Network simulator NS-2

Aim

To install the network simulator NS-2 in the Linux operating system and simulate wired and wireless scenarios.

Theory

NS

NS(network simulator) is a name for a series of discrete event network simulators, more specifically ns-1, ns-2, ns-3 and ns-4. All of these are discrete-event computer network simulators, primarily used in research and teaching. It is an open-source simulation tool that runs on Linux, targeted at networking research and provides substantial support for simulation of routing, multicast protocols and IP protocols, such as UDP, TCP, RTP and SRM over wired and wireless (local and satellite) networks. It has many advantages that make it a useful tool, such as support for multiple protocols and the capability of graphically detailing network traffic.

Installation

1. **STEP 1** : Download the package for ns2 from *https://sourceforge.net/projects/nsnam/files/latest/downloadhere*. Extraction of the zip file results in the extraction into a folder called *"ns-allinone-2.35"*. A few packages and the GCC version 4.3 are prerequisites for ns2 for its proper working.

2. **STEP 2** : Necessary dependencies are to be installed and configured as follows :
   • *sudo apt-get install build-essential autoconf automake libxmu-dev*
   • One of the dependencies mentioned is the compiler GCC, which can be installed as follows
   sudo apt-get install gcc-[your-require-version]
   • Now in the file named *"ls.h"*, goto line 137and change the word *"erase"* to *"this->erase"*. To specify the version of gcc, change the Makefile.in file, by changing *CC= @CC@* to *CC=gcc-[your-version]*.
3. **STEP 3** : Now continue with the installation process by running the following command.
   • *sudo su cd /ns-allinone-2.35/./install*
4. **STEP 4** : The final step is to set the environment path using the *".bashrc"* file. In that file, we need
   to add a few lines at the bottom corresponding to the environment paths. Make sure you replace them with
   your path. For example, if you have installed it in a folder *"/home/username"*, then replace *"/home/abc/ns-allinone-2.35/otcl-1.14"* with *"/home/username/ns-allinone-2.35/otcl-1.14".*
5. **STEP 5** : After completing the above steps, you can try running ns in the terminal. A % denotes successful installation.

TCL

It is a high-level, general-purpose, interpreted, dynamic programming language. It casts everything into the
mold of a command, including variable assignment and procedure definition. It supports multiple programming

paradigms, including object-oriented, imperative and functional programming or procedural styles.

Code

Wired

```
# Filename: out.tcl
#||-Event scheduler object creation||{#
set ns [new Simulator]
#|||-creating trace objects|||||-#
set nt [open out.tr w]
$ns trace-all $nt
#|||-creating nam objects|||||-#
set nf [open out.nam w]
$ns namtrace-all $nf
#|||-Setting color ID|||||-#
$ns color 1 darkmagenta
$ns color 2 yellow
$ns color 3 blue
$ns color 4 green
$ns color 5 black
#|||- Creating Network|||||-#
set totalNodes 3
for {set i 0} {$i < $totalNodes} {incr i} {
set node_($i) [$ns node]
}
set server 0
set router 1
set client 2
#|||- Creating Duplex Link|||||-#
$ns duplex-link $node_($server) $node_($router) 2Mb 50ms DropTail
$ns duplex-link $node_($router) $node_($client) 2Mb 50ms DropTail
$ns duplex-link-op $node_($server) $node_($router) orient right
$ns duplex-link-op $node_($router) $node_($client) orient right
#||||Labelling|||||-#
$ns at 0.0 "$node_($server) label Server"
$ns at 0.0 "$node_($router) label Router"
$ns at 0.0 "$node_($client) label Client"
$ns at 0.0 "$node_($server) color blue"
$ns at 0.0 "$node_($client) color blue"
$node_($server) shape hexagon
$node_($client) shape hexagon
#||||Data Transfer between Nodes|||||-#
# Defining a transport agent for sending
set tcp [new Agent/TCP]
# Attaching transport agent to sender node
$ns attach-agent $node_($server) $tcp
# Defining a transport agent for receiving
set sink [new Agent/TCPSink]
# Attaching transport agent to receiver node
$ns attach-agent $node_($client) $sink
#Connecting sending and receiving transport agents
$ns connect $tcp $sink
#Defining Application instance
set ftp [new Application/FTP]
# Attaching transport agent to application agent
$ftp attach-agent $tcp
# Setting flow color
$tcp set fid_ 4
# data packet generation starting time
$ns at 1.0 "$ftp start"
```

```
# data packet generation ending time
$ns at 6.0 "$ftp stop"
#|||finish procedure||{#
proc finish {} {
global ns nf nt
$ns flush-trace
close $nf
close $nt
puts "running nam..."
exec nam out.nam &
exit 0
}
#Calling finish procedure
$ns at 10.0 "finish"
$ns run
#||| Execution ||{#
ns out.tcl
```

wireless

```
set   val(chan)             val(stop)               ;# channel type
set   val(prop)             Propagation/TwoRayGround ;# radiopropagation model
set   val(netif)            Phy/WirelessPhy         ;# network interface type
set   val(netif)            Mac/802_11              ;# MAC type
set   val(ifq)              Queue/DropTail/PriQueue ;# interface queue type
set   val(ifq)              LL                      ;# link layer type
set   val(ant)              Antenna/OmniAntenna     ;# antenna model
set   val(ifqlen)           50                      ;# max packet in ifq
set   val(nn)               3                       ;# number of mobilenodes
set   val(rp)               DSDV                    ;# routing protocol
set   val(x)                500                     ;#
set   val(y)                400                     ;#
set   val(stop)             150                     ;#
```

```
set ns
[new Simulator]
#creating trace file and nam file
set tracefd
[open wireless1.tr w]
set windowVsTime2 [open win.tr w]
set namtrace
[open wireless1.nam w]
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
# set up topography object
set topo
[new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)
# configure the nodes
      $ns node-config -adhocRouting $val(rp) \
            -llType $val(ll) \
            -macType $val(mac) \
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen) \
            -antType $val(ant) \
            -propType $val(prop) \
            -phyType $val(netif) \
            -channelType $val(chan) \
```

```
            -topoInstance $topo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace OFF \
            -movementTrace ON
      for {set i 0} {$i < $val(nn) } { incr i } {
            set node_($i) [$ns node]
      }
# Provide initial location of mobilenodes
$node_(0)  set X_ 5.0
$node_(0)  set Y_ 5.0
$node_(0)  set Z_ 0.0

$node_(1) set X_ 490.0
$node_(1) set Y_ 285.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 150.0
$node_(2) set Y_ 240.0
$node_(2) set Z_ 0.0

# Generation of movements
$ns at 10.0 "$node_(0) setdest 250.0 250.0 3.0"
$ns at 15.0 "$node_(1) setdest 45.0 285.0 5.0"
$ns at 19.0 "$node_(2) setdest 480.0 300.0 5.0"
# Set a TCP connection between node_(0) and node_(1)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(1) $tcp
$ns attach-agent $node_(2) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"
# Printing the window size
proc plotWindow {tcpSource file} {
global ns
set time 0.01
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 10.0 "plotWindow $tcp $windowVsTime2"
# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}
```

```
# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
$ns at $val(stop) "$node_($i) reset";
}
# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 150.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
      global ns tracefd namtrace
      $ns flush-trace
      close $tracefd
      close $namtrace
      exec nam wireless1.nam &
      }
$ns run
```

Output

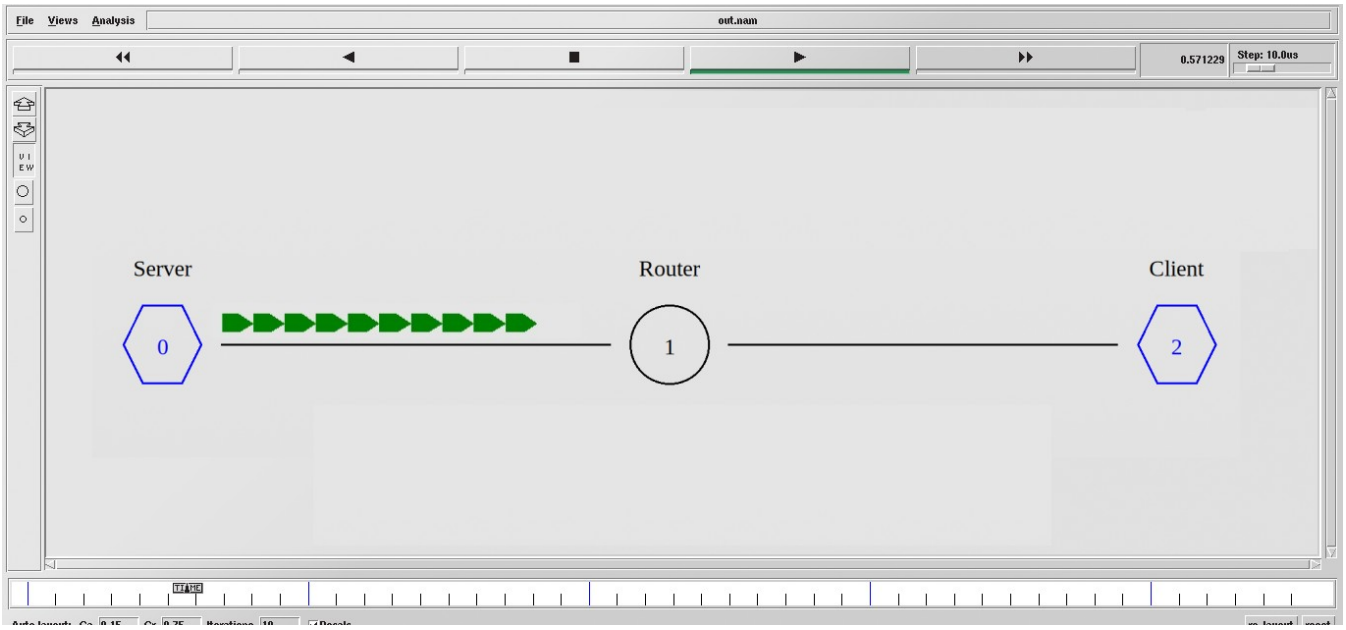

Figure 1: Wired network

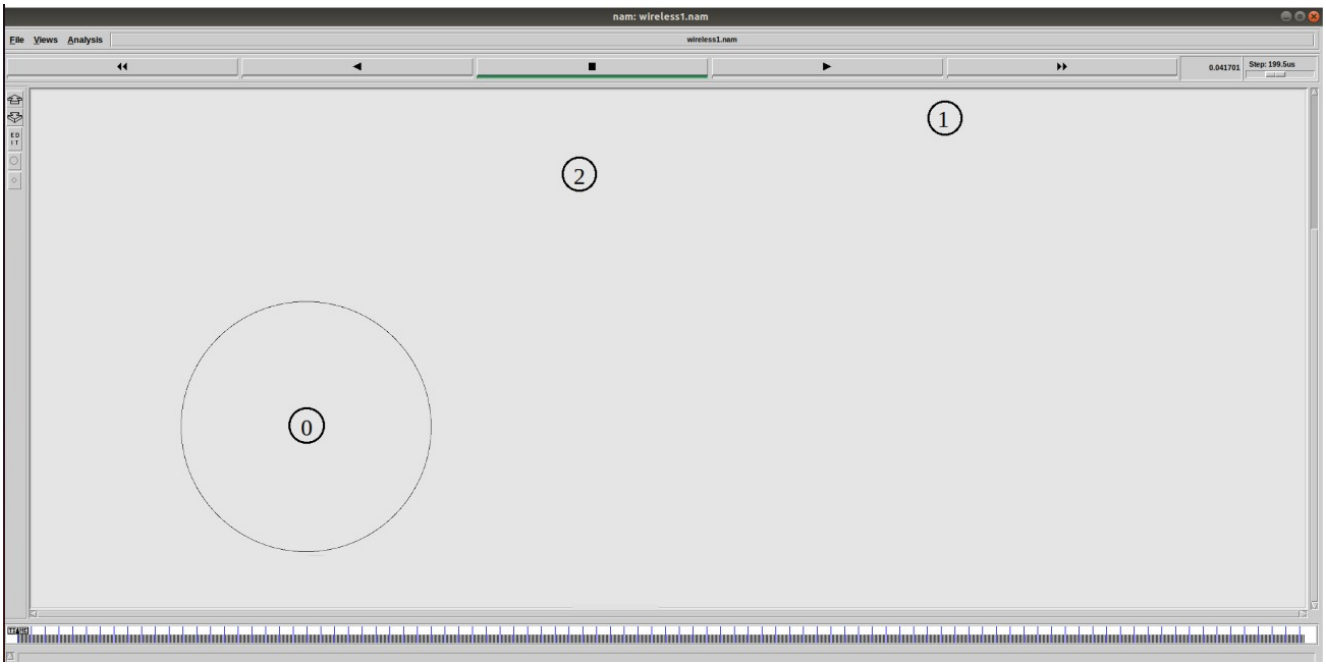

Figure 2: Wireless network

Result

Installed the network simulator NS-2 in the Linux operating system and simulate wired and wireless scenarios.