Sri Venkateshwara College of Engineering

NH 7, Vidyanagar, KIA Road, Bengaluru - 562 157.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Accredited by NBA)

A LABORATORY MANUAL

FOR

COMPUTER NETWORK LABORATORY (18CSL57)

(V SEMESTER)

Prepared By:

1. Prof. Pallavi R

COMPUTER NETWORK LABORATORY

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

Subject Code	18CSL57	IA Marks	40
Number of Lecture Hours/Week	01I + 02P	Exam Marks	60
Total Number of Lecture Hours	40	Exam Hours	03

CREDITS – 02

Course objectives: This course will enable student

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

Description (If any):

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

Lab Experiments:

PART A

1. Implement three nodes point – to – point network wi th 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 progr am 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.

Study Experiment / Project: NIL
Course outcomes: The students should be able to:

- Analyze and Compare various networking protocols.
- Demonstrate the working of different concepts of networking
- Implement, analyze and evaluate networking protocols in NS2 / NS3

Conduction of Practical Examination:

1. All laboratory experiments are to be included for practical examination.
2. Students are allowed to pick one experiment from part A and part B with lot.
3. Strictly follow the instructions as printed on the cover page of answer script
4. Marks distribution: Procedure + Conduction + Viva: 100

   Part A: 8+35+7 =50

   Part B: 8+35+7 =50
5. Change of experiment is allowed only once and marks allotted to the procedure part to be made zero

TABLE OF CONTENTS

PART - A

Simulation Experiments are conducted using
Network Simulator version 2.34 (NS 2.34)

NS2 stands for Network Simulator version 2.

- NS2: Is a discrete event simulator for networking.
- Works at packet level.
- Provides support to simulate protocols such as TCP, UDP, FTP, HTTP and DSR.
- Simulate wired and wireless network.
- Use TCL as its scripting language.

Procedure to execute all Simulation Programs

Step 1 :	Deploy the required number of nodes.
Step 2 :	Create the links among the deployed nodes.
Step 3 :	Attach the specific Agents on to the corresponding nodes.
Step 4 :	Link the source agent with respective destination agent.
Step 5 :	Application should be associated with the source agents.
Step 6 :	Set the time for simulation.
Step 7 :	Save the Simulation program as <filename>.ns
Step 8 :	Run the Simulation using ns command Ex:- $ns filename.ns
Step 9 :	Check for the trace file (<filename>.tr) generated.
Step 10:	Measure the required performance using suitable filters

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

Aim: To understand and design the point-to-point network and to realize that changing the queue size and the bandwidth will have its effect on the flow of data.

```
set ns [new Simulator]

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

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

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


$ns duplex-link $n0 $n2 10Mb 300ms DropTail
$ns duplex-link $n1 $n2 10Mb 300ms DropTail
$ns duplex-link $n2 $n3 1Mb 300ms DropTail

$ns set queue-limit $n0 $n2 10
$ns set queue-limit $n1 $n2 10
$ns set queue-limit $n2 $n3 5

set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
set null3 [new Agent/Null]
$ns attach-agent $n3 $null3
set udp1 [ new Agent/UDP]
$ns attach-agent $n1 $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1

$udp0 set class_ 1
$udp1 set class_ 2

$ns connect $udp0 $null3
$ns connect $udp1 $null3

$cbr1 set packetSize_ 500 mb
```

```
$cbr1 set interval_ 0.005
proc finish { } {
        global ns nf tf
        $ns flush-trace
        #execute nam on the trace file
        exec nam lab1.nam &
        #close the trace file
        close $tf
        close $nf
        exit 0
}

$ns at 0.1 "$cbr0 start"
$ns at 0.1 "$cbr1 start"
$ns at 10.0 "finish"
$ns run
```

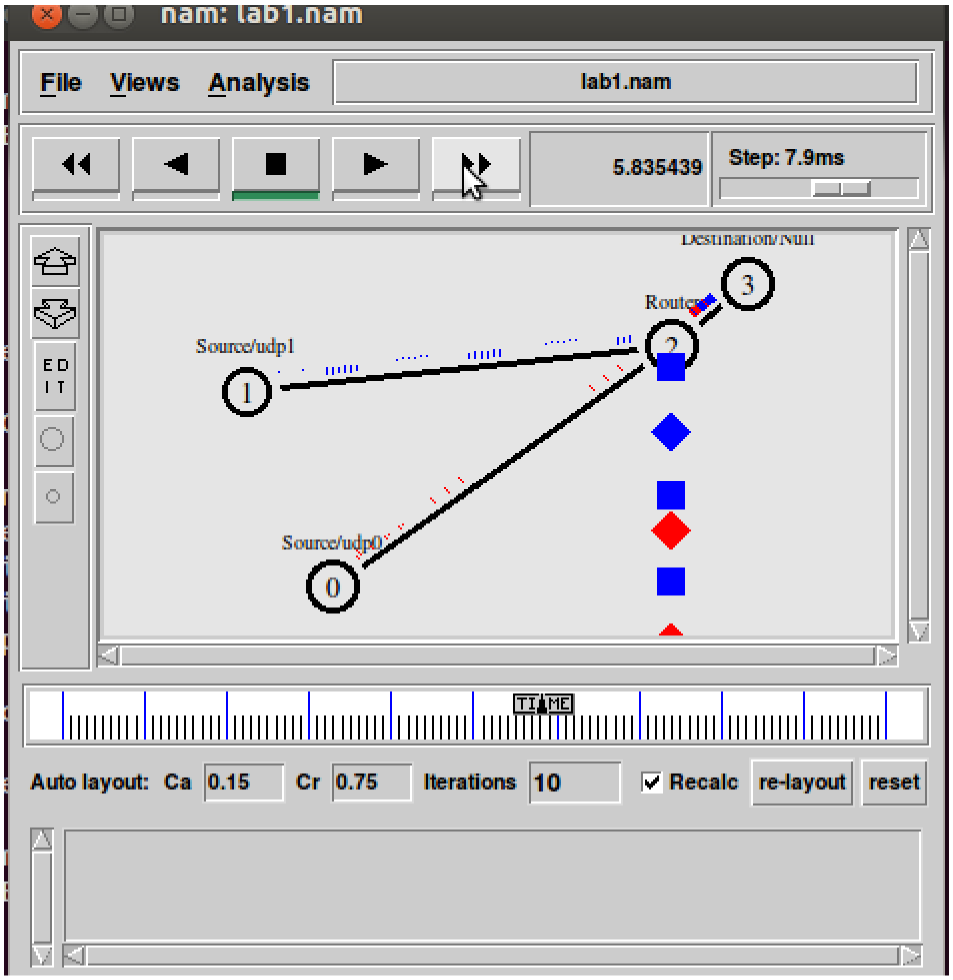
Expected output: Animated 3 node structure is displayed. We need to see the trace file to understand what has happened to the data flow.

AWK Script:

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

Output:

ns lab1.tcl



awk –f lab1.awk lab1.tr
The Total no of packets Dropped due to congestion: 4560

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

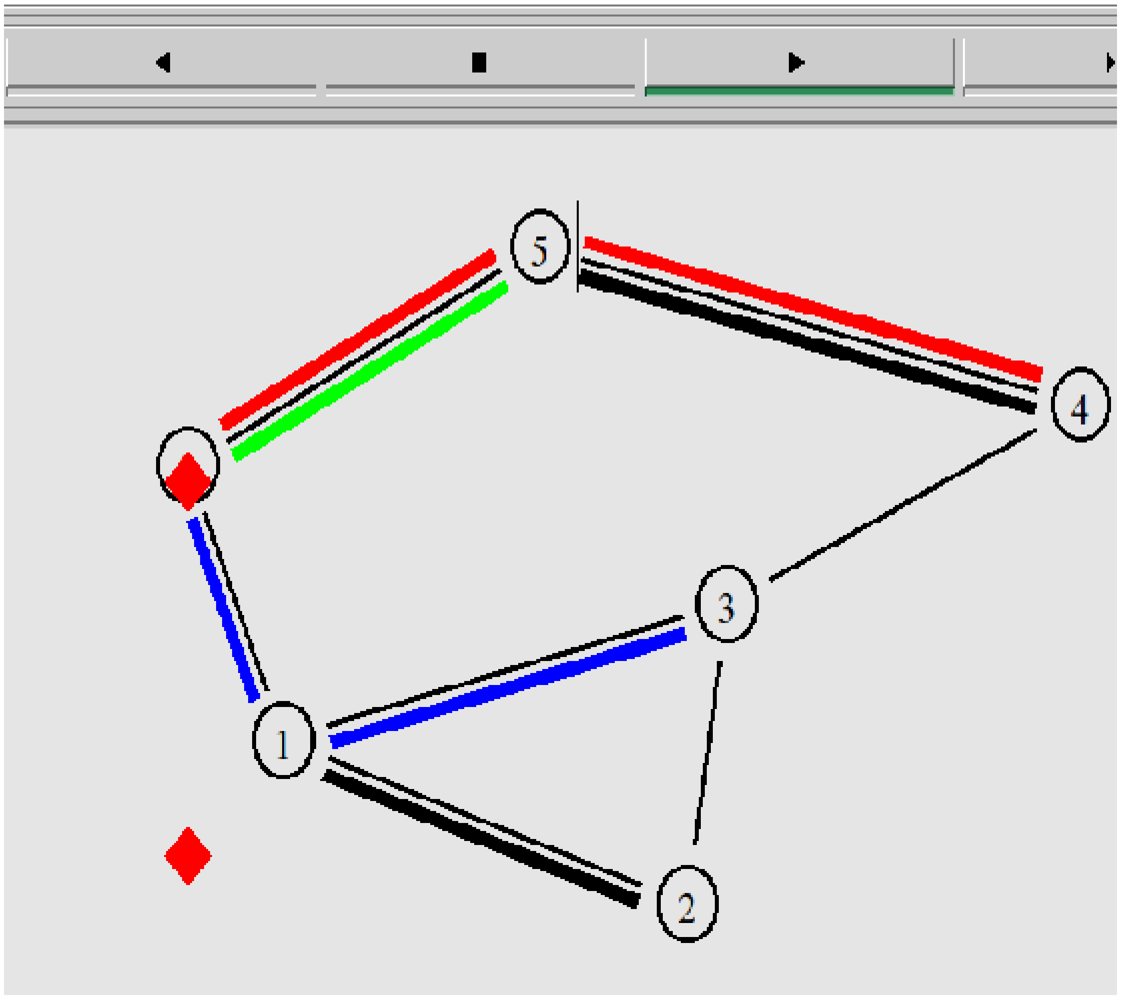
Aim: To understand and design the ping message topology and observe how the packets get dropped due to congestion

```
set ns [new Simulator]

set f [open ex2.tr w]
$ns trace-all $f
set nf [open ex2.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]

$ns duplex-link $n0 $n1 0.01Mb 10ms DropTail
$ns duplex-link $n1 $n2 0.01Mb 10ms DropTail
$ns duplex-link $n2 $n3 0.01Mb 10ms DropTail
$ns duplex-link $n3 $n4 0.01Mb 10ms DropTail
$ns duplex-link $n4 $n5 0.01Mb 10ms DropTail
$ns duplex-link $n0 $n5 0.01Mb 10ms DropTail

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

set p(0) [new Agent/Ping]
$ns attach-agent $n0 $p(0)

set p(1) [new Agent/Ping]
$ns attach-agent $n1 $p(1)

set p(2) [new Agent/Ping]
$ns attach-agent $n2 $p(2)

set p(3) [new Agent/Ping]
$ns attach-agent $n3 $p(3)

set p(4) [new Agent/Ping]
$ns attach-agent $n4 $p(4)

set p(5) [new Agent/Ping]
```

```
$ns attach-agent $n5 $p(5)

$ns connect $p(0) $p(1)
$ns connect $p(1) $p(2)
$ns connect $p(2) $p(3)
$ns connect $p(3) $p(4)
$ns connect $p(4) $p(5)
$ns connect $p(5) $p(0)

for {set i 0} {$i < 50 } {incr i} {
$ns at 0.2 "$p(0) send"
$ns at 0.2 "$p(1) send"
$ns at 0.2 "$p(2) send"
$ns at 0.2 "$p(3) send"
$ns at 0.2 "$p(4) send"
$ns at 0.2 "$p(5) send"
}

proc finish {} {
        global ns f nf
        $ns flush-trace
        #close the trace file
        close $f
        close $nf
        puts "running nam..."
        exec nam ex2.nam &
        exit 0
}

$ns at 1.0 "finish"
$ns run
```

Output:

ns lab2.tcl



```
Node 0 Received Ping answer from 2 with round trip time 122.4ms
Node 0 Received Ping answer from 2 with round trip time 327.2ms
Node 0 Received Ping answer from 2 with round trip time 532.0ms
Node 0 Received Ping answer from 2 with round trip time 736.8ms
```

```
grep “^d” ex2.tr | wc -l
The Total no of packets dropped due to congestion: 14
```

3. Simulate an Ethernet LAN using N nodes and set multiple traffic nodes and plot congestion window for different source/destination.

Aim: To understand and design the LAN network structure. and set different applications for observing how congestion occurs.

```
set ns [new Simulator]

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

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

$ns make-lan "$n0 $n1 $n2 $n3" 10Mb 10ms LL Queue/DropTail Mac/802_3

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3
$ns connect $tcp0 $sink3

set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
set sink1 [new Agent/TCPSink]
$ns attach-agent $n1 $sink1
$ns connect $tcp2 $sink1

set file1 [open file1.tr w]
$tcp0 attach $file1
$tcp0 trace cwnd_ $tcp0
set maxcwnd_ 10

set file2 [open file2.tr w]
$tcp2 attach $file2
$tcp2 trace cwnd_ $tcp2
set maxcwnd_ 10

proc finish { } {
        global nf tf ns
        $ns flush-trace
        exec nam lab3.nam &
        close $nf
        close $tf
        exit 0
}

$ns at 0.1 "$ftp0 start"
$ns at 1.5 "$ftp0 stop"
$ns at 2 "$ftp0 start"
$ns at 3 "$ftp0 stop"
$ns at 0.2 "$ftp2 start"
$ns at 2 "$ftp2 stop"
$ns at 2.5 "$ftp2 start"
$ns at 4 "$ftp2 stop"
$ns at 5.0 "finish"

$ns run
```

Note: The throughput can analyzed by changing the data rate and error rate as shown below.

I. Now Fix the error rate to 0.1 and vary the data rate then throughput increases as shown below.

Error rate	Data rate	Throughput
0.1	0.1	78Mbps
0.2	0.01	89Mbps
0.1	0.001	100Mbps
0.1	0.0001	148Mbps

AWK Script:

```
BEGIN{

#include <stdio.h>

}

{

if($6=="cwnd_")
```

printf("%f\t%f\n",$1,$7);

}

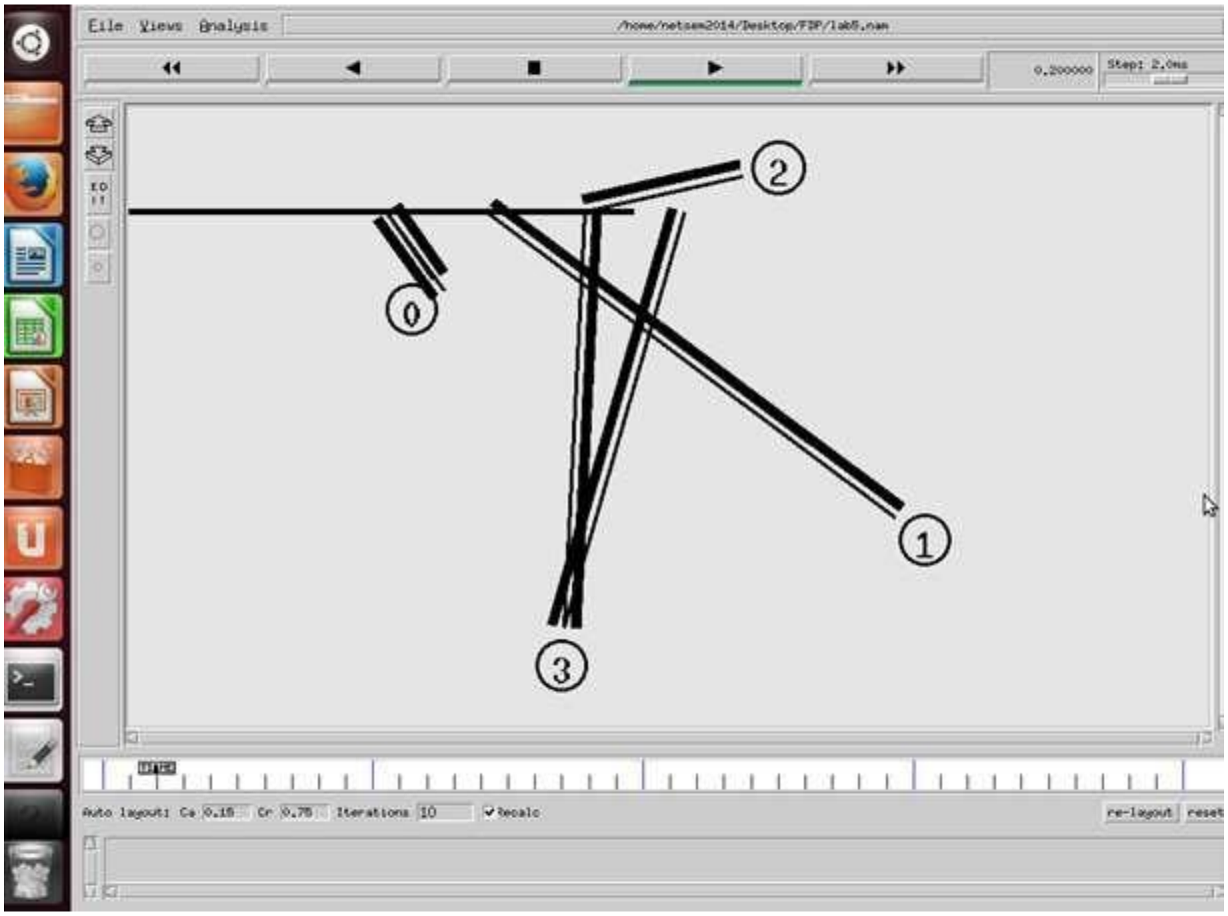
END{

puts "DONE";

}

Output:

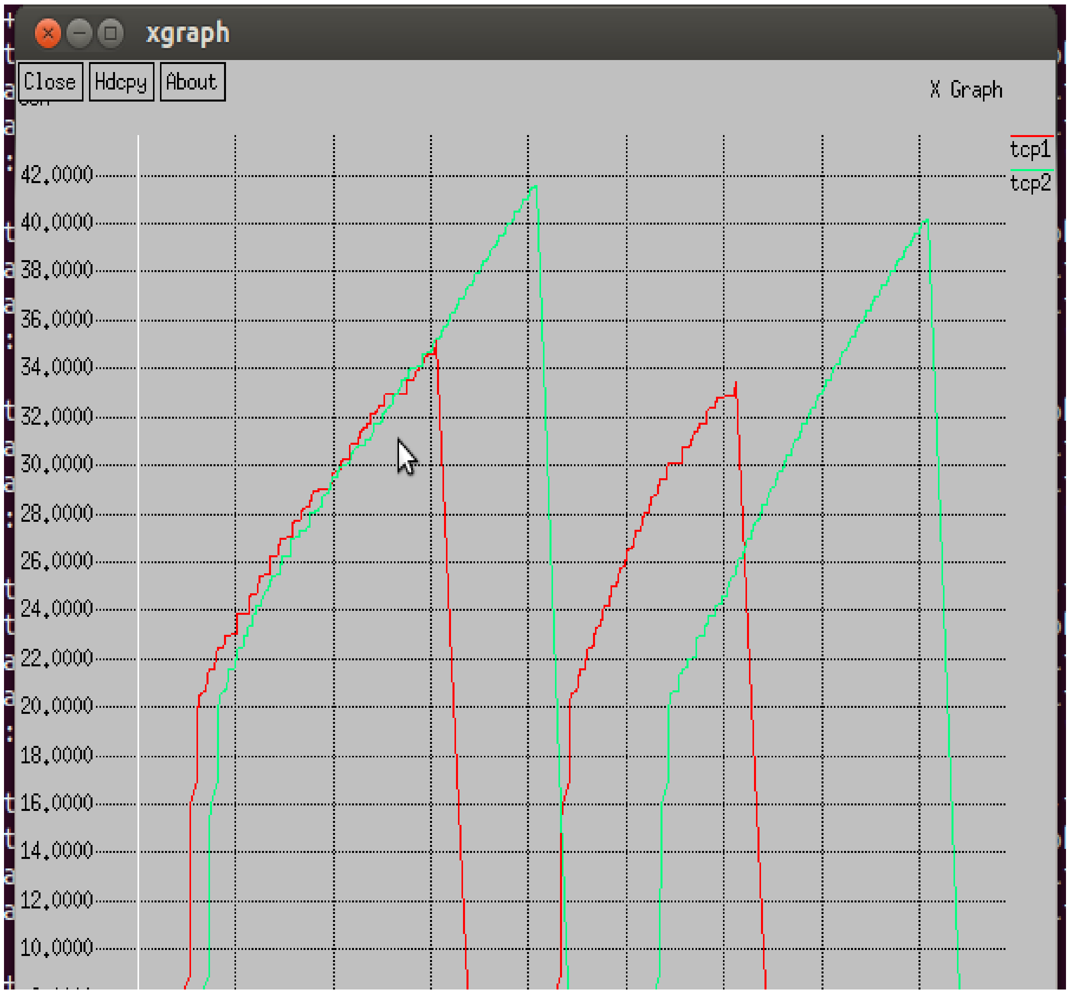
ns lab3.tcl


File
Views
Analysis
/home/netsem2014/Desktop/FIP/lab5.nam
0.200000
Step: 2.0ms
2
0
1
3
Auto layout: Ca 0.15 Cr 0.75 Iterations 10
Recalc
re-layout
reset

awk –f lab3.awk file1.tr>tcp1

awk –f lab3.awk file2.tr>tcp2

xgraph –x time –y congestion tcp1 tcp2



4. Simulate simple ESS and with transmitting nodes in wireless LAN by simulation and determine the performance with respect to transmission of packets.

Aim: To understand and design the wireless LAN network structure and the default setting required to simulate the wireless LAN.

```
set val(chan)           Channel/WirelessChannel      ;
set val(prop)           Propagation/TwoRayGround;
set val(netif)          Phy/WirelessPhy        ;
set val(mac)            Mac/802_11;
set val(ifq)            Queue/DropTail/PriQueue      ;
set val(ll)             LL;
set val(ant)            Antenna/OmniAntenna;
set val(ifqlen)         50;

set ns [new Simulator]

set tf [open lab4.tr w]
$ns trace-all $tf
set nf [open lab4.nam w]
$ns namtrace-all-wireless $nf 100 100


$ns node-config             -llType $val(ll) \
                                 -macType $val(mac) \
                                 -ifqType $val(ifq) \
                                  -ifqLen $val(ifqlen) \
                                  -antType $val(ant) \
                                  -propType $val(prop) \
                                  -phyType $val(netif) \
                                  -channelType $val(chan) \

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 lan [$ns newLan "$n0 $n1 $n2 $n3 $n4 $n5" 0.5Mb 30ms LL Queue/DropTail Mac/802_11
Channel/WirelessChannel]


set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0

set ftp0 [new Application/FTP]
```

```
$ftp0 attach-agent $tcp0

set sink0 [new Agent/TCPSink]
$ns attach-agent $n5 $sink0

$ns connect $tcp0 $sink0

$ns at 5.000000 "$ftp0 start"

proc finish { }             {
global ns tf nf
$ns flush-trace
close $tf
close $nf
exec nam lab4.nam &
exit 0
}
$ns at 6.0000 "finish"
$ns run
```

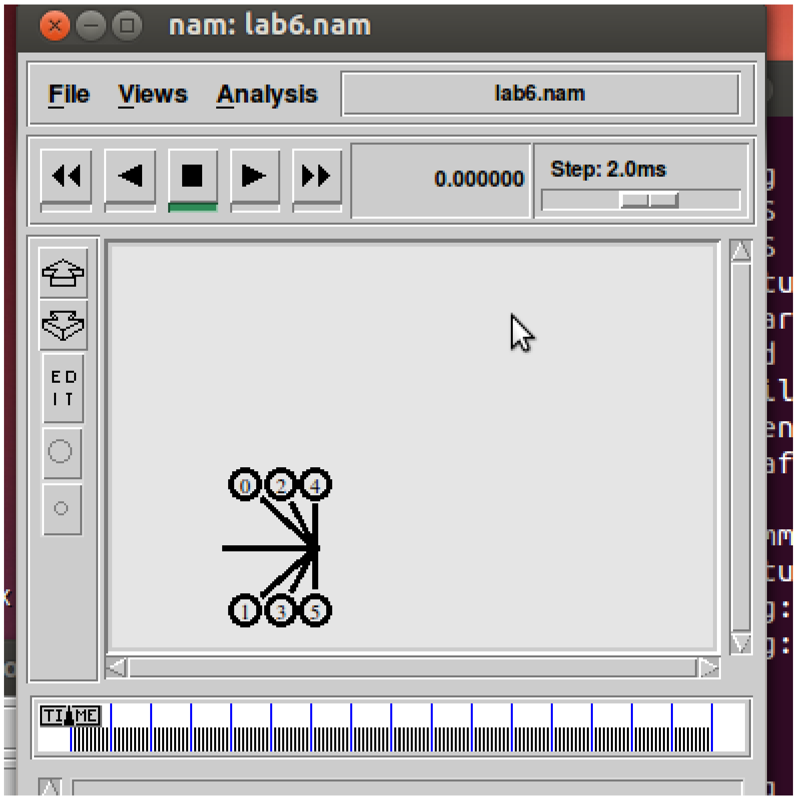
Expected output: Animated 6 node structure is displayed. We need to see the trace file to understand what has happened to the data flow and packet dropped.

AWK Script:

```
BEGIN{
        #include<stdio.h>
pkt=0;
time=0;
}
{
  if($1="r" && $3==”0” && $4==”6”)
  {
    pkt=pkt+$6;
    time=$2;
  }
}
END{
        printf(“throughput=%dMbps\n\n”,(pkt/time)*(8/100000));
}
```

Output:

ns lab4.tcl



```
awk –f lab4.awk lab4.tr
Throughput: 3.783302
```

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

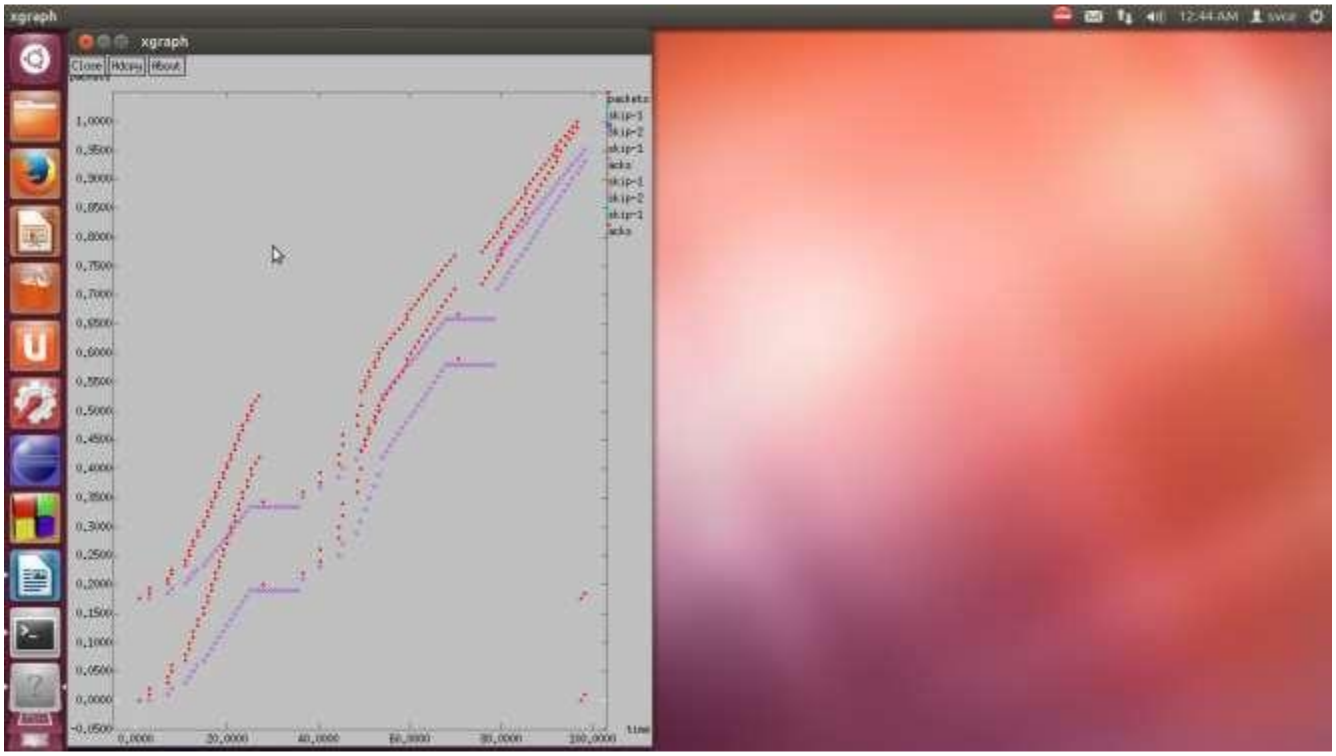
```
set opt(title) zero ;

set opt(stop) 100     ;

set opt(ecn) 0        ;

set opt(type) gsm    ;

set opt(secondDelay) 55 ;

set opt(minth) 30 ;

set opt(maxth) 0 ;

set opt(adaptive) 1 ;

set opt(flows) 0 ;

set opt(window) 30 ;

set opt(web) 2  ;

set opt(quiet) 0  ;

set opt(wrap) 100 ;

set opt(srcTrace) is ;

set opt(dstTrace) bs2 ;

set opt(gsmbuf) 10 ;

set bwDL(gsm) 9600

set bwUL(gsm) 9600

set propDL(gsm) .500

set propUL(gsm) .500

set buf(gsm) 10

set ns [new Simulator]

set tf [open 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)

}

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


switch $opt(type) {

gsm -

gsm -

umts {cell_topo}

}

set_link_params $opt(type)


$ns insert-delayer $nodes(ms) $nodes(bs1) [new Delayer]

$ns insert-delayer $nodes(bs1) $nodes(ms) [new Delayer]
```

```
$ns insert-delayer $nodes(ms) $nodes(bs2) [new Delayer]

$ns insert-delayer $nodes(bs2) $nodes(ms) [new Delayer]

if {$opt(flows) == 0} {

set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]

set ftp1 [[set tcp1] attach-app FTP]

$ns at 0.8 "[set ftp1] start"

}

if {$opt(flows) > 0} {

set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]

set ftp1 [[set tcp1] attach-app FTP]

$tcp1 set window_ 100

$ns at 0.0 "[set ftp1] start" $ns at 3.5 "[set ftp1] stop"

set tcp2 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]

set ftp2 [[set tcp2] attach-app FTP]

$tcp2 set window_ 3

$ns at 1.0 "[set ftp2] start"

$ns at 8.0 "[set ftp2] stop"

}

proc stop { } {

global nodes opt nf

set wrap $opt(wrap)

set sid [$nodes($opt(srcTrace)) id]

set did [$nodes($opt(dstTrace)) id]

if {$opt(srcTrace) == "is"} {
```

```
set a "-a out.tr"

} else {


set a "out.tr"

}

set GETRC "ns-allinone-2.34/ns-2.34/bin/getrc"

set RAW2XG "ns-allinone-2.34/ns-2.34/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

if {!$opt(quiet)} {

exec xgraph -bb -tk -nl -m -x time -y packets plot.xgr &

}

exit 0

}

$ns at $opt(stop) "stop"

$ns run
```

OUTPUT:


xgraph
Close
Hdcpy
About
packets
skip-1
skip-2
skip-1
acks
skip-1
skip-2
skip-1
acks
time

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

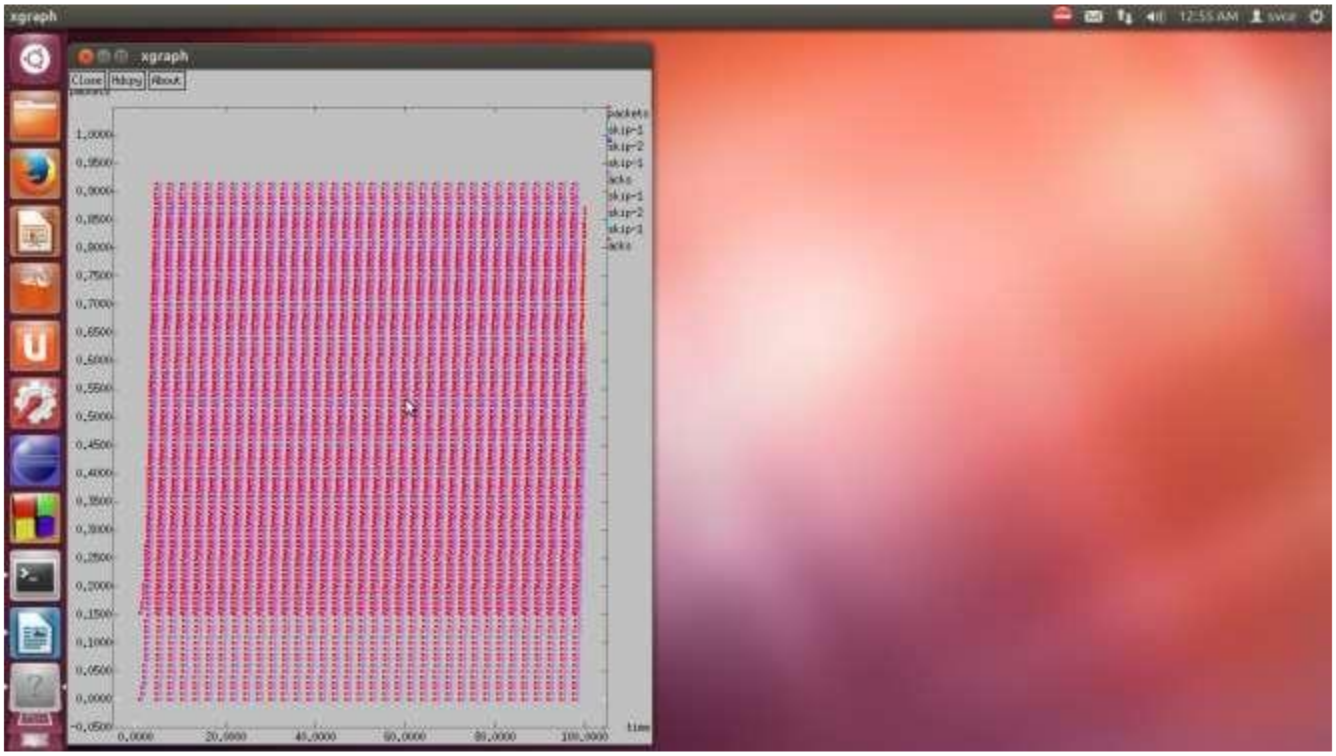
```
set opt(title) zero;

set opt(stop) 100;

set opt(ecn) 0;

set opt(type) umts ;

set opt(bwUL) 0;

set opt(bwDL) 0;

set opt(propUL) 0 ;

set opt(propDL) 0 ;

set opt(secondDelay) 55;

set opt(maxth) 0;

set opt(queue) DT;

set opt(qsize) 0;

set opt(flows) 1;

set opt(shortflows) 0;

set opt(webers) 0;

set opt(window) 30;

set opt(smallpkt) 10;

set opt(web) 2;

set opt(pagesize) 10;

set opt(objSize) 60;

set opt(shape) 1.05;

set opt(interpage) 1;
set opt(quiet) 0;

set opt(wrap) 90;

set opt(srcTrace) is;

set opt(dstTrace) bs;

set opt(umtsbuf) 20;

set opt(tfrcFB) 1;

set bwDL(umts) 384000;

set bwUL(umts) 64000;

set propDL(umts) .150;


set propUL(umts) .150;

set buf(umts) 20;


proc cell_topo {} {

global ns nodes qm

$ns duplex-link $nodes(bs) $nodes(ms) 1 1 $qm

$ns duplex-link $nodes(lp) $nodes(ms) 3Mbps 10ms DropTail

$ns duplex-link $nodes(bs) $nodes(is) 3Mbps 50ms DropTail

puts "cell topology"

}

proc set_link_params {t} {

global ns nodes bwUL bwDL propUL propDL buf

$ns bandwidth $nodes(bs) $nodes(ms) $bwDL($t) simplex

$ns bandwidth $nodes(ms) $nodes(bs) $bwUL($t) simplex

$ns delay $nodes(bs) $nodes(ms) $propDL($t) simplex
```

```
$ns delay $nodes(ms) $nodes(bs) $propDL($t) simplex

$ns queue-limit $nodes(bs) $nodes(ms) $buf($t)

$ns queue-limit $nodes(ms) $nodes(bs) $buf($t)

}

if {$opt(queue) == "DT"} {

set qm DropTail

}

if {$opt(queue) == "RED"} {

set qm RED

}

set tcpTick_ 0.01

set pktsize 1460


Agent/TCP set tcpTick_ $tcpTick_

set out $opt(title)

proc stop {} {

global nodes opt

set wrap $opt(wrap)

set sid [$nodes($opt(srcTrace)) id]

set did [$nodes($opt(dstTrace)) id]

if {$opt(srcTrace) == "bs"} {

set a "-a umts.tr"

} else {

set a "umts.tr"
}


set GETRC "ns-allinone-2.34/ns-2.34/bin/getrc"
set RAW2XG "ns-allinone-2.34/ns-2.34/bin/raw2xg"
exec $GETRC -s $sid -d $did -f 0 umts.tr | \
$RAW2XG -s 0.01 -m $wrap -r > plot.xgr
exec $GETRC -s $did -d $sid -f 0 umts.tr | \
$RAW2XG -a -s 0.01 -m $wrap >> plot.xgr
exec $GETRC -s $sid -d $did -f 1 umts.tr | \
$RAW2XG -s 0.01 -m $wrap -r >> plot.xgr
if {!$opt(quiet)} {
exec xgraph -bb -tk -nl -m -x time -y packets plot.xgr &
}
exit 0
}


set ns [new Simulator]
variable delayerUL
variable delayerDL


Queue/RED set summarystats_ true
Queue/DropTail set summarystats_ true
Queue/RED set maxthresh_ $opt(maxth)
Queue/DropTail set shrink_drops_ true
Agent/TCP set ecn_ $opt(ecn)
```

```
Agent/TCP set packetSize_ $pktsize


Agent/TCP set window_ $opt(window)

DelayLink set avoidReordering_ true


set tf [open umts.tr w]

$ns trace-all $tf

set nodes(lp) [$ns node]

set nodes(ms) [$ns node]

set nodes(bs) [$ns node]

set nodes(is) [$ns node]

switch $opt(type) {

umts {cell_topo}

}

set_link_params $opt(type)

set delayerDL [new Delayer]

set delayerUL [new Delayer]

$ns insert-delayer $nodes(ms) $nodes(bs) $delayerUL

$ns insert-delayer $nodes(bs) $nodes(ms) $delayerDL

if {$opt(flows) > 0} {

set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]

set ftp1 [[set tcp1] attach-app FTP]

$ns at 0.8 "[set ftp1] start"

}

if {$opt(shortflows) > 0} {
```

```
set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]

set ftp1 [[set tcp1] attach-app FTP]

$tcp1 set window_ 100

$ns at 0.0 "[set ftp1] start"

$ns at 3.5 "[set ftp1] stop"

set tcp2 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]

set ftp2 [[set tcp2] attach-app FTP]

$tcp2 set window_ 3

$ns at 1.0 "[set ftp2] start"

$ns at 8.0 "[set ftp2] stop"

}


set req_trace_ 0

set count $opt(webers)

if ($count) {

add_web_traffic $opt(secondDelay) $opt(web) $opt(interpage) $opt(pagesize)

$opt(objSize) $opt(shape) 1

add_web_traffic $opt(secondDelay) [expr $opt(web)/2] $opt(interpage) $opt(pagesize)

$opt(objSize) $opt(shape) 0

}

$ns at $opt(stop) "stop"

$ns run
```

Output:


xgraph
Close
Hdcpy
About
packets
skip-1
skip-2
skip-1
acks
skip-1
skip-2
skip-1
acks
time

Extra Programs:

1. Simulate simple BSS and with transmitting nodes in wireless LAN by simulation and determine the performance with respect to transmission of packets.

```
set ns [new Simulator]

set val(chan) Channel/WirelessChannel;# channel type

set val(prop) Propagation/TwoRayGround;# radio-propagation model

set val(netif) Phy/WirelessPhy;# network interface type

set val(mac) Mac/802_11;# MAC type

set val(ifq) Queue/DropTail/PriQueue;# interface queue type

set val(ll) LL;# link layer type

set val(ant) Antenna/OmniAntenna;# antenna model

set val(ifqlen) 50;# max packet in ifq

set val(nn) 2;# number of mobilenodes

set val(rp) DSDV;# routing protocol

set val(x) 1000.0;

set val(y) 1000.0;

set tf [open lab6.tr w]

$ns trace-all $tf

set topo [new Topography]

$topo load_flatgrid 1000 1000

set nf [open lab6.nam w]

$ns namtrace-all-wireless $nf 1000 1000

$ns node-config -adhocRouting $val(rp) \
-llType $val(ll) \

-macType $val(mac) \


-ifqType $val(ifq) \

-ifqLen $val(ifqlen) \

-antType $val(ant) \

-propType $val(prop) \

-phyType $val(netif) \

-channelType $val(chan) \

-topoInstance $topo \

-agentTrace ON \

-routerTrace ON \

-macTrace 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"#The below code is used to give the initial node positions.

$n0 set X_ 250

$n0 set Y_ 250

$n0 set Z_ 0

$n1 set X_ 300

$n1 set Y_ 300
```

```
$n1 set Z_ 0

$n2 set X_ 600

$n2 set Y_ 600

$n2 set Z_ 0

$ns at 0.1 "$n0 set dest 250 250 15"

$ns at 0.1 "$n1 set dest 300 300 25"

$ns at 0.1 "$n2 set dest 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"

#The below code is used to provide the node movements.

$ns at 100 "$n1 setdest 550 550 15"
```

```
$ns at 190 "$n1 setdest 70 70 15"

proc finish {} { global ns nf tf


$ns flush-trace

exec nam lab6.nam &

close $tf

exit 0}

$ns at 250 "finish"

$ns run
```

Expected output: Animated 6 node structure is displayed. We need to see the trace file to understand what has happened to the data flow and packet dropped.

AWK Script:

```
 BEGIN{

#include<stdio.h>

}

{

if($6=="cwnd_")

printf("%f \t %f \n", $1,$7);

}

END{

puts "DONE";

}
```

Output:

```
ns lab7.tcl

awk –f lab7.awk lab7.tcl

Throughput: 3.783302
```

2. Simulate an Ethernet LAN using N nodes (6-10), change error rate and data rate and compare throughput.

```
 set ns [new Simulator]
 set f [open ex5.tr w]
 $ns trace-all $f
set nf [open ex5.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 lan [$ns newLan "$n0 $n1 $n2 $n3 $n4 $n5" 0.5Mb 10ms LL Queue/DropTail Channel]

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
```

```
set sink0 [new Agent/TCPSink]

$ns attach-agent $n5 $sink0


$ns connect $tcp0 $sink0


$ns at 5.000000 "$ftp0 start"


proc finish {} {

global ns f nf

$ns flush-trace

close $f

close $nf

puts "running nam..."

exec nam ex5.nam &

exit 0

}

 $ns at 60.0000 "finish"

 $ns run
```

Expected output:

Animated 6 node structure is displayed. We need to see the trace file to understand what has happened to the data flow.

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.

Awk Script:

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

Output:

```
 ns lab4.tcl
awk –f lab4.awk lab4.tr
Throughput: 148Mbps
```

PART - B

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

Program-Name : crc.java

```
import java.io.*;
class crc
{
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];
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++)
{
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]);

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");

}

}

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:

```
Enter number 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

Dividend (after appending 0's) are : 101100100


CRC code :

101100111

Enter CRC code of 9 bits :

1 0 1 1 0 0 1 1 1

No Error
```

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

```
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:

```
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 destination vertex

2
Distance of Source2 to 1 is 5

Distance of Source2 to 2 is 0

Distance of Source2 to 3 is 3

Distance of Source2 to 4 is 4
```

9. Using TCP/IP sockets, write a client-server program to make client sending the file name and make the server to send back the contents of the requested file if present.
Implement the above program using as message queues or FIFOs as IPC channels.

Client side Program

```
import java.net.*;

import java.io.*;

public class ContentClient

{

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

  {

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

     System.out.print("Enter the file name");

     BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));

     String fname = keyRead.readLine();

     OutputStream ostream = sock.getOutputStream( );

     PrintWriter pwrite = new PrintWriter(ostream, true);

     pwrite.println(fname);

     InputStream istream = sock.getInputStream();

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

     String str;

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

     {

         System.out.println(str);

     }

     pwrite.close(); socketRead.close(); keyRead.close();
  }
}
```

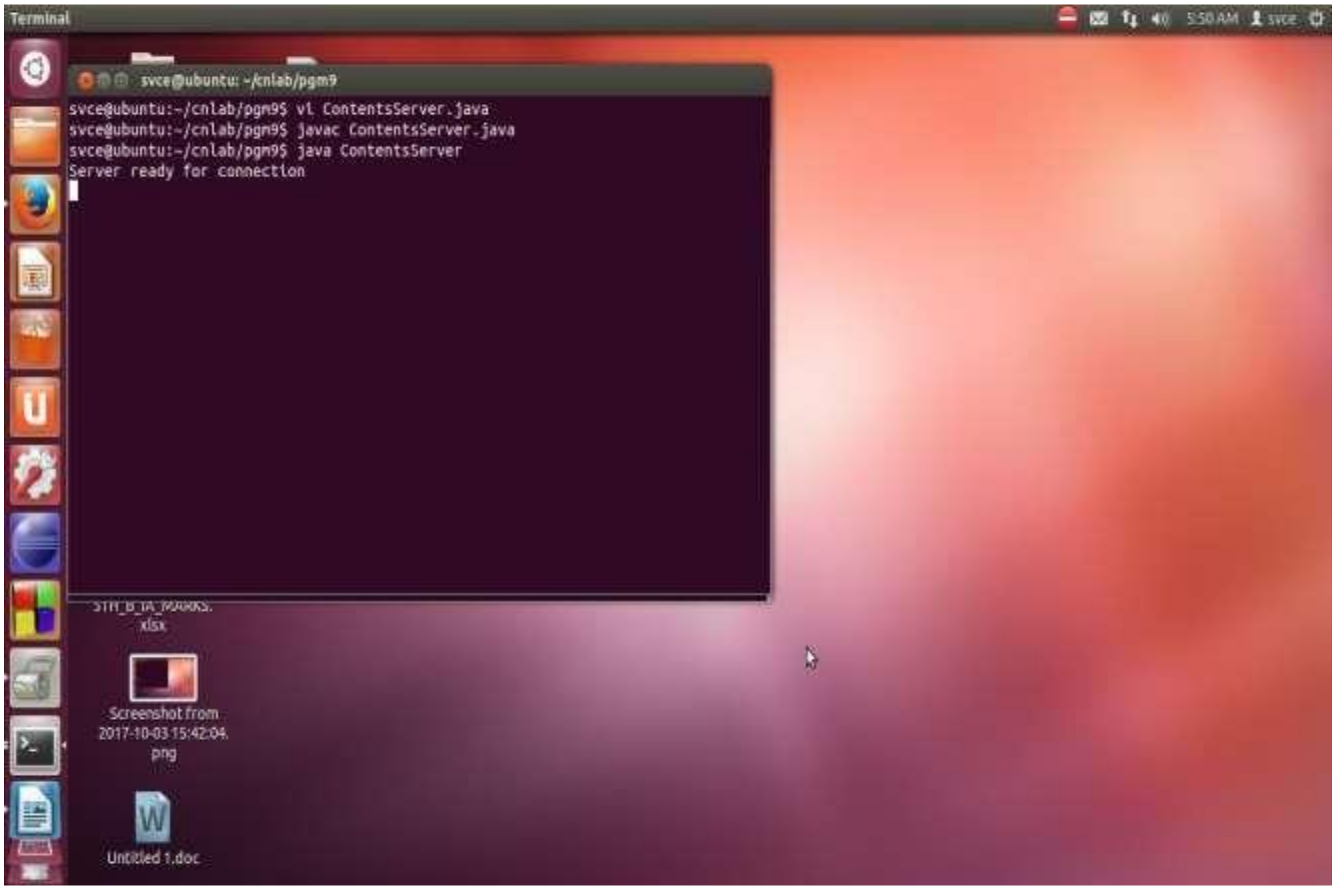
Server side program

```
import java.net.*;
import java.io.*;
public class ContentsServer
{
  public static void main(String args[]) throws Exception
  {
     ServerSocket sersock = new ServerSocket(4000);
     System.out.println("Server ready for connection");
     Socket sock = sersock.accept();
     System.out.println("Connection is successful and wating for chatting");
     InputStream istream = sock.getInputStream( );
     BufferedReader fileRead =new BufferedReader(new InputStreamReader(istream));
     String fname = fileRead.readLine( );
     BufferedReader contentRead = new BufferedReader(new FileReader(fname) );
     OutputStream ostream = sock.getOutputStream( );
     PrintWriter pwrite = new PrintWriter(ostream, true);
     String str;
     while((str = contentRead.readLine()) != null)
     {
pwrite.println(str);
}
     sock.close();  sersock.close();
```

```
    pwrite.close(); fileRead.close(); contentRead.close();
  }
}
```

Output:

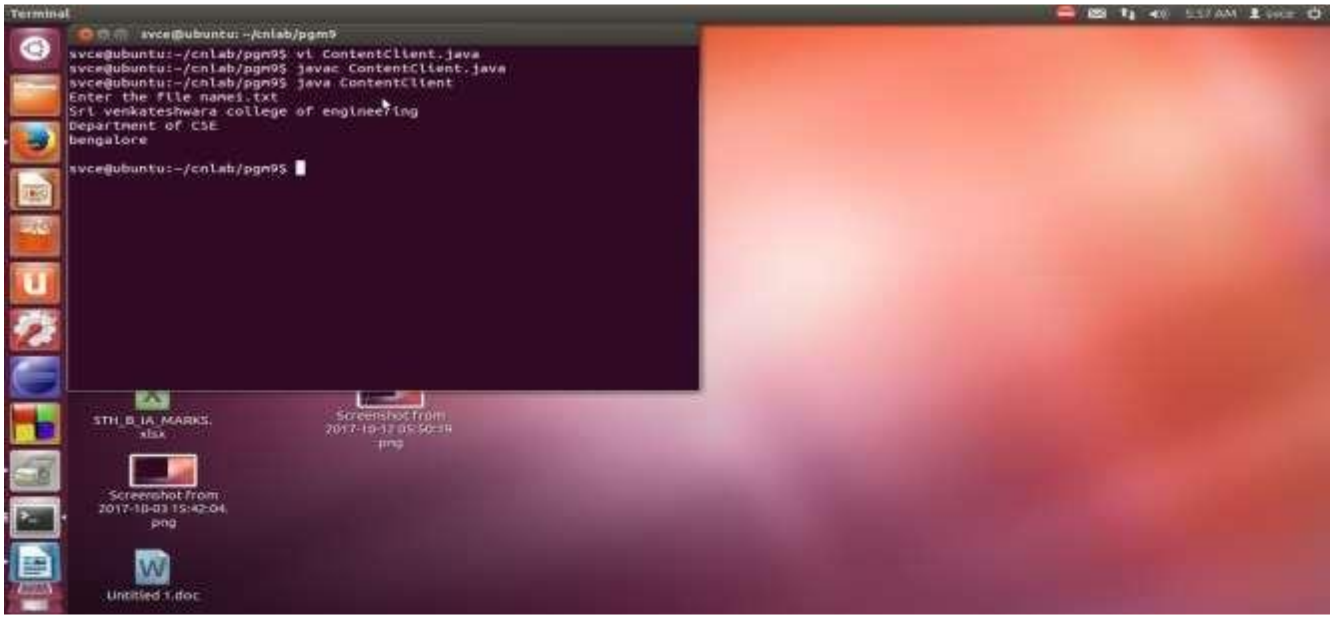
Then Run the server side Terminal



```
svce@ubuntu:~/cnlab/pgm9$ vi ContentsServer.java
svce@ubuntu:~/cnlab/pgm9$ javac ContentsServer.java
svce@ubuntu:~/cnlab/pgm9$ java ContentsServer
Server ready for connection
```

Terminal
svce@ubuntu: ~/cnlab/pgm9
5:50 AM
svce
Screenshot from 2017-10-03 15:42:04.png
Untitled 1.doc

To Run the Client side terminal



```
svce@ubuntu:~/cnlab/pgm9$ vi ContentClient.java
svce@ubuntu:~/cnlab/pgm9$ javac ContentClient.java
svce@ubuntu:~/cnlab/pgm9$ java ContentClient
Enter the file name1.txt
Sri venkateshwara college of engineering
Department of CSE
bengalore

svce@ubuntu:~/cnlab/pgm9$ 
```

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

UDP client

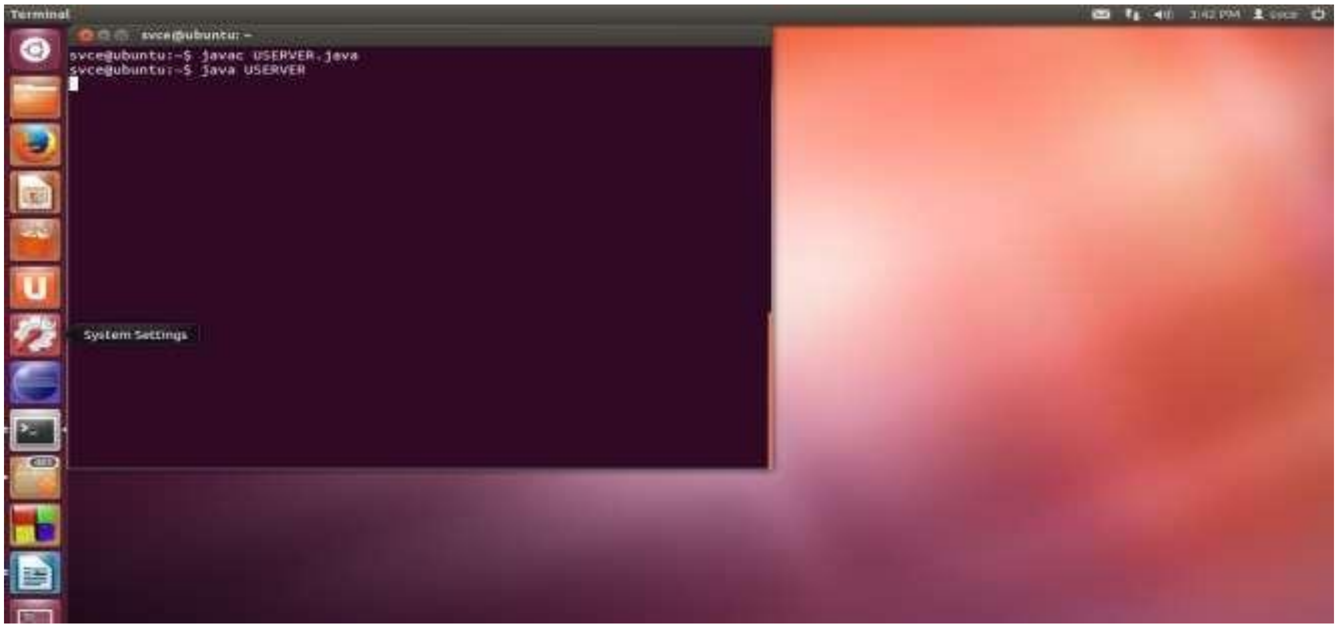
```
import java.io.*;

import java.net.*;

public class UDPC

{

 public static void main(String[] args)

  {

   DatagramSocket skt;

   try

      {

        skt=new DatagramSocket();

        String msg="text message";

        byte[] b=msg.getBytes();

        InetAddress host=InetAddress.getByName("127.0.0.1");

        int serverSocket=6788;

        DatagramPacket request=new DatagramPacket(b,b.length,host,serverSocket);skt.send(request);

        byte[] buffer=new byte[1000];

        DatagramPacket reply=new DatagramPacket(buffer,buffer.length);skt.receive(reply);

        System.out.println("client received:"+new String(reply.getData()));skt.close();

      }

    catch(Exception ex)

         { }

       } }
```

UDP Server

```
import java.io.*;

import java.net.*;

public class USERVER

{

 public static void main(String[] args)

  {

   DatagramSocket skt=null;

   try

     {

       skt=new DatagramSocket(6788);

       byte[] buffer=new byte[1000];

       while(true)

        {

         DatagramPacket request=new DatagramPacket(buffer,buffer.length);

         skt.receive(request);

         String[] message=(new String(request.getData())).split(" ");

         byte[] sendMsg=(message[1]+"svce").getBytes();

         DatagramPacket reply=new
DatagramPacket(sendMsg,sendMsg.length,request.getAddress(),request.getPort());

         skt.send(reply);

        }

    }

  catch(Exception ex)

    { } { }
```

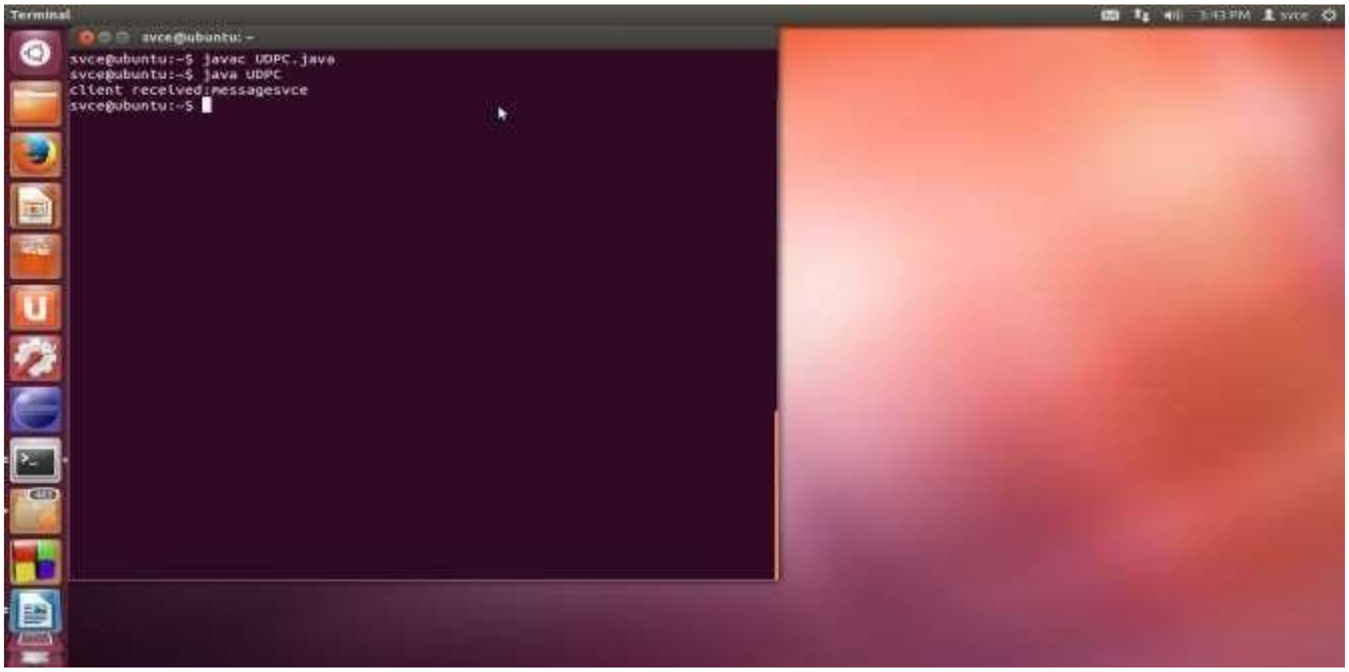
Output:

To run the server side terminal



```
svce@ubuntu:~$ javac USERVER.java
svce@ubuntu:~$ java USERVER
```

To run the client side terminal



```
svce@ubuntu:~$ javac UDPC.java
svce@ubuntu:~$ java UDPC
client received:messagesvce
svce@ubuntu:~$
```

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

```
import java.io.DataInputStream;

import java.io.IOException;

import java.math.BigInteger;

import java.util.Random;

public class RSA

{

    private BigInteger p;

    private BigInteger q;

    private BigInteger N;

    private BigInteger phi;

    private BigInteger e;

    private BigInteger d;

    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;
    }
    @SuppressWarnings("deprecation")
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: svce


Encrypting String: svce

String in Bytes: 1069711897

Decrypting Bytes: 1069711897

Decrypted String: svce
```

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

```
import java.util.Scanner;
public class leakybucket
{
 static int front=-1;
 static int rear=-1;
 static int max=10;
static int[]q=new int[10];

public static void delete()
  {
    if(front==rear)
    front=rear=-1;
    else
     front=(front+1)%max;
  }

public static void insert(int i)
 {
  if(front==-1)
  front=rear=0;
  else
  rear=(rear+1)%max;
  q[rear]=i;
 }

public static void main(String[] args)
 {
  int t=0,i=1,j=1,cap,n,a,q=1;
  int b=0;
  System.out.println("enter the bucket capacity");
  Scanner S=new Scanner(System.in);
  cap=S.nextInt();
  System.out.println("enter the rate at which packet must be sent:");
  b=S.nextInt();
  System.out.println("enter the number of count for which output should be shown");
  n=S.nextInt();
  for(a=0;a<n;a++)
  {
   t=(++t)%b;
   System.out.print("\t t:"+t);

   if(t==0)
    {
     delete();
System.out.print(j+"delivered\n");
     j++;
    }
 else {
       if((q-j)>=cap)
        {
         System.out.print(i+"discarded\n");
         i++;
        }
     else
       {
         System.out.print(i+"queue\n");
         insert(i);
         i++;
         q++;
        }
      }
    }
  }
}
```

Output:

```
Enter the bucket capacity

5

enter the rate at which packet must be sent:

2

enter the number of count for which output should be shown

5


 t:1 1 queue

 t:0 1 delivered

 t:1 2 queue

 t:0 2 delivered

 t:1 3 queue
```

Extra Programs:

1. Java program for Dijkstra's single source shortest path algorithm

```
// The program is for adjacency matrix representation of the graph

import java.util.*;

import java.lang.*;

import java.io.*;


class ShortestPath {

    // A utility function to find the vertex with minimum distance value,

    // from the set of vertices not yet included in shortest path tree

    static final int V = 9;

    int minDistance(int dist[], Boolean sptSet[])

    {

        // Initialize min value

        int min = Integer.MAX_VALUE, min_index = -1;


        for (int v = 0; v < V; v++)

            if (sptSet[v] == false && dist[v] <= min) {

                min = dist[v];

                min_index = v;

            }


        return min_index;
```

```
}

// A utility function to print the constructed distance array
void printSolution(int dist[], int n)
{
   System.out.println("Vertex Distance from Source");
   for (int i = 0; i < V; i++)
      System.out.println(i + " tt " + dist[i]);
}

// Funtion that implements Dijkstra's single source shortest path
// algorithm for a graph represented using adjacency matrix
// representation
void dijkstra(int graph[][], int src)
{
   int dist[] = new int[V]; // The output array. dist[i] will hold
   // the shortest distance from src to i

   // sptSet[i] will true if vertex i is included in shortest
   // path tree or shortest distance from src to i is finalized
   Boolean sptSet[] = new Boolean[V];

   // Initialize all distances as INFINITE and stpSet[] as false
   for (int i = 0; i < V; i++) {
      dist[i] = Integer.MAX_VALUE;
```

```
sptSet[i] = false;
        }

        // Distance of source vertex from itself is always 0
        dist[src] = 0;

        // Find shortest path for all vertices
        for (int count = 0; count < V - 1; count++) {
            // Pick the minimum distance vertex from the set of vertices
            // not yet processed. u is always equal to src in first
            // iteration.
            int u = minDistance(dist, sptSet);

            // Mark the picked vertex as processed
            sptSet[u] = true;

            // Update dist value of the adjacent vertices of the
            // picked vertex.
            for (int v = 0; v < V; v++)

                // Update dist[v] only if is not in sptSet, there is an
                // edge from u to v, and total weight of path from src to
                // v through u is smaller than current value of dist[v]
                if (!sptSet[v] && graph[u][v] != 0 &&
                dist[u] != Integer.MAX_VALUE && dist[u] + graph[u][v] < dist[v])
                 dist[v] = dist[u] + graph[u][v];
        }

        // print the constructed distance array
        printSolution(dist, V);
    }

    // Driver method
    public static void main(String[] args)
    {
        /* Let us create the example graph discussed above */
        int graph[][] = new int[][] { { 0, 4, 0, 0, 0, 0, 0, 8, 0 },
                                      { 4, 0, 8, 0, 0, 0, 0, 11, 0 },
                                      { 0, 8, 0, 7, 0, 4, 0, 0, 2 },
                                      { 0, 0, 7, 0, 9, 14, 0, 0, 0 },
                                      { 0, 0, 0, 9, 0, 10, 0, 0, 0 },
                                      { 0, 0, 4, 14, 10, 0, 2, 0, 0 },
                                      { 0, 0, 0, 0, 0, 2, 0, 1, 6 },
                                      { 8, 11, 0, 0, 0, 0, 1, 0, 7 },
                                      { 0, 0, 2, 0, 0, 0, 6, 7, 0 } };
        ShortestPath t = new ShortestPath();
        t.dijkstra(graph, 0);

  }
```

```
}
```

Output:

```
Vertex Distance from Source 0 tt 0 ,1 tt 4 ,2 tt 12 ,3 tt 19 ,4 tt 21 ,5 tt 11 ,6 tt 9 ,7 tt 8 , and 8 tt 14
```

COMMON VIVA QUESTIONS

1) What are functions of different layers?
2) Differentiate between TCP/IP Layers and OSI Layers
3) Differentiate between TCP and UDP.
4) Differentiate between Connectionless and connection oriented connection.
5) What is meant by subnet?
6) What is meant by Gateway?
7) What is an IP address?
8) What is MAC address?
9) Define Raw Socket
10) What is a fork command?
11) Why IP address is required when we have MAC address?
12) What is meant by port?
13) What are ephemerical port number and well known port numbers?
14) What is a socket?
15) What are the parameters of socket()?
16) Describe bind(), listen(), accept(),connect(), send() and recv().
17) What are system calls? Mention few of them.
18) What is IPC? Name three techniques.
19) What type of protocol is BGP?
20) What type of protocol is OSPF?

21) Difference between ARP and RARP.

22) What is Distance Vector Routing

23) What is flooding?

24) What are three way handshakes?

25) Disadvantages of Stop and wait protocol.

26) Differentiate bridges from switches.

27) What is a Router?

28) What is routing?

29) What is the role of DNS?

30) What type of transport protocol is used for DNS ?