CN lab notes

Index:

PART A:

Program 1 - Point to Point network:

```
#============= Simulation parameters setup ================

set val(stop)   2.0                      ;# global time of simulation end

#============= Initialization ================
#Create a ns simulator
#objects are created using command set and operator new (to allocate memory)
set ns [new Simulator] ;# object ns is created under class Simulator using operator new and command set

#Open the NS trace file
set tracefile [open out.tr w] ;# trace file object is created in the write mode and named as out.tr
$ns trace-all $tracefile ;# moves data from object ns to trace file

#Open the NAM trace file
set namfile [open out.nam w] ;# name file object is created in the write mode and named as out.nam
$ns namtrace-all $namfile ;# moves data from object ns to nam file

#============== Nodes Definition ==============
#Create 3 nodes
set n0 [$ns node] ;# setting nodes of type node using command set
set n1 [$ns node]
set n2 [$ns node]

#========== Links Definition ==========
#Createlinks between nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail

# 1Mb bandwidth (channel capacity)
# propogation delay time is 10ms (time taken to send packets from one node to the other)
# queue type is drop tail - at the source end packets will be dropped when the queue is full

$ns queue-limit $n0 $n1 5
# queue limit is the queue size, here 5 packets can be sent from node0 to node1 at a time

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

# Give node position (for NAM)
$ns duplex-link-op $n0 $n1 orient right-down ;# position of node n1 wrt to n0
$ns duplex-link-op $n1 $n2 orient left-down ;# position of node n2 wrt to n1

#======== Agents Definition ========

#Setup a TCP connection
set tcp0 [new Agent/TCP]  ;# tcp0 is an object under the class tcp agent - tcp is a subclass under the class agent
$ns attach-agent $n0 $tcp0  ;# characteristics of the tcp protocol is acquired by the node0 by attaching tcp0 to n0
set sink2 [new Agent/TCPSink]  ;# sink2 is an object under the class tcpsink agent - tcpsink is a subclass under the class agent
$ns attach-agent $n1 $sink2   ;# characteristics of the tcpsink protocol is acquired by the node1 by attaching sink2 to n1
$ns connect $tcp0 $sink2  ;# tcp0 and sink2 is connected using method connect
$tcp0 set packetSize_ 1500  ;# default packet size is set to 1500 for tcp connection
```

```
#note: In tcp connection - sender is the source node and receiver is the sink

#Setup a TCP connection
set tcp1 [new Agent/TCP]
$ns attach-agent $n2 $tcp1
set sink3 [new Agent/TCPSink]
$ns attach-agent $n1 $sink3
$ns connect $tcp1 $sink3
$tcp1 set packetSize_ 1500

#======== Applications Definition ========
# Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP] ;# ftp0 object is created under the class application FTP - FTP is the subclass of class Application
$ftp0 attach-agent $tcp0 ;# characteristics of the ftp protocol is acquired by tcp0 which is connected to node0

# now node0 has ftp0 and tcp0 protocols

$ns at 0.0 "$ftp0 start" ;# for node n0 and n1 connection start time is set to 0 - packets are sent as soon as simulation starts
$ns at 2.0 "$ftp0 stop" ;# stop time of packet transmission

#Setup a FTP Application over TCP connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns at 1.0 "$ftp1 start"
$ns at 2.0 "$ftp1 stop"

#======== Termination =================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile ;# namefile, tracefile and ns are global objects
    $ns flush-trace ;# flushing the trace file
    close $tracefile ;# closing the files
    close $namfile
    exec nam out.nam & ;# animation file runs at the backend - indicated by &
    exit 0
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)" ;# stop nam file
$ns at $val(stop) "finish"            ;# stop finish procedure
$ns at $val(stop) "puts \"done\" ; $ns halt"        ;# halting simulation process at val(stop) time period
$ns run                 ;# run new simulator process
```

Program 2 - Ping messages:

```
#========Simulation Parameters Setup========

set val(stop) 10.0 ;           #Time of simulation end.

#Initialization

#Create a ns simulator
set ns [new Simulator]

#Open the NS Trace File
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile

#========Nodes Definition========

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n0 label "ping0"
$n1 label "ping1"
$n2 label "R1"
$n3 label "R2"
$n4 label "ping4"
$n5 label "ping5"
$ns color 1 red
$ns color 2 blue
$ns color 3 green
$ns color 4 orange

#========Links definition========

#create link between nodes
$ns duplex-link $n0 $n2 0.4Mb 10ms DropTail
```

```
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 4kb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n3 $n5 1Mb 10ms DropTail

#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down

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

set ping1 [new Agent/Ping]
$ns attach-agent $n1 $ping1

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

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

$ns connect $ping0 $ping4
$ns connect $ping1 $ping5

proc sendPingPacket {} {
  global ns ping0 ping1
  set intervalTime 0.001
  set now [$ns now]
  $ns at [expr $now + $intervalTime] "$ping0 send"
  $ns at [expr $now + $intervalTime] "$ping1 send"
  $ns at [expr $now + $intervalTime] "sendPingPacket"
}

#rtt = round trip time (packet travel from src to dest and back to src)
Agent/Ping instproc recv {from rtt} {
  global seq
  $self instvar node_
  puts "The node [$node_ id] received an ACK from the node $from with RTT $rtt ms"
}
$ping0 set class_ 1
$ping1 set class_ 2
$ping4 set class_ 3
$ping5 set class_ 4
#end

#======== Termination ================
#Define a 'finish' procedure

proc finish {} {
  global ns tracefile namfile
  $ns flush-trace
  close $tracefile
  close $namfile
  exec nam out.nam &
  exit 0
}

#add manually
$ns at 0.01 "sendPingPacket"
$ns at 10.0 "finish"
$ns run;   # run new simulator process
```

Program 3 - Ethernet LAN:

```
set val(stop) 10.0      ;# time of simulation end

set ns [new Simulator]  ;# create a ns object and open nam and tracefiles
set tracefile [open 5.tr w]
$ns trace-all $tracefile
set namfile [open 5.nam w]
$ns namtrace-all $namfile

set wf0 [open WinFile0 w]  ;# set winfile
set wf1 [open WinFile1 w]

proc PlotWindow {tcpSource file} { ;# plot window procedure
  global ns
  set time 0.1
  set now [$ns now]
```

```
  set cwnd [$tcpSource set cwnd_] ;#cwnd -> TCP state variable that limits the amount of data that can be sent into the network
#before receiving an ACK
  puts $file "$now $cwnd"
  $ns at [expr $now+$time] "PlotWindow $tcpSource $file"
}

set n0 [$ns node] ;# create nodes
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n0 label "Source0" ;# set labels
$n1 label "Source1"
$n2 label "R1"
$n3 label "R2"
$n4 label "Dest0"
$n5 label "Dest1"
$ns color 1 "red" ;# set colors
$ns color 2 "green"

# set wired ethernet connection - 802.3 (wireless ethernet connection - 802.11)
set lan [$ns newLan "$n0 $n1 $n2" 0.5Mb 40ms LL Queue/DropTail MAC/802_3 Channel] ;#node 0,1,2 are connected to one ethernet
$ns duplex-link $n2 $n3 10Mb 100ms DropTail
$ns duplex-link-op $n2 $n3 queuePos 0.5 ;# queue position 0.5 -> vertically appended queue (if 1 then it is a horizontally appended queue)
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/802_3 Channel] ;#node 2,3,4 are connected to one ethernet

# create error packet and send from node2 to node3 - error packet is dropped by destination node
set loss_module [new ErrorModel]
$loss_module ranvar [new RandomVariable/Uniform]
$loss_module drop-target [new Agent/Null]
$ns lossmodel $loss_module $n2 $n3

set tcp0 [new Agent/TCP] ;# tcp to node 0 and 4
$ns attach-agent $n0 $tcp0
set sink2 [new Agent/TCPSink]
$ns attach-agent $n4 $sink2
$ns connect $tcp0 $sink2
$tcp0 set packetSize_ 1500

set tcp1 [new Agent/TCP] ;# tcp to node 1 and 5
$ns attach-agent $n1 $tcp1
set sink3 [new Agent/TCPSink]
$ns attach-agent $n5 $sink3
$ns connect $tcp1 $sink3
$tcp1 set packetSize_ 1500

set ftp0 [new Application/FTP] ;#ftp to tcp0 -> node 0 and 4
$ftp0 attach-agent $tcp0
$ns at 0.1 "$ftp0 start"
$ns at 9.8 "$ftp0 stop"

set ftp1 [new Application/FTP] ;#ftp to tcp1 -> node 1 and 5
$ftp1 attach-agent $tcp1
$ns at 1 "$ftp1 start"
$ns at 9.9 "$ftp1 stop"

$ns at 0.1 "PlotWindow $tcp0 $wf0" ;#plot window procedure is called from here
$ns at 0.5 "PlotWindow $tcp1 $wf1"
$tcp0 set class_ 1
$tcp1 set class_ 2

proc finish {} { ;# finish procedure
  global ns tracefile namfile
  $ns flush-trace
  close $tracefile
  close $namfile
  exec nam 5.nam &
  exec xgraph WinFile0 WinFile1 & ;# &-> run winfile in background
  exit 0
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run`
```

Program 4 - ESS:

```
#4  Implement simple ESS and with transmitting nodes in wire-less LAN by simulation
#and determine the performance with respect to transmission of packets.
# Here we are trying to simulate the behaviour of mobile networks
```

```
# While executing the tcl file we pass a command line argument representing the number of nodes


if {$argc != 1} {
  error "Commad: <ScriptName>.tcl <Number_of_Nodes>"
  exit 0
}

set val(chan)   Channel/WirelessChannel    ;    #Creating a wireless channel
set val(prop)   Propagation/TwoRayGround   ;    #Using the TwoRayGround propagation as it provides 5.15GHz of carrier frequency
set val(netif)  Phy/WirelessPhy            ;    #Physical Link - Wireless
set val(mac)    Mac/802_11                 ;    #Ethernet - 802.11
set val(ifq)    Queue/DropTail/PriQueue    ;    #Creating a priority queue for storing the packets to send
set val(ll)     LL                         ;    #Link Layer
set val(ant)    Antenna/OmniAntenna        ;    #Omni means a 360 degree rotatable antenna (Uni-90, Bi-180)
set val(ifqlen) 50                         ;    #Limit for overflow
set val(nn)     [lindex $argv 0]           ;    #Taking the first argument of the command
set val(rp)     AODV                       ;    #Ad-hoc On-demand Distance Vector routing protocol
set val(x)      750                        ;    #Grid size
set val(y)      750                        ;
set val(stop)   100.0                      ;    #Simulation time


set ns [new Simulator]


set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)
#GOD -  stands for general operation director- To keep track of the moving nodes - contains data about the position of nodes


#Opening trace and nam files
set tracefile [open out.tr w]
$ns trace-all $tracefile


set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];               #Create wireless channel

#The below steps are to set-up the variables with the values we've defined for them earlier and are done always for wireless connections
$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) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      OFF \
                -movementTrace OFF

for {set i 0} {$i < $val(nn)} {incr i} {
  set n($i) [$ns node]
}


#setting random position for the nodes
for {set i 0} {$i < $val(nn)} {incr i} {
  set XX [expr rand()*750]
  set YY [expr rand()*750]
  $n($i) set X_ $XX
  $n($i) set Y_ $YY
}

$ns at 0.0 "destination"

#setting up initial positions
for {set i 0} {$i < $val(nn)} {incr i} {
  $ns initial_node_pos $n($i) 50
}

proc destination {} {
  global ns val n
  set now [$ns now]
  set time 3.0
  for {set i 0} {$i < $val(nn)} {incr i} {
    set XX [expr rand()*750]
    set YY [expr rand()*750]
    $ns at [expr $now + $time] "$n($i) setdest $XX $YY 20.0";
    # $XX $YY - XandY co-ordinates to place the node and 20.0 is the speed at which the nodes will change the position
```

```
  }
  $ns at [expr $now + $time] "destination"
}


set tcp0 [new Agent/TCP]
$ns attach-agent $n(0) $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n(5) $sink1
$ns connect $tcp0 $sink1
$tcp0 set packetSize_ 1500

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"



proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exec awk -f 4.awk out.tr &      # To execute the awk, file which calculates the tthroughput
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$n($i) reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

```
#4.awk

BEGIN{
  PacketRcvd=0;
  Throughput=0.0;
}
{
  if(($1=="r")&&($3=="_5_")&&($4=="AGT")&&($7=="tcp")&&($8>1000))
  {
    PacketRcvd++;
  }
}
END {
  Throughput=((PacketRcvd*1500*8)/(99.0*1000000));
  # *8 to convert the bytes to bits, and *1000000 to convert seconds to milliseconds
  # Thus we now have the throughput in Mbits/second (Mbps)
  printf "the throughput is:%f\n",Throughput;
}
```

Program 5 - GSM:

```

#===================================
# Simulation parameters setup
#===================================
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) 6 ;# number of mobilenodes
set val(rp) AODV ;# routing protocol
set val(x) 1052 ;# X dimension of topography
set val(y) 600 ;# Y dimension of topography
set val(stop) 10.0 ;# time of simulation end

#===================================
# Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo [new Topography]
```

```
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel

#===================================
# Mobile node parameter setup
#===================================
$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) \
-channel $chan \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace ON \
-movementTrace ON

#===================================
# Nodes Definition
#===================================
#Create 6 nodes
set n0 [$ns node]
$n0 set X_ 303
$n0 set Y_ 302
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 527
$n1 set Y_ 301
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 748
$n2 set Y_ 300
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 952
$n3 set Y_ 299
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 228
$n4 set Y_ 500
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 305
$n5 set Y_ 72
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20

#===================================
# Generate movement
#===================================
$ns at 2 " $n5 setdest 900 72 75 "

#===================================
# Agents Definition
#===================================
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n4 $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n5 $sink1
$ns connect $tcp0 $sink1
$tcp0 set packetSize_ 1500


#===================================
# Applications Definition
#===================================
#Setup a FTP Application over TCP connection
```

```
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 10.0 "$ftp0 stop"


#====================================
# Termination
#====================================
#Define a 'finish' procedure
proc finish {} {
global ns tracefile namfile
$ns flush-trace
close $tracefile
close $namfile
exec nam out.nam &
exec awk -f 5.awk out.tr &
exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
$ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

```
#5.awk
BEGIN{
  count1=0
  pack1=0
  time1=0
}
{
  if($1=="r" && $3=="_5_" && $4=="AGT")
  {
    count1++
    pack1=pack1+$8
    time1=$2
  }
}
END{
  printf("The Throughput from n4 to n5: %f Mbps \n", ((count1*pack1*8)/(time1*1000000)));
}
```

Program 6 - CDMA:

```
# CDMA is a coding technology that focuses/emphasis on:
# 1. How you connect the nodes?
# 2. How nodes are communicating with each other?
# 3. How data is sent to every node?
# 4. How the handover is happening?
puts "Enter number of nodes"
set tnn [gets stdin]  ;#Accpets input from standard input
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(x) 1500             ;#add manually   -> X dimension of topography
set val(y) 1500             ;#add manually   -> Y dimension of topography
set val(ifqlen) 1000        ;#add manually   -> max packet in ifq
set val(adhocRouting) AODV  ;#add manually   -> routing protocol
set val(nn) $tnn                       ; # number of mobilenodes
set val(stop) 10.0
#add manually
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)
#end

# To determine the performance of this wireless cellular network, we calculate three parameters like
# throughput, packet loss and end-to-end delay.
# Here we have 3 files to determine the above said parameters:
```

```
# 1. f0: contains trace-file information, I,e received packets.
# 2. f1: stores the information about the packet loss.
# 3. f2: stores the information about end-to-end delay.
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)

# Here we are creating grid using X, Y value.
$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]
$ns_ color 0 red

# Node configuration
$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 $topo \
 -agentTrace ON \
 -routerTrace ON \
 -macTrace OFF
#add manually
for {set i 0} {$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"

# Since we are generating CBR traffic (UDP connection) , we use 3 Variables to hold the values of
# previous iteration.
# 1. holdtime : stores the last packet arrival time.
# 2. holdseq : stores the no. of packets (count) received.
# 3. holdrate: sores the no. of bytes received.
set holdtime 0
set holdseq 0
set holdrate1 0

#start
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

    # Again we are going to define 4 more variables that stores the values of current iteration, they are
    # 4. bw0 : holds the no, of bytes transmitted in the current iteration/time.
    # 5. bw1 : holds the no, of packets lost during current iteration.
    # 6. bw2 : this variable is going to hold/store the current time of the last packet received of the
    # current iteration.
    # 7. bw3 : holds the no. of packets received during current iteration.

    set bw0 [$sink set bytes_]
    set bw1 [$sink set nlost_]
    set bw2 [$sink set lastPktTime_]
    set bw3 [$sink set npkts_]

    # The 7 variables keep updating for every iteration during simulation where holdtime, holdseq and
    # holdrate gets updated with the previous values and the variables bw0 , bw1 , bw2 , bw3 holds the
    # current values.
    # With these variable’s values, we calculate the 3 performance parameters such as throughput,
    # packet loss rate and end–to-end delay

    set now [$ns now]
    # Record Bit Rate/ Throughput in Trace Files
    # Throughput: (successful packet received) / time in Mbps (mega bits/sec)
    # bw0 -> Current bytes received
    # holdrate1 -> Previous bytes received
    # 8 -> Converting bytes to bits
    # time -> bw0 time and holdrate time
    # 1000000 -> Converting to Megabits/sec
    puts $f0 "$now [expr (($bw0+$holdrate1)*8)/(2*$time*1000000)]"

    # Record Packet Loss Rate in File
    # packet loss rate will be measured by taking the no. of packets lost / sec
    # bw1 -> holds the no. of packet lost in that current time/iteration
    puts $f1 "$now [expr $bw1/$time]"

    # Record end-to-end delay:  the amount of time the required for packets to be
    # transmitted from source to destination.
    # calculate the average delay of each packet within that time interval.
    # bw2  -> current time of the last packet received.
    # holdrate -> previous time of the last packet received.
    # bw3 -> no. of packets received at the current time.
    # holdseq -> no. of packets received during previous iteration.

    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
}
#end


# Start Recording at Time 0
$ns_ at 0.0 "record"
source link.tcl

# A graph for throughput values, packet loss rates and end-to- end delays against the
# time is plotted using xgraph API.
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\""
```

PART B:

Program 7 - CRC-CCITT (16- bits) :

```
import java.util.*;
class crc
{
    void div(int[] a,int k) //crc 16bit divison
    {
        int[] gp={1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1}; //intial expression
        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++]; //long division
               }
               count=0;
            }
        }
   }
public static void main(String args[])
{
   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 (enter 1 bit per line):");
   for(int i=0;i<len;i++)
     a[i]=sc.nextInt();        //user's message
   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);                    //division operation on client's side
   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]+" ");    //message sent from client's side
    System.out.println();
    System.out.println("Enter the Received Data: ");
    for(int i=0;i<len;i++)
      a[i]=sc.nextInt();        //message received on server's side
    ob.div(a, k);           //divison operation on server's side
    for(int i=0;i<len;i++)
    {
      if(a[i]!=0)
      {
        flag=1;         //after divison if the message contains 1 then, there is an error in the message
        break;
      }
    }
    if(flag==1)
      System.out.println("error in data");
    else
      System.out.println("no error");
    sc.close();
  }
}
```

Program 8 - bellman-ford algorithm:

```
import java.util.*;
public class bellman_ford{
  public static void main(String args[]){
    Scanner in = new Scanner(System.in);
    int graph[][] = new int [100][100];
    //Enter number of nodes
    System.out.println("Enter no of nodes: ");
    int n = in.nextInt();
    //Enter Adjacency Matrix of the Graph
    System.out.println("Enter Adjacency Matrix: ");
    for(int j=0;j<n;j++){
      for(int k=0;k<n;k++){
          graph[j][k] = in.nextInt();
      }
    }
    //Enter Source Node
    System.out.println("Enter Source node: ");
    int src = in.nextInt();
    //Calculate Distance from src to all other nodes
    calc_distance(graph,n,src);
  }
  public static void calc_distance(int[][] g,int n,int src){
    int i=0,j=0,k=0;
    int d[] = new int[n];
    //Initialize Distance vector to all destination nodes 999
    for(i=0;i<n;i++)
      d[i] = 999;
    d[src] = 0;
    //Bellman Ford Algorithm
    //Run outer loop n-1 times
    for(i=0;i<n-1;i++){
      for(j=0;j<n;j++){
        for(k=0;k<n;k++){
          //Bellman Ford Algorithm condition
          if (d[k] > d[j] + g[j][k])
          d[k] = d[j]+g[j][k];
        }
      }
    }
  //Checking for a negative cycle. If the above iteration runs for the nth time and the resulting
  //distance vector is different from the distance vector obtained in the (n-1)th iteration,
  //the graph is said to have a negative cycle. Bellman Ford Algorithm fails when the graph has a negative cycle.
    int c[] = new int[n];
    c = d;
    for(j=0;j<n;j++){
      for(k=0;k<n;k++){
        if (g[j][k]!=999 && c[k] > c[j] + g[j][k]){
          System.out.println("Graph contains a negative Cycle. Bellman Ford Algorithm Cannot be Applied");
          return;
        }
      }
    }
//Print the distance to each destination node from the source node.
    for(i=0;i<n;i++){
      if(i==src)
      continue;
      System.out.println(src+"->"+i+"="+d[i]);
    }
    return;
  }
}
```

Program 9 - TCP Client Server:

```
//Prog-9 TCP-client
import java.net.*;
import java.io.*;
import java.io.FileWriter;
import java.util.Scanner;

public class Client
{
  public static void main(String args[]) throws Exception
  {
    int Entry = 1;
    Scanner Read = new Scanner(System.in);
```

```
    System.out.println("Enter Server Address: ");
    //127.0.0.1 denotes the address of the current system, i.e.
    //irrespective of the actual IP address of the system,
    //127.0.0.1 will always point towards the system that is executing the program.
    String address = Read.nextLine();
    while (Entry == 1)
    {
      Socket sock = new Socket(address, 5119);
      System.out.println("Enter file name to send. Enter 'exit' to exit");
      System.out.print("Client /> ");
      String Command = Read.nextLine();           //Reading the file name here
      if (Command.equals("exit"))
      {
        Entry = 0;
        break;
      }
      OutputStream ostream = sock.getOutputStream();       //Setting up the output streams
      PrintWriter pwrite = new PrintWriter(ostream, true);
      pwrite.println(Command);

      InputStream istream = sock.getInputStream();
      BufferedReader socketRead = new BufferedReader(new InputStreamReader(istream));
      String FileContent="";
      String Temp;

      //Reading file till end of content and storing the contents on in FileContent string variable
      while ((Temp = socketRead.readLine()) != null)
      {
        pwrite.println(Temp);
        FileContent+=Temp;
      }
      System.out.println("File : "+Command+" Received.");
      //Setting the name of the file to "Client" + command(user's input-file name)
      FileWriter Writer = new FileWriter("Client"+Command);  //argument:new file name
      Writer.write(FileContent);      //Writing the filecontents on to the new file
      Writer.close();
      pwrite.close();
      socketRead.close();
      sock.close();

    }
    Read.close();
  }
}
```

```
//Prog-9 TCP-server

import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

public class Server
{
  public static void main(String args[]) throws Exception
  {
    int data = 1;
    ServerSocket sersock = new ServerSocket(5119);
    while(data == 1)
    {
      System.out.println("Server ready for connection....");
      Socket sock = sersock.accept();                  //Server Socket waiting to accept requests
      System.out.println("Connection Established\nWaiting for Client Request.");
      InputStream istream = sock.getInputStream( );
      BufferedReader br =new BufferedReader(new InputStreamReader(istream));
      String fname = br.readLine();
      if(fname.equals("exit"))
      {
         continue;
      }
      BufferedReader contentRead = new BufferedReader(new FileReader(fname));
      OutputStream ostream = sock.getOutputStream( );
      PrintWriter pwrite = new PrintWriter(ostream, true);
      String str;
      while((str = contentRead.readLine()) != null)          //Reads the file and writes using writer object pwrite
      {
        pwrite.println(str);
      }
      System.out.println("\nFile Contents sent successfully\n\n");
      pwrite.close();
      br.close();
      contentRead.close();
      sock.close();
    }
```

```
    sersock.close();
  }
}
```

Program 10 - UDP Client Server:

```
// Java program to illustrate UDP Client side
import java.io.*;
import java.net.*;
class udpclient
{
  public static DatagramSocket clientsocket;
  public static DatagramPacket dp;
  public static BufferedReader br;
  public static InetAddress ia;
  public static byte buf[] = new byte[1024];
  public static int cport = 3000, sport = 8000;
  public static void main(String[] args) throws IOException
  {
    clientsocket = new DatagramSocket(cport);                //create socket
    dp = new DatagramPacket(buf, buf.length);                //data packet
    br = new BufferedReader(new InputStreamReader(System.in));
    ia = InetAddress.getLocalHost();
    System.out.println("Client is Running...");
    System.out.println("Type some text if you want, else 'exit' to quit");
    while(true)
    {
      String str1 = new String(br.readLine());
      buf = str1.getBytes();               // string is converted into bytes format
      if(str1.equals("exit"))          // data = exit then terminates connection
      {
        System.out.println("Terminated..");
        clientsocket.send(new DatagramPacket(buf,str1.length(),ia,sport));
        break;
      }
      clientsocket.send(new DatagramPacket(buf,str1.length(), ia, sport));  // sends data packet
      clientsocket.receive(dp);                                             //recieves data packet from server
      String str4 = new String(dp.getData(), 0, dp.getLength());
      System.out.println("Server said : " + str4);
    }
  }
}
```

```
// Java program to illustrate UDP Server side
import java.io.*;
import java.net.*;
class udpserver
{
  public static DatagramSocket serversocket;
  public static DatagramPacket dp;
  public static BufferedReader br;            // bufferedReader class - read the text from a character-based input stream
  public static InetAddress ia;
  public static byte buf[] = new byte[1024];
  public static int cport = 3000,sport=8000;
  public static void main(String[] args) throws IOException
  {
    serversocket = new DatagramSocket(sport);                     //create new socket
    dp = new DatagramPacket(buf,buf.length);
    br = new BufferedReader (new InputStreamReader(System.in));   // entered input stream is read by bufferedReader
    ia = InetAddress.getLocalHost();            // class returns the instance of InetAddress containing local host name and address
    System.out.println("Server is Running...");
    while(true)
    {
      serversocket.receive(dp);           // recieves datapkt from client
      String str2 = new String(dp.getData(), 0, dp.getLength());  // get data
      if(str2.equals("exit"))
      {
        System.out.println("Terminated...");
        break;
      }
      System.out.println("Client said : " + str2);      // display msg
      String str3 = new String(br.readLine());      // reads line of a text
      buf = str3.getBytes();            // into bytes
      serversocket.send(new DatagramPacket(buf,str3.length(), ia, cport));  // send data packet to client
    }
  }
}
```

Program 11 - RSA algorithm:

```
import java.util.*;
public class prog11
{
  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,q,n,e,d,phi,i,j=0;
    int[] num=new int[100];
    int[] encrypted=new int[100];
    int[] decrypted=new int[100];
    String message ;

    Scanner sc=new Scanner(System.in);

    System.out.println("Enter the message to be encrypted :");
    message = sc.nextLine();
    System.out.println("\nEnter value of p and q : ");
    p=sc.nextInt();
    q=sc.nextInt();

    n=p*q;
    phi=(p-1)*(q-1);

    for(e=2;e<phi;e++)
    {
      if(gcd(e,phi)==1)
        break;
    }
    for(i=2;i<phi;i++)
    {
      if((e*i)%phi==1)
        break;
    }
    d=i;
    for(i=0;i<message.length();i++)
    {
      char c = message.charAt(i);
      int a = (int)c;
      num[i] = a-97; //97 is any number to add or subtract
    }

    for(i=0;i<message.length();i++)
    {
      encrypted[i]=1;
      for(j=0;j<e;j++)
        encrypted[i] =(encrypted[i]*num[i])%n;
    }

    System.out.println("\nEncrypted message in nums: ");
    for(i=0;i<message.length();i++)
    {
      System.out.print(encrypted[i]);
    }
    System.out.println("\n\nEncrypted message in characters: ");
    for(i=0;i<message.length();i++)
    {
      System.out.print((char)(encrypted[i]+97));
    }
    for(i=0;i<message.length();i++)
    {
      decrypted[i]=1;
      for(j=0;j<d;j++)
        decrypted[i]=(decrypted[i]*encrypted[i])%n;
    }
    System.out.println("\n\nDecrypted message: ");
    for(i=0;i<message.length();i++)
    {
      System.out.print((char)(decrypted[i]+97));
    }
    sc.close();
```

```
  }
}
```

Program 12 - leaky bucket algorithm:

```
import java.util.*;
public class Leaky_Bucket{
    public static void main(String args[]){
        Scanner in = new Scanner(System.in);
        System.out.println("Enter Buffer Size and Output Rate: ");
        int buffer = in.nextInt();
        int out_rate = in.nextInt();
        int curr_storage=0; // No. of bytes stored currently in the buffer.
        int overflow = 0;   // No. of bytes overflowing
        int remaining_buffer = 0;  // Remaining space in the buffer after the incoming bytes are stored and outgoing bytes are removed
        int req_storage = 0; // total storage required to store the current bytes in the buffer and the incoming bytes.
        int option = 1;
        while(option>0){
            System.out.println("Enter Input Rate: ");
            int inp_rate = in.nextInt();
            if(buffer < curr_storage + inp_rate){
            // Condition for the scenario when the required storage exceeds the buffer size (Overflow condition)
                req_storage = curr_storage+inp_rate;
                overflow = req_storage-buffer;
                curr_storage = req_storage - overflow - out_rate;
            }else{
            // Condition for the scenario when the required storage does not exceed the buffer size (No Overflow condition)
                req_storage = curr_storage + inp_rate;
                overflow = 0;
                curr_storage = req_storage - out_rate;
            }
            System.out.println("Input Rate: "+inp_rate+"\tTotal Required Storage: "+
            req_storage+"\tOverflow: "+overflow+"\tCurrent Storage after transmission: "+curr_storage);
            System.out.println("Enter 0 to exit or 1 to continue: ");
            option = in.nextInt();
        }

    }
}
```