CITY ENGINEERING COLLEGE

Doddakallsandra, Kanakapura Road, Bangalore – 560061

[As per Choice Based Credit System (CBCS) scheme]

LAB MANUAL

COMPUTER NETWORK LABORATORY (18CSL57)

For

V SEMESTER B.E (CSE/ISE)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CITY ENGINEERING COLLEGE

BANGALORE- 560061

COMPUTER NETWORK LABORATORY

Subject Code 18CSL57

[As per Choice Based Credit System (CBCS) scheme]

SEMESTER – V

Lab Experiments:

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.

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. Implement the above program using as message queues or FIFOs as IPC channels.

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.

PART A - SIMULATION USING NS-2

Introduction to NS-2

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

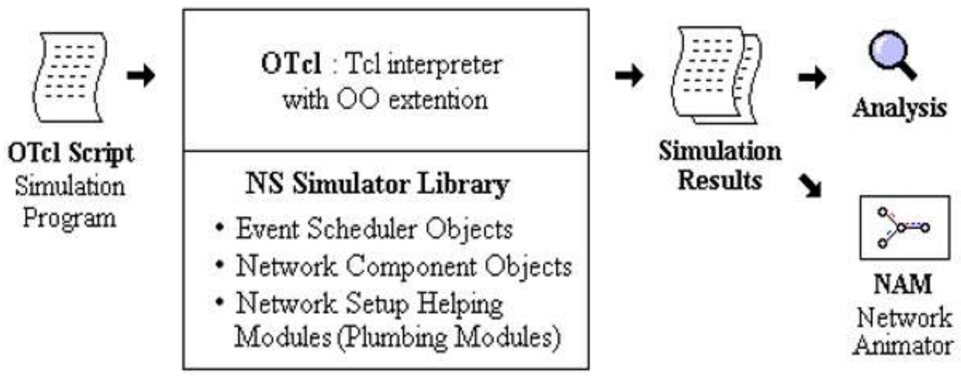
 Widely known as NS2, is simply an event driven simulation tool.

 Useful in studying the dynamic nature of communication networks.

 Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2.

 In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors.

Basic Architecture of NS2


OTcl Script
Simulation Program
OTcl : Tcl interpreter with OO extention
NS Simulator Library
• Event Scheduler Objects
• Network Component Objects
• Network Setup Helping Modules (Plumbing Modules)
Simulation Results
Analysis
NAM
Network Animator

Features of NS2

NS2 can be employed in most unix systems and windows. Most of the NS2 code is in C++. It uses TCL as its scripting language, Otcl adds object orientation to TCL.NS(version 2) is an object oriented, discrete event driven network simulator that is freely distributed and open source.
• Traffic Models: CBR, VBR, Web etc
• Protocols: TCP, UDP, HTTP, Routing algorithms, MAC etc
• Error Models: Uniform, bursty etc
• Misc: Radio propagation, Mobility models , Energy Models
• Topology Generation tools
• Visualization tools (NAM), Tracing

Structure of NS

- NS is an object oriented discrete event simulator
  - Simulator maintains list of events and executes one event after another

   - Single thread of control: no locking or race conditions
- Back end is C++ event scheduler
   - Protocols mostly
   - Fast to run, more control
- Front end is OTCL
   - Creating scenarios, extensions to C++ protocols
   - fast to write and change

Platforms
It can be employed in most unix systems(FreeBSD, Linux, Solaris) and Windows.

Source code
Most of NS2 code is in C++

Scripting language
It uses TCL as its scripting language OTcl adds object orientation to TCL.

Protocols implemented in NS2
Transport layer(Traffic Agent) – TCP, UDP
Network layer(Routing agent)
Interface queue – FIFO queue, Drop Tail queue, Priority queue
Logic link control layer – IEEE 802.2, AR

TCL – Tool Command Language
- 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

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 Procedure:

STEP 1: Start

STEP 2: Create the simulator object ns for designing the given simulation

STEP 3: Open the trace file and nam file in the write mode

STEP 4: Create the nodes of the simulation using the ‘set’ command

STEP 5: Create links to the appropriate nodes using $ns duplex-link command

STEP 6: Set the orientation for the nodes in the simulation using ‘orient’ command

STEP 7: Create TCP agent for the nodes and attach these agents to the nodes

STEP 8: The traffic generator used is FTP for both node0 and node1

STEP 9: Configure node1 as the sink and attach it

STEP10: Connect node0 and node1 using ‘connect’ command

STEP 11: Setting color for the nodes

STEP 12: Schedule the events for FTP agent 10 sec

STEP 13: Schedule the simulation for 5 minutes

Initialization and Termination of TCL Script in NS-2

An ns simulation starts with the command

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

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

#Open the Trace file

```
set tracefile1 [open out.tr w]

$ns trace-all $tracefile1
```

#Open the NAM trace file

```
set namfile [open out.nam w]

$ns namtrace-all $namfile
```

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

#Define a 'finish' procedure

```
Proc finish { } {

global ns tracefile1 namfile

$ns flush-trace

Close $tracefile1

Close $namfile

Exec nam out.nam &

Exit 0

}
```

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

```
$ns at 125.0 "finish"
```

will be used to call "finish" at time 125sec. Indeed, the at method of the simulator allows us to schedule events explicitly.

The simulation can then begin using the command

```
$ns run
```

Definition of a network of links and nodes

The way to define a node is

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

The node is created which is printed by the variable n0. When we shall refer to that node in the script we shall thus write $n0. Once we define several nodes, we can define the links that connect them.

Creating link between nodes:

$ns <link_type> $n0 $n1 <bandwidth> <delay><queue-type>

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 "simplexlink".

In NS, an output queue of a node is implemented as a part of each link whose input is that node. The definition of the link then includes the way to handle overflow at that queue. In our case, if the buffer capacity of the output queue is exceeded then the last packet to arrive is dropped. Many alternative options exist, such as the RED (Random Early Discard) mechanism, the FQ (Fair Queuing), the DRR (Deficit Round Robin), the stochastic Fair Queuing (SFQ) and the CBQ (which including a priority and a round-robin scheduler).

In ns, an output queue of a node is implemented as a part of each link whose input is that node. We should also define the buffer capacity of the queue related to each link. An example would be:

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

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

Agents and Applications

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

FTP over TCP

TCP is a dynamic reliable congestion control protocol. It uses Acknowledgements created by the destination to know whether packets are well received. There are number variants of the TCP protocol, such as Tahoe, Reno, NewReno, Vegas. The type of agent appears in the first line:

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

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

The command

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

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

#Setup a UDP connection

```
set udp [new Agent/UDP]

$ns attach-agent $n1 $udp

set null [new Agent/Null]

$ns attach-agent $n5 $null

$ns connect $udp $null

$udp set fid_2
```

#setup a CBR over UDP connection

The below shows the definition of a CBR application using a UDP agent
The command **$ns attach-agent $n4 $sink** defines the destination node.
The command **$ns connect $tcp $sink** finally makes the TCP connection between the source and destination nodes.

```
set cbr [new
Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set packetsize_ 100

$cbr set rate_ 0.01Mb

$cbr set random_ false
```

TCP has many parameters with initial fixed defaults values that can be changed if mentioned explicitly. For example, the default TCP packet size has a size of 1000bytes. This can be changed to another value, say 552bytes, using the command $tcp set packetSize_ 552.

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"

$ns at 124.5 "$cbr stop"
```

Structure of Trace Files

When tracing into an output ASCII file, the trace is organized in 12 fields as follows in fig shown below,

The meaning of the fields are:

Event	Time	From Node	To Node	PKT Type	PKT Size	Flags	Fid	Src Addr	Dest Addr	Seq Num	Pkt id

1. The first field is the event type. It is given by one of four possible symbols r, +, -, d which correspond respectively to receive (at the output of the link), enqueued, dequeued and dropped.

2. The second field gives the time at which the event occurs.

3. Gives the input node of the link at which the event occurs.

4. Gives the output node of the link at which the event occurs.

5. Gives the packet type (eg CBR or TCP)

6. Gives the packet size

7. Some flags

8. This is the flow id (fid) of IPv6 that a user can set for each flow at the input OTcl script one can further use this field for analysis purposes; it is also used when specifying stream color for the NAM display.

9. This is the source address given in the form of "node.port".

10. This is the destination address, given in the same form.

11. This is the network layer protocol’s packet sequence number. Even though UDP implementations in a real network do not use sequence number, ns keeps track of UDP packet sequence number for analysis purposes

12. The last field shows the Unique id of the packet.

XGRAPH

The xgraph program draws a graph on an x-display given data read from either data file or from standard input if no files are specified. It can display upto 64 independent data sets using different colors and line styles for each set. It annotates the graph with a title, axis labels, grid lines or tick marks, grid labels and a legend.

Syntax:

```
Xgraph [options] file-name
```

Options are listed here;

/-bd <color> (Border)

This specifies the border color of the xgraph window.

/-bg <color> (Background)

This specifies the background color of the xgraph window.

/-fg<color> (Foreground)

This specifies the foreground color of the xgraph window.

/-lf <fontname> (LabelFont)

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

/-t<string> (Title Text)

This string is centered at the top of the graph.

/-x <unit name> (XunitText)

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

/-y <unit name> (YunitText)

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

Awk- An Advanced

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

Syntax:

```
awk option ‘selection_criteria {action}’ file(s)
```

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

THE –f OPTION: STORING awk PROGRAMS IN A FILE

awk programs are stored in separate file and provide them with the awk extension for easier identification. After simulation is completed run awk file to see the output.

Example:

```
awk -f  filename.awk  filename.tr
```

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.

Experiment No: 1 **THREE NODE POINT TO POINT NETWORK**

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

Program: prog1.tcl

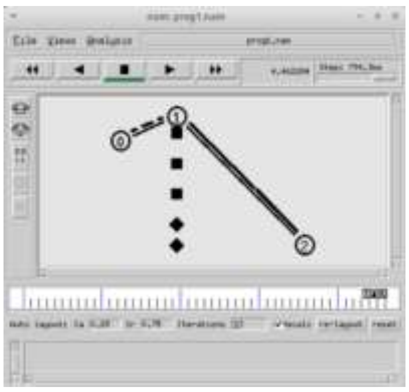
```
set ns [new Simulator]                         # Letter S is capital

$ns color 0 Blue                               # specify packet color  from Node n0
set nf [open prog1.nam w]                      # open a nam trace file in write mode
$ns namtrace-all $nf                           # nf  nam filename
set nd [open prog1.tr w]                       # nd  trace filename
$ns trace-all $nd

proc finish { } {
global ns nf nd
$ns flush-trace                        # clears trace file contents
close $nf
close $nd
exec nam prog1.nam &
exit 0
}

set n0 [$ns node]                              # creates 3 nodes
set n1 [$ns node]
set n2 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 512Kb 10ms DropTail
$ns queue-limit $n1 $n2 5

set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_  500
$cbr0 set interval_  0.005
$cbr0 attach-agent $udp0

set sink [new Agent/Null]                      # establishing links
$ns attach-agent $n2 $sink
$ns connect $udp0 $sink

$ns at 0.2 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```

AWK file: (Open a new editor using "gedit command" and write awk file and save with ".awk" extension) #immediately after BEGIN should open braces „{„

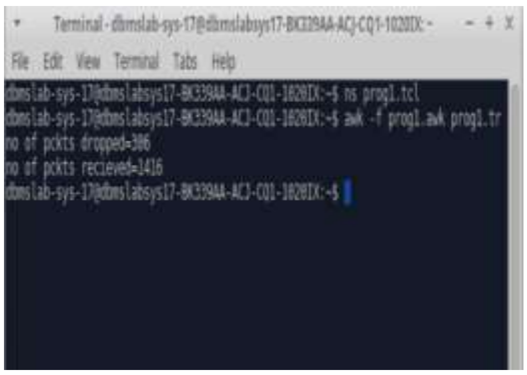
```
BEGIN {
            dcount = 0;
            rcount = 0;
            }
{
  event = $1;
  if(event == "d")
  {
   dcount++;
  }
  if(event == "r")
  {
    rcount++;
  }}
END { printf("no of pckts dropped=%d\n",dcount);
           printf("no of pckts recieved=%d\n",rcount);
    }
```

Steps for execution:

- Open geditor and type program. Save the program using extension " .tcl "
  [root@localhost ~]# gedit prog1.tcl
- Open geditor and type awk program. Save the program using extension ".awk "
  [root@localhost ~]# gedit prog1.awk
- Run the simulation program
  [root@localhost~]# ns prog1.tcl
- Here "ns" indicates network simulator. We get the topology shown in the snapshot.
- Now press the play button in the simulation window and the simulation will begins.
- After simulation is completed run awk file to see the output,
  [root@localhost~]# awk –f prog1.awk prog1.tr
- To see the trace file contents open the file as,
  [root@localhost~]# gedit prog1.tr



Topology
Contents of Trace
File



```
Terminal - dbmslab-sys-17@dbmslabsys17-BK339AA-ACJ-CQ1-1020IX: -
File  Edit  View  Terminal  Tabs  Help
dbmslab-sys-17@dbmslabsys17-BK339AA-ACJ-CQ1-1020IX:~$ ns prog1.tcl
dbmslab-sys-17@dbmslabsys17-BK339AA-ACJ-CQ1-1020IX:~$ awk -f prog1.awk prog1.tr
no of pckts dropped=386
no of pckts recieved=1416
dbmslab-sys-17@dbmslabsys17-BK339AA-ACJ-CQ1-1020IX:~$
```

Output

Experiment No: 2 **Transmission of Ping Messages over a Six Nodes Network Topology**

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

Program:Prog2.tcl

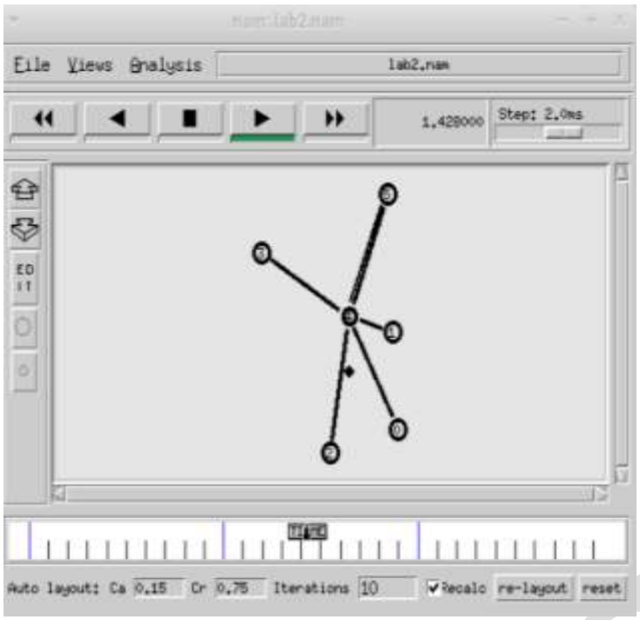
```
set ns [ new Simulator ]
set nf [ open prog2.nam w ]
$ns namtrace-all $nf
set tf [ open prog2.tr w ]
$ns trace-all $tf

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$ns duplex-link $n0 $n4 1005Mb 1ms DropTail
$ns duplex-link $n1 $n4 50Mb 1ms DropTail
$ns duplex-link $n2 $n4 2000Mb 1ms DropTail
$ns duplex-link $n3 $n4 200Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 1ms DropTail

set p1 [new Agent/Ping]
$ns attach-agent $n0 $p1
$p1 set packetSize_ 50000
$p1 set interval_ 0.0001

set p2 [new Agent/Ping]
$ns attach-agent $n1 $p2

set p3 [new Agent/Ping]
$ns attach-agent $n2 $p3
$p3 set packetSize_ 30000
$p3 set interval_ 0.00001

set p4 [new Agent/Ping]
$ns attach-agent $n3 $p4

set p5 [new Agent/Ping]
$ns attach-agent $n5 $p5



$ns queue-limit $n0 $n4 5
$ns queue-limit $n2 $n4 3
```

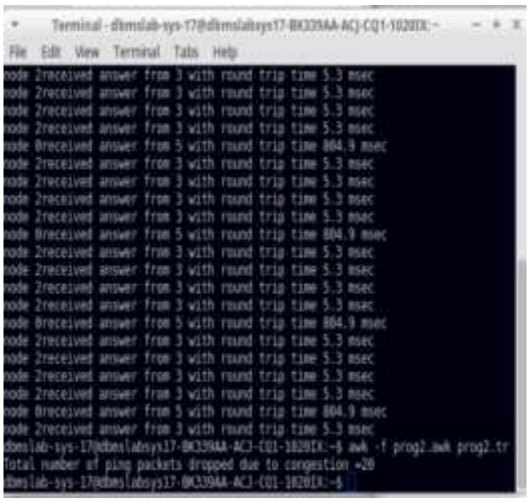
```
$ns queue-limit $n4 $n5 2

Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id]received answer from $from with round trip time $rtt msec"
}

$ns connect $p1 $p5
$ns connect $p3 $p4
proc finish { } {
        global ns nf tf
        $ns flush-trace
        close $nf
        close $tf
        exec nam lab2.nam &
        aexit 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:

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


nam: lab2.nam
File Views Analysis
lab2.nam
1.428000
Step: 2.0ms
EDIT
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc re-layout reset



```
Terminal - dbmslab-sys-17@dbmslabsys17-BK339AA-ACJ-CQ1-1020IX: ~
File  Edit  View  Terminal  Tabs  Help
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 804.9 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 804.9 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 804.9 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 804.9 msec
node 2received answer from 3 with round trip time 5.3 msec
dbmslab-sys-17@dbmslabsys17-BK339AA-ACJ-CQ1-1020IX:~$ awk -f prog2.awk prog2.tr
Total number of ping packets dropped due to congestion =20
dbmslab-sys-17@dbmslabsys17-BK339AA-ACJ-CQ1-1020IX:~$
```

Topology

Output

Note: Vary the bandwidth and queue size between the nodes n0-n2 , n2-n4. n6-n2 and n2- n5 and see the number of packets dropped at the nodes.

Experiment No: 3	**Ethernet LAN**
Aim: Simulate an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.	

Program:prog3.tcl

```
set ns [new Simulator]
set tf [open prog3.tr w]
$ns trace-all $tf
set nf [open prog3.nam w]
$ns namtrace-all $nf

set n0 [$ns node]
$n0 color "magenta"
$n0 label "src1"
set n1 [$ns node]
set n2 [$ns node]
$n2 color "magenta"
$n2 label "src2"
set n3 [$ns node]
$n3 color "blue"
$n3 label "dest2"
set n4 [$ns node]
set n5 [$ns node]
$n5 color "blue"
$n5 label "dest1"

$ns make-lan "$n0 $n1 $n2 $n3 $n4" 100Mb 100ms LL Queue/DropTail Mac/802_3
$ns duplex-link $n4 $n5 1Mb 1ms DropTail

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.0001
set sink5 [new Agent/TCPSink]
$ns attach-agent $n5 $sink5

$ns connect $tcp0 $sink5

set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ftp2 set packetSize_ 600
$ftp2 set interval_ 0.001
```

```
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3

ns connect $tcp2 $sink3
set file1 [open file1.tr w]
$tcp0 attach $file1
set file2 [open file2.tr w]
$tcp2 attach $file2

$tcp0 trace cwnd_
$tcp2 trace cwnd_

proc finish { } {
global ns nf tf
$ns flush-trace
close $tf
close $nf
exec nam prog3.nam &
exit 0
}

$ns at 0.1 "$ftp0 start"
$ns at 5 "$ftp0 stop"
$ns at 7 "$ftp0 start"
$ns at 0.2 "$ftp2 start"
$ns at 8 "$ftp2 stop"
$ns at 14 "$ftp0 stop"
$ns at 10 "$ftp2 start"
$ns at 15 "$ftp2 stop"
$ns at 16 "finish"
$ns run
```

AWK file :

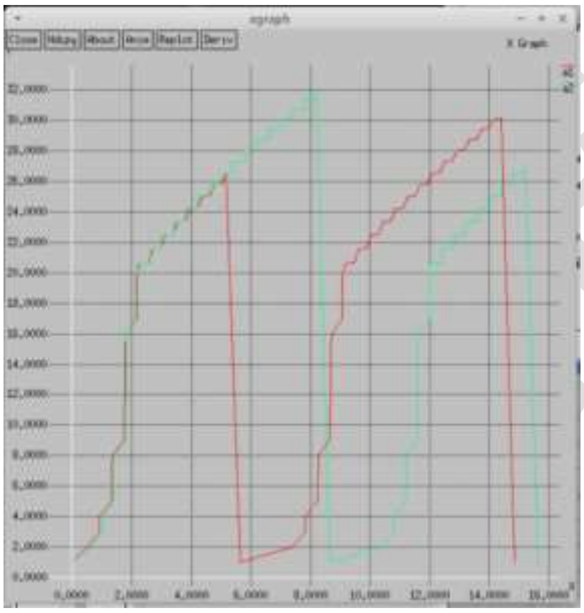
cwnd:- means congestion window

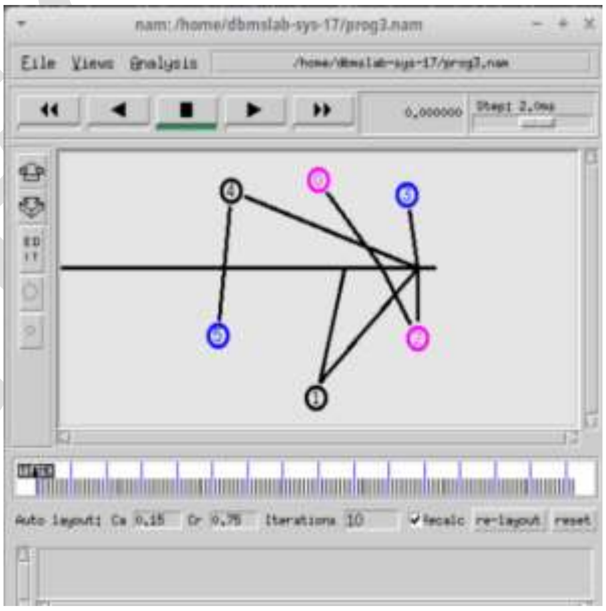
```
 BEGIN { } {
  if($6= ="cwnd_")                          / * don’t leave space after writing cwnd_ */
 printf("%f\t%f\t\n",$1,$7);     /*  put \n in printf */
 } END { }
```

Steps for execution:

- Open geditor and type program. Save the program using extension " .tcl "
  [root@localhost ~]# gedit prog3.tcl
- Open geditor and type awk program. Save the program using extension ".awk "
  [root@localhost ~]# gedit prog3.awk
- Run the simulation program
  [root@localhost~]# ns prog3.tcl
- Here "ns" indicates network simulator. We get the topology shown in the snapshot.
- Now press the play button in the simulation window and the simulation will begins.
- After simulation is completed run awk file to see the output,

  i. [root@localhost~]# awk –f prog3.awk file1.tr > a1
  ii. [root@localhost~]# awk –f prog3.awk file2.tr > a2
  iii. [root@localhost~]# xgraph a1 a2
- Here we are using the congestion window trace files i.e. file1.tr and file2.tr and we are redirecting the contents of those files to new files say a1 and a2 using output redirection operator (>).
- To see the trace file contents open the file as ,
  [root@localhost~]# gedit prog3.tr




nam: /home/dbmslab-sys-17/prog3.nam
File
Views
Analysis
/home/dbmslab-sys-17/prog3.nam
0.000000
Step: 2.0ms
Auto layout: Ca 0.15 Cr 0.75 Iterations 10
Recalc
re-layout
reset

Topology Graph

Experiment No: 4	**Simple ESS with transmitting nodes in WLAN**
Aim: Simulate simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.	

Program: prog4.tcl

```
set ns [new Simulator]
set tf [open prog4.tr w]
$ns trace-all $tf
set topo [new Topography]
$topo load_flatgrid 1000 1000
set nf [open prog4.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 \
                -prrootype 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 lab8.nam  &
        close $tf
        exit 0
}
$ns at 250 "finish"
$ns run
```

AWK file:

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

Experiment No: 5	**GSM**
Aim: Simulate and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.	

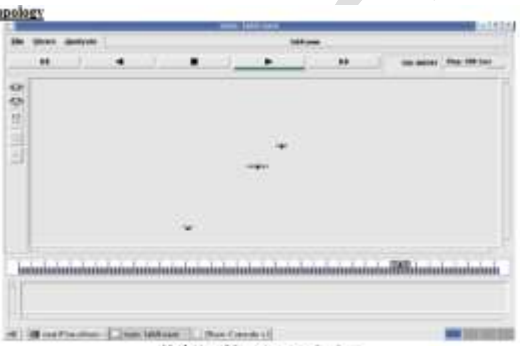
Second Generation (2G) technology is based on the technology known as global system for mobile communication (GSM). This technology enabled various networks to provide services like text messages, picture messages and MMS.

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).

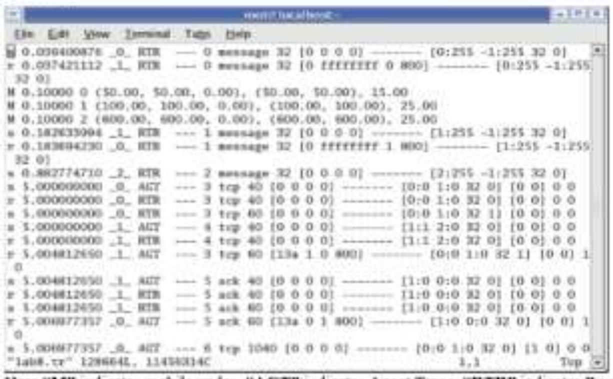
GSM can be implemented on all the versions of NS2 (Since year 2004: ns-2.27, and later versions of NS2)

Topology



Node 1 and 2 are communicating

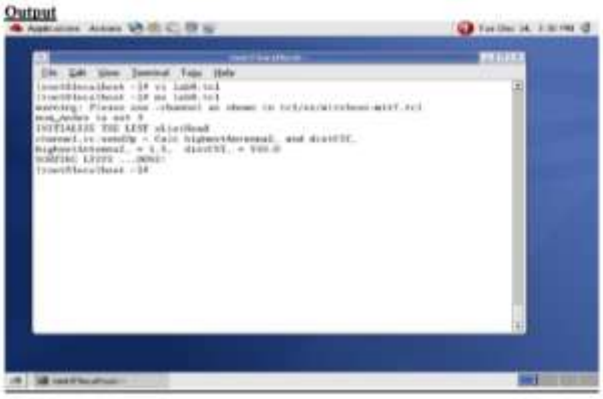
Trace file



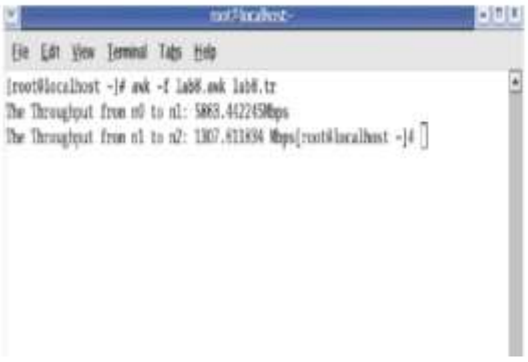
```
s 0.038400876 _0_ RTR  --- 0 message 32 [0 0 0 0] ------- [0:255 -1:255 32 0]
r 0.037421112 _1_ RTR  --- 0 message 32 [0 ffffffff 0 800] ------- [0:255 -1:255
 32 0]
M 0.10000 0 (50.00, 50.00, 0.00), (50.00, 50.00), 15.00
M 0.10000 1 (100.00, 100.00, 0.00), (100.00, 100.00), 25.00
M 0.10000 2 (600.00, 600.00, 0.00), (600.00, 600.00), 25.00
s 0.182633004 _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 5.004877357 _0_ AGT  --- 5 ack 60 [13a 0 1 800] ------- [1:0 0:0 32 0] [0 0] 1
 0
s 5.004877357 _0_ AGT  --- 6 tcp 1040 [0 0 0 0] ------- [0:0 1:0 32 0] [1 0] 0 0
"lab8.tr" 128664L, 11450314C                                  1,1          Top
```

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

Output

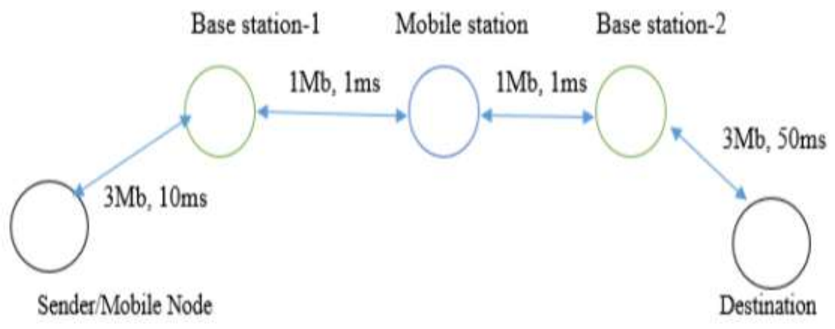


```
[root@localhost ~]# vi lab8.tcl
[root@localhost ~]# ns lab8.tcl
warning: Please use -channel as shown in tcl/ex/wireless-mitf.tcl
num_nodes is set 3
INITIALIZE THE LIST xListHead
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
[root@localhost ~]#
```



```
[root@localhost ~]# awk -f lab8.awk lab8.tr
The Throughput from n0 to n1: 5863.442245Mbps
The Throughput from n1 to n2: 1307.611834 Mbps[root@localhost ~]#
```

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
```

Experiment No: 6	**CDMA**
Aim: Simulate 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)

```
# 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
```

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).

Experiment No: 1	Error Detecting Code Using CRC-CCITT (16-bit)
Aim: A program for error detecting code using CRC-CCITT (16- bits).	

Theory

The cyclic redundancy check, or CRC, is a technique for detecting errors in digital data, but not for making corrections when errors are detected. It is used primarily in data transmission. In the CRC method, a certain number of check bits, often called a checksum, are appended to the message being transmitted. The receiver can determine whether or not the check bits agree with the data, to ascertain with a certain degree of probability whether or not an error occurred in transmission.

Error detection with CRC

Consider a message represented by the polynomial M(x)

Consider a *generating polynomial* G(x)

This is used to generate a CRC = C(x) to be appended to M(x).

Note this G(x) is prime.

Steps:

1. Multiply M(x) by highest power in G(x). i.e. Add So much zeros to M(x).
2. Divide the result by G(x). The remainder = C(x).

   Special case: This won't work if bitstring =all zeros. We don't allow such an M(x).But M(x) bitstring = 1 will work, for example. Can divide 1101 into 1000.
3. If: x div y gives remainder c

   that means: x = n y + c

   Hence (x-c) = n y

   (x-c) div y gives remainder 0

   Here (x-c) = (x+c)

   Hence (x+c) div y gives remainder 0
4. Transmit: T(x) = M(x) + C(x)
5. Receiver end: Receive T(x). Divide by G(x), should have remainder 0.

Note: If G(x) has order n - highest power is x^n, then G(x) will cover (n+1) bits and the *remainder* will cover n bits. i.e. Add n bits (Zeros) to message.

Figure 10.6: Division in CRC encoder

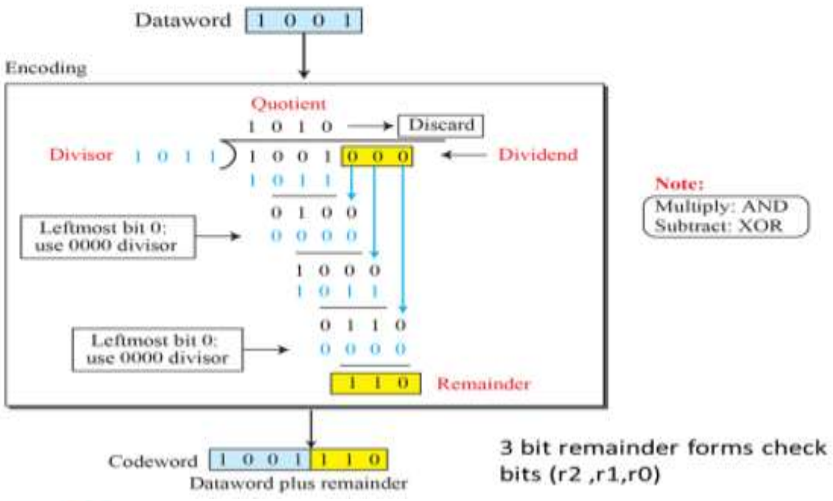
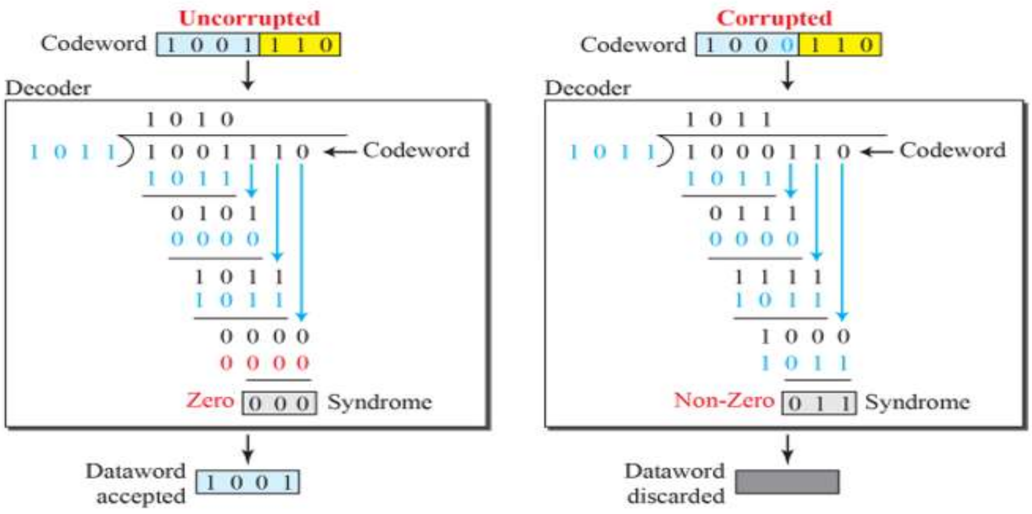


Figure 10.7: Division in the CRC decoder for two cases



Source Code:

```
import java.io.*;
class crc_gen
{
        public static void main(String args[]) throws IOException
        {
                BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
                int[] data;
               int[] div;
               int[] divisor;
              int[] rem;
              int[] crc;
              int data_bits, divisor_bits, tot_length;
```

```
                System.out.println("Enter number of data bits : ");
                data_bits=Integer.parseInt(br.readLine());
                data=new int[data_bits];

                System.out.println("Enter data bits : ");
                for(int i=0; i<data_bits; i++)
                data[i]=Integer.parseInt(br.readLine());

                System.out.println("Enter number of bits in divisor : ");
                divisor_bits=Integer.parseInt(br.readLine());
                divisor=new int[divisor_bits];

                System.out.println("Enter Divisor bits : ");
                for(int i=0; i<divisor_bits; i++)
                divisor[i]=Integer.parseInt(br.readLine());
                System.out.print("Data bits are : ");

                for(int i=0; i< data_bits; i++)
                System.out.print(data[i]);
                 System.out.println();

                System.out.print("divisor bits are : ");
                for(int i=0; i< divisor_bits; i++)
                System.out.print(divisor[i]);
                System.out.println();

                tot_length=data_bits+divisor_bits-1;
               div=new int[tot_length];
               rem=new int[tot_length];
              crc=new int[tot_length];
 /*------------------ CRC GENERATION-----------------------*/

               for(int i=0;i<data.length;i++)
              div[i]=data[i];

                System.out.print("Dividend (after appending 0's) are : ");
                for(int i=0; i< div.length; i++)
                System.out.print(div[i]);
                System.out.println();

                for(int j=0; j<div.length; j++)
                {
                        rem[j] = div[j];
                }

                 rem=divide(div, divisor, rem);
                for(int i=0;i<div.length;i++)//append dividend and ramainder
                 {
                        crc[i]=(div[i]^rem[i]);
                 }

                System.out.println();
                System.out.println("CRC code : ");
                for(int i=0;i<crc.length;i++)
                System.out.print(crc[i]);

/*-------------------ERROR DETECTION---------------------*/
                System.out.println();
                System.out.println("Enter CRC code of "+tot_length+" bits : ");
                for(int i=0; i<crc.length; i++)
                crc[i]=Integer.parseInt(br.readLine());

                /* System.out.print("crc bits are : ");
                for(int i=0; i< crc.length; i++)
                 System.out.print(crc[i]);
                System.out.println();
                */
                for(int j=0; j<crc.length; j++)
                {
                        rem[j] = crc[j];
                }

                 rem=divide(crc, divisor, rem);
                for(int i=0; i< rem.length; i++)
                {
                        if(rem[i]!=0)
                        {
                                 System.out.println("Error");
                                break;
                        }
                         if(i==rem.length-1)
                        System.out.println("No Error");
                 }

                System.out.println("THANK YOU.... :)");
        }

  static int[] divide(int div[],int divisor[], int rem[])
  {
        int cur=0;
        while(true)
        {
                for(int i=0;i<divisor.length;i++)
                 rem[cur+i]=(rem[cur+i]^divisor[i]);
                while(rem[cur]==0 && cur!=rem.length-1)
```

```
                    cur++;
                    if((rem.length-cur)<divisor.length)
                   break;
            }
           return rem;
    }

}
```

```
OUTPUT :
Enternumber of data bits :
7
Enter data bits :
1
0
1
1
0
0
1
Enter number of bits in divisor :
3
Enter Divisor bits :
1
0
1
Data bits are : 1011001
divisor bits are : 101
Dividend (after appending 0's) are : 101100100

CRC code :
101100111
Enter CRC code of 9 bits :
1
0
1
1
0
0
1
0
1
crc bits are : 101100101
Error
```

Experiment No: 8	Bellman-ford algorithm
Aim: Implement 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

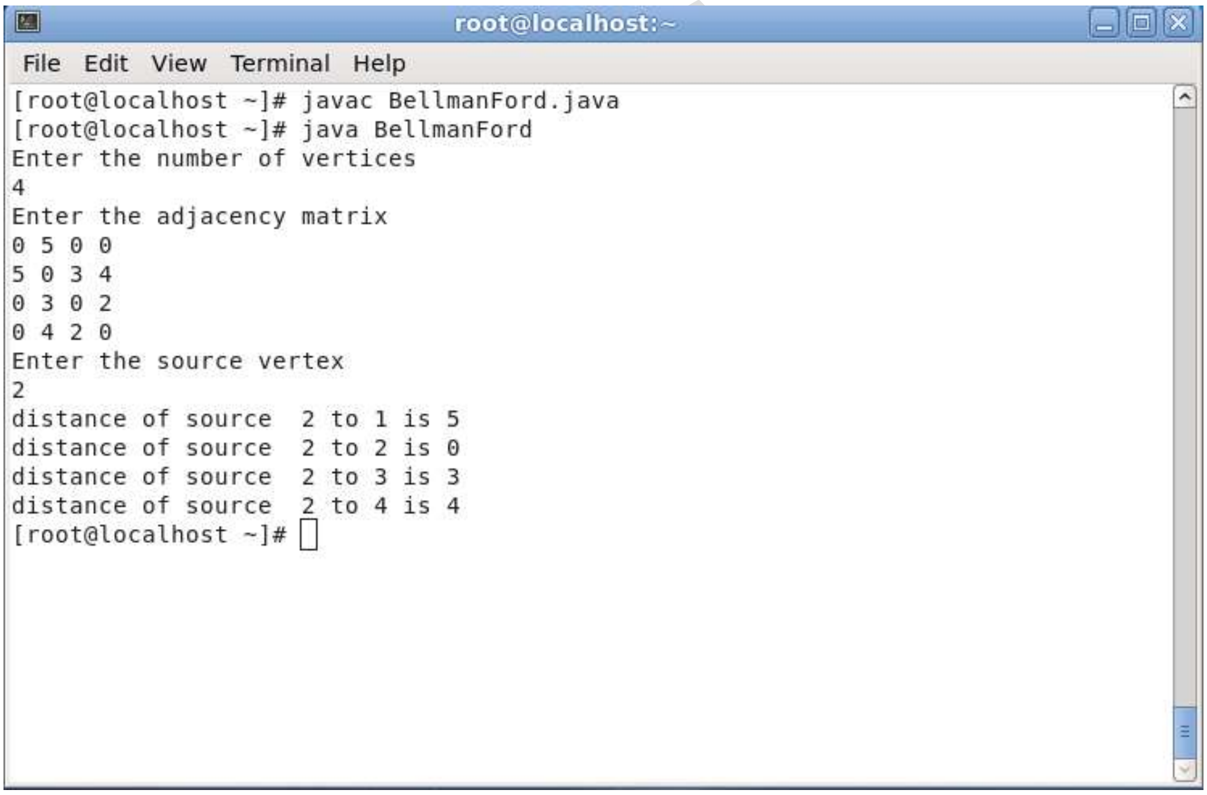
Program:

```
import java.util.Scanner;
public class BellmanFord
{
        private int D[]; private 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:



```
[root@localhost ~]# javac BellmanFord.java
[root@localhost ~]# java BellmanFord
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
[root@localhost ~]#
```

Experiment No: 9	Client-server program
Aim: Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present. Implement the above program using as message queues or FIFOs as IPC channels.	

A socket is an endpoint of a two-way communication link between two programs running on the network. Socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent.

Sockets are a protocol independent method of creating a connection between processes.
Use of sockets:

- Connection–based sockets communicate client-server: the server waits for a connection from the client
- Connectionless sockets are peer-to-peer: each process is symmetric.

A simple Server Program in Java

The steps for creating a simple server program are:
1. Open the Server Socket:
```
    ServerSocket server = new ServerSocket( PORT );
```
2. Wait for the Client Request:
```
    Socket client = server.accept();
```
3. Create I/O streams for communicating to the client
```
    DataInputStream is = new
    DataInputStream(client.getInputStream());
    DataOutputStream os = new
    DataOutputStream(client.getOutputStream());
```
4. Perform communication with client
```
    Receive from client: String line = is.readLine();
    Send to client: os.writeBytes(“Hello\n”);
```
5. Close socket:
```
    server.close();
```

A simple Client Program in Java

The steps for creating a simple client program are:
1. Create a Socket Object:
```
  Socket client = new Socket(server, port_id);
```
2. Create I/O streams for communicating with the server.
```
   is = new DataInputStream(client.getInputStream());
   os = new DataOutputStream(client.getOutputStream());
```
3. Perform I/O or communication with the server:
```
  Receive data from the server: String line = is.readLine();
   Send data to the server: os.writeBytes(“Hello\n”);
```
4. Close the socket when done:
```
  client.close();
```

Client Program:

```
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.EOFException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStreamReader;
import java.net.Socket;
import java.util.Scanner;
class Client
{
        public static void main(String args[])throws Exception
        {
                String address = "";
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter Server Address: ");
                address=sc.nextLine();
//create the socket on port 5000
                Socket s=new Socket(address,5000);
                DataInputStream din=new DataInputStream(s.getInputStream());
                DataOutputStream dout=new DataOutputStream(s.getOutputStream());
                BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
                System.out.println("Send Get to start...");
                String str="",filename="";
                try
                {
                        while(!str.equals("start"))
                        str=br.readLine();
                        dout.writeUTF(str);
                        dout.flush();
                        filename=din.readUTF();
                        System.out.println("Receving file: "+filename);
                        filename="client"+filename;
                        System.out.println("Saving as file: "+filename);
                        long sz=Long.parseLong(din.readUTF());
                        System.out.println ("File Size: "+(sz/(1024*1024))+" MB");
                        byte b[]=new byte [1024];
                        System.out.println("Receving file..");
                        FileOutputStream fos=new FileOutputStream(new
File(filename),true);
                        long bytesRead;
                do
                {
                        bytesRead = din.read(b, 0, b.length);
                        fos.write(b,0,b.length);
                }while(!(bytesRead<1024));
                        System.out.println("Comleted");
                        fos.close();
                        dout.close();
                        s.close();
                }
                catch(EOFException e)
                {
                //do nothing
                }
        }
}
```

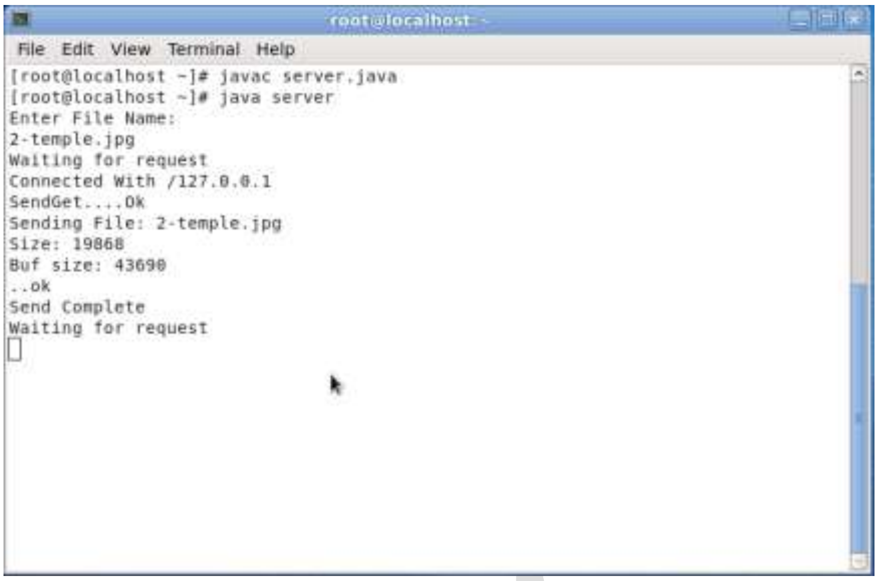
Server program:

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
class Server
{
        public static void main(String args[])throws Exception
        {
                String filename;
                System.out.println("Enter File Name: ");
                Scanner sc=new Scanner(System.in);
                filename=sc.nextLine();
                sc.close();
                while(true)
                {
//create server socket on port 5000
                        ServerSocket ss=new ServerSocket(5000);
                        System.out.println ("Waiting for request");
                        Socket s=ss.accept();
                        System.out.println ("Connected With "+s.getInetAddress().toString());
                        DataInputStream din=new DataInputStream(s.getInputStream());
                        DataOutputStream dout=new DataOutputStream(s.getOutputStream());
                        try
                        {
                                String str="";
                                str=din.readUTF();
                                System.out.println("SendGet....Ok");
                                if(!str.equals("stop"))
                                {
                                        System.out.println("Sending File: "+filename);
                                        dout.writeUTF(filename);
                                        dout.flush();
                                        File f=new File(filename);
                                        FileInputStream fin=new FileInputStream(f);
                                        long sz=(int) f.length();
                                        byte b[]=new byte [1024];
                                        int read;
                                        dout.writeUTF(Long.toString(sz));
                                        dout.flush();
                                        System.out.println ("Size: "+sz);
                                        System.out.println ("Buf size:
"+ss.getReceiveBufferSize());
                                        while((read = fin.read(b)) != -1)
                                        {
                                                dout.write(b, 0, read);
                                                dout.flush();
                                        }
                                        fin.close();
                                        System.out.println("..ok");
                                        dout.flush();
                                }
                                dout.writeUTF("stop");
                                System.out.println("Send Complete");
                                dout.flush();
                        }
                        catch(Exception e)
                        {
                                e.printStackTrace();
                                System.out.println("An error occured");
                        }
                        din.close();
                        s.close();
                        ss.close();
                }
        }
}
```

Note: Create two different files Client.java and Server.java. Follow the steps given:
1. Open a terminal run the server program and provide the filename to send
2. Open one more terminal run the client program and provide the IP address of the server. We can give localhost address "127.0.0.1" as it is running on same machine or give the IP address of the machine.
3. Send any start bit to start sending file.
4. Refer https://www.tutorialspoint.com/java/java_networking.htm for all the parameters, methods description in socket communication.

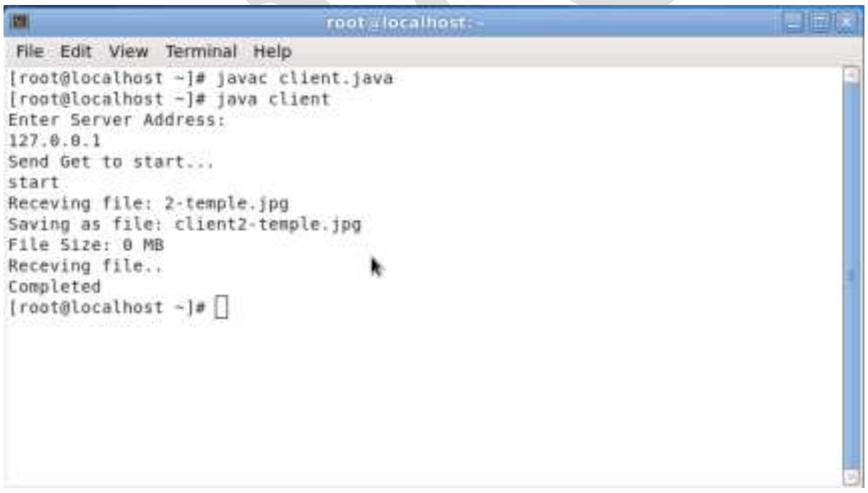
Output:

At server Side:



```
[root@localhost ~]# javac server.java
[root@localhost ~]# java server
Enter File Name:
2-temple.jpg
Waiting for request
Connected With /127.0.0.1
SendGet....Ok
Sending File: 2-temple.jpg
Size: 19868
Buf size: 43690
..ok
Send Complete
Waiting for request
```

At client side:



```
[root@localhost ~]# javac client.java
[root@localhost ~]# java client
Enter Server Address:
127.0.0.1
Send Get to start...
start
Receving file: 2-temple.jpg
Saving as file: client2-temple.jpg
File Size: 0 MB
Receving file..
Completed
[root@localhost ~]#
```

Experiment No: 10	Client-server program
Aim: 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.

Source Code:

UDP Client

```
import java.net.*;

public class UDPClient {
        public static void main(String[] args) throws Exception{
                DatagramSocket socket=new DatagramSocket(4000);
                byte data[]=new byte[1000];

                while(true) {
                        DatagramPacket request=new DatagramPacket(data, data.length);
                        socket.receive(request);
                        String str=new String(request.getData());
                        System.out.println(str);
                }
        }
}
```

UDP Server

```
import java.net.*;
import java.util.Scanner;

public class UDPServer {
        public static void main(String[] args) throws Exception{
                Scanner in = new Scanner(System.in);
                DatagramSocket socket=new DatagramSocket();
                String msg=in.nextLine();

                byte code[]=msg.getBytes();
                InetAddress iAddress=InetAddress.getByName("127.0.0.1");
                DatagramPacket request=new
DatagramPacket(code,code.length,iAddress,4000);
                socket.send(request);
                }
}
```

Note: Create two different files UDPC.java and UDPS.java. Follow the following steps:
1. Open a terminal run the server program.
2. Open one more terminal run the client program, the sent message will be received.

Output:

At Server side:
hello

At Client side:
hello

Experiment No: 11	Simple RSA algorithm
Aim: Implement a program for simple RSA algorithm to encrypt and decrypt the data.	

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.

```
import java.io.DataInputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Random;
public class RSA
{
   private BigInteger d, e, p, q, N, phi;
   private int bitlength = 1024;
   private Random r;

   public RSA()
   {
      r = new Random();
      p = BigInteger.probablePrime(bitlength, r);
      q = BigInteger.probablePrime(bitlength, r);
      N = p.multiply(q);
      phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
      e = BigInteger.probablePrime(bitlength / 2, r);
      while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0)
         e.add(BigInteger.ONE);
      d = e.modInverse(phi);
   }

   public RSA(BigInteger e, BigInteger d, BigInteger N)
   {
      this.e = e;
      this.d = d;
      this.N = N;
   }

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

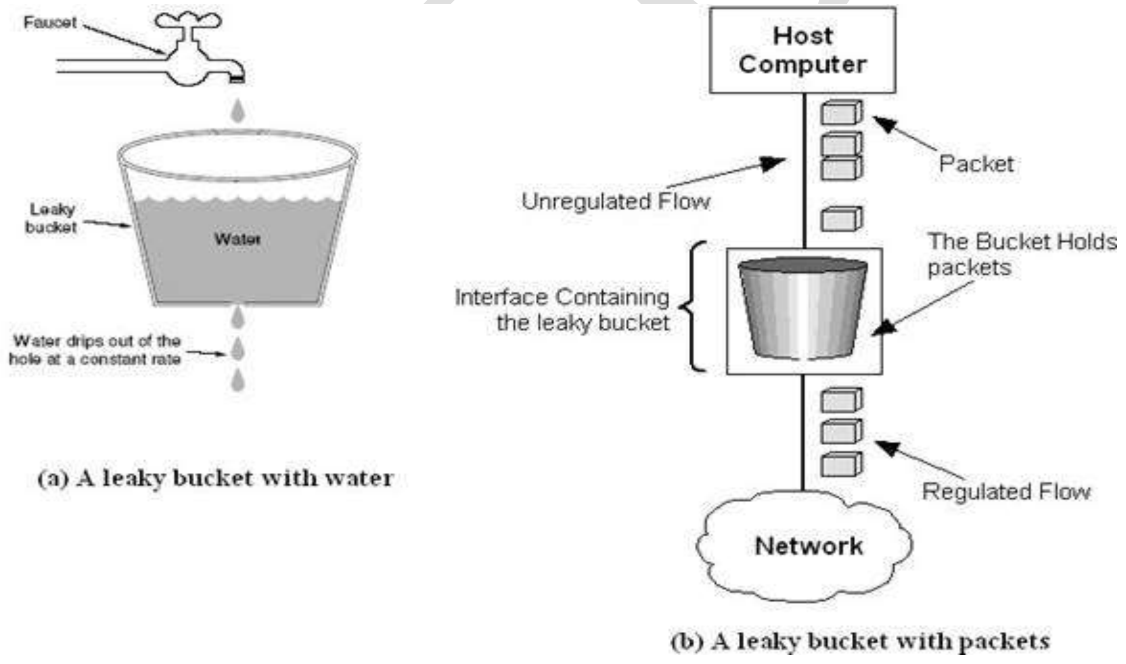
```
    {
        RSA rsa = new RSA();
        DataInputStream in = new DataInputStream(System.in);
        String teststring;
        System.out.println("Enter the plain text:");
        teststring = in.readLine();
        System.out.println("Encrypting String: " + teststring+" ");
        System.out.println("String in Bytes:  " + bytesToString(teststring.getBytes()));

        byte[] encrypted = rsa.encrypt(teststring.getBytes());

        byte[] decrypted = rsa.decrypt(encrypted);
        System.out.println("Decrypting Bytes: " + bytesToString(decrypted));
        System.out.println("Decrypted String: " + new String(decrypted));
    }

    private static String bytesToString(byte[] encrypted)
    {
        String test = "";
        for(byte b : encrypted)
            test += Byte.toString(b);
        return test;
    }

    public byte[] encrypt(byte[] message)
    {
        return (new BigInteger(message)).modPow(e, N).toByteArray();
    }

    public byte[] decrypt(byte[] message)
    {
        return (new BigInteger(message)).modPow(d, N).toByteArray();
    }
}
```

OUTPUT:

```
Enter the plain text:
hi I am student
Encrypting String: hi I am student
String in Bytes:  1041053273329710932115116117100101110116
Decrypting Bytes: 1041053273329710932115116117100101110116
Decrypted String: hi I am student
```

Experiment No: 12	Leaky bucket algorithm
Aim: Implement 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.

Implementation Algorithm:

Steps:

```
1. Read The Data For Packets
2. Read The Queue Size
3. Divide the Data into Packets
4. Assign the random Propagation delays for each packets to input into the bucket
   (input_packet).
5. wlile((Clock++<5*total_packets)and
         (out_packets< total_paclets))
    a. if (clock == input_packet)
        i. insert into Queue
    b. if (clock % 5 == 0 )
        i. Remove packet from Queue
6. End
```

```
import java.util.Scanner;
import java.lang.*;
public class LeakyBucket{
        public static void main(String[] args)
        {
                int i;
                int a[]=new int[20];
                int buck_rem=0,buck_cap=4,rate=3,sent,recv;

                Scanner in=new Scanner(System.in);

                System.out.println("enter the number of packates");
                int n=in.nextInt();
                System.out.println("enter the packates");
                for(i=1;i<=n;i++)
                        a[i]=in.nextInt();
        System.out.println("clock \t packet size \t accept \t sent \remaining");
                for(i=1;i<=n;i++) {
                        if(a[i]!=0) {
                                if(buck_rem+a[i]>buck_cap)
                                        recv=1;
                                else {
                                        recv=a[i];
                                        buck_rem+=a[i];
                                }
                        }
```

```
                    else
                            recv=0;
                    if(buck_rem!=0)
                    {
                            if(buck_rem<rate)
                            {sent=buck_rem;
                            buck_rem=0;
                                            }
                            else
                            {
                                    sent=rate;
                                    buck_rem=buck_rem-rate;
                            }
                    }
                    else
                            sent=0;
                    if(recv==-1)
System.out.println(+i+ "\t\t" +a[i]+ "\t dropped \t" +sent+ "\t" +buck_rem);
                    else
System.out.println(+i+ "\t\t" +a[i]+ "\t\t" +recv+ "\t" +sent+ "\t" +buck_rem);
                }

        }
        }
```

OUTPUT:

```
enter the number of packates
5
enter the packates
2
4
1
5
3
clock    packet size   accept   sent   Remaining
1             2          2        2       0
2             4          4        3       1
3             1          1        2       0
4             5          1        0       0
5             3          3        3       0
```

VIVA QUESTION AND ANSWER

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 anonymous FTP?
Anonymous FTP is a way of granting user access to files in public servers. Users that are allowed access to data in these servers do not need to identify themselves, but instead log in as an anonymous guest.

9) What is subnet mask?

A subnet mask is combined with an IP address in order to identify two parts: the extended network address and the host address. Like an IP address, a subnet mask is made up of 32 bits.

10) 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.

11) 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.

12) What is the job of the Network Layer under the OSI reference model?

The Network layer is responsible for data routing, packet switching and control of network congestion. Routers operate under this layer.

13) 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.

14) 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.

15) 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.

16) 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.

17) What is the importance of the OSI Physical Layer?

The physical layer does the conversion from data bits to electrical signal, and vice versa. This is where network devices and cable types are considered and setup.

18) How many layers are there under TCP/IP?

There are four layers: the Network Layer, Internet Layer, Transport Layer and Application Layer.

19) What are proxy servers and how do they protect computer networks?

Proxy servers primarily prevent external users who identifying the IP addresses of an internal network. Without knowledge of the correct IP address, even the physical location of the network cannot be identified. Proxy servers can make a network virtually invisible to external users.

20) 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.

21) 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.

22) 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.

23) How can you identify the IP class of a given IP address?

By looking at the first octet of any given IP address, you can identify whether it's Class A, B or C. If the first octet begins with a 0 bit, that address is Class A. If it begins with bits 10 then that address is a Class B address. If it begins with 110, then it's a Class C network.

24) What are firewalls?

Firewalls serve to protect an internal network from external attacks. These external threats can be hackers who want to steal data or computer viruses that can wipe out data in an instant. It also prevents other users from external networks from gaining access to the private network.

25) Describe at one disadvantage of a peer to peer network.

When you are accessing the resources that are shared by one of the workstations on the network, that workstation takes a performance hit.

26) What is Hybrid Network?

A hybrid network is a network setup that makes use of both client-server and peer-to-peer architecture.

27) 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.

28) How can you manage a network using a router?

Routers have built in console that lets you configure different settings, like security and data logging. You can assign restrictions to computers, such as what resources it is allowed access, or what particular time of the day they can browse the internet. You can even put restrictions on what websites are not viewable across the entire network.

29) What protocol can be applied when you want to transfer files between different platforms, such between UNIX systems and Windows servers?

Use FTP (File Transfer Protocol) for file transfers between such different servers. This is possible because FTP is platform independent.

30) What is Ping?

Ping is a utility program that allows you to check connectivity between network devices on the network. You can ping a device by using its IP address or device name, such as a computer name.

31) What is peer to peer?

Peer to peer are networks that does not reply on a server. All PCs on this network act as individual workstations.

32) What is DNS?

DNS is Domain Name System. The main function of this network service is to provide host names to TCP/IP address resolution.

33) 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.
34) What protocols fall under the Application layer of the TCP/IP stack?
The following are the protocols under TCP/IP Application layer: FTP, TFTP, Telnet and SMTP.
35) You need to connect two computers for file sharing. Is it possible to do this without using a hub or router?
Yes, you can connect two computers together using only one cable. A crossover type cable can be use in this scenario. In this setup, the data transmit pin of one cable is connected to the data receive pin of the other cable, and vice versa.
36) What is ipconfig?
Ipconfig is a utility program that is commonly used to identify the addresses information of a computer on a network. It can show the physical address as well as the IP address.

37) 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.
38) 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.
39) When you move the NIC cards from one PC to another PC, does the MAC address gets transferred as well?
Yes, that's because MAC addresses are hard-wired into the NIC circuitry, not the PC. This also means that a PC can have a different MAC address when the NIC card was replace by another one.
40) In a network that contains two servers and twenty workstations, where is the best place to install an Anti-virus program?
An anti-virus program must be installed on all servers and workstations to ensure protection. That's because individual users can access any workstation and introduce a computer virus when plugging in their removable hard drives or flash drives.
41) 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.
42) 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.
43) What is SMTP?
SMTP is short for Simple Mail Transfer Protocol. This protocol deals with all Internal mail, and provides the necessary mail delivery services on the TCP/IP protocol stack.
44) What is multicast routing?
Multicast routing is a targeted form of broadcasting that sends message to a selected group of user, instead of sending it to all users on a subnet.

45) 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.
46) What is the role of IEEE in computer networking?
IEEE, or the Institute of Electrical and Electronics Engineers, is an organization composed of engineers that issues and manages standards for electrical and electronic devices. This includes networking devices, network interfaces, cablings and connectors.
47) What is IPv6?
IPv6, or Internet Protocol version 6, was developed to replace IPv4. At present, IPv4 is being used to control internet traffic, but is expected to get saturated in the near future. IPv6 was designed to overcome this limitation.
48) 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.

49) **What is Leaky bucket algorithm?**
The leaky bucket algorithm is a method of temporarily storing a variable number of requests and organizing them into a set-rate output of packets in an asynchronous transfer mode (ATM) network. The leaky bucket is used to implement traffic policing and traffic shaping in Ethernet and cellular data networks.

50) What is a socket?

A socket is used to connect an application to a network protocol. A socket enables communication between a client and a server.

51) Define Datagram vs. stream

Stream can be considered as a pipe that allows full duplex connection. A datagram or a packet on the other hand, has a source and a destination

52) What is a stream socket?

A stream socket provides two way communications between a client and server. This communication is reliable and sequenced.

53)What does a socket consists of?
The combination of an IP address and a port number is called a socket.

54) What are some advantages and disadvantages of Java Sockets?

Advantages of Java Sockets:
Sockets are flexible and sufficient. Efficient socket based programming can be easily implemented for general communications.
Sockets cause low network traffic. Unlike HTML forms and CGI scripts that generate and transfer whole web pages for each new request, Java applets can send only necessary updated information.

Disadvantages of Java Sockets:
Security restrictions are sometimes overbearing because a Java applet running in a Web browser is only able to establish connections to the machine where it came from, and to nowhere else on the network
Despite all of the useful and helpful Java features, Socket based communications allows only to send packets of raw data between applications. Both the client-side and server-side have to provide mechanisms to make the data useful in any way.
Since the data formats and protocols remain application specific, the re-use of socket based implementations is limited.

55) What is the difference between a NULL pointer and a void pointer?
A NULL pointer is a pointer of any type whose value is zero. A void pointer is a pointer to an object of an unknown type, and is guaranteed to have enough bits to hold a pointer to any object. A void pointer is not guaranteed to have enough bits to point to a function.

56) What is encapsulation technique?
Hiding data within the class and making it available only through the methods. This technique is used to protect your class against accidental changes to fields, which might leave the class in an inconsistent state.

57) What is ESS?

An extended service set (**ESS**) is one or more interconnected basic service sets (BSSs) and their associated LANs. Each BSS consists of a single access point (AP) together with all wireless client devices (stations, also called STAs) creating a local or enterprise 802.11 wireless LAN (**WLAN**).

58) What is Ethernet?

Ethernet is the most widely installed local area network (LAN) technology. Ethernet is a link layer protocol in the TCP/IP stack, describing how networked devices can format data for transmission to other network devices on the same network segment, and how to put that data out on the network connection.

59) Which command would you use to ping a system in a loop from a Windows PC.

Ping -t 192.168.1.100 can be used to ping the IP address 192.168.1.100 in a loop

60) Which protocol does ping use at at the network layer
ICMP

61) What type of ICMP packet is send when a ping request is initiated?
ICMP request.

62) If a system is not responding to ping requests, what could be a possible reason
A firewall would be blocking ping requests.

63) What is the similarity between ping and tracert
Both use ICMP for communication

64) Which protocol does ping use – TCP or UDP
Ping does not use TCP or UDP. It uses ICMP.

65) What is the port number used by ping.
Ping does not use any port number as it does not work on TCP or UDP.

66) What are the standard data rates for Ethernet?
The standard data rates for Ethernet are 10 Mbps, 100 Mbps, and 1 Gbps

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

68) What is GSM?

Short form of Global System for Mobile Communications, is a wireless network system A standard for digital cellular mobile communications International roaming arrangements are enabled among mobile network operators, by providing the subscribers to use their personal mobile phones anywhere in the world.

69) What is the maximum data rate supported by a GSM system?

The maximum data rate supported by a GSM system is 9.6 kbps.

70) What is CDMA?
CDMA stands for Code Division Multiple Access. CDMA is a wireless technology used in transmission of signal from places with high Security and noise reduction.

71) What is the difference between CDMA and GSM?
Data Transfer Speed: CDMA is faster than GSM
Roaming: GSM carriers have wider coverage of more rural areas, where as CDMA may not cover rural areas compared to GSM carriers
International Roaming: GSM has facility to offer more international roaming, as the number of connections in world market dominate GSM network.
CDMA phones do not have the capacity; however, there are more countries that use CDMA networks.