

THE OXFORD COLLEGE OF ENGINEERING
Hosur Road, Bommanahalli, Bengaluru-560068
2021-2022

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING



SUBJECT NAME/CODE: COMPUTER NETWORK LABORATORY (15CSL37)

SCHEME : 2018 Scheme

SEMESTER : V

FACULTY INCHARGES: Ms. P KOKILA, Asst. Professor, ISE

TABLE OF CONTENTS

SL No.	<u>TITLE</u>
I	LIST OF PROGRAMS AS PER VTU SYLLABUS
II	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.
III	PART – B Implement the following in Java:
7.	Write a program for error detecting code using CRC-CCITT (16- bits).
8.	Write a program to find the shortest path between vertices using bellman-ford algorithm.
9.	Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.
10.	Write a program on datagram socket for client/server to display the messages on client side, typed at the server side.
11.	Write a program for simple RSA algorithm to encrypt and decrypt the data.
12.	Write a program for congestion control using leaky bucket algorithm.
IV	VIVA QUESTIONS AND ANSWERS
REFERENCES	

I. LIST OF PROGRAMS AS PER VTU SYLLABUS

COMPUTER NETWORK LABORATORY

Subject Code: 18CSL57

Number of Lecture Hours/Week 01I + 02P

Total Number of Lecture Hours 40

IA Marks 40

Exam Marks 60

Exam Hours 03

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.
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

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.

Program:

```
#creates a new simulator
set ns [new Simulator]
set nf [open prog1.nam w]
$ns namtrace-all $nf
set nd [open prog1.tr w]
$ns trace-all $nd

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

#creating nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]

$ns color 1 "red"

#linking nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 512Kb 10ms DropTail

#setting queue size of the link
$ns queue-limit $n1 $n2 5

#creating a udp connection in network simulator
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

#set up CBR over udp
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set sink [new Agent/Null]
$ns attach-agent $n2 $sink

$udp0 set class_ 1

$ns connect $udp0 $sink

#scheduling events
$ns at 0.2 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
```

\$ns run

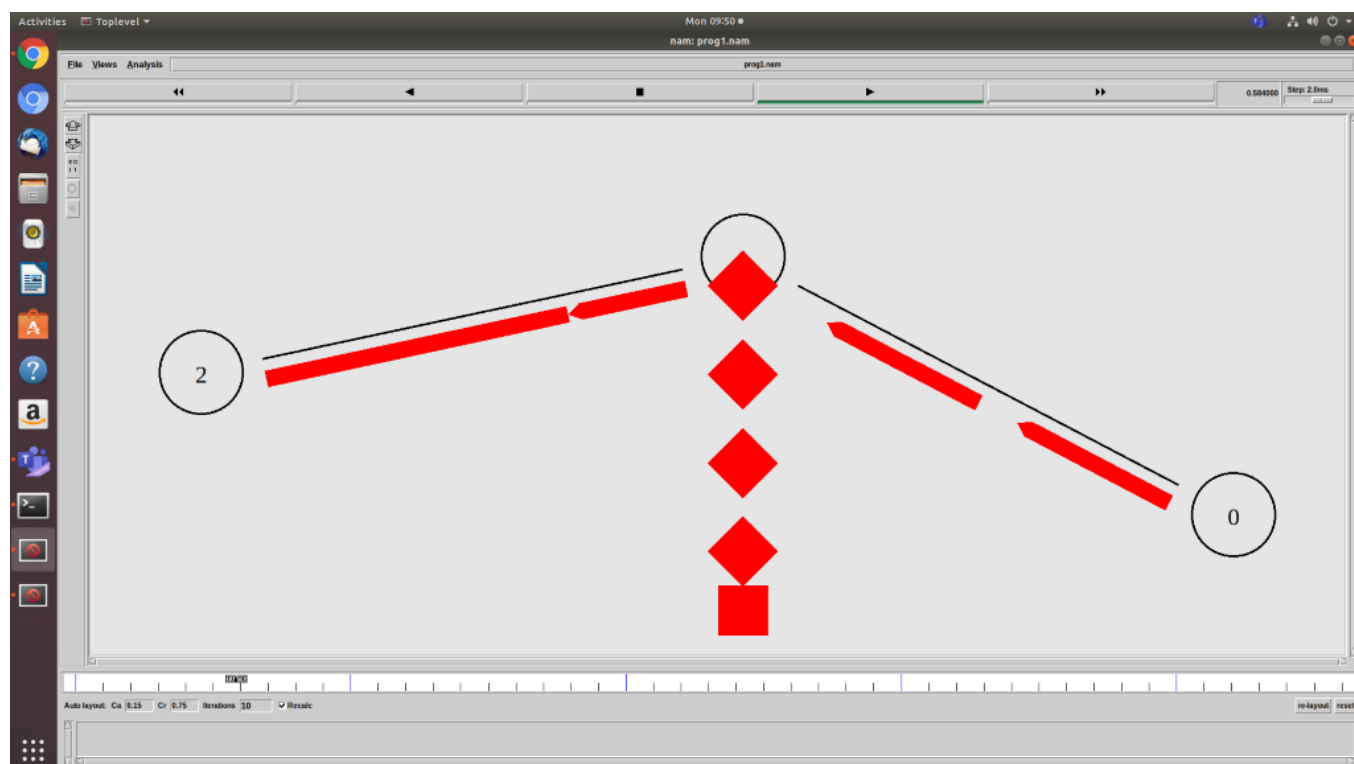
prog1.awk

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

Steps for Execution:-

1. [root@localhost ~]# vi prog1.tcl
Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press Enter key.
2. [root@localhost ~]# vi prog1.awk
Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press Enter key.
3. Run the simulation program
[root@localhost~]# ns prog1.tcl
4. After simulation is completed run awk file to see the output ,
[root@localhost~]# awk -f prog1.awk prog1.tr

Topology:-



```
ise@ise-OptiPlex-3070:~$ awk -f progml.awk prog1.tr
```

The no.of packets dropped : 306

The no.of packets recieved : 1416

```
ise@ise-OptiPlex-3070:~$ gedit prog1.tr
```

```

+ 0.2 0 1 cbr 500 ----- 1 0.0 2.0 0 0
- 0.2 0 1 cbr 500 ----- 1 0.0 2.0 0 0
+ 0.205 0 1 cbr 500 ----- 1 0.0 2.0 1 1
- 0.205 0 1 cbr 500 ----- 1 0.0 2.0 1 1
+ 0.21 0 1 cbr 500 ----- 1 0.0 2.0 2 2
- 0.21 0 1 cbr 500 ----- 1 0.0 2.0 2 2
r 0.214 0 1 cbr 500 ----- 1 0.0 2.0 0 0
+ 0.214 1 2 cbr 500 ----- 1 0.0 2.0 0 0
- 0.214 1 2 cbr 500 ----- 1 0.0 2.0 0 0
+ 0.215 0 1 cbr 500 ----- 1 0.0 2.0 3 3
- 0.215 0 1 cbr 500 ----- 1 0.0 2.0 3 3
r 0.219 0 1 cbr 500 ----- 1 0.0 2.0 1 1
+ 0.219 1 2 cbr 500 ----- 1 0.0 2.0 1 1
- 0.22 0 1 cbr 500 ----- 1 0.0 2.0 4 4
+ 0.22 0 1 cbr 500 ----- 1 0.0 2.0 4 4
- 0.221813 1 2 cbr 500 ----- 1 0.0 2.0 1 1
r 0.224 0 1 cbr 500 ----- 1 0.0 2.0 2 2
+ 0.224 1 2 cbr 500 ----- 1 0.0 2.0 2 2
- 0.225 0 1 cbr 500 ----- 1 0.0 2.0 5 5
+ 0.225 0 1 cbr 500 ----- 1 0.0 2.0 5 5
r 0.229 0 1 cbr 500 ----- 1 0.0 2.0 3 3
- 0.229 1 2 cbr 500 ----- 1 0.0 2.0 3 3
+ 0.229625 1 2 cbr 500 ----- 1 0.0 2.0 2 2
- 0.23 0 1 cbr 500 ----- 1 0.0 2.0 6 6
+ 0.23 0 1 cbr 500 ----- 1 0.0 2.0 6 6
r 0.231813 1 2 cbr 500 ----- 1 0.0 2.0 0 0
- 0.234 0 1 cbr 500 ----- 1 0.0 2.0 4 4
+ 0.234 1 2 cbr 500 ----- 1 0.0 2.0 4 4
- 0.235 0 1 cbr 500 ----- 1 0.0 2.0 7 7
+ 0.235 0 1 cbr 500 ----- 1 0.0 2.0 7 7
- 0.237438 1 2 cbr 500 ----- 1 0.0 2.0 3 3
r 0.239 0 1 cbr 500 ----- 1 0.0 2.0 5 5
+ 0.239 1 2 cbr 500 ----- 1 0.0 2.0 5 5
r 0.239625 1 2 cbr 500 ----- 1 0.0 2.0 1 1
+ 0.24 0 1 cbr 500 ----- 1 0.0 2.0 8 8
- 0.24 0 1 cbr 500 ----- 1 0.0 2.0 8 8
r 0.244 0 1 cbr 500 ----- 1 0.0 2.0 6 6
+ 0.244 1 2 cbr 500 ----- 1 0.0 2.0 6 6
- 0.245 0 1 cbr 500 ----- 1 0.0 2.0 9 9
+ 0.245 0 1 cbr 500 ----- 1 0.0 2.0 9 9
- 0.24525 1 2 cbr 500 ----- 1 0.0 2.0 4 4
r 0.247438 1 2 cbr 500 ----- 1 0.0 2.0 2 2
r 0.249 0 1 cbr 500 ----- 1 0.0 2.0 7 7
+ 0.249 1 2 cbr 500 ----- 1 0.0 2.0 7 7
- 0.25 0 1 cbr 500 ----- 1 0.0 2.0 10 10
+ 0.25 0 1 cbr 500 ----- 1 0.0 2.0 10 10
- 0.253063 1 2 cbr 500 ----- 1 0.0 2.0 5 5
r 0.254 0 1 cbr 500 ----- 1 0.0 2.0 8 8
+ 0.254 1 2 cbr 500 ----- 1 0.0 2.0 8 8
- 0.255 0 1 cbr 500 ----- 1 0.0 2.0 11 11
+ 0.255 0 1 cbr 500 ----- 1 0.0 2.0 11 11
r 0.25525 1 2 cbr 500 ----- 1 0.0 2.0 3 3
r 0.259 0 1 cbr 500 ----- 1 0.0 2.0 9 9
- 0.259 1 2 cbr 500 ----- 1 0.0 2.0 9 9
+ 0.26 0 1 cbr 500 ----- 1 0.0 2.0 12 12

```

Result:-

Thus three nodes point – to – point network with duplex links between them was implemented and output was verified successfully.

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.

Program:

```
set ns [new Simulator]
set tf [open lab2.tr w]
$ns trace-all $tf
set nf [open lab2.nam w]
$ns namtrace-all $nf

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]

$n0 label "Ping0"
$n4 label "Ping4"
$n5 label "Ping5"
$n6 label "Ping6"
$n2 label "Router"

$ns color 1 "red"
$ns color 2 "green"

$ns duplex-link $n0 $n2 100Mb 300ms DropTail
$ns duplex-link $n2 $n6 1Mb 300ms DropTail
$ns duplex-link $n5 $n2 100Mb 300ms DropTail
$ns duplex-link $n2 $n4 1Mb 300ms DropTail
$ns duplex-link $n3 $n2 1Mb 300ms DropTail
$ns duplex-link $n1 $n2 1Mb 300ms DropTail

$ns queue-limit $n0 $n2 5
$ns queue-limit $n2 $n6 2
$ns queue-limit $n2 $n4 3
$ns queue-limit $n5 $n2 5

#The below code is used to connect between the ping agents
#to the node n0, n4, n5 and n6.

set ping0 [new Agent/Ping]
$ns attach-agent $n0 $ping0

set ping4 [new Agent/Ping]
$ns attach-agent $n4 $ping4

set ping5 [new Agent/Ping]
$ns attach-agent $n5 $ping5

set ping6 [new Agent/Ping]
$ns attach-agent $n6 $ping6
```

```
$ping0 set packetSize_ 50000
$ping0 set interval_ 0.0001
$ping5 set packetSize_ 60000
$ping5 set interval_ 0.00001
```

```
$ping0 set class_ 1
$ping5 set class_ 2
```

```
$ns connect $ping0 $ping4
$ns connect $ping5 $ping6
```

```
#The below function is executed when the ping agent receives
#a reply from the destination
```

```
Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts " The node [$node_ id] received a reply from $from with round trip time of $rtt"
}
```

```
proc finish {} {
global ns nf tf
exec nam lab2.nam &
$ns flush-trace
close $tf
close $nf
exit 0
}
```

```
#The below code makes the link down(failure) at 0.9 from n2
#to n6 and when the time becomes 1.5 the link between n2 to
#n6 is enabled.
```

```
$ns rtmodel-at 0.9 down $n2 $n6
$ns rtmodel-at 1.5 up $n2 $n6
$ns at 0.1 "$ping0 send"
$ns at 0.2 "$ping0 send"
$ns at 0.3 "$ping0 send"
$ns at 0.4 "$ping0 send"
$ns at 0.5 "$ping0 send"
$ns at 0.6 "$ping0 send"
$ns at 0.7 "$ping0 send"
$ns at 0.8 "$ping0 send"
$ns at 0.9 "$ping0 send"
$ns at 1.0 "$ping0 send"
$ns at 1.1 "$ping0 send"
$ns at 1.2 "$ping0 send"
$ns at 1.3 "$ping0 send"
$ns at 1.4 "$ping0 send"
$ns at 1.5 "$ping0 send"
$ns at 1.6 "$ping0 send"
$ns at 1.7 "$ping0 send"
$ns at 1.8 "$ping0 send"
$ns at 0.1 "$ping5 send"
$ns at 0.2 "$ping5 send"
$ns at 0.3 "$ping5 send"
$ns at 0.4 "$ping5 send"
```



```
$ns at 0.5 "$ping5 send"
$ns at 0.6 "$ping5 send"
$ns at 0.7 "$ping5 send"
$ns at 0.8 "$ping5 send"
$ns at 0.9 "$ping5 send"
$ns at 1.0 "$ping5 send"
$ns at 1.1 "$ping5 send"
$ns at 1.2 "$ping5 send"
$ns at 1.3 "$ping5 send"
$ns at 1.4 "$ping5 send"
$ns at 1.5 "$ping5 send"
$ns at 1.6 "$ping5 send"
$ns at 1.7 "$ping5 send"
$ns at 1.8 "$ping5 send"
$ns at 5.0 "finish"
$ns run
```

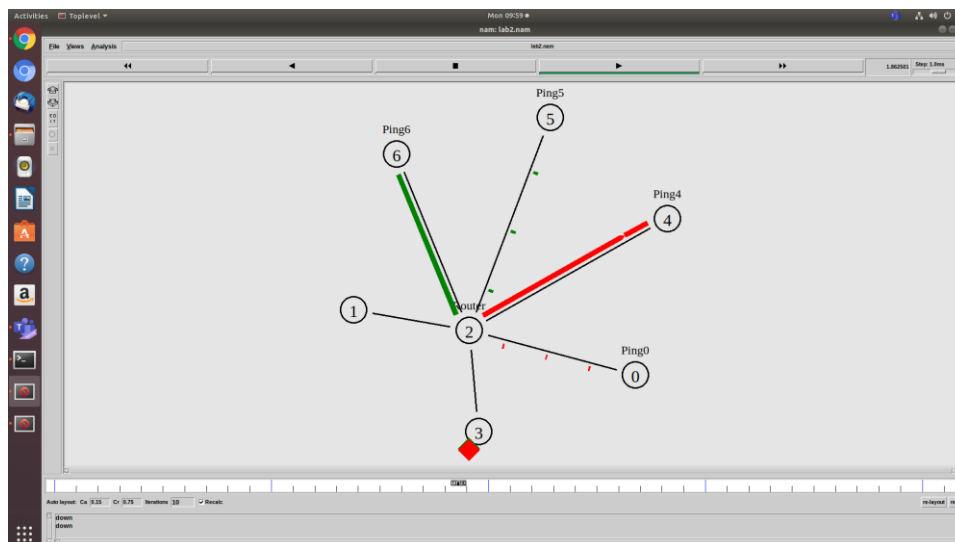
AWK Script:-

```
BEGIN{
#include<stdio.h>
count=0;
}
{
if($1=="d")
count++
}
END{
printf("The Total no of Packets Dropped due to Congestion:%d", count)
}
```

Steps for Execution:-

1. **[root@localhost ~]# vi lab2.tcl**
Save the program by pressing “**ESC key**” first, followed by “**Shift and :**” keys simultaneously and type “**wq**” and press **Enter key**.
2. **[root@localhost ~]# vi lab2.awk**
Save the program by pressing “**ESC key**” first, followed by “**Shift and :**” keys simultaneously and type “**wq**” and press **Enter key**.
3. Run the simulation program
[root@localhost~]# ns lab2.tcl
4. After simulation is completed run **awk file** to see the output ,
[root@localhost~]# awk -f lab2.awk lab2.tr

Topology:-



Output:-

ise@ise-OptiPlex-3070:~\$ ns lab2.tcl.txt

The node 0 received a reply from 4 with round trip time of 1604.5
 The node 0 received a reply from 4 with round trip time of 1904.5
 The node 0 received a reply from 4 with round trip time of 2204.5
 The node 5 received a reply from 6 with round trip time of 1685.3
 The node 0 received a reply from 4 with round trip time of 2404.5
 The node 0 received a reply from 4 with round trip time of 2304.5
 The node 5 received a reply from 6 with round trip time of 2065.3
 The node 0 received a reply from 4 with round trip time of 2404.5
 The node 5 received a reply from 6 with round trip time of 2145.3
 The node 0 received a reply from 4 with round trip time of 2404.5

ise@ise-OptiPlex-3070:~\$ awk -f lab2.awk.txt lab2.tr

The Total no of Packets Dropped due to Congestion: 20

ise@ise-OptiPlex-3070:~\$ gedit lab2.tr

```

+ 0.1 0 2 ping 50000 ..... 1 0.0 4.0 -1 0
+ 0.1 0 2 ping 50000 ..... 1 0.0 4.0 -1 0
+ 0.1 5 2 ping 60000 ..... 2 5.0 6.0 -1 1
+ 0.1 5 2 ping 60000 ..... 2 5.0 6.0 -1 1
+ 0.2 0 2 ping 50000 ..... 1 0.0 4.0 -1 2
+ 0.2 0 2 ping 50000 ..... 1 0.0 4.0 -1 2
+ 0.2 5 2 ping 60000 ..... 2 5.0 6.0 -1 3
+ 0.2 5 2 ping 60000 ..... 2 5.0 6.0 -1 3
+ 0.3 0 2 ping 50000 ..... 1 0.0 4.0 -1 4
+ 0.3 0 2 ping 50000 ..... 1 0.0 4.0 -1 4
+ 0.3 5 2 ping 60000 ..... 2 5.0 6.0 -1 5
+ 0.3 5 2 ping 60000 ..... 2 5.0 6.0 -1 5
+ 0.4 0 2 ping 50000 ..... 1 0.0 4.0 -1 6
+ 0.4 0 2 ping 50000 ..... 1 0.0 4.0 -1 6
+ 0.4 5 2 ping 60000 ..... 2 5.0 6.0 -1 7
+ 0.4 5 2 ping 60000 ..... 2 5.0 6.0 -1 7
+ 0.404 0 2 ping 50000 ..... 1 0.0 4.0 -1 0
+ 0.404 2 4 ping 50000 ..... 1 0.0 4.0 -1 0
+ 0.404 2 4 ping 50000 ..... 1 0.0 4.0 -1 0
+ 0.4048 0 2 ping 60000 ..... 2 5.0 6.0 -1 1
+ 0.4048 2 6 ping 60000 ..... 2 5.0 6.0 -1 1
+ 0.4048 2 6 ping 60000 ..... 2 5.0 6.0 -1 1
+ 0.5 0 2 ping 50000 ..... 1 0.0 4.0 -1 8
+ 0.5 0 2 ping 50000 ..... 1 0.0 4.0 -1 8
+ 0.5 5 2 ping 60000 ..... 2 5.0 6.0 -1 9
+ 0.5 5 2 ping 60000 ..... 2 5.0 6.0 -1 9
+ 0.504 0 2 ping 50000 ..... 1 0.0 4.0 -1 2
+ 0.504 2 4 ping 50000 ..... 1 0.0 4.0 -1 2
+ 0.5048 0 2 ping 60000 ..... 2 5.0 6.0 -1 3
+ 0.5048 2 6 ping 60000 ..... 2 5.0 6.0 -1 3
+ 0.6 0 2 ping 50000 ..... 1 0.0 4.0 -1 10
+ 0.6 0 2 ping 50000 ..... 1 0.0 4.0 -1 10
+ 0.6 5 2 ping 60000 ..... 2 5.0 6.0 -1 11
+ 0.6 5 2 ping 60000 ..... 2 5.0 6.0 -1 11
+ 0.604 0 2 ping 50000 ..... 1 0.0 4.0 -1 4
+ 0.604 2 4 ping 50000 ..... 1 0.0 4.0 -1 4
+ 0.6048 0 2 ping 60000 ..... 2 5.0 6.0 -1 5
+ 0.6048 2 6 ping 60000 ..... 2 5.0 6.0 -1 5
+ 0.6048 2 6 ping 60000 ..... 2 5.0 6.0 -1 5
+ 0.7 0 2 ping 50000 ..... 1 0.0 4.0 -1 12
+ 0.7 0 2 ping 50000 ..... 1 0.0 4.0 -1 12
+ 0.7 5 2 ping 60000 ..... 2 5.0 6.0 -1 13
+ 0.7 5 2 ping 60000 ..... 2 5.0 6.0 -1 13
+ 0.704 0 2 ping 50000 ..... 1 0.0 4.0 -1 6
+ 0.704 2 4 ping 50000 ..... 1 0.0 4.0 -1 6
+ 0.7048 0 2 ping 60000 ..... 2 5.0 6.0 -1 7
+ 0.7048 2 6 ping 60000 ..... 2 5.0 6.0 -1 7
+ 0.7048 2 6 ping 60000 ..... 2 5.0 6.0 -1 7
+ 0.8 0 2 ping 50000 ..... 1 0.0 4.0 -1 14
+ 0.8 0 2 ping 50000 ..... 1 0.0 4.0 -1 14
+ 0.8 5 2 ping 60000 ..... 2 5.0 6.0 -1 15
+ 0.8 5 2 ping 60000 ..... 2 5.0 6.0 -1 15
+ 0.804 2 4 ping 50000 ..... 1 0.0 4.0 -1 2
+ 0.804 0 2 ping 50000 ..... 1 0.0 4.0 -1 0
  
```

Result:-

Thus the transmission of ping messages/trace route over a network topology consisting of 6 nodes was implemented and the number of packets dropped due to congestion was found successfully.

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

```
#Make a NS simulator
set ns [new Simulator]
set tf [open lab3.tr w]
$ns trace-all $tf

set nf [open lab3.nam w]
$ns namtrace-all $nf

# Create the nodes,color and label
set n0 [$ns node]
$n0 color "magenta"
$n0 label "src1"
set n1 [$ns node]
$n1 color "red"
set n2 [$ns node]
$n2 color "magenta"
$n2 label "src2"
set n3 [$ns node]
$n3 color "blue"
$n3 label "dest2"
set n4 [$ns node]
$n4 shape square
set n5 [$ns node]
$n5 color "blue"
$n5 label "dest1"
$ns make-lan "$n0 $n1 $n2 $n3 $n4" 50Mb 100ms LL Queue/DropTail Mac/802_3

# Create the link
$ns duplex-link $n4 $n5 1Mb 1ms DropTail
# Create the node position
$ns duplex-link-op $n4 $n5 orient right

# Add a TCP sending module to node n0
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
# Setup a FTP traffic generator on "tcp0"
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.0001

# Add a TCP receiving module to node n5
set sink0 [new Agent/TCPSink]
$ns attach-agent $n5 $sink0
# Direct traffic from "tcp0" to "sink1"
$ns connect $tcp0 $sink0
```

```
# Add a TCP sending module to node n2
set tcp1 [new Agent/TCP]
$ns attach-agent $n2 $tcp1
# Setup a FTP traffic generator on "tcp1"
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set packetSize_ 600
$ftp1 set interval_ 0.001
```

```
# Add a TCP receiving module to node n3
set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
```

```
# Direct traffic from "tcp1" to "sink1"
$ns connect $tcp1 $sink1
set file1 [open file1.tr w]
$tcp0 attach $file1
set file2 [open file2.tr w]
$tcp1 attach $file2
$tcp0 trace cwnd_
$tcp1 trace cwnd_
```

```
# Define a 'finish' procedure
proc finish { } {
    global ns nf tf
    $ns flush-trace
    close $tf
    close $nf
    exec nam lab3.nam &
    exit 0
}
# Schedule start/stop times
$ns at 0.1 "$ftp0 start"
$ns at 5 "$ftp0 stop"
$ns at 7 "$ftp0 start"
$ns at 0.2 "$ftp1 start"
$ns at 8 "$ftp1 stop"
$ns at 14 "$ftp0 stop"
$ns at 10 "$ftp1 start"
$ns at 15 "$ftp1 stop"
```

```
# Set simulation end time
$ns at 16 "finish"
$ns run
```

AWK file:-

```
BEGIN {
}
{
    if($6=="cwnd_")
        printf("%f\t%f\t\n",$1,$7);
```

```

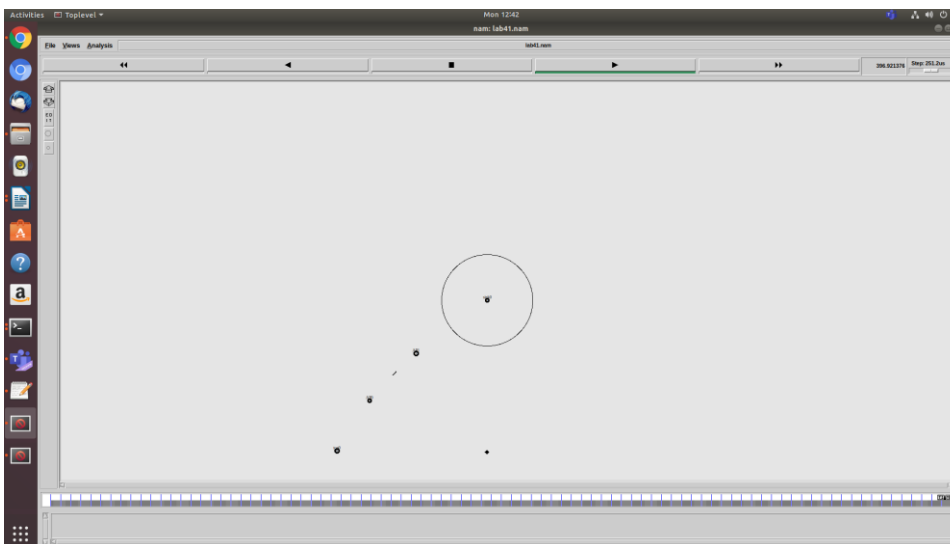
}
END {
}

```

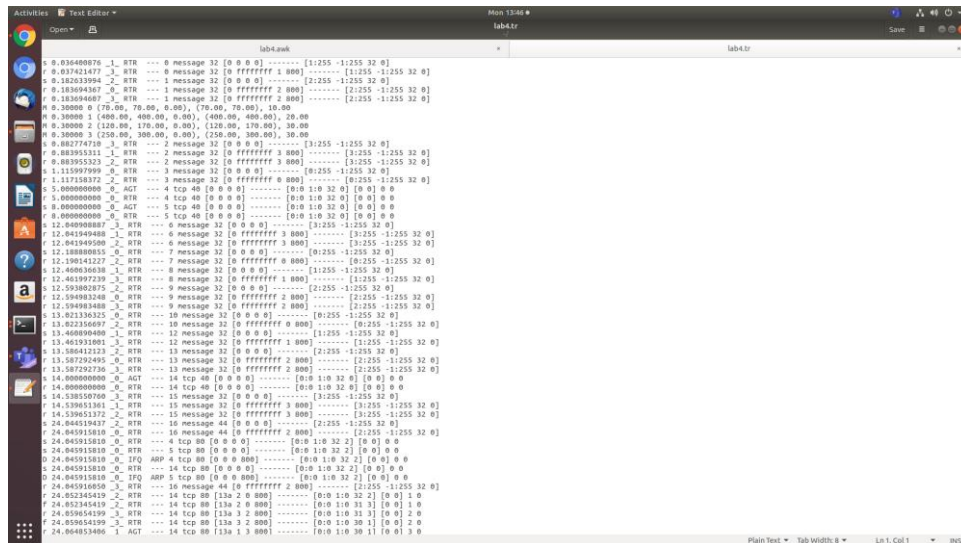
Steps for Execution:-

1. **[root@localhost ~]# vi lab3.tcl**
Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press **Enter key**.
2. **[root@localhost ~]# vi lab3.awk**
Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press **Enter key**.
3. Run the simulation program
[root@localhost~]# ns lab3.tcl
4. After simulation is completed run **awk file** to see the output ,
 - i. **[root@localhost~]# awk -f lab3.awk file1.tr > a1**
 - ii. **[root@localhost~]# awk -f lab3.awk file2.tr > a2**
 - iii. **[root@localhost~]# xgraph a1 a2**

Topology:-



Output:- ise@ise-OptiPlex-3070:~\$ gedit lab3.tr



Result:-

Thus an Ethernet LAN using n nodes and set multiple traffic nodes was implemented and congestion window for different source / destination was plotted successfully

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

```
#set ns simulator
set ns [new Simulator]
#open trace file
set nt [open lab41.tr w]
$ns trace-all $nt
set topo [new Topography]
$topo load_flatgrid 1000 1000
#open nam trace file
set nf [open lab41.nam w]
$ns namtrace-all-wireless $nf 1000 1000
```

```
$ns node-config -adhocRouting DSDV \-llType LL \-macType Mac/802_11 \-ifqType Queue/DropTail \-ifqLen 20 \-phyType Phy/WirelessPhy \-channelType Channel/WirelessChannel \-propType Propagation/TwoRayGround \-antType Antenna/OmniAntenna \-topoInstance $topo \-agentTrace ON \-routerTrace ON
```

```
create-god 4
#create nodes
```

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
#label the nodes
$n0 label "tcp0"
$n1 label "sink0"
```

```
$n2 label "AP0"
$n3 label "AP1"
$n0 set X_ 70
$n0 set Y_ 70
$n0 set Z_ 0
$n1 set X_ 400
$n1 set Y_ 400
$n1 set Z_ 0
$n2 set X_ 120
$n2 set Y_ 170
$n2 set Z_ 0
$n3 set X_ 250
$n3 set Y_ 300
$n3 set Z_ 0
$ns at 0.3 "$n0 setdest 70 70 10"
$ns at 0.3 "$n1 setdest 400 400 20"
$ns at 0.3 "$n2 setdest 120 170 30"
$ns at 0.3 "$n3 setdest 250 300 30"
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
$ns at 5 "$ftp0 start"
$ns at 120 "$n1 setdest 250 250 20"
$ns at 100 "$n0 setdest 60 60 10"
$ns at 125 "$n1 setdest 300 350 20"
$ns at 105 "$n0 setdest 50 50 10"
$ns at 130 "$n1 setdest 390 400 20"
$ns at 110 "$n0 setdest 40 40 10"
$ns at 135 "$n1 setdest 450 450 20"
$ns at 115 "$n0 setdest 30 30 10"
```

```
proc finish { } {
global ns nt nf
$ns flush-trace
exec nam lab41.nam &
close $nt
close $nf
exit 0
}
```

```
$ns at 400 "finish"
$ns run
```

AWK Script:-

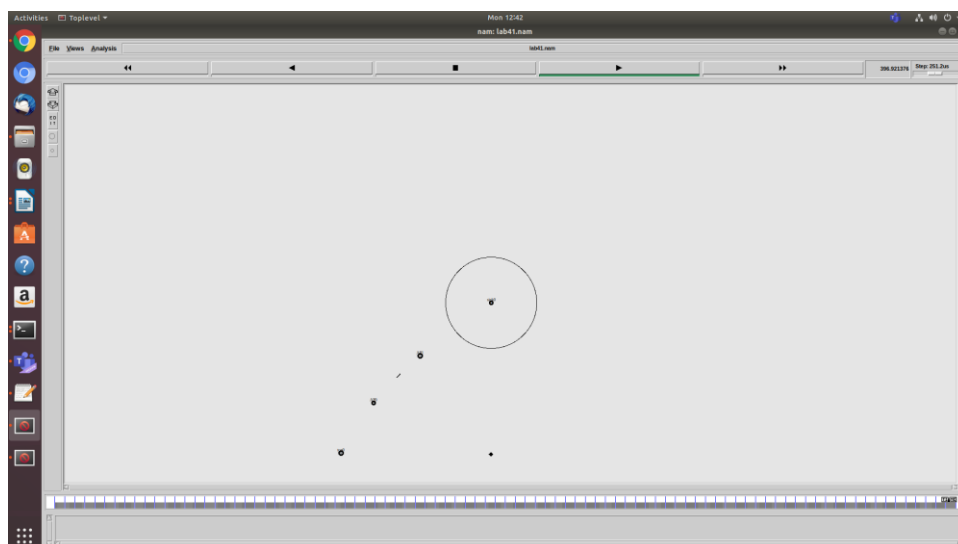
```
BEGIN{
count1=0
count2=0
pack1=0
pack2=0
time1=0
time2=0
}
{
if($1=="r"&&$3=="_1_"&&$4=="AGT")
```

```
{
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:%fMbps\n",((count1*pack1*8)/(time1*1000000)));
printf("The Throughput from n1 to n2:%fMbps\n",((count2*pack2*8)/(time2*1000000)));
}
```

Steps for Execution:-

1. **[root@localhost ~]# vi lab4.tcl**
Save the program by pressing “**ESC key**” first, followed by “**Shift and :**” keys simultaneously and type “**wq**” and press **Enter key**.
2. **[root@localhost ~]# vi lab4.awk**
Save the program by pressing “**ESC key**” first, followed by “**Shift and :**” keys simultaneously and type “**wq**” and press **Enter key**.
3. Run the simulation program
[root@localhost~]# ns lab4.tcl
4. After simulation is completed run **awk file** to see the output ,
[root@localhost~]# awk -f lab4.awk lab4.tr

Topology:-



Output:-

ise@ise-OptiPlex-3070:~\$ awk -f lab4.awk lab41.tr

The Throughput from n0 to n1:1747.285732 Mbps

The Throughput from n1 to n2:1307.611834 Mbps

ise@ise-OptiPlex-3070:~\$ gedit lab41.tr

Result:-

Thus a simple ESS with transmitting nodes in wire-less LAN was simulated and the performance with respect to transmission of packets was determined successfully.

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

Program:

```
# General Parameters
set opt(ecn) 0 ;
set opt(window) 30 ;
# Topology
set opt(type) gsm ; #type of link:
# AQM parameters
set opt(minth) 5 ;
set opt(maxth) 10 ;
set opt(adaptive) 1 ; # 1 for Adaptive RED, 0 for plain RED
#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]
set nf [open out1.nam w]
$ns trace-all $tf
$ns namtrace-all $nf
```

```
set nodes(s) [$ns node]
set nodes(bs1) [$ns node]
set nodes(ms) [$ns node]
set nodes(bs2) [$ns node]
set nodes(d) [$ns node]
proc cell_topo {} {
  global ns nodes
  $ns duplex-link $nodes(s) $nodes(bs1) 3Mbps 10ms DropTail
  $ns duplex-link $nodes(bs1) $nodes(ms) 1Mbps 1ms RED
  $ns duplex-link $nodes(ms) $nodes(bs2) 1Mbps 1ms RED
  $ns duplex-link $nodes(bs2) $nodes(d) 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
#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
set tcp1 [$ns create-connection TCP/Sack1 $nodes(s) TCPSink/Sack1 $nodes(d) 0]
set ftp1 [[set tcp1] attach-app FTP]
$ns at 0.5 "$ftp1 start"
proc stop {} {
  global nodes ns opt nf tf
  $ns flush-trace
```

```
close $nf
close $tf
exec nam out1.nam &
exit 0
}
$ns at 100 "stop"
$ns run
```

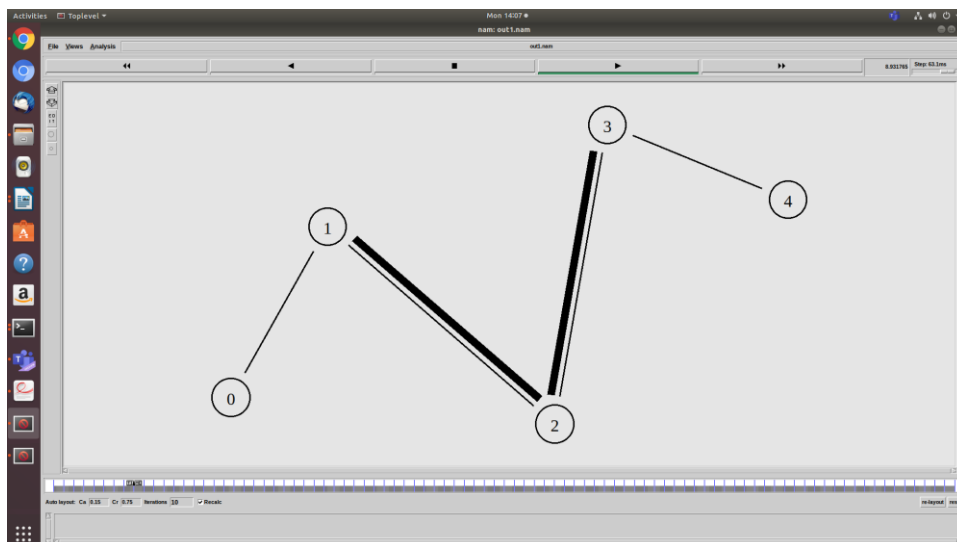
AWK Script:-

```
BEGIN {
PacketRcvd=0;
Throughput=0.0;
}
{
if(($1=="r") && ($5=="tcp") && ($10=4.0))
{
PacketRcvd++;
}
}
END {
Throughput=((PacketRcvd*1000*8)/(95.0*1000000));
printf("Packet received:%f\n", PacketRcvd);
printf( "The throughput is:%f\n",Throughput);
}
```

Steps for Execution:-

1. [root@localhost ~]# vi lab5.tcl
Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press Enter key.
2. [root@localhost ~]# vi lab5.awk
Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press Enter key.
3. Run the simulation program
[root@localhost~]# ns lab5.tcl
4. After simulation is completed run awk file to see the output ,
[root@localhost~]# awk -f lab5.awk out.tr

Topology:-



GSM Trace File:-

ise@ise-OptiPlex-3070:~\$ gedit out.tr

```

0.5 0 1 tcp 40 ----- 0.0 0.0 0.0
0.5 0 1 tcp 40 ----- 0.0 0.0 0.0
0.510107 0 1 tcp 40 ----- 0.0 0.0 0.0
0.510107 1 2 tcp 40 ----- 0.0 0.0 0.0
0.510107 1 2 tcp 40 ----- 0.0 0.0 0.0
1.04344 1 2 tcp 40 ----- 0.0 0.0 0.0
1.04344 2 3 tcp 40 ----- 0.0 0.0 0.0
1.04344 2 3 tcp 40 ----- 0.0 0.0 0.0
1.570773 2 3 tcp 40 ----- 0.0 0.0 0.0
1.570773 3 4 tcp 40 ----- 0.0 0.0 0.0
1.570773 3 4 tcp 40 ----- 0.0 0.0 0.0
1.62088 3 4 tcp 40 ----- 0.0 0.0 0.0
1.62088 4 3 ack 40 ----- 0.0 0.0 0.1
1.62088 4 3 ack 40 ----- 0.0 0.0 0.1
1.670907 4 3 ack 40 ----- 0.0 0.0 0.1
1.670907 3 2 ack 40 ----- 0.0 0.0 0.1
1.670907 3 2 ack 40 ----- 0.0 0.0 0.1
2.21032 3 2 ack 40 ----- 0.0 0.0 0.1
2.21032 2 1 ack 40 ----- 0.0 0.0 0.1
2.21032 2 1 ack 40 ----- 0.0 0.0 0.1
2.743033 2 1 ack 40 ----- 0.0 0.0 0.1
2.743033 1 0 ack 40 ----- 0.0 0.0 0.1
2.743033 1 0 ack 40 ----- 0.0 0.0 0.1
2.75376 1 0 ack 40 ----- 0.0 0.0 0.1
2.75376 0 1 tcp 1040 ----- 0.0 0.0 1.2
2.75376 0 1 tcp 1040 ----- 0.0 0.0 1.2
2.75376 0 1 tcp 1040 ----- 0.0 0.0 2.3
2.756533 0 1 tcp 1040 ----- 0.0 0.0 2.3
2.760533 0 1 tcp 1040 ----- 0.0 0.0 1.2
2.760533 1 2 tcp 1040 ----- 0.0 0.0 1.2
2.769307 0 1 tcp 1040 ----- 0.0 0.0 2.3
2.769307 1 2 tcp 1040 ----- 0.0 0.0 2.3
3.0332 1 2 tcp 1040 ----- 0.0 0.0 2.3
4.1332 1 2 tcp 1040 ----- 0.0 0.0 1.2
4.1332 2 3 tcp 1040 ----- 0.0 0.0 1.2
4.999807 2 3 tcp 1040 ----- 0.0 0.0 2.3
4.999807 2 3 tcp 1040 ----- 0.0 0.0 2.3
5.499807 2 3 tcp 1040 ----- 0.0 0.0 1.2
5.499807 3 4 tcp 1040 ----- 0.0 0.0 1.2
5.499807 4 3 ack 1040 ----- 0.0 0.0 1.2
5.5204 3 4 tcp 1040 ----- 0.0 0.0 1.2
5.5204 4 3 ack 40 ----- 0.0 0.0 1.4
5.602747 4 3 ack 40 ----- 0.0 0.0 1.4
5.602747 3 2 ack 40 ----- 0.0 0.0 1.4
5.602747 3 2 ack 40 ----- 0.0 0.0 1.4
6.13008 3 2 ack 40 ----- 0.0 0.0 1.4
6.13008 2 1 ack 40 ----- 0.0 0.0 1.4
6.360533 2 3 tcp 1040 ----- 0.0 0.0 2.3
6.360533 3 4 tcp 1040 ----- 0.0 0.0 2.3
6.360533 3 4 tcp 1040 ----- 0.0 0.0 2.3
Loading file "/home/ise/out.tr"

```

Output:

ise@ise-OptiPlex-3070:~\$ awk -f lab5.awk out.tr

Packet received:410.000000

|The throughput is:0.034526

Result:-

Thus the performance of GSM on NS2/NS3 (Using MAC layer) was studied and implemented successfully.

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

Program:

```

# General Parameters
set opt(ecn) 0 ;
set opt(window) 30 ;
# Topology
set opt(type) umts ; #type of link:
# AQM parameters

```

```
set opt(minth) 5 ;
set opt(maxth) 10 ;
set opt(adaptive) 1 ; # 1 for Adaptive RED, 0 for plain RED
#default downlink bandwidth in bps
set bwDL(umts) 9600
#default uplink bandwidth in bps
set bwUL(umts) 9600
#default downlink propagation delay in seconds
set propDL(umts) .500
#default uplink propagation delay in seconds
set propUL(umts) .500
#default buffer size in packets
set buf(umts) 10

set ns [new Simulator]
set tf [open out1.tr w]
set nf [open out2.nam w]
$ns trace-all $tf
$ns namtrace-all $nf
set nodes(s) [$ns node]
set nodes(bs1) [$ns node]
set nodes(ms) [$ns node]
set nodes(bs2) [$ns node]
set nodes(d) [$ns node]
proc cell_topo {} {
    global ns nodes
    $ns duplex-link $nodes(s) $nodes(bs1) 3Mbps 10ms DropTail
    $ns duplex-link $nodes(bs1) $nodes(ms) 1Mbps 1ms RED
    $ns duplex-link $nodes(ms) $nodes(bs2) 1Mbps 1ms RED
    $ns duplex-link $nodes(bs2) $nodes(d) 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
```

```
#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
set tcp1 [$ns create-connection TCP/Sack1 $nodes(s) TCPSink/Sack1 $nodes(d) 0]
set ftp1 [[set tcp1] attach-app FTP]
$ns at 0.5 "$ftp1 start"
proc stop {} {
global nodes ns opt nf tf
$ns flush-trace
close $nf
close $tf
exec nam out1.nam &
exit 0
}
$ns at 100 "stop"
$ns run
```

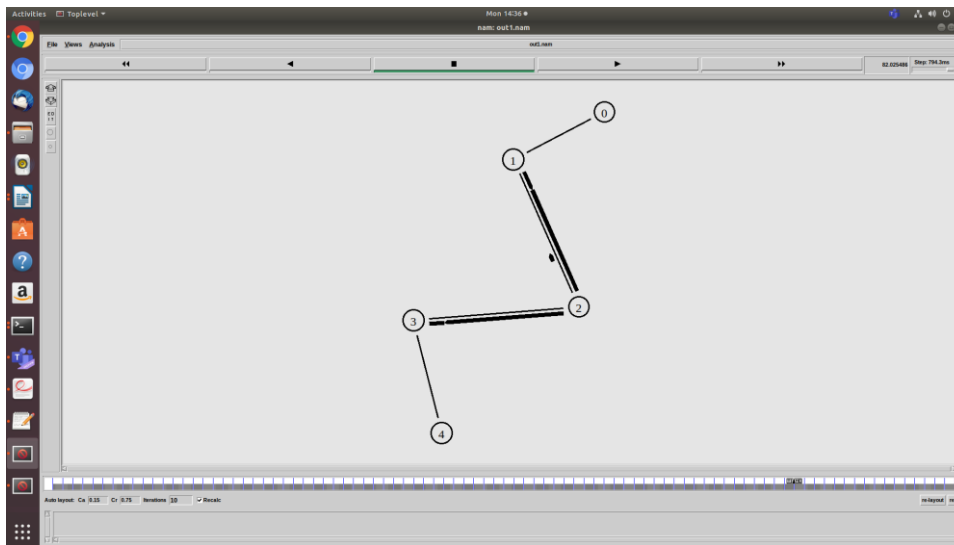
AWK Script:-

```
BEGIN {
PacketRcvd=0;
Throughput=0.0;
}
{
if(($1=="r") && ($5=="tcp") && ($10=4.0))
{
PacketRcvd++;
}
}
END {
Throughput=((PacketRcvd*1000*8)/(95.0*1000000));
printf("Packet received:%f\n", PacketRcvd);
printf( "The throughput is:%f\n",Throughput);
}
```

Steps for Execution:-

1. **[root@localhost ~]# vi lab6.tcl**
Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press **Enter key**.
2. **[root@localhost ~]# vi lab6.awk**
Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press **Enter key**.
3. Run the simulation program
[root@localhost~]# ns lab6.tcl
4. After simulation is completed run **awk file** to see the output ,
[root@localhost~]# awk -f lab6.awk out1.tr

Topology:



CDMA Trace file:

```

0.0 0 1 tcp 40 ----- 0 0.0 4.0 0.0
0.1 0 1 tcp 40 ----- 0 0.0 4.0 0.0
0.510187 0 1 tcp 40 ----- 0 0.0 4.0 0.0
0.510187 1 2 tcp 40 ----- 0 0.0 4.0 0.0
0.510187 1 2 tcp 40 ----- 0 0.0 4.0 0.0
1.04344 1 2 tcp 40 ----- 0 0.0 4.0 0.0
1.04344 2 3 tcp 40 ----- 0 0.0 4.0 0.0
1.04344 2 3 tcp 40 ----- 0 0.0 4.0 0.0
1.576773 2 3 tcp 40 ----- 0 0.0 4.0 0.0
1.576773 3 4 tcp 40 ----- 0 0.0 4.0 0.0
1.576773 3 4 tcp 40 ----- 0 0.0 4.0 0.0
1.62888 3 4 tcp 40 ----- 0 0.0 4.0 0.0
1.62888 4 3 ack 40 ----- 0 4.0 0.0 1
1.62888 4 3 ack 40 ----- 0 4.0 0.0 1
1.676987 4 3 ack 40 ----- 0 4.0 0.0 1
1.676987 3 2 ack 40 ----- 0 4.0 0.0 1
1.676987 3 2 ack 40 ----- 0 4.0 0.0 1
2.21832 3 2 ack 40 ----- 0 4.0 0.0 1
2.21832 2 1 ack 40 ----- 0 4.0 0.0 1
2.21832 2 1 ack 40 ----- 0 4.0 0.0 1
2.743653 2 1 ack 40 ----- 0 4.0 0.0 1
2.743653 1 0 ack 40 ----- 0 4.0 0.0 1
2.743653 1 0 ack 40 ----- 0 4.0 0.0 1
2.75376 1 0 tcp 1040 ----- 0 0.0 4.0 1.2
2.75376 0 1 tcp 1040 ----- 0 0.0 4.0 1.2
2.75376 0 1 tcp 1040 ----- 0 0.0 4.0 2.3
2.756533 0 1 tcp 1040 ----- 0 0.0 4.0 2.3
2.766533 0 1 tcp 1040 ----- 0 0.0 4.0 1.2
2.766533 1 2 tcp 1040 ----- 0 0.0 4.0 1.2
2.766533 1 2 tcp 1040 ----- 0 0.0 4.0 2.3
2.769387 0 1 tcp 1040 ----- 0 0.0 4.0 2.3
2.769387 1 2 tcp 1040 ----- 0 0.0 4.0 2.3
3.6332 1 2 tcp 1040 ----- 0 0.0 4.0 2.3
4.1332 1 2 tcp 1040 ----- 0 0.0 4.0 1.2
4.1332 2 3 tcp 1040 ----- 0 0.0 4.0 1.2
4.1332 2 3 tcp 1040 ----- 0 0.0 4.0 2.3
4.999867 2 3 tcp 1040 ----- 0 0.0 4.0 2.3
4.999867 2 3 tcp 1040 ----- 0 0.0 4.0 2.3
5.499867 3 4 tcp 1040 ----- 0 0.0 4.0 1.2
5.499867 3 4 tcp 1040 ----- 0 0.0 4.0 1.2
5.55264 3 4 tcp 1040 ----- 0 0.0 4.0 1.2
5.55264 4 3 ack 40 ----- 0 4.0 0.0 1.4
5.55264 4 3 ack 40 ----- 0 4.0 0.0 1.4
5.682747 4 3 ack 40 ----- 0 4.0 0.0 1.4
5.682747 3 2 ack 40 ----- 0 4.0 0.0 1.4
6.13088 3 2 ack 40 ----- 0 4.0 0.0 1.4
6.13088 2 1 ack 40 ----- 0 4.0 0.0 1.4
6.13088 2 1 ack 40 ----- 0 4.0 0.0 1.4
6.366533 2 3 tcp 1040 ----- 0 0.0 4.0 2.3
6.366533 3 4 tcp 1040 ----- 0 0.0 4.0 2.3
6.366533 3 4 tcp 1040 ----- 0 0.0 4.0 2.3

```

Output:

ise@ise-OptiPlex-3070:~\$ awk -f lab6.awk out1.tr

Packet received:410.000000

The throughput is:0.034526

Result:-

Thus the performance of CDMA on NS2/NS3 (Using stack called Call net) was studied and implemented successfully.

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

The CCITT, now known as the ITU-T (for Telecommunication Standardization Sector of the International Telecommunications Union), is the primary international body for fostering cooperative standards for telecommunications equipment and systems. It is located in Geneva, Switzerland.

Algorithm:-

1. Given a bit string, append 0 s to the end of it (the number of 0 s is the same as the degree of the generator polynomial) let $B(x)$ be the polynomial corresponding to B.
2. Divide $B(x)$ by some agreed on polynomial $G(x)$ (generator polynomial) and determine the remainder $R(x)$. This division is to be done using Modulo 2 Division.
3. Define $T(x) = B(x) - R(x)$
($T(x)/G(x) \Rightarrow$ remainder 0)
4. Transmit T, the bit string corresponding to $T(x)$.
5. Let T' represent the bit stream the receiver gets and $T'(x)$ the associated polynomial. The receiver divides $T'(x)$ by $G(x)$. If there is a 0 remainder, the receiver concludes $T = T'$ and no error occurred otherwise, the receiver concludes an error occurred and requires a retransmission.

Program:

```
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());

        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 remainder
{
    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());

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:

ise@ise-OptiPlex-3070:~\$ gedit crc_gen.java

ise@ise-OptiPlex-3070:~\$ javac crc_gen.java

ise@ise-OptiPlex-3070:~\$ java crc_gen

Enter number of data bits :

4

Enter data bits :

1

0

0

1

Enter number of bits in divisor :

4

Enter Divisor bits :

1

0

1

1

Dividend (after appending 0's) are : 1001000

CRC code :

1001110

Enter CRC code of 7 bits :

1

0

0

1

1

1

0

No Error

ise@ise-OptiPlex-3070:~\$ java crc_gen

Enter number of data bits :

4

Enter data bits :

1

0

0

1

Enter number of bits in divisor :

4
Enter Divisor bits :
1
0
1
1
Dividend (after appending 0's) are : 1001000

CRC code :
1001110
Enter CRC code of 7 bits :
1
0
0
0
1
1
0
Error

Result:

Thus the program for error detecting code using CRC-CCITT (16- bits) was implemented successfully.

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

Algorithm:

At each node, x:

```
1  Initialization:
2    for all destinations y in N:
3       $D_x(y) = c(x,y)$  /* if y is not a neighbor then  $c(x,y) = \infty$  */
4    for each neighbor w
5       $D_w(y) = ?$  for all destinations y in N
6    for each neighbor w
7      send distance vector  $D_x = [D_x(y): y \text{ in } N]$  to w
8
9  loop
10   wait (until I see a link cost change to some neighbor w or
11         until I receive a distance vector from some neighbor w)
12
13   for each y in N:
14      $D_x(y) = \min_v \{c(x,v) + D_v(y)\}$ 
15
16   if  $D_x(y)$  changed for any destination y
17     send distance vector  $D_x = [D_x(y): y \text{ in } N]$  to all neighbors
18
19  forever
```

Program:

```
import java.util.Scanner;

class BellmanFord
{
    static int n, dest;
    static double[] prevDistanceVector, distanceVector;
    static double[][] adjacencyMatrix;
    public static void main(String[] args)
    {
```

```
Scanner scanner = new Scanner(System.in);
System.out.println("Enter number of nodes");
n = scanner.nextInt();
adjacencyMatrix = new double[n][n];
System.out.println("Enter Adjacency Matrix (Use 'Zero' for No Link)");
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        adjacencyMatrix[i][j] = scanner.nextDouble();

System.out.println("Enter destination vertex");
dest = scanner.nextInt();

distanceVector = new double[n];
for (int i = 0; i < n; i++)
    distanceVector[i] = Double.POSITIVE_INFINITY;
distanceVector[dest - 1] = 0;

bellmanFordAlgorithm();

System.out.println("Distance Vector");
for (int i = 0; i < n; i++) {
    if (i == dest - 1) {
        continue;
    }
    System.out.println("Distance from " + (i + 1) + " is " + distanceVector[i]);
}
System.out.println();
}

static void bellmanFordAlgorithm() {
    for (int i = 0; i < n - 1; i++) {
        prevDistanceVector = distanceVector.clone();
        for (int j = 0; j < n; j++) {
            double min = Double.POSITIVE_INFINITY;
            for (int k = 0; k < n; k++) {
                if (min > adjacencyMatrix[j][k] + prevDistanceVector[k]) {
                    min = adjacencyMatrix[j][k] + prevDistanceVector[k];
                }
            }
            distanceVector[j] = min;
        }
    }
}
}
```

OUTPUT:

```
ise@ise-OptiPlex-3070:~$ gedit BellmanFord.java
```

```
ise@ise-OptiPlex-3070:~$ javac BellmanFord.java
```

```
ise@ise-OptiPlex-3070:~$ java BellmanFord
```

```
Enter number of nodes
```

```
3
```

```
Enter Adjacency Matrix (Use 'Zero' for No Link)
```

```
0 2 3
```

```
2 0 1
3 1 0
Enter destination vertex
2
Distance Vector
Distance from 1 is 2.0
Distance from 3 is 1.0
```

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.

Server Program: serv.java

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
public class serv
{
    public static void main(String args[]) throws Exception
    {
        // establishing the connection with the server
        ServerSocket sersock = new ServerSocket(4000);
        System.out.println("Server ready for connection");
        Socket sock = sersock.accept(); // binding with port: 4000
        System.out.println("Connection is successful and waiting for chatting"); // reading the file name from
        client
        InputStream istream = sock.getInputStream( );
        BufferedReader fileRead =new BufferedReader(new InputStreamReader(istream));
        String fname = fileRead.readLine( ); // reading file contents
        BufferedReader contentRead = new BufferedReader(new FileReader(fname) ); // keeping output
        stream ready to send the contents
        OutputStream ostream = sock.getOutputStream( );
        PrintWriter pwrite = new PrintWriter(ostream, true);
        String str;
        while((str = contentRead.readLine()) != null) // reading line-by-line from file
        {
            pwrite.println(str); // sending each line to client
        }
        sock.close();
        sersock.close(); // closing network sockets
        pwrite.close();
        fileRead.close();
        contentRead.close();
    }
}
```

Client Program: clnt.java

```
import java.net.Socket;
```

```
import java.io.*;
public class clnt
{
public static void main( String args[ ] ) throws Exception
{
Socket sock = new Socket( "127.0.0.1", 4000); // reading the file name from keyboard. Uses input
stream
System.out.print("Enter the file name");
BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));
String fname = keyRead.readLine(); // sending the file name to server. Uses PrintWriter
OutputStream ostream = sock.getOutputStream( );
PrintWriter pwrite = new PrintWriter(ostream, true);
pwrite.println(fname); // receiving the contents from server. Uses input stream
InputStream istream = sock.getInputStream();
BufferedReader socketRead = new BufferedReader(new InputStreamReader(istream));
String str;
while((str = socketRead.readLine()) != null) // reading line-by-line
{
System.out.println(str);
}
}
}
```

Note: Create two different files Client.java and Server.java.

Run server program first then run client program.

Follow the steps:

1. Open a terminal run the server program.
2. Open one more terminal run the client program

Output: Server side

```
ise@ise-OptiPlex-3070:~$ javac serv.java
```

```
ise@ise-OptiPlex-3070:~$ java serv
```

Server ready for connection

Connection is successful and wating for chatting

Output: Client side

```
ise@ise-OptiPlex-3070:~$ javac clnt.java
```

```
ise@ise-OptiPlex-3070:~$ java clnt
```

Enter the file name

hi.txt

hello 5B students

Result:-

Thus 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 was implemented successfully

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

Program:**Server Program: udpserver.java**

```
import java.io.*;
import java.net.*;
class udpserver
{
    public static void main(String [] a) throws IOException
    {
        int i=2000;
        String fooString1 = new String("exit");
        while(true)
        {
            InetAddress clientIP=InetAddress.getLocalHost();
            int clientPort=i;
            byte buf[]=new byte[1024];
            DatagramSocket ds=new DatagramSocket();
            BufferedReader dis =new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Server is Running.....");
            System.out.println("Enter the message to client:");
            String str1=new String(dis.readLine());
            if(str1.equals(fooString1))
            {
                ds.close();
                break;
            }
            else
            {
                buf=str1.getBytes();
                DatagramPacket packet=new DatagramPacket(buf,str1.length(),clientIP,clientPort);
                ds.send(packet);
                i=i+1;
            }
            ds.close();
        }
    }
}
```

Client Program: udpclient.java

```
import java.io.*;
import java.net.*;
class udpclient
{
    public static void main(String[] a) throws IOException
    {
        int i=2000;
        while(true)
        {
            byte buf[]=new byte[1024];
            DatagramSocket ds=new DatagramSocket(i);
            DatagramPacket dp=new DatagramPacket(buf,buf.length);
            ds.receive(dp);
            String str2=new String(dp.getData(),0,dp.getLength());
            System.out.println("Server cnnected,\nThe message from server:"+str2);
        }
    }
}
```

```
        System.out.println("*****");
        i=i+1;
    }
}
```

Output: Server side

```
ise@ise-OptiPlex-3070:~$ gedit udpserver.java
ise@ise-OptiPlex-3070:~$ javac udpserver.java
ise@ise-OptiPlex-3070:~$ java udpserver
Server is Running.....
Enter the message to client:
hello 5B
Server is Running.....
Enter the message to client:
exit
```

Output: Client side

```
ise@ise-OptiPlex-3070:~$ gedit udpclient.java
ise@ise-OptiPlex-3070:~$ javac udpclient.java
ise@ise-OptiPlex-3070:~$ java udpclient
Server connected,
The message frn server:hello 5B
*****
```

Result:-

Thus the program on datagram socket for client/server to display the messages on client side, typed at the server side was implemented successfully

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

1. Key Generation Algorithm

1. Generate two large random primes, p and q , of approximately equal size such that their product $n = pq$ is of the required bit length, e.g. 1024 bits.
2. Compute $n = pq$ and (ϕ) $\phi = (p-1)(q-1)$.
3. Choose an integer e , $1 < e < \phi$, such that $\gcd(e, \phi) = 1$. [See note 2].
4. Compute the secret exponent d , $1 < d < \phi$, such that $ed \equiv 1 \pmod{\phi}$. [See note 3].
5. The public key is (n, e) and the private key is (n, d) . The values of p , q , and ϕ should also be kept secret.
 - n is known as the *modulus*.
 - e is known as the *public exponent* or *encryption exponent*.
 - d is known as the *secret exponent* or *decryption exponent*.

2. Encryption

Sender A does the following:-

1. Obtains the recipient B's public key (n, e) .
2. Represents the plaintext message as a positive integer m [see note 4].
3. Computes the ciphertext $c = m^e \bmod n$.
4. Sends the ciphertext c to B.

3. Decryption

Recipient B does the following:-

1. Uses his private key (n, d) to compute $m = c^d \bmod n$.

2. Extracts the plaintext from the integer representative m.

Program: rsa.java

```
import java.util.*;
import java.io.*;
public class rsa
{
    static int gcd(int m,int n)
    {
        while(n!=0)
        {
            int r=m%n; m=n;
            n=r;
        }
        return m;
    }
    public static void main(String args[])
    {
        int p=0,q=0,n=0,e=0,d=0,phi=0;
        int nummes[]=new int[100];
        int encrypted[]=new int[100];
        int decrypted[]=new int[100];
        int i=0,j=0,nofelem=0;
        Scanner sc=new Scanner(System.in);
        String message ;
        System.out.println("Enter the Message to be encrypted.");
        message= sc.nextLine();
        System.out.println("Enter value of p and q");
        p=sc.nextInt();
        q=sc.nextInt();
        n=p*q;
        phi=(p-1)*(q-1);

        for(i=2;i<phi;i++) if(gcd(i,phi)==1) break; e=i;

        for(i=2;i<phi;i++) if((e*i-1)%phi==0)
            break;
        d=i;
        for(i=0;i<message.length();i++)
        {
            char c = message.charAt(i); int a =(int)c;
            nummes[i]=c-96;
        }
        nofelem=message.length(); for(i=0;i<nofelem;i++)
        {
            encrypted[i]=1; for(j=0;j<e;j++)
                encrypted[i] =(encrypted[i]*nummes[i])%n;
        }
        System.out.println("\n Encrypted message\n");
        for(i=0;i<nofelem;i++)
        {
            System.out.print(encrypted[i]);
            System.out.print((char)(encrypted[i]+96));
        }
        for(i=0;i<nofelem;i++)
        {
            decrypted[i]=1; for(j=0;j<d;j++)
```

```
                decrypted[i]=(decrypted[i]*encrypted[i])%n;
            }
            System.out.println("\n Decrypted message ");
            for(i=0;i<nofelem;i++)
                System.out.print((char)(decrypted[i]+96));
            return;
        }
    }
}
```

Output:

```
ise@ise-OptiPlex-3070:~$ javac rsa.java
ise@ise-OptiPlex-3070:~$ java rsa
Enter the Message to be encrypted:
ise
Enter value of p and q
5
7
Encrypted message
4d24x10j
Decrypted message
ise
```

Result:

Thus the program for simple RSA algorithm to encrypt and decrypt the data was implemented successfully.

12 .Write a program for congestion control using leaky bucket algorithm.**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. while((Clock++<5*total_packets)and (out_packets< total_packets))
 - a. if (clock == input_packet)
i.insert into Queue
 - b. if (clock % 5 == 0)
i.Remove packet from Queue
6. End

Program: leakybucket.java

```
import java.io.*;
import java.util.Scanner;
class leakybucket
{
    public static void main(String args[])throws InterruptedException
    {
        int n,in_pkt, out_pkt, size, store=0;
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the number of packets, incoming packet size, bucket size, outgoing rate");
        n=sc.nextInt();
        in_pkt=sc.nextInt();
```

```
        size=sc.nextInt();
        out_pkt=sc.nextInt();
        while (n!=0)
        {
            System.out.println("Incoming packet size is: "+in_pkt);
            if (in_pkt<=(size-store))
            {
                store=store+in_pkt;
                System.out.println("Buffer size is "+store+" out of: "+size);
            }
            else
            {
                System.out.println("Packet loss is: "+(in_pkt-(size-store)));
                store=size;
                System.out.println("Buffer size is "+store+"out of: "+size);
            }
            store=store-out_pkt;
            System.out.println("After outgoing we have "+store+"packets left out of "+size+"to buffer \n");
            n=n-1;
            Thread.sleep(3000);
        }
    }
}
```

Output:

ise@ise-OptiPlex-3070:~\$ gedit leakybucket .java

ise@ise-OptiPlex-3070:~\$ javac leakybucket.java

ise@ise-OptiPlex-3070:~\$ java leakybucket

Enter the number of packets, incoming packet size, bucket size, outgoing rate

3

200

300

100

Incoming packet size is: 200

Buffer size is 200out of: 300

After outgoing we have 100 packets left out of 300 to buffer

Incoming packet size is: 200

Buffer size is 300out of: 300

After outgoing we have 200 packets left out of 300 to buffer

Incoming packet size is: 200

Packet loss is: 100


Buffer size is 300out of: 300


After outgoing we have 200 packets left out of 300 to buffer


Result:-


Thus the program for congestion control using leaky bucket algorithm was implemented successfully.


VIVA QUESTIONS AND ANSWERS

1. An error-detecting code inserted as a field in a block of data to be transmitted is known as
 - A. Frame check sequence 
 - B. Error detecting code
 - C. Checksum
 - D. flow control
 - E. None of the above

2. The communication mode that supports two-way traffic but only one direction at a time is
 - A. simplex
 - B. duplex
 - C. half duplex 
 - D. multiplex
 - E. None of the above

3. Demodulation is a process of
 - A. converting analog to digital signals 
 - B. converting digital to analog signals
 - C. multiplexing various signals into one high speed line signals
 - D. performing data description.

4. Which of the following layer protocols are responsible for user and the application programme support such as passwords, resource sharing, file transfer and network management?
 - A. Layer 7 protocols 
 - B. Layer 6 protocols
 - C. Layer 5 protocols
 - D. Layer 4 protocols
 - E. None of the above

5. Which of the following layer protocols are responsible for user and the application programme support such as passwords, resource sharing, file transfer and network management?
 - A. Layer 7 protocols 
 - B. Layer 6 protocols
 - C. Layer 5 protocols
 - D. Layer 4 protocols
 - E. None of the above

6. Which of the following statement is incorrect?
 - A. The Address Resolution Protocol, ARP, allows a host to find the physical address of a target host on the same physical network, given only the target IP address.
 - B. The sender's IP - to- physical address binding is included in every ARP broadcast; receivers update the IP-to-Physical address binding information in their cache before processing an ARP packet.

C. ARP is a low-level protocol that hides the underlying network physical addressing, permitting us to assign IP-addresses of our choice to every machine.

D. All of the above 

E. None of the above

7. Modulation is the process of

A. converting analog signals to digital signals


B. converting digital signals to analog signals 

C. Multiplexing various signals into high speed line signals

D. performing data encryption.

8. Which of the following technique is used for encapsulation?

A. a technique used in best-effort delivery systems to avoid endlessly looping packets.

B. a technique used by protocols in which a lower level protocol accepts a message from a higher level protocol and places it in the data portion of the low level frame. 

C. One of the pieces that results when an IP gateway divides an IP datagram into smaller pieces for transmission across a network that cannot handle the original datagram size

D. All of the above

E. None of the above

9. Error detection at a data link level is achieved by

A. bit stuffing

B. cyclic redundancy codes 


C. Hamming codes

D. equalization

10. Which address is the loopback address?

A. 0.0.0.1

B. 127.0.0.0

C. 127.0.0.1 

D. 255.255.255.255


E. None of the above

11. Error rate is

A. an error-detecting code based on a summation operation performed on the bits to be checked.


B. a check bit appended to an array of binary digits to make the sum of all the binary digits.

C. a code in which each expression conforms to specific rules of construction, so that if certain errors occur in an expression, the resulting expression will not conform to the rules of construction and thus the presence of the errors is detected.

D. the ratio of the number of data units in error to the total number of data units 

E. None of the above

12. Which of the following technique is used for Time-To-Line (TTL)?


A. a technique used in best-effort delivery system to avoid endlessly looping packets. 

B. a technique used by protocols in which a lower level protocol accepts a message from a higher level protocol and places it in the data portion of the low level frame


C. One of the pieces that results when an IP gateway divides an IP datagram into smaller pieces for transmission across a network that cannot handle the original datagram size.

- D. All of the above
- E. None of the above


13. Which of the following is the address of the router?

- A. The IP address
- B. The TCP address
- C. The subnet mask
- D. The default gateway 
- E. None of the above


14. What is the first octet range for a class C IP address?

- A. 192 - 255
- B. 192 - 223 
- C. 192 - 226
- D. 128 - 191
- E. 1 - 126


15. If the ASCII character G is sent and the character D is received, what type of error is this?

- A. single-bit
- B. multiple-bit
- C. burst 
- D. recoverable


16. Because the configuration information for a DHCP client is received dynamically, you must use which utility to read the current configuration to verify the settings?

- A. PING
- B. TRACERT
- C. ARP
- D. IPCONFIG 
- E. None of the above


17. TCP is:

- A. Operates at the Data Link layer
- B. Connection orientated and unreliable
- C. Connection orientated and reliable 
- D. Connectionless and unreliable

18. The physical layer, in reference to the OSI model, defines


- A. data link procedures that provide for the exchange of data via frames that can be sent and received
- B. the interface between the X.25 network and packet mode device 
- C. the virtual circuit interface to packet-switched service
- D. All of the above
- E. None of the above

19. HDLC (High-level Data Link Control) is


- A. a method of determining which device has access to the transmission medium at any time
- B. a method access control technique for multiple-access transmission media
- C. a very common bit-oriented data link protocol issued by ISO. 

- D. network access standard for connecting stations to a circuit-switched network
- E. None of the above


20. Which of the following device copies electrical signals from one Ethernet to another?

- A. bridge
- B. repeater 
- C. hub
- D. passive hub
- E. None of the above


21. The transport layer protocol is connectionless.

- A. NVT
- B. FTP
- C. TCP
- D. UDP 
- E. None of the above


22. Communication between computers is almost always

- A. serial 
- B. parallel
- C. seriesparallel
- D. direct
- E. None of the above


23. MAC is

- A. a method of determining which device has access to the transmission medium at any time 
- B. a method access control technique or multiple-access transmission media
- C. a very common bit-oriented data link protocol issued to ISO.
- D. network access standard for connecting stations to a circuit-switched network
- E. None of the above


24. Which application allows a user to access and change remote files without actual transfer?

- A. TELNET
- B. NFS 
- C. FTP
- D. DNS
- E. None of the above


25. Parity bit is

- A. an error-detecting code based on a summation operation performed on the bits to be checked.
- B. a check bit appended to an array of binary digits to make the sum of all the binary digits. 
- C. a code in which each expression conforms to specific rules of construction, so that if certain errors occur in an expression, the resulting expression will not conform to the rules of construction and thus the presence of the errors is detected
- D. the ratio of the number of data units in error to the total number of data units
- E. None of the above


26. Which IP address class has few hosts per network?

- A. D
- B. C 
- C. B
- D. A
- E. None of the above


27. Which of the following specifies the network address and host address of the computer?

- A. The IP address 
- B. The TCP address
- C. The subnet mask
- D. The default gateway
- E. None of the above


28. Alex is required to provide information on how many people are using the network at any one time. Which network will enable him to do so?

- A. Server-based 
- B. Token-Ring
- C. Ethernet
- D. Star
- E. Peer-to-peer


29. The main difference between TCP and UDP is

- A. UDP is connection oriented where as TCP is datagram service
- B. TCP is an Internet protocol where as UDP is an ATM protocol
- C. UDP is a datagram where as TCP is a connection oriented service 
- D. All of the above


30. Which of the following is not a transmission medium?

- A. telephone lines
- B. coaxial cable
- C. modem 
- D. microwave systems
- E. satellite systems

31. Which of the following statement is incorrect?


- A. Gateways are assumed to know correct routes; hosts begin with minimal routing information and learn new routes from gateways.
- B. Layered protocols are designed so that layer n at the destination receives exactly the same object sent by layer n at the source.
- C. Application programs as well as all protocol software from the Internet layer upward use only IP addresses; the network interface layer handles physical addresses.
- D. All of the above 
- E. None of the above

32. An error detecting code is which code is the remainder resulting from dividing the bits to be checked by a predetermined binary number, is known as


- A. Cyclic redundancy check 

- B. Checksum
- C. Error detecting code
- D. Error rate
- E. None of the above


33. In OSI model, which of the following layer provides error-free delivery of data?

- A. Data link
- B. Network
- C. Transport 
- D. Session
- E. None of the above


34. Error detection at the data link level is achieved by?

- A. Bit stuffing
- B. Hamming codes
- C. Cyclic Redundancy codes 
- D. Equalization
- E. None of the above


35. The _____ layer is the layer closest to the transmission medium.

- A. transport
- B. network
- C. data link
- D. physical 
- E. None of the above


36. In a _____ topology, if there are n devices in a network, each device has $n - 1$ ports for cables.

- A. ring
- B. bus
- C. star
- D. mesh 
- E. None of the above


37. What is the first octet range for a class A IP address?

- A. 1 - 126 
- B. 192 - 255
- C. 192 - 223
- D. 1 - 127
- E. 128 - 191


38. The 32-bit internet address 10000000 00001010 00000010 00011110 will be written in dotted decimal notation as

- A. 148.20.2.30
- B. 164.100.9.61
- C. 210.20.2.64
- D. 128.10.2.30 
- E. None of the above


39. Error detection at a data link level is achieved by

- A. bit stuffing
- B. cyclic redundancy codes 
- C. Hamming codes
- D. equalization
- E. None of the above


40. End-to-end connectivity is provided from host-to-host in:

- A. Network layer
- B. Session layer
- C. Data link layer
- D. Transport layer 
- E. None of the above


41. CSMA (Carrier Sense Multiple Access) is

- A. a method of determining which device has access to the transmission medium at any time
- B. a method access control technique for multiple-access transmission media. 
- C. a very common bit-oriented data link protocol issued by ISO.
- D. network access standard for connecting stations to a circuit-switched network
- E. None of the above


42. The main difference between synchronous and asynchronous transmission is

- A. the clocking is derived from the data in synchronous transmission 
- B. the clocking is mixed with the data in asynchronous transmission
- C. the pulse height is different.
- D. the bandwidth required is different
- E. None of the above

43. Which of the following uses network address translation?

- A. Routers
- B. Network adapter drivers 
- C. Hubs
- D. Windows 95

44. A communication network which is used by large organizations over regional, national or global area is called

- A. LAN
- B. WAN 
- C. MAN
- D. Intranet
- E. None of the above

45. Which of the following TCP/IP protocol is used for file transfer with minimal capability and minimal overhead?

- A. RARP
- B. FTP
- C. TFTP 
- D. TELNET

E. None of the above

46. Alice is setting up a small network in her home so that she can study for her MCSE exams. She doesn't have a lot of money to spend on hardware, so she wants to use a network topology that requires the least amount of hardware possible. Which topology should she select?

A. Star

B. Ring

C. Token-Ring

D. Ethernet

E. Bus 

47. Which of the following technique is used for allocating capacity on a satellite channel using fixed-assignment FDM?

A. Amplitude modulation

B. Frequency-division multiple access 

C. Frequency modulation

D. Frequency-shift keying

E. None of the above

48. Which of the following allows a simple email service and is responsible for moving messages from one mail server to another?

A. IMAP

B. DHCP

C. SMTP 

D. FTP


E. POP3

49. A modulator converts a _____ signal to a(n) _____ signal.

A. FSK; PSK

B. PSK; FSK

C. analog; digital

D. digital; analog 

E. None of the above

50. Who originally designed TCP/IP?

A. The Department of Defense 

B. Novell

C. IBM

D. Xerox

REFERENCES

1. **Communication Networks: Fundamental Concepts and Key Architectures** - Alberto Leon, Garcia and Indra Widjaja, 3rd Edition, Tata McGraw- Hill, 2004.
2. **Data and Computer Communication**, William Stallings, 8th Edition, Pearson Education, 2007.
3. **Computer Networks: A Systems Approach** - Larry L. Peterson and Bruce S. David, 4th Edition, Elsevier, 2007.
4. **Introduction to Data Communications and Networking** – Wayne Tomasi, Pearson Education, 2005.
5. **Communication Networks – Fundamental Concepts and Key architectures** – Alberto Leon- Garcia and Indra Widjaja:, 2rd Edition, Tata McGraw-Hill, 2004
6. **Computer and Communication Networks** – Nader F. Mir:, Pearson Education, 2007.
7. <https://www.isi.edu/nsnam/ns/>
8. <http://aknetworks.webs.com/e-books>