("Enter primitive root of "+p);
 int g=sc.nextInt();
 System.out.println("Choose 1st secret no(Alice)");
 int a=sc.nextInt();
 System.out.println("Choose 2nd secret no(BOB)");
 int b=sc.nextInt();

 int A = (int)Math.pow(g,a)%p;
 int B = (int)Math.pow(g,b)%p;

 int S_A = (int)Math.pow(B,a)%p;
 int S_B =(int)Math.pow(A,b)%p;

 if(S_A==S_B)
 {
 System.out.println("ALice and Bob can communicate with each other!!!");
 System.out.println("They share a secret no = "+S_A);
 }
  else
 {
 System.out.println("ALice and Bob cannot communicate with each other!!!");
 }
 }
```

}

Output:

Enter modulo(p)
7
Enter primitive root of 7
17
Choose 1st secret no(Alice)
8
Choose 2nd secret no(BOB)
4
ALice and Bob can communicate with each other!!!
They share a secret no =1