EXPERIMENT NO. : 10

NETWORK SIMULATOR NS2 IN LINUX OPERATING SYSTEM & SIMULATION OF A PROGRAM

AIM

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

Description

Basics of Computer Network Simulation:

A simulation can be thought of as a flow process of network entities (e.g., nodes, packets). As these entities move through the system, they interact with other entities, join certain activities, trigger events, cause some changes to the state of the system, and leave the process. From time to time, they contend or wait for some type of resources. This implies that there must be a logical execution sequence to cause all these actions to happen in a comprehensible and manageable way.

Introduction to Network Simulator 2 (NS2)

Network Simulator (Version 2), widely known as NS2, is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors.

Basic Architecture

NS2 provides users with an executable command ns which takes on input argument, the name of a Tcl simulation scripting file. In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation. NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend).

Ns2 Installation On Ubuntu 16.04

1) Download *'ns-allinone-2.35'* from :

http://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.35/ns-allinone-2.35.tar.gz/download

2) Extract the downloaded zip file 'ns-allinone-2.35.tar.gz file' to desktop.

3) Now you need to download some essential packages for ns2.Type the below lines one by one on the terminal window

"sudo apt-get update"

"sudo apt-get dist-upgrade"

"sudo apt-get update"

"sudo apt-get gcc"

"sudo apt-get install build-essential autoconf automake"

"sudo apt-get install tcl8.5-dev tk8.5-dev"

"sudo apt-get install perl xgraph libxt-dev libx11-dev libxmu-dev"

4) Now change your directory(here i have already extracted the downloaded files to desktop,so my location is desktop) type the following codes in the command window to install NS2. cd Desktop

cd ns-allinone-2.35

./install

The installation procedure will take a few minutes.....

5) After compleating the installation type the following command in the command window

gedit ~/.bashrc

6) Now an editor window appears,please copy and paste the following codes in the end of the text file (note that '/home/abhiram/Desktop/ns-allinone-2.35/otcl-1.14' in each line in the below code should be replaced with your location where the 'ns-allinone-2.35.tar.gz'file is extracted) #

LD_LIBRARY_PATH

OTCL_LIB=/home/abhiram/Desktop/ns-allinone-2.35/otcl-1.14

NS2_LIB=/home/abhiram/Desktop/ns-allinone-2.35/lib

X11_LIB=/usr/X11R6/lib

USR_LOCAL_LIB=/usr/local/lib

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:$USR_LOC AL

_LIB

TCL_LIBRARY

TCL_LIB=/home/abhiram/Desktop/ns-allinone-2.35/tcl8.5.10/library

USR_LIB=/usr/lib

export TCL_LIBRARY=$TCL_LIB:$USR_LIB

PATH

XGRAPH=/home/abhiram/Desktop/ns-allinone-2.35/bin:/home/abhiram/Desktop/ns-allinone 2.35/tcl8.5.10/unix:/home/abhiram/Desktop/ns-allinone-2.35/tk8.5.10/unix

NS=/home/abhiram/Desktop/ns-allinone-2.35/ns-2.35/

NAM=/home/abhiram/Desktop/ns-allinone-2.35/nam-1.15/

PATH=$PATH:$XGRAPH:$NS:$NAM

7) Save and close the text editor and then type the following command on the terminal

source ~/.bashrc

8) Close the terminal window and start a new terminal window and now change the directory to ns 2.35 and validate ns-2.35 by executing the following command (it takes 30 to 45 minutes) cd ns-2.35

./validate

9) If the installation is successful, then you will be able to see % at the command prompt while typing the following command

ns

10) Now type

exit

Ns2 Commands

Creating wired node

Simulator class has a procedure node which return a node.command

[simulator-instance] node

Changing shape of Node

Shape of node can be changed using shape procedure of node class. Shapes in ns2 available are

box, hexagon and circle. Default shape is circle. Once a shape is defined then it can't be changed after simulation starts.

Syntax:

[node-instance] shape <circle|hexagon|box>

Reset all agents of Node

reset command is used to reset all agents attached to node.

Syntax:

[node-instance] reset

Fetch id of Node

Node id can be fetched in ns2.

Syntax:

[node-instance] id

Attach agent to node on a specific port

Agent can be attached to node on a specified port number.

Syntax:

[node-instance] attach-agent [agent-instance] optional:<Port_no

Attach label to node

Labels can be attached to nodes in NAM for better understanding.

Syntax:

[node-instance] label [label]

Fetch next available port number

alloc-port is used to access available port numbers on a node for new agents.

Syntax:

[node-instance] alloc_port null_agent

Fetch list of neighbors

Every node maintains a list of neighbors which are connected to that node.

Syntax:

[node-instance] neighbor

Link Commands

In ns2, nodes can be connected in two ways, simplex and duplex. Simplex connection allows one-way communication and duplex connection allows two-way communication. Each type requires bandwidth, delay and type of queue for configuration.

Bandwidth is specified in Mbps(Mb) and delay is specified in milli seconds (ms).

Type of Queue available in ns2: DropTail, RED, CBQ, FQ, SFQ, DRR.

Syntax:

$ns simplex-link/duplex-link [node-instance1] [node-instance2] bandwidth delay Q-Type

Setting orientation of links in NAM

Position of links can be defined for better representation of network in NAM(Network AniMator).

Syntax : *$ns simplex-link-op/duplex-link-op [node-instance1] [node-instance2] orient [orientation]*

Orientation can be right, left, up, down, up-right, up-left etc.

Agent Class

For every node transport mechanism need to be defined to send data. These transport mechanism in ns2 defined using agent. For example FTP application requires TCP transport protocol, that's why TCP agent need to be associated with sending node. All agents are subclass of Agent class

Attach agent to node

Before attaching a agent to node, instance of agent need to created. Instance of above given agent can be instantiated as

$ [simulator-instance] attach-agent [node-instance] [agent-instance]

Fetch port number to which agent is attached

Every agent store port number assigned to it. This port number can be accessed by using following command

Syntax:

$ [agent-instance] port

Connect agent to another agent

For communication agent need to be connected to another agent to which it wants to send data.

. Syntax : *$ [simulator-instance] connect [agent-instance] [agent-instance]*

Ranjal 1/8/22

Result

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

PROGRAM

NETWORK SIMULATOR

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

#Define different colors for data flows (for NAM)
$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 NAM 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 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
```

```
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail



#Set Queue Size of link (n2-n3) to 10


$ns queue-limit $n2 $n3 10


#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

#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5

#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
```

```
#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false

#Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

#Detach tcp and sink agents (not really necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Print CBR packet size and interval
```

```
puts "CBR packet size = [$cbr set packet_size_]"

puts "CBR interval = [$cbr set interval_]"

#Run the simulation
$ns run
```

OUTPUT