RAMCO INSTITUTE OF TECHNOLOGY
Approved by AICTE, New Delhi & Affiliated to Anna University
Accredited by NAAC & An ISO 9001:2015 Certified Institution
NBA Accredited UG Programs: CSE, EEE, ECE and MECH

Department of Artificial Intelligence and Data Science
Academic Year 2022-2023
CS3591 - Computer Networks

01.04.2023

NETWORK SIMULATOR (NS-2)

MANUAL

Prepared by : Dr.M.Kaliappan, Professor/AI & DS

NS-2 INSTALLATION

- Copy ns-allinone-2.34.tar.gz package to **/root** folder
- Right click & do **extract here**
  - It will create ns-allinone-2.34 folder
- Open the terminal and go to ns-allinone-2.34 folder
  - **cd /root/ ns-allinone-2.34**
- To install type the command
  - **./install**
- Open another terminal : set the environment variable open the /etc/profile file
  - **gedit /etc/profile**
- Enter these variable in end of profile
  - **export PATH=$PATH**
  - **export LD_LIBRARY_PATH=**
  - **export TCL_ LIBRARY=**

From this output itself we have to copy the value of PATH, LD_LIBRARY_PATH, TCL_LIBRARY environment variables and paste in /etc/profile file.

Example:

- **set the environment variable**
  - **export PATH=$PATH:/root/ns-allinone-2.34/tcl8.4.15/unix:/root/ns-allinone-2.34/tk8.4.15/unix: /root/ns-allinone-2.34/bin**
  - **export LD_LIBRARY_PATH=/root/ ns-allinone-2.34/lib: /root/ns-allinone-2.34/otcl-1.13**
  - **export TCL_ LIBRARY=/root/ns-allinone-2.34/tcl8.4.15/library**

WIRED SIMULATION - Exercise

hello.tcl

```
set ns [new Simulator]
$ns at 1 "puts \"Hello World!!\""
$ns at 1.5 "exit"
$ns run
```

extcl.tcl

```
# Writing a procedure called "test"
proc test {} {
    set a 43
    set b 27
    set c [expr $a + $b]
    set d [expr [expr $a - $b] * $c]
    for {set k 0} {$k < 10} {incr k} {
     if {$k < 5} {
         puts "k < 5, pow = [expr pow($d, $k)]"
     } else {
         puts "k >= 5, mod = [expr $d % $k]"
     }
    }
}
# Calling the "test" procedure created above
```

test

exotcl.tcl

```
# Create a class call "mom" and
# add a member function call "greet"
Class mom
mom instproc greet {} {
    $self instvar age_
    puts "$age_ years old mom say:
    How are you doing?"
}

# Create a child class of "mom" called "kid"
# and overide the member function "greet"
Class kid -superclass mom
kid instproc greet {} {

    $self instvar age_
    puts "$age_ years old kid say:
    What's up, dude?"
}

# Create a mom and a kid object set each age
set a [new mom]
$a set age_ 45
set b [new kid]
$b set age_ 15

# Calling member function "greet" of each object
$a greet
$b greet
```

---

ex1.tcl

```
#Create a simulator object
set ns [new Simulator]

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

#Define a 'finish' procedure
proc finish {} {
        global ns nf
        $ns flush-trace
     #Close the trace file
        close $nf
     #Execute nam on the trace file
```

```
        exec nam out.nam &
        exit 0
}

#Create two nodes
set n0 [$ns node]
set n1 [$ns node]


#Create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail

#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0


# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0


#Create a Null agent (a traffic sink) and attach it to node n1
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0


#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0


#Schedule events for the CBR agent
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"


#Run the simulation
$ns run
```

---

ex2.tcl

```
#Create a simulator object
set ns [new Simulator]

#Define different colors for data flows
$ns color 1 Blue
$ns color 2 Red


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



#Define a 'finish' procedure
proc finish {} {
        global ns nf
        $ns flush-trace
     #Close the trace file
        close $nf
     #Execute nam on the trace file
        exec nam out.nam &
        exit 0

}

#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]


#Create links between the nodes
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms SFQ

$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

#Monitor the queue for the link between node 2 and node 3
$ns duplex-link-op $n2 $n3 queuePos 0.5


#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
```

```
$udp0 set class_ 1
$ns attach-agent $n0 $udp0


# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0


#Create a UDP agent and attach it to node n1
set udp1 [new Agent/UDP]
$udp1 set class_ 2
$ns attach-agent $n1 $udp1


# Create a CBR traffic source and attach it to udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

#Create a Null agent (a traffic sink) and attach it to node n3
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0


#Connect the traffic sources with the traffic sink
$ns connect $udp0 $null0
$ns connect $udp1 $null0

#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Run the simulation
$ns run
```

---

ex3.tcl

```
#Create a simulator object
set ns [new Simulator]
```

```
#Tell the simulator to use dynamic routing
$ns rtproto DV


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


#Define a 'finish' procedure
proc finish {} {

        global ns nf
        $ns flush-trace
     #Close the trace file
       close $nf
     #Execute nam on the trace file
        exec nam out.nam &
        exit 0
}

#Create seven nodes
for {set i 0} {$i < 7} {incr i} {
        set n($i) [$ns node]
}

#Create links between the nodes
for {set i 0} {$i < 7} {incr i} {
        $ns duplex-link $n($i) $n([expr ($i+1)%7]) 1Mb 10ms
DropTail
}

#Create a UDP agent and attach it to node n(0)
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0



# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

```
#Create a Null agent (a traffic sink) and attach it to node
n(3)
set null0 [new Agent/Null]
$ns attach-agent $n(3) $null0

#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0

#Schedule events for the CBR agent and the network dynamics
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n(1) $n(2)
$ns rtmodel-at 2.0 up $n(1) $n(2)
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

---

mcast.tcl

```
set ns [new Simulator]
$ns multicast
set f [open out.tr w]
$ns trace-all $f
$ns namtrace-all [open out.nam w]

$ns color 1 red
# prune/graft packets
$ns color 30 purple
$ns color 31 green
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]


# Use automatic layout
$ns duplex-link $n0 $n1 1.5Mb 10ms DropTail
$ns duplex-link $n1 $n2 1.5Mb 10ms DropTail
$ns duplex-link $n1 $n3 1.5Mb 10ms DropTail

$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n1 $n3 orient right-down
$ns duplex-link-op $n0 $n1 queuePos 0.5

set mrthandle [$ns mrtproto DM {}]
```

```
set group1 [Node allocaddr]
set cbr0 [new Application/Traffic/CBR]
set udp0 [new Agent/UDP]
$cbr0 attach-agent $udp0
$ns attach-agent $n1 $udp0
$udp0 set dst_addr_ $group1
$udp0 set dst_port_ 0

set group2 [Node allocaddr]
set cbr1 [new Application/Traffic/CBR]
set udp1 [new Agent/UDP]
$cbr1 attach-agent $udp1
$udp1 set dst_addr_ $group2
$udp1 set dst_port_ 0
$udp1 set class_ 1
$ns attach-agent $n3 $udp1

set rcvr [new Agent/LossMonitor]
#$ns attach-agent $n3 $rcvr
$ns at 1.2 "$n2 join-group $rcvr $group2"
$ns at 1.25 "$n2 leave-group $rcvr $group2"
$ns at 1.3 "$n2 join-group $rcvr $group2"
$ns at 1.35 "$n2 join-group $rcvr $group1"

$ns at 1.0 "$cbr0 start"
$ns at 1.1 "$cbr1 start"

$ns at 2.0 "finish"
proc finish {} {

     global ns
     $ns flush-trace
     puts "running nam..."
     exec nam out.nam &
     exit 0
}

$ns run
```

Graph

graph1.tcl

```
exec xgraph -m -bb  -bg white -tk -P  -t " Maximal Energy Vs
No of Nodes" -x "No of Nodes" -y "Maximal Energy(Joule)" PCLQ
PCRE ECTR -geometry 800X400
```

---

exgraph.tcl

```
#Create a simulator object

set ns [new Simulator]


#Open the output files
set f0 [open out0.tr w]
set f1 [open out1.tr w]
set f2 [open out2.tr w]

#Create 5 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]


#Connect the nodes
$ns duplex-link $n0 $n3 1Mb 100ms DropTail
$ns duplex-link $n1 $n3 1Mb 100ms DropTail
$ns duplex-link $n2 $n3 1Mb 100ms DropTail

$ns duplex-link $n3 $n4 1Mb 100ms DropTail

#Define a 'finish' procedure

proc finish {} {

     global f0 f1 f2

     #Close the output files

     close $f0

     close $f1

     close $f2

     #Call xgraph to display the results

     exec xgraph out0.tr out1.tr out2.tr -geometry 800x400 &

        exit 0
```

```
}


#Define a procedure that attaches a UDP agent to a previously
created node
#'node' and attaches an Expoo traffic generator to the agent
with the
#characteristic values 'size' for packet size 'burst' for burst
time,
#'idle' for idle time and 'rate' for burst peak rate. The
procedure connects
#the source with the previously defined traffic sink 'sink'
and returns the
#source object.

proc attach-expoo-traffic { node sink size burst idle rate }
{

     #Get an instance of the simulator

     set ns [Simulator instance]

     #Create a UDP agent and attach it to the node
     set source [new Agent/UDP]
     $ns attach-agent $node $source

     #Create an Expoo traffic agent and set its configuration
parameters
     set traffic [new Application/Traffic/Exponential]
     $traffic set packetSize_ $size
     $traffic set burst_time_ $burst
     $traffic set idle_time_ $idle
     $traffic set rate_ $rate



        # Attach traffic source to the traffic generator

        $traffic attach-agent $source

     #Connect the source and the sink

     $ns connect $source $sink

     return $traffic

}
```

```
#Define a procedure which periodically records the bandwidth
received by the

#three traffic sinks sink0/1/2 and writes it to the three files
f0/1/2.

proc record {} {

        global sink0 sink1 sink2 f0 f1 f2

     #Get an instance of the simulator

     set ns [Simulator instance]

     #Set the time after which the procedure should be called
again

        set time 0.5

     #How many bytes have been received by the traffic sinks?

        set bw0 [$sink0 set bytes_]

        set bw1 [$sink1 set bytes_]

        set bw2 [$sink2 set bytes_]

#Get the current time

        set now [$ns now]

#Calculate the bandwidth (in MBit/s) and write it to the files

        puts $f0 "$now [expr $bw0/$time*8/1000000]"

        puts $f1 "$now [expr $bw1/$time*8/1000000]"

        puts $f2 "$now [expr $bw2/$time*8/1000000]"

#Reset the bytes_ values on the traffic sinks

        $sink0 set bytes_ 0

        $sink1 set bytes_ 0

        $sink2 set bytes_ 0

#Re-schedule the procedure
```

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

#Create three traffic sinks and attach them to the node n4
set sink0 [new Agent/LossMonitor]
set sink1 [new Agent/LossMonitor]
set sink2 [new Agent/LossMonitor]
$ns attach-agent $n4 $sink0
$ns attach-agent $n4 $sink1
$ns attach-agent $n4 $sink2

#Create three traffic sources
set source0 [attach-expoo-traffic $n0 $sink0 200 2s 1s 100k]
set source1 [attach-expoo-traffic $n1 $sink1 200 2s 1s 200k]
set source2 [attach-expoo-traffic $n2 $sink2 200 2s 1s 300k]

#Start logging the received bandwidth
$ns at 0.0 "record"
#Start the traffic sources
$ns at 10.0 "$source0 start"
$ns at 10.0 "$source1 start"
$ns at 10.0 "$source2 start"
#Stop the traffic sources
$ns at 50.0 "$source0 stop"
```

```
$ns at 50.0 "$source1 stop"

$ns at 50.0 "$source2 stop"

#Call the finish procedure after 60 seconds simulation time

$ns at 60.0 "finish"


#Run the simulation

$ns run
```

LAB – 1 TCL COMMANDS

```
set a 10
set b 15
puts $a
puts "the value of a = $a"
puts "the value of b = $b"
……..
set c [expr $a + $b]
puts "the value of c =$c"
set d [expr [expr $b-$a] * $c]
puts "the value of d=$d"
……
proc display { } {
puts "this is the testing message"
}
display
……
```

```
proc add {x y} {
set z [expr $x+$y]
puts "the value of $x + $y=$z"
}
add 10 20
……..
proc print {k} {
for {set i 0} {$i < $k} {incr i} {
      puts "node_($i)"
      puts "n$i"
}
}
print 10
……
set test [open file1.txt w]
puts $test "Testing message"
close $test
……
set test1 [open file1.txt r]
while { [ eof $test1]==0} {
gets $test1 line
puts $line
}
```

WIRELESS LAB – 1 NODE CREATION

```
set val(x)              500 ;# X dimension of the topography
set val(y)              500 ; # y dimension of the topography
set val(nn)             3 ; # how many nodes
set val(stop)           200.0 ; # simulation time
set val(routing)        AODV
set qtype               Queue/DropTail/PriQueue
# set qtype             CMUPriQueue


set ns_           [new Simulator]
set topo          [new Topography]
$topo load_flatgrid  $val(x)  $val(y)
set tracefd       [open wireless.tr w]


$ns_     trace-all $tracefd
$ns_     use-newtrace
set namtrace      [open wireless.nam w]
$ns_   namtrace-all-wireless $namtrace $val(x) $val(y)

Set god_ [create-god $val(nn)]
#define how node should be created – global node setting
$ns_   node-config      - adhocRouting $val (routing) \
                        - llType LL \
                        - macType Mac/802_11 \
                        - ifqType $qtype \
                        - ifqLen 50 \
                        - antType Antenna/OmniAntenna \
                        - propType Propagation/TwoRayGround \
                        - phyType Phy/WirelessPhy \
                        - channelType Channel/WirelessChannel\
                        - topoInstance $topo \
                        - agentTrace ON \
                        - routerTrace ON \
                        - macTrace OFF

#create the specified number of nodes

for {set i 0 } {$i < val(nn)}{incr i} {
     set node_($i) [$ns_ node]
}
```

```
#Define node position

for {set i 0}{$i < $val(nn)}{incr i}{
       $ns_  initial_node_pos $node_($i) 30
}
$ns_ at $val(stop).000 "$ns_  halt"
$ns_ run
```

WIRELESS LAB – 2 MOBILE NODE POSITION

#Static (Fixed) Topology

```
$node_(0)   set  X_ 50.0
$node_(0)   set  Y_ 50.0
$node_(0)   set  Z_ 0.0

$node_(1)   set  X_150.0
$node_(1)   set  Y_50.0
$node_(1)   set  Z_0.0

$node_(2)   set  X_300.0
$node_(2)   set  Y_50.0
$node_(2)   set  Z_0.0
```

WIRELESS LAB – 3 UDP DATA TRAFFIC

```
#create UDP Source
set udp0 [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp0

#create UDP Destination
set null0 [new Agent/Null]
$ns_ attach-agent $node_(0) $null0

#connecting  UDP source & Destination
$ns_ connect $udp0 $null0

#create application traffic
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 512
$cbr0 set rate_ 100Kb
#$cbr0 set interval_ 0.1
#Application start time
$ns_ at 10.0 "$cbr0 start"
```

#Application stop time

```
$ns_ at 50.0 "$cbr0 stop"
```

WIRELESS LAB – 4 TCP DATA TRAFFIC

#create TCP Source

```
set tcp0 [new Agent/TCP]
$ns_ attach-agent $node_(0) $tcp0
```

#create TCP Destination

```
set sink0 [new Agent/TCPSink]
$ns_ attach-agent $node_(1) $sink0
$ns_ connect $tcp0 $sink0
```

#create application traffic

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

#Application start & stop time

```
$ns_ at 60.0 "$ftp0 start"
$ns_ at 90.0 "$ftp0 stop"
```

WIRELESS LAB – 5 MOVEMENT GENERATION

#Mobile Node Manual Movement

```
$ns_ at 3.0 "$node_(2) setdest 450 100 50"
$ns_ at 3.0 "$node_(1) setdest 250 100 50"
```

#Random Topology

#To apply random movement we need **setdest** command

#Syntax

setdest **–n** number of node **–p** pause time **–M** max_speed **–t** total_simulation_time **–x** x-axis-value **–y** y-axis-value

#Example command

```
setsest> ./setdest -n 20 -p 10 -M 20 -t 100 -x 800 -y 800 >scen-20-20

setdest -n 30 -p 10 -M 20 -t 100 -x 800 -y 800 >scen-30-20

setdest -n 40 -p 10 -M 20 -t 100 -x 800 -y 800 >scen-40-20
```

Traffic generator and run the cbrgen.tcl

Go to cmu-scen-gen folder which is in the following folder ns-allinone-2.34/ns-2.34/indep-utils/cmu-scen-gen

ns cbrgen.tcl -type cbr -nn 20 -seed 8 -mc 15 -rate 4.5 >cbr-20-15

Then store the traffic generator and scenario files into nsallinone-2.34/ns-2.34/tcl/mobility/scene folder

WIRELESS LAB – 6 Trace File Analysis

- New wireless Trace format
  - s –t 163.001503520 –Hs 0 –Hd -2 –Ni 0 –Nx 300.00 –Ny 500.00 –Nz 0.0 –Ne -1.000000 –Nl AGT –Nw --- –Ma 0 –Md 0 –Ms 0 –Mt 0 –Is 0.0 –Id 2.0 –It cbr –Il 200 –If 1 –Ii 77 –Iv 32 –Pn cbr –Pi 32 –Pf 0 –Po 0

- Field 0: Event type
  - s:send r:receive d:drop f:forward
- Field 1: General tag
  - -t: time
- Field 3: Next hop info
  - -Hs: id for this node
  - -Hd: id for next hop towards the destination

Field 4: Node property type tag

-Ni: node id

-Nx –Ny –Nz: node's x/y/z coordinate

-Ne: Node energy level

-Nl: trace level, such as AGT, RTR, MAC

-Nw: reason for the event

Field 5: Packet info at MAC level

-Ma: duration

-Md: dest's Ethernet address

-Ms: src's Ethernet address

-Mt: Ethernet type

Field 6: Packet information at IP level

-Is: source address, source port number

-Id: dest address, dest port number

-It: packet type

-Il: packet size

-If: flow id

-Ii: unique id

-Iv: TTL value

Field 7:

- Packet info at "Application level" which consists of the type of application like ARP, TCP, the type of Adhoc routing protocol like DSDV, DSR, AODV etc. The field consists of a leading **–P** and the list of tags for different applications.

RUN AWK SCRIPT

```
#gawk –f wireless.awk wireless.tr
```

WIRELESS LAB – 7 – protocol comparison

Performance comparison of Adhoc routing protocol (AODV, DSDV, DSR)

1. **By varying number of mobiles nodes**

   By varying the number of nodes in the given topology area. Analyze the protocols behavior.

   Topology area – 800m x 800m, Max speed – 20ms, Pause time – 10s, Simulation time 200secs. Create UDP Data source for node 5 to node 10.

AODV

No of nodes	Packet Delivery Ratio(PDR)	End-to-end delay	Overhead
20			
30			
40			
50			

DSDV

No of nodes	Packet Delivery Ratio(PDR)	End-to-end delay	Overhead
20			

30			
40			
50			

DSR

No of nodes	Packet Delivery Ratio(PDR)	End-to-end delay	Overhead
20			
30			
40			
50			

creating scenario for various no of mobile nodes

```
setdest  -n 20 -p 10 -m 20 -t 200 -x 800 -y >scen-20-20

setdest  -n 30 -p 10 -m 20 -t 200 -x 800 -y >scen-30-20

setdest  -n 40 -p 10 -m 20 -t 200 -x 800 -y >scen-40-20

setdest  -n 50 -p 10 -m 20 -t 200 -x 800 -y >scen-50-20
```

2. By varying speed of the mobiles nodes

By varying speed of the mobile nodes in the given topology area and for fixed number of mobile nodes analyse the protocols behavior.

Topology area – 800m x 800m, Number of nodes – 50, pause time – 10s, Simulation Time 200secs. Create UDP data source for node 5 to node 10.

AODV

Speed (m/s)	Packet Delivery Ratio(PDR)	End-to-end delay	Overhead
5			
10			
15			
20			

DSDV

Speed (m/s)	Packet Delivery Ratio(PDR)	End-to-end delay	Overhead
5			

10			
15			
20			

DSR

Speed (m/s)	Packet Delivery Ratio(PDR)	End-to-end delay	Overhead
5			
10			
15			
20			

#creating scenario for various Mobile speed-No.of.nodes as 50

```
setdest  -n 50 -p 10 -m 5 -t 200 -x 800 -y 800 >scen-50-5

setdest  -n 50 -p 10 -m 10 -t 200 -x 800 -y 800 >scen-50-10

setdest  -n 50 -p 10 -m 15 -t 200 -x 800 -y 800 >scen-50-15

setdest  -n 50 -p 10 -m 20 -t 200 -x 800 -y 800 >scen-50-20
```

Xgraph

```
xgraph –x “No.of.Nodes” –y “End-to-End Delay” –t “title”
–geometry “600 x 600” -lw 2 aodv_pdr  dsdv_pdr
```

WIRELESS LAB – 8 ENERGY MODEL

```
$ns_   node-config      - adhocRouting $val (routing) \
                        - llType LL \
                        - macType Mac/802_11 \
                        - ifqType Queue/DropTail/PriQueue \
                        - ifqLen 100 \
                        - antType Antenna/OmniAntenna \
                        - propType Propagation/TwoRayGround \
                        - phyType Phy/WirelessPhy \
                        - channelType Channel/WirelessChannel\
                        - energyModel EnergyModel \
                        - rxPower 0.3 \
                        - txPower 0.6  \
                        - initialEnergy 1 \
                        - topoInstance $topo \
                        - agentTrace ON \
                        - routerTrace ON \
```

NS-2 Modification

LAB - 1 ADD AGENT

Create the testagent to understand the behavior of C++ and otcl interaction.

Write your C++ code test.cc in ns-allinone-2.34/ns-2.34/**test** folder

Test.cc

```
#include<stdio.h>
#include<string.h>
#include "agent.h"

Class TestAgent : public Agent  {
public:
          TestAgent ();
protected:
          int command(int argc,const char*const* argv);
private:
           int var1;
           double var2;
           void test_function(void);
};

static class TestAgentClass : public TclClass   {
public:
        TestAgentClass() : TclClass("Agent/TestAgentOtcl") {}
        TclObject* create(int, const char*const*)  {
        return(new TestAgent());
   }
} class_test_agent;
```

```
TestAgent::TestAgent() : Agent(PT_UDP) {
bind ("var1_otcl", &var1);
bind ("var2_otcl", &var2);
}

int TestAgent::command(int argc, const char*const* argv) {
if(argc == 2) {
          if(strcmp(argv[1], "call_function") == 0) {
          test_function();
          return(TCL_OK);
   }
}
if(argc == 4) {
     if(strcmp(argv[1], "add") == 0) {
     int x= atoi(argv[2]) + atoi(argv[3]);
     printf("\nThe result of ADD %d",x);
     return(TCL_OK);
   }
}
   return(Agent::command(argc,argv));
}

void TestAgent::test_function(void) {
printf("\nThe value of var1=%d", var1);
printf("\nThe value of var2=%d", var2);
}
```

test.tcl

```
set testagent [new Agent/TestAgentOtcl]
#Set configurable parameters of TestAgent
$testagent set var1_otcl2
$testagent set var2_otcl 3.14
```

```
# Give a command to MyAgent
$testagent call_function
$testagent add 2000 4000
```

……

ADD C++ code into NS 2

To include c++ code into NS-2

STEP 1: Edit /usr/local/ns-allinone-2.34/ns-2.34/**makefile.in**

Add following line in **OBJ_CC** section

**test/test.o **

STEP 2: recompiling NS-2

go to ~/ns-allinone-2.34/ns-2.34/directory and do

./configure

make

make install

STEP 3: run your **test.tcl** file

LAB – 2 ADD NEW PROTOCOL IN NS-2

Write your c++ code in /root/ns-allinone-2.34/ns-2.34/**newrp** folder

To include c++ code into NS-2

STEP 1: Edit /root/ns-allinone-2.34/ns-2.34/**makefile.in**

Add following line in **OBJ_CC** section

**newrp/newrp.o **

STEP 2: Edit /root/ns-allinone-2.34/ns-2.34/**common/packet.h**

To define new packet type we have to modify **packet.h file**

a) add the following line **packet_t** section

```
    static const packet_t PT_NEWRP = 62;
static packet_t  PT_NTYPE = 63;// this MUST be the LAST one
```

b) add the following line in **class p_info** section

```
static packetClass classify(packet_t type) {
if (type == PT_DSR ||
     type == PT_MESSAGE  ||
     type == PT_TORA ||
     type == PT_ADOV ||
     type == PT_NEWRP)
```

c) add the following line in **class p_info** section

```
name_[PT_AOMDV]="AOMDV";
name_[PT_NEWRP]="NEWRP";
name_[PT_NTYPE]="undefined";
```

STEP 3: Edit /root/ns-allinone-2.34/ns-2.34/tcl/lib/ns-packet.tcl

To configure routing agent, ADD protocol Name (line no: 175)

NEWRP

STEP 4: Edit /root/ns-allinone-2.34/ns-2.34//tcl/lib/ns-lib.tcl

ADD the following line in **switch –exact $routingAgent**_section line no:632)

```
NEWRP {
  set ragent [$self create-newrp-agent $node]
}
```

The following code should be added in same **ns-lib.tcl** at the end of the file.

```
Simulator instproc create-newrp-agent { node } {
# Create NEWRP routing agent
set ragent [new Agent/NEWRP [$node node-addr]]
$self at 0.0 "$ragent start"
$node set ragent $ragent
return $ragent
}
```

STEP 5: Edit /root/ns-allinone-2.34/ns-2.34/tcl/lib/ns-agent.tcl

Add following code at the end of the file

```
Agent/NEWRP instproc init args {
$self next $args

Agent/NEWRP set sport_    0
Agent/NEWRP set dport_    0
```

STEP 6: Recompiling NS-2

Go to /root/ns-allinone-2.34/ns-2.34/ directory and do

```
./configure
make clean
make
make install
```

STEP:7 Test

Edit wdemo.tcl script

Change routing protocol name as **NEWRP**

Run the tcl file and check the output.