**SYLLABUS**

**COMPUTER NETWORK LABORATORY**

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

**(Effective from the academic year 2018 -2019)**

**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 students to

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

**PART - A**

**Description:**

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

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

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

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

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

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

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

**PART B**

**Implement the following in Java:**

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

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

9. Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.

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

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

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

**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: 10+25+5 =40

Part B: 10+25+5 =60

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

made zero.

**Course outcomes:** The students should be 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

**Table of contents**

|  |  |  |
| --- | --- | --- |
| **Exp. No.** | **Experiments** | **Page No.** |
| 1 | Three nodes point – to – point network with duplex links | 1 |
| 2 | Transmission of ping messages | 4 |
| 3 | Implement an Ethernet LAN | 8 |
| 4 | Implement simple ESS | 12 |
| 5 | Implement and study the performance of GSM | 16 |
| 6 | Implement and study the performance of CDMA | 20 |
| 7 | Error detecting code using CRC-CCITT | 24 |
| 8 | Implement Bellman-Ford algorithm | 28 |
| 9 | Implement client – server program using TCP/IP | 31 |
| 10 | Implement client – server program using UDP | 35 |
| 11 | Implement RSA algorithm | 37 |
| 12 | Implement Congestion control using leaky bucket | 39 |

**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.**

**TCL Program:**

set ns [new Simulator]

set nf [open lab1.nam w]

$ns namtrace-all $nf

set tf [open lab1.tr w]

$ns trace-all $tf

proc finish { } {

global ns nf tf

$ns flush-trace

close $nf

close $tf

exec nam lab1.nam & exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

$ns duplex-link $n0 $n2 200Mb 10ms DropTail

$ns duplex-link $n1 $n2 100Mb 5ms DropTail

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

$ns queue-limit $n0 $n2 10

$ns queue-limit $n1 $n2 10

set udp0 [new Agent/UDP]

$ns attach-agent $n0 $udp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.005

$cbr0 attach-agent $udp0

set udp1 [new Agent/UDP]

$ns attach-agent $n1 $udp1

set cbr1 [new Application/Traffic/CBR]

$cbr1 attach-agent $udp1

set udp2 [new Agent/UDP]

$ns attach-agent $n2 $udp2

set cbr2 [new Application/Traffic/CBR]

$cbr2 attach-agent $udp2

set null0 [new Agent/Null]

$ns attach-agent $n3 $null0

$ns connect $udp0 $null0

$ns connect $udp1 $null0

$ns at 0.1 "$cbr0 start"

$ns at 0.2 "$cbr1 start"

$ns at 1.0 "finish"

$ns run

**AWK Program:**

BEGIN { c=0;

}

{

If ($1= ="d")

{

c++;

printf("%s\t%s\n",$5,$11);

}

}

END{

printf("The number of packets dropped =%d\n",c);

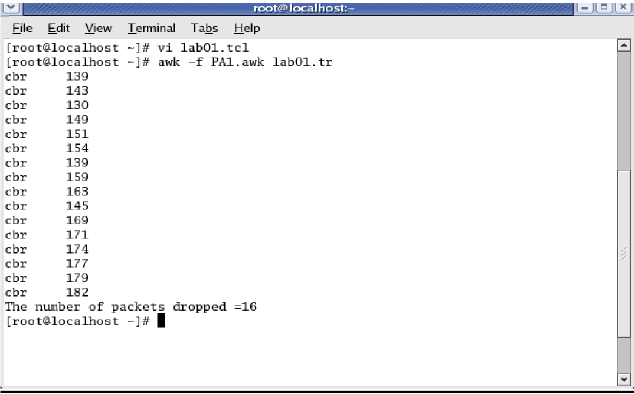
}

**OUTPUT:**

**[root@localhost ~]# ns lab1.tcl**

****

**[root@localhost ~]# awk –f lab1.awk lab1.tr**



**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.**

**TCL Program:**

set ns [ new Simulator ]

set nf [ open lab2.nam w ]

$ns namtrace-all $nf

set tf [ open lab2.tr w ]

$ns trace-all $tf

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

$n4 shape box

$ns duplex-link $n0 $n4 1005Mb 1ms DropTail

$ns duplex-link $n1 $n4 50Mb 1ms DropTail

$ns duplex-link $n2 $n4 2000Mb 1ms DropTail

$ns duplex-link $n3 $n4 200Mb 1ms DropTail

$ns duplex-link $n4 $n5 1Mb 1ms DropTail

set p1 [new Agent/Ping]

$ns attach-agent $n0 $p1

$p1 set packetSize\_ 50000

$p1 set interval\_ 0.0001

set p2 [new Agent/Ping]

$ns attach-agent $n1 $p2

set p3 [new Agent/Ping]

$ns attach-agent $n2 $p3

$p3 set packetSize\_ 30000

$p3 set interval\_ 0.00001

set p4 [new Agent/Ping]

$ns attach-agent $n3 $p4

set p5 [new Agent/Ping]

$ns attach-agent $n5 $p5

$ns queue-limit $n0 $n4 5

$ns queue-limit $n2 $n4 3b

$ns queue-limit $n4 $n5 2

Agent/Ping instproc recv {from rtt} {

$self instvar node\_

puts "node [$node\_ id] received answer from $from with round trip time $rtt msec"

}

$ns connect $p1 $p5

$ns connect $p3 $p4

proc finish { } {

global ns nf tf

$ns flush-trace

close $nf

close $tf

exec nam lab2.nam &

exit 0

}

$ns at 0.1 "$p1 send"

$ns at 0.2 "$p1 send"

$ns at 0.3 "$p1 send"

$ns at 0.4 "$p1 send"

$ns at 0.5 "$p1 send"

$ns at 0.6 "$p1 send"

$ns at 0.7 "$p1 send"

$ns at 0.8 "$p1 send"

$ns at 0.9 "$p1 send"

$ns at 1.0 "$p1 send"

$ns at 1.1 "$p1 send"

$ns at 1.2 "$p1 send"

$ns at 1.3 "$p1 send"

$ns at 1.4 "$p1 send"

$ns at 1.5 "$p1 send"

$ns at 1.6 "$p1 send"

$ns at 1.7 "$p1 send"

$ns at 1.8 "$p1 send"

$ns at 1.9 "$p1 send"

$ns at 2.0 "$p1 send"

$ns at 2.1 "$p1 send"

$ns at 2.2 "$p1 send"

$ns at 2.3 "$p1 send"

$ns at 2.4 "$p1 send"

$ns at 2.5 "$p1 send"

$ns at 2.6 "$p1 send"

$ns at 2.7 "$p1 send"

$ns at 2.8 "$p1 send"

$ns at 2.9 "$p1 send"

$ns at 0.1 "$p3 send"

$ns at 0.2 "$p3 send"

$ns at 0.3 "$p3 send"

$ns at 0.4 "$p3 send"

$ns at 0.5 "$p3 send"

$ns at 0.6 "$p3 send"

$ns at 0.7 "$p3 send"

$ns at 0.8 "$p3 send"

$ns at 0.9 "$p3 send"

$ns at 1.0 "$p3 send"

$ns at 1.1 "$p3 send"

$ns at 1.2 "$p3 send"

$ns at 1.3 "$p3 send"

$ns at 1.4 "$p3 send"

$ns at 1.5 "$p3 send"

$ns at 1.6 "$p3 send"

$ns at 1.7 "$p3 send"

$ns at 1.8 "$p3 send"

$ns at 1.9 "$p3 send"

$ns at 2.0 "$p3 send"

$ns at 2.1 "$p3 send"

$ns at 2.2 "$p3 send"

$ns at 2.3 "$p3 send"

$ns at 2.4 "$p3 send"

$ns at 2.5 "$p3 send"

$ns at 2.6 "$p3 send"

$ns at 2.7 "$p3 send"

$ns at 2.8 "$p3 send"

$ns at 2.9 "$p3 send"

$ns at 3.0 "finish"

$ns run

**AWK Program:**

BEGIN{

drop=0;

}

{

if($1=="d" )

{

drop++;

}

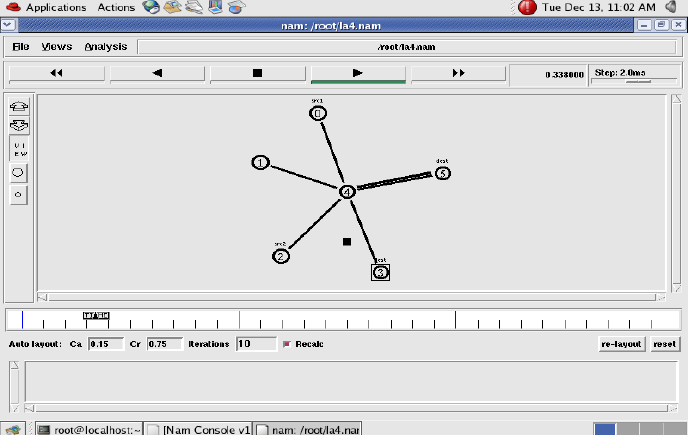
} END{

printf("Total number of %s packets dropped due to congestion =%d\n",$5,drop);

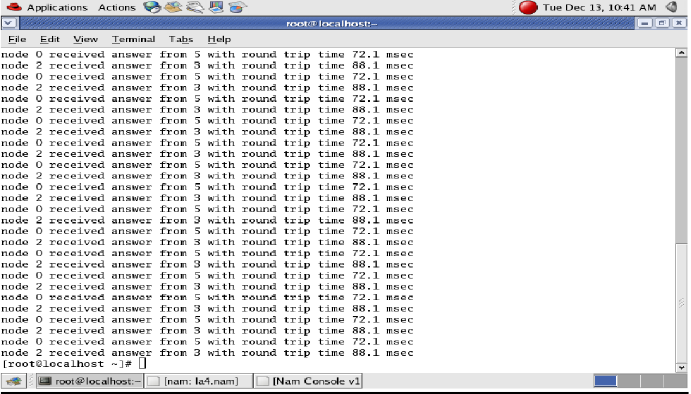
}

**OUTPUT:**

**[root@localhost ~]# ns lab2.tcl**



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



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

**TCL Program:**

set ns [new Simulator]

set tf [open lab3.tr w]

$ns trace-all $tf

set nf [open lab3.nam w]

$ns namtrace-all $nf

set n0 [$ns node]

$n0 color "magenta"

$n0 label "src1"

set n1 [$ns node]

set n2 [$ns node]

$n2 color "magenta"

$n2 label "src2"

set n3 [$ns node]

$n3 color "blue"

$n3 label "dest2"

set n4 [$ns node]

set n5 [$ns node]

$n5 color "blue"

$n5 label "dest1"

LanRouter set debug\_ 0

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

$ns duplex-link $n4 $n5 1Mb 1ms DropTail

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

$ftp0 set packetSize\_ 500

$ftp0 set interval\_ 0.0001

set sink5 [new Agent/TCPSink]

$ns attach-agent $n5 $sink5

$ns connect $tcp0 $sink5

set tcp2 [new Agent/TCP]

$ns attach-agent $n2 $tcp2

set ftp2 [new Application/FTP]

$ftp2 attach-agent $tcp2

$ftp2 set packetSize\_ 600

$ftp2 set interval\_ 0.001

set sink3 [new Agent/TCPSink]

$ns attach-agent $n3 $sink3

$ns connect $tcp2 $sink3

set file1 [open file1.tr w]

$tcp0 attach $file1

set file2 [open file2.tr w]

$tcp2 attach $file2

$tcp0 trace cwnd\_

$tcp2 trace cwnd\_

proc finish { } {

global ns nf tf

$ns flush-trace

close $tf

close $nf

exec nam lab3.nam &

exit 0

}

$ns at 0.1 "$ftp0 start"

$ns at 5 "$ftp0 stop"

$ns at 7 "$ftp0 start"

$ns at 0.2 "$ftp2 start"

$ns at 8 "$ftp2 stop"

$ns at 14 "$ftp0 stop"

$ns at 10 "$ftp2 start"

$ns at 15 "$ftp2 stop"

$ns at 16 "finish"

$ns run

**AWK Program:**

﻿BEGIN {

}

{

if($6=="cwnd\_")

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

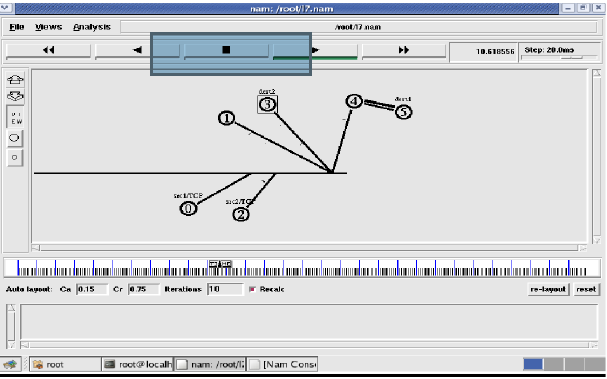
}

END {

}

**OUTPUT:**

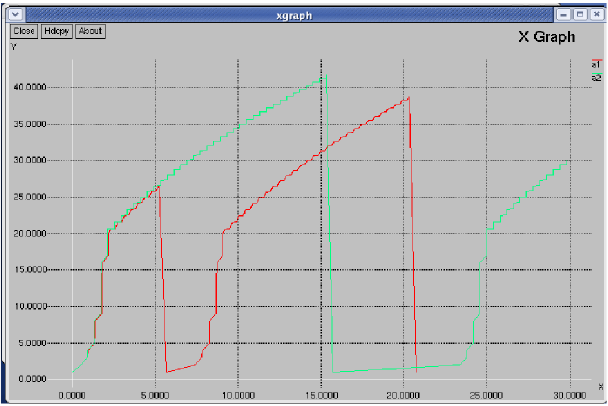
**[root@localhost ~]# ns lab3.tcl**



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

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

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



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

**TCL Program:**

﻿set ns [new Simulator]

set tf [open lab4.tr w]

$ns trace-all $tf

set topo [new Topography]

$topo load\_flatgrid 1000 1000

set nf [open lab4.nam w]

$ns namtrace-all-wireless $nf 1000 1000

$ns node-config -adhocRouting DSDV \

-llType LL \

-macType Mac/802\_11 \

-ifqType Queue/DropTail \

-ifqLen 50 \

-phyType Phy/WirelessPhy \

-channelType Channel/WirelessChannel \

-propType Propagation/TwoRayGround \

-antType Antenna/OmniAntenna \

-topoInstance $topo \

-agentTrace ON \

-routerTrace ON

create-god 3

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

$n0 label "tcp0"

$n1 label "sink1/tcp1"

$n2 label "sink2"

$n0 set X\_ 50

$n0 set Y\_ 50

$n0 set Z\_ 0

$n1 set X\_ 100

$n1 set Y\_ 100

$n1 set Z\_ 0

$n2 set X\_ 600

$n2 set Y\_ 600

$n2 set Z\_ 0

$ns at 0.1 "$n0 setdest 50 50 15"

$ns at 0.1 "$n1 setdest 100 100 25"

$ns at 0.1 "$n2 setdest 600 600 25"

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

set sink1 [new Agent/TCPSink]

$ns attach-agent $n1 $sink1

$ns connect $tcp0 $sink1

set tcp1 [new Agent/TCP]

$ns attach-agent $n1 $tcp1

set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1

set sink2 [new Agent/TCPSink]

$ns attach-agent $n2 $sink2

$ns connect $tcp1 $sink2

$ns at 5 "$ftp0 start"

$ns at 5 "$ftp1 start"

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

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

proc finish { } {

global ns nf tf

$ns flush-trace

exec nam lab4.nam &

close $tf

exit 0

}

$ns at 250 "finish"

$ns run

**AWK Program:**

BEGIN{

count1=0

count2=0

pack1=0

pack2=0

time1=0

time2=0

}

{

if($1=="r"&& $3=="\_1\_" && $4=="AGT")

{

count1++

pack1=pack1+$8

time1=$2

}

if($1=="r" && $3=="\_2\_" && $4=="AGT")

{

count2++

pack2=pack2+$8

time2=$2

}

}

END{

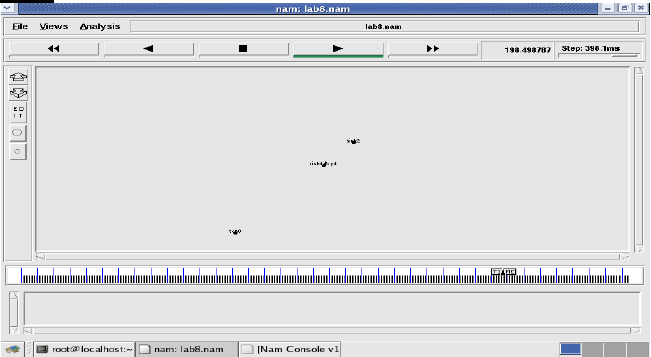
printf("The Throughput from n0 to n1: %f Mbps \n", ((count1\*pack1\*8)/(time1\*1000000)));

printf("The Throughput from n1 to n2: %f Mbps", ((count2\*pack2\*8)/(time2\*1000000)));

}

**OUTPUT:**

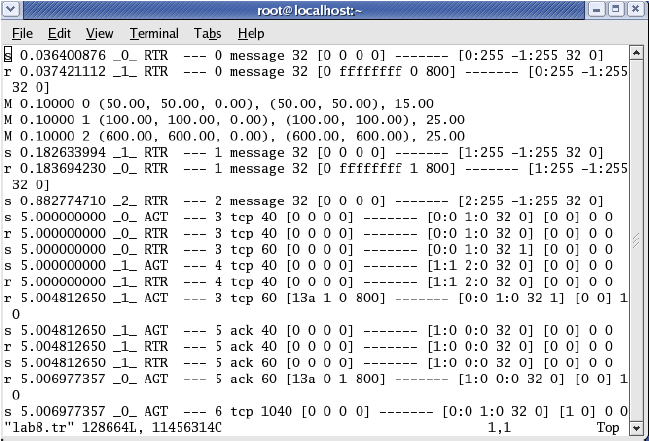
**[root@localhost ~]# ns lab4.tcl**



**[root@localhost~]# awk –f lab4.awk lab4.tr**

The Throughput from n0 to n1: 5863.442245Mbps

The Throughput from n1 to n2: 1307.611834Mbps



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

**TCL Program:**

set val(chan) Channel/WirelessChannel

set val(type) GSM

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(x) 1500

set val(y) 1500

set val(ifqlen) 1000

set val(adhocRouting) AODV

set val(nn) 10

set val(stop) 5.0

set f0 [open out02.tr w]

set f1 [open lost02.tr w]

set f2 [open delay02.tr w]

set ns\_ [new Simulator]

set topo [new Topography]

set tracefd [open out.tr w]

set namtrace [open out.nam w]

$ns\_ trace-all $tracefd

$ns\_ namtrace-all-wireless $namtrace $val(x) $val(y)

$topo load\_flatgrid $val(x) $val(y)

set god\_ [create-god $val(nn)]

$ns\_ color 0 red

$ns\_ node-config -adhocRouting AODV \

-llType $val(ll) \

-macType $val(mac) \

-ifqType $val(ifq) \

-ifqLen $val(ifqlen) \

-antType $val(ant) \

-propType $val(prop) \

-phyType $val(netif) \

-channelType $val(chan) \

-energyModel EnergyModel \

-initialEnergy 100 \

-rxPower 0.3 \

-txPower 0.6 \

-topoInstance $topo \

-agentTrace ON \

-routerTrace ON \

-macTrace OFF

for {set i 0} {$i < $val(nn) } {incr i} {

set node\_($i) [$ns\_ node]

}

set X1(0) 1035.201

set Y1(0) 444.699

set X1(1) 244.365

set Y1(1) 521.418

set X1(2) -18.1268

set Y1(2) 300.612

set X1(3) 723.89

set Y1(3) 343.533

set X1(4) 122.34

set Y1(4) 311.755

set X1(5) 373.498

set Y1(5) 472.206

set X1(6) 548.549

set Y1(6) 361.062

set X1(7) 389.995

set Y1(7) 381.178

set X1(8) 494.798

set Y1(8) 477.771

set X1(9) 275.01

set Y1(9) 381.99

for {set i 0} {$i < $val(nn) } {incr i} {

$node\_($i) set X\_ $X1($i)

$node\_($i) set Y\_ $Y1($i)

$node\_($i) set Z\_ 0.0

}

puts "---------------------------------------"

set m 0

puts "----------------------------------------"

puts "| Node | One hop neighbour |"

puts "----------------------------------------"

for {set i 0} {$i < $val(nn) } {incr i} {

set k 0

for {set j 0} {$j < $val(nn) } {incr j} {

set a [ expr $X1($j)-$X1($i)]

set b [ expr $a\*$a]

set c [ expr $Y1($j)-$Y1($i)]

set d [ expr $c\*$c]

set e [ expr $b+$d]

set f 0.5

set g [expr pow($e,$f)]

#puts "Distance from node($i) --to--node($j)----------->$g"

if {$g <= 200 && $i != $j} {

puts "| node($i) | node($j) |"

set nei($m) $j

set k [expr $k+1]

set m [ expr $m+1]

}

}

puts "----------------------------------------"

}

puts "Loading connection pattern..."

puts "Loading scenario file..."

for {set i 0} {$i < $val(nn) } {incr i} {

$ns\_ initial\_node\_pos $node\_($i) 45

}

for {set i 0} {$i < $val(nn) } {incr i} {

$ns\_ at $val(stop).0 "$node\_($i) reset";

}

set udp\_(0) [new Agent/UDP]

$ns\_ attach-agent $node\_(2) $udp\_(0)

set sink [new Agent/LossMonitor]

$ns\_ attach-agent $node\_(3) $sink

set cbr1\_(0) [new Application/Traffic/CBR]

$cbr1\_(0) set packetSize\_ 1000

$cbr1\_(0) set interval\_ 0.1

$cbr1\_(0) set maxpkts\_ 1000

$cbr1\_(0) attach-agent $udp\_(0)

$ns\_ connect $udp\_(0) $sink

$ns\_ at 1.00 "$cbr1\_(0) start"

set holdtime 0

set holdseq 0

set holdrate1 0

proc record {} {

global sink f0 f1 f2 holdtime holdseq holdrate1

set ns [Simulator instance]

set time 0.9 ;#Set Sampling Time to 0.9 Sec

set bw0 [$sink set bytes\_]

set bw1 [$sink set nlost\_]

set bw2 [$sink set lastPktTime\_]

set bw3 [$sink set npkts\_]

set now [$ns now]

# Record Bit Rate in Trace Files

puts $f0 "$now [expr (($bw0+$holdrate1)\*8)/(2\*$time\*1000000)]"

# Record Packet Loss Rate in File

puts $f1 "$now [expr $bw1/$time]"

if { $bw3 > $holdseq } {

puts $f2 "$now [expr ($bw2 - $holdtime)/($bw3 - $holdseq)]"

} else {

puts $f2 "$now [expr ($bw3 - $holdseq)]"

}

$sink set bytes\_ 0

$sink set nlost\_ 0

set holdtime $bw2

set holdseq $bw3

set holdrate1 $bw0

$ns at [expr $now+$time] "record" ;# Schedule Record after $time interval sec

}

# Start Recording at Time 0

$ns\_ at 0.0 "record"

source link.tcl

proc stop {} {

global ns\_ tracefd f0 f1 f2

# Close Trace Files

close $f0

close $f1

close $f2

exec nam out.nam

# Plot Recorded Statistics

exec xgraph out02.tr -geometry -x TIME -y thr -t Throughput 800x400 &

exec xgraph lost02.tr -geometry -x TIME -y loss -t Packet\_loss 800x400 &

exec xgraph delay02.tr -geometry -x TIME -y delay -t End-to-End-Delay 800x400 &

$ns\_ flush-trace

}

$ns\_ at $val(stop) "stop"

$ns\_ at $val(stop).0002 "puts \"NS EXITING...\" ; $ns\_ halt"

puts $tracefd "M 0.0 nn $val(nn) x $val(x) y $val(y) rp "

puts $tracefd "M 0.0 prop $val(prop) ant $val(ant)"

puts "Starting Simulation..."

$ns\_ run

**Link.tcl**

$ns\_ at 0.1 "$node\_(0) setdest 786 813 20"

$ns\_ at 0.1 "$node\_(1) setdest 895 890 20"

$ns\_ at 0.1 "$node\_(2) setdest 633 669 20"

$ns\_ at 0.1 "$node\_(3) setdest 1375 712 20"

$ns\_ at 0.1 "$node\_(4) setdest 773 680 20"

$ns\_ at 0.1 "$node\_(5) setdest 1024 841 20"

$ns\_ at 0.1 "$node\_(6) setdest 1199 730 20"

$ns\_ at 0.1 "$node\_(7) setdest 1041 750 20"

$ns\_ at 0.1 "$node\_(8) setdest 1146 846 20"

$ns\_ at 0.1 "$node\_(9) setdest 926 751 20"

$ns\_ at 0.5 "$node\_(2) add-mark m blue square"

$ns\_ at 0.5 "$node\_(3) add-mark m blue square"

$ns\_ at 0.5 "$node\_(2) label source"

$ns\_ at 0.5 "$node\_(3) label Destination"

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

**TCL Program:**

puts "Enter number of nodes"

set tnn [gets stdin]

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(x) 1500

set val(y) 1500

set val(ifqlen) 1000

set val(adhocRouting) AODV

set val(nn) $tnn

set val(stop) 10.0

Mac/802\_11 set cdma\_code\_bw\_start\_ 0 ;# cdma code for bw request (start)

Mac/802\_11 set cdma\_code\_bw\_stop\_ 63 ;# cdma code for bw request (stop)

Mac/802\_11 set cdma\_code\_init\_start\_ 64 ;# cdma code for initial request (start)

Mac/802\_11 set cdma\_code\_init\_stop\_ 127 ;# cdma code for initial request (stop)

Mac/802\_11 set cdma\_code\_cqich\_start\_ 128 ;# cdma code for cqich request (start)

Mac/802\_11 set cdma\_code\_cqich\_stop\_ 195 ;# cdma code for cqich request (stop)

Mac/802\_11 set cdma\_code\_handover\_start\_ 196 ;# cdma code for handover request (start)

Mac/802\_11 set cdma\_code\_handover\_stop\_ 255 ;# cdma code for handover request (stop)

set f0 [open out02.tr w]

set f1 [open lost02.tr w]

set f2 [open delay02.tr w]

set ns\_ [new Simulator]

set topo [new Topography]

set tracefd [open out.tr w]

set namtrace [open out.nam w]

$ns\_ trace-all $tracefd

$ns\_ namtrace-all-wireless $namtrace $val(x) $val(y)

$topo load\_flatgrid $val(x) $val(y)

set god\_ [create-god $val(nn)]

$ns\_ color 0 red

$ns\_ node-config -adhocRouting AODV \

-llType $val(ll) \

-macType $val(mac) \

-ifqType $val(ifq) \

-ifqLen $val(ifqlen) \

-antType $val(ant) \

-propType $val(prop) \

-phyType $val(netif) \

-channelType $val(chan) \

-energyModel EnergyModel \

-initialEnergy 100 \

-rxPower 0.3 \

-txPower 0.6 \

-topoInstance $topo \

-agentTrace ON \

-routerTrace ON \

-macTrace OFF

for {set i 0} {$i < $val(nn) } {incr i} {

set node\_($i) [$ns\_ node]

$node\_($i) set X\_ [expr rand() \* 500]

$node\_($i) set Y\_ [expr rand() \* 500]

$node\_($i) set Z\_ 0.000000000000;

}

for {set i 0} {$i < $val(nn) } {incr i} {

set xx [expr rand() \* 1500]

set yy [expr rand() \* 1000]

$ns\_ at 0.1 "$node\_($i) setdest $xx 4yy 5"

}

puts "Loading connection pattern..."

puts "Loading scenario file..."

for {set i 0} {$i < $val(nn) } {incr i} {

$ns\_ initial\_node\_pos $node\_($i) 55

}

for {set i 0} {$i < $val(nn) } {incr i} {

$ns\_ at $val(stop).0 "$node\_($i) reset";

}

puts "Enter source node"

set source [gets stdin]

puts "Enter destination node"

set dest [gets stdin]

set udp\_(0) [new Agent/UDP]

$ns\_ attach-agent $node\_($source) $udp\_(0)

set sink [new Agent/LossMonitor]

$ns\_ attach-agent $node\_($dest) $sink

set cbr1\_(0) [new Application/Traffic/CBR]

$cbr1\_(0) set packetSize\_ 1000

$cbr1\_(0) set interval\_ 0.1

$cbr1\_(0) set maxpkts\_ 10000

$cbr1\_(0) attach-agent $udp\_(0)

$ns\_ connect $udp\_(0) $sink

$ns\_ at 1.00 "$cbr1\_(0) start"

set holdtime 0

set holdseq 0

set holdrate1 0

proc record {} {

global sink f0 f1 f2 holdtime holdseq holdrate1

set ns [Simulator instance]

set time 0.9 ;#Set Sampling Time to 0.9 Sec

set bw0 [$sink set bytes\_]

set bw1 [$sink set nlost\_]

set bw2 [$sink set lastPktTime\_]

set bw3 [$sink set npkts\_]

set now [$ns now]

# Record Bit Rate in Trace Files

puts $f0 "$now [expr (($bw0+$holdrate1)\*8)/(2\*$time\*1000000)]"

# Record Packet Loss Rate in File

puts $f1 "$now [expr $bw1/$time]"

if { $bw3 > $holdseq } {

puts $f2 "$now [expr ($bw2 - $holdtime)/($bw3 - $holdseq)]"

} else {

puts $f2 "$now [expr ($bw3 - $holdseq)]"

}

$sink set bytes\_ 0

$sink set nlost\_ 0

set holdtime $bw2

set holdseq $bw3

set holdrate1 $bw0

$ns at [expr $now+$time] "record" ;# Schedule Record after $time interval sec

}

# Start Recording at Time 0

$ns\_ at 0.0 "record"

source link.tcl

proc stop {} {

global ns\_ tracefd f0 f1 f2

# Close Trace Files

close $f0

close $f1

close $f2

exec nam out.nam

exec xgraph out02.tr -geometry -x TIME -y thr -t Throughput 800x400 &

exec xgraph lost02.tr -geometry -x TIME -y loss -t Packet\_loss 800x400 &

exec xgraph delay02.tr -geometry -x TIME -y delay -t End-to-End-Delay 800x400 &

$ns\_ flush-trace

}

$ns\_ at $val(stop) "stop"

$ns\_ at $val(stop).0002 "puts \"NS EXITING...\" ; $ns\_ halt"

puts $tracefd "M 0.0 nn $val(nn) x $val(x) y $val(y) rp "

puts $tracefd "M 0.0 prop $val(prop) ant $val(ant)"

puts "Starting Simulation..."

$ns\_ run

**Link.tcl**

$ns\_ at 0.5 "$node\_($source) add-mark m blue square"

$ns\_ at 0.5 "$node\_($dest) add-mark m magenta square"

$ns\_ at 0.5 "$node\_($source) label SENDER"

$ns\_ at 0.5 "$node\_($dest) label RECEIVER"

$ns\_ at 0.01 "$ns\_ trace-annotate \"Network Deployment\""

**graph.tcl**

exec xgraph ACO-thr.tr -x Time -y Throughout -t Throughout -m -bg white -geometry 800\*500 &

exec xgraph ACO-delay.tr -x Time -y Delay -t Delay -m -bg white -geometry 800\*500 &

exec xgraph ACO-overhead.tr -x Time -y overhead -t Overhead -m -bg white -geometry 800\*500 &

exec xgraph ACO-pdr.tr -x Time -y pdr -t PDR -m -bg white -geometry 800\*500 &

**PART B**

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

import java.util.\*;

public class CRC

{

void div(int a[],int k)

{

int gp[]={1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1};

int count=0;

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

{

if(a[i]==gp[0])

{

for(int j=i;j<17+i;j++)

{

a[j]=a[j]^gp[count++];

}

count=0;

}

}

}

/\*\*

\* @param args

\*/

public static void main(String[] args)

{

// TODO Auto-generated method stub

int a[]=new int[100];

int b[]=new int[100];

int len,k;

CRC ob=new CRC();

System.out.println("Enter the length of Data Frame:");

Scanner sc=new Scanner(System.in);

len=sc.nextInt();

int flag=0;

System.out.println("Enter the Message:");

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

{

a[i]=sc.nextInt();

}

for(int i=0;i<16;i++)

{

a[len++]=0;

}

k=len-16;

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

{

b[i]=a[i];

}

ob.div(a,k);

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

a[i]=a[i]^b[i];

System.out.println("Data to be transmitted: ");

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

{

System.out.print(a[i]+" ");

}

System.out.println();

System.out.println("Enter the Reveived Data: ");

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

{

a[i]=sc.nextInt();

}

ob.div(a, k);

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

{

if(a[i]!=0)

{

flag=1;

break;

}

}

if(flag==1)

System.out.println("error in data");

else

System.out.println("no error");

}

}

**OUTPUT**

**RUN 1**

Enter the length of Data Frame:

4

Enter the Message:

1 1 0 1

Data to be transmitted:

1 1 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 1 0 1

Enter the Received Data:

1 1 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 1 0 1

no error

**RUN 2**

Enter the length of Data Frame:

4

Enter the Message:

1 1 0 1

Data to be transmitted:

1 1 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 1 0 1

Enter the Received Data:

1 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 1 0 1

error in data

**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 nov;

public static final int MAX\_VALUE = 999;

public BellmanFord(int nov)

{

this.nov = nov;

d = new int[nov + 1];

}

public void BellmanFordEvaluation(int s, int am[][])

{

for (int n = 1; n <= nov; n++)

{

d[n] = MAX\_VALUE;

}

d[s] = 0;

for (int n = 1; n <= nov - 1; n++)

{

for (int sn = 1; sn <= nov; sn++)

{

for (int dn = 1; dn <= nov; dn++)

{

if (am[sn][dn] != MAX\_VALUE)

{

if (d[dn] > d[sn] + am[sn][dn])

d[dn] = d[sn] + am[sn][dn];

}

}

}

}

for (int sn = 1; sn <= nov; sn++)

{

for (int dn = 1; dn <= nov; dn++)

{

if (am[sn][dn] != MAX\_VALUE)

{

if(d[dn] > d[sn] + am[sn][dn])

{

System.out.println("The Graph contains negative egde cycle");

break;

}

}

}

}

for (int v = 1; v <= nov; v++)

{

System.out.println("Distance of source " + s + " to " + v + " is " + d[v]);

}

}

public static void main(String args[])

{

int nov = 0;

int s;

Scanner scanner = new Scanner(System.in);

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

nov = scanner.nextInt();

int am[][] = new int[nov + 1][nov + 1];

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

for (int sn = 1; sn <= nov; sn++)

{

for (int dn = 1; dn <= nov; dn++)

{

am[sn][dn] = scanner.nextInt();

if (sn == dn)

{

am[sn][dn] = 0;

continue;

}

if (am[sn][dn] == 0)

{

am[sn][dn] = MAX\_VALUE;

}

}

}

System.out.println("Enter the source vertex");

s = scanner.nextInt();

BellmanFord bellmanford = new BellmanFord(nov);

bellmanford.BellmanFordEvaluation(s, am);

scanner.close();

}

}

OUTPUT:

[root@localhost ~]# javac BellmanFord.java

[root@localhost ~]# java BellmanFord

Enter the number of vertices

4

Enter the adjacency matrix

0 5 0 0

5 0 3 4

0 3 0 2

0 4 2 0

Enter the source vertex

2

Distance of source 2 to 1 is 5

Distance of source 2 to 2 is 0

Distance of source 2 to 3 is 3

Distance of source 2 to 4 is 4

**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.**

TCP Server:

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

class TCPServer

{

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

{

while(true)

{

ServerSocket ss=new ServerSocket(5000);

System.out.println ("Waiting for request");

Socket s=ss.accept();

System.out.println ("Connected With "+s.getInetAddress().toString());

DataInputStream din=new DataInputStream(s.getInputStream());

DataOutputStream dout=new DataOutputStream(s.getOutputStream());

try

{

String filename="";

filename=din.readUTF();

System.out.println("Receiving file name");

System.out.println("SendGet....Ok");

File f=new File(filename);

FileInputStream fin=new FileInputStream(f);

long sz=(int) f.length();

byte b[]=new byte [1024];

int read;

dout.writeUTF(Long.toString(sz));

dout.flush();

System.out.println ("File Size: "+sz+" Bytes");

System.out.println ("Receive Buffer size: "+ss.getReceiveBufferSize());

System.out.println("Sending file contents");

while((read = fin.read(b)) != -1)

{

dout.write(b, 0, read);

dout.flush();

}

fin.close();

dout.flush();

System.out.println("Send Complete");

}

catch(Exception e)

{

e.printStackTrace();

System.out.println("An error occured");

}

din.close();

s.close();

ss.close();

}

}

}

**TCP Client:**

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

class TCPClient

{

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

{

String address = "";

Scanner sc=new Scanner(System.in);

System.out.println("Enter Server Address: ");

address=sc.nextLine();

Socket s=new Socket(address,5000);

DataInputStream din=new DataInputStream(s.getInputStream());

DataOutputStream dout=new DataOutputStream(s.getOutputStream());

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

String filename,rcvfile;

System.out.println("Enter File Name: ");

filename=sc.nextLine();

sc.close();

try

{

dout.writeUTF(filename);

dout.flush();

rcvfile="client"+filename;

System.out.println("Saving file as: "+rcvfile);

FileOutputStream fos=new FileOutputStream(new File(rcvfile),true);

long bytesRead;

System.out.println("Receving file..");

long sz=Long.parseLong(din.readUTF());

System.out.println ("File Size: "+sz+" Bytes");

byte b[]=new byte [1024];

do

{

bytesRead = din.read(b, 0, b.length);

fos.write(b,0,b.length);

}while(!(bytesRead<1024));

System.out.println("Completed");

fos.close();

dout.close();

s.close();

}

catch(EOFException e)

{

//do nothing

}

}

}

**OUTPUT:**

**TCPServer**

Waiting for request

Connected With /127.0.0.1

Receiving file name

SendGet....Ok

File Size: 17 Bytes

Receive Buffer size: 8192

Sending file contents

Send Complete

Waiting for request

**TCPClient**

Enter Server Address:

127.0.0.1

Enter File Name:

sample.txt

Saving file as: clientsample.txt

Receiving file..

File Size: 17 Bytes

Completed

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

**UDP Server:**

import java.net.\*;

public class UDPServer

{

public static void main(String[] args)

{

DatagramSocket skt=null;

try

{

skt=new DatagramSocket(6788);

byte[] buffer = new byte[1000];

System.out.println("Listening on port 6788");

while(true)

{

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

skt.receive(request);

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

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

DatagramPacket reply = new DatagramPacket(sendMsg,sendMsg.length,

request.getAddress(),request.getPort());

skt.send(reply);

}

}

catch(Exception ex)

{

}

}

}

**UDP Client:**

import java.net.\*;

public class UDPClient

{

public static void main(String[] args)

{

DatagramSocket skt;

try

{

skt=new DatagramSocket();

String msg= "Computer Networks ";

byte[] b = msg.getBytes();

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

int serverPort=6788;

DatagramPacket request =new DatagramPacket (b,b.length,host,serverPort);

skt.send(request);

byte[] buffer =new byte[1000];

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

skt.receive(reply);

String s1 = new String(reply.getData());

System.out.println("Client received:" + s1.trim());

skt.close();

}

catch(Exception ex)

{

}

}

}

**OUTPUT:**

**UDPServer**

Listening on port 6788

server received request

server sending response

**UDPClient**

Enter Message: networks

Client received: NETWORKS

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

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 tobe encrypted:");

message= sc.nextLine();

System.out.println("Enter value of p and q\n");

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;

nofelem=message.length();

for(i=0;i< nofelem;i++)

{

char c = message.charAt(i);

nummes[i]=c-96;

}

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\n ");

for(i=0;i<nofelem;i++)

System.out.print((char)(decrypted[i]+96));

}

}

**OUTPUT:**

Enter the Message to be encrypted:

Computer Networks Lab

Enter value of p and q

7

17

Encrypted message

-7136136721903186-64-86319010936864466-64-90132

Decrypted message

Computer Networks Lab

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

import java.util.Scanner;

public class LeakyBucket

{

public static void main(String args[])

{

int n, outgoing, store, bucketSize;

int incoming[];

Scanner scan = new Scanner(System.in);

System.out.println("Enter number of inputs");

n = scan.nextInt();

incoming = new int[n];

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

{

System.out.println("Enter incoming packet size of "+(i+1)+" packet");

incoming[i] = scan.nextInt();

}

System.out.println("Enter bucket size");

bucketSize = scan.nextInt();

System.out.println("Enter outgoing rate");

outgoing = scan.nextInt();

store = 0;

int i = 0;

System.out.println("Packet Recieved | Packet Dropped | Packet Sent | Packet Left");

do

{

int pktReceived = 0, pktSent = 0,pktDrop = 0;

if(i < n)

{

pktReceived = incoming[i];

if(pktReceived <= (bucketSize - store))

{

store += pktReceived ;

}

else

{

pktDrop = pktReceived -(bucketSize - store);

store = bucketSize;

}

}

if(store > outgoing)

{

store -= outgoing;

pktSent = outgoing;

}

else

{

pktSent = store;

store = 0;

}

System.out.println(pktReceived +"\t\t"+pktDrop+"\t\t"+pktSent+"\t\t"+store);

try

{

Thread.sleep(2000);

}

catch(Exception e)

{

}

i++;

}while(store != 0 || i < n);

}

}

**OUTPUT**

Enter number of inputs

5

Enter incoming packet size 1

20

Enter incoming packet size 2

60

Enter incoming packet size 3

40

Enter incoming packet size 4

75

Enter incoming packet size 5

30

Enter bucket size

50

Enter outgoing rate

25

Packet Recieved | Packet Dropped | Packet Sent | Packet Left

20 0 20 0

60 10 25 25

40 15 25 25

75 50 25 25

30 5 25 25

0 0 25 0