Finolex Academy of Management and Technology, Ratnagiri

Department of Information Technology

Subject:	**Networking Lab (ITL401)**		
Class:	**SE IT / Semester – IV (CBCGS) / Academic year: 2017-18**		
Name of Student:	**Kazi Jawwad A Rahim**		
Roll No:	**28**	**Date of performance (DOP) :**	
Experiment No:	**06**	**Date of checking (DOC) :**	
Title: Graphical simulation of routing protocol using NAM with UDP.			
Marks:		**Teacher's Signature:**	

1. Aim: To implementation a network topology with respect to a routing protocol and observe graphical simulation in NAM with data transfer through UDP protocol.

2. Prerequisites:

Knowledge of

1. TCL programming
2. NS2 commands
3. Network Layers and protocols

3. Hardware Requirements:

1. PC with minimum 2GB RAM

4. Software Requirements:

1. Linux (Ubuntu 10.04)
2. ns-2.34 package
3. Text editor

5. Learning Objectives:

1. To understand the network simulator environment and visualize a network topology.
2. To understand the behavior of network protocols.
3. Understand the wired network using NS2.

6. Course Objectives Applicable: LO 3

7. Program Outcomes Applicable: PO2, PO4

8. Program Education Objectives Applicable: 1

9. Theory:

Steps to write a program using UDP

1. Initialization and Termination of TCL Script in NS-2

   set ns [new Simulator]
   An ns simulation starts with the command

2. Open Trace and NAM file:

   set tracefile1 [open out.tr w]
   $ns trace-all $tracefile1

The above statements creates a data trace file called "out.tr" and a nam visualization trace file called "out.nam".Within the tcl script, these files are not called explicitly by their names, but instead by pointers that are declared above and called "tracefile1" and "namfile" respectively.
Remark that they begin with a # symbol. The second line open the file "out.tr" to be used for writing, declared with the letter "w". The third line uses a simulator method called trace-all that have as parameter the name of the file where the traces will go.

The last line tells the simulator to record all simulation traces in NAM input format. It also gives the file name that the trace will be written to later by the command $ns flush-trace. In our case, this will be the file pointed at by the pointer "$namfile", i.e the file "out.tr".

3. Define a 'finish" procedure : The termination of the program is done using 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

The word proc declares a procedure in this case called finish and without arguments. The word global is used to tell that we are using variables declared outside the procedure. The simulator method "flush-trace" will dump the traces on the respective files. The tcl command "close" closes the trace files defined before and exec executes the nam program for visualization. The command exit will ends the application and return the number 0 as status to the system. Zero is the default for a clean exit. Other values can be used to say that is a exit because something fails. At the end of ns program we should call the procedure "finish".

4. Definition of a network of links and nodes

The way to define a node is
set n0 [$ns node]

The node is created which is printed by the variable n0. When we shall refer to that node in the script

we shall thus write $n0.

Define Links: we can define the links that connect them. An example of a definition of a link is:

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

Which means that $n0 and $n2 are connected using a bi-directional link that has 10ms of propagation delay and a capacity of 10Mb per sec for each direction.

To define a directional link instead of a bi-directional one, we should replace "duplex-link" by "simplex-link".

5. Attach Agents and Applications

We need to define routing (sources, destinations) the agents (protocols) the application that use them.

#setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set packetsize_ 100

The command $ns attach-agent $n4 $sink defines the destination node.
The command $ns connect $tcp $sink finally makes the TCP connection between the source and destination nodes.
CBR over UDP
A UDP source and destination is defined in a similar way as in the case of TCP.

Define the rate in the command
$cbr set rate_ 0.01Mb

The packet size can be set to some value using
$cbr set packetSize_1000

The beginning and end of the CBR application can be done through the following command
$ns at 0.1 "$cbr start"

Steps to create and execute tcl script:
Step 1: Open any text editor (vi, nano).
Step 2: Write the program using ns2 tcl script and save with extension as "filename.tcl"
Step 3: Execute tcl script as "ns filename.tcl"
Step 4: Press the "play button" and Observe the output.

Code:-

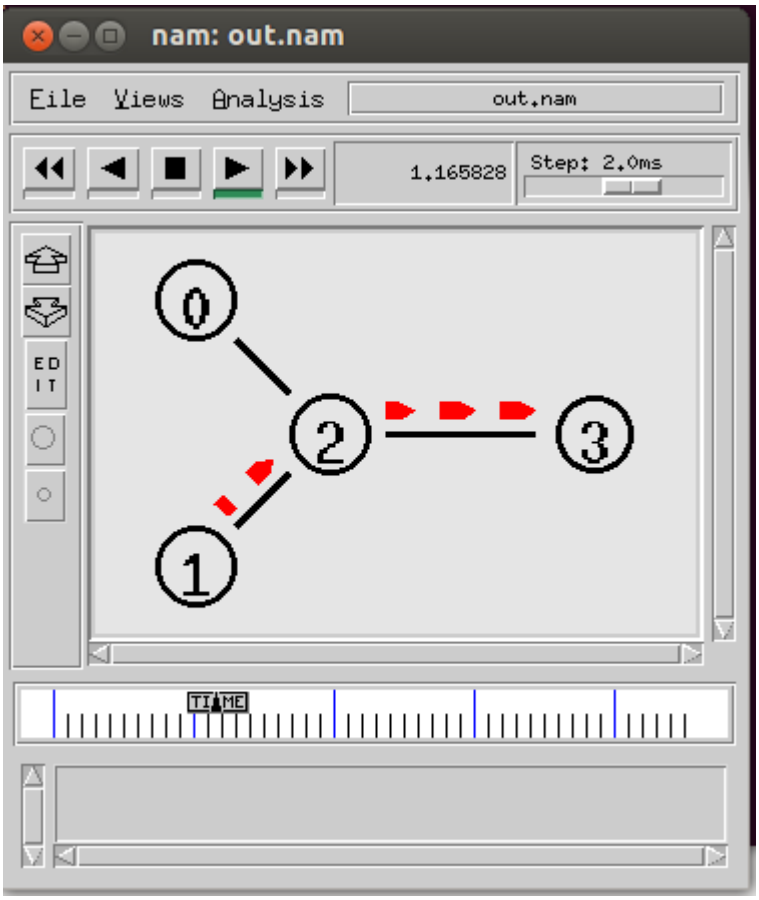
```
#Create a simulator object
set ns [new Simulator]
#Define different colours for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red
#Open the NAM trace file
set nf [open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
global ns nf
$ns flush-trace
#Close the NAM trace 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 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 agent
$ns at 0.1 "$cbr start"
$ns at 4.5 "$cbr stop"
#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:

```
students@ubuntu:~$ ns exp6.tcl
CBR packet size = 1000
CBR interval = 0.0080000000000000002
students@ubuntu:~$
```



13. Experiment/Assignment Evaluation

SR	Parameters	Weight	Excellent	Good	Average	Poor	Not as per requirement
		Scale Factor ->	5	4	3	2	0
1	Technical Understanding	25					
2	Performance / Execution	25					
3	Question Answers	20					
4	Punctuality	20					
5	Presentation	10					
	Total out of 100 --> #(to be converted as per term-work evaluation applicable to the subject)		**∑ (Weight * Scale Factor)/5 = _____**				

References:

[1] http://www.jgyan.com/ns2/trace%20file.php
[2] https://www.tcl.tk/man/tcl8.5/tutorial/Tcl1.html
[3] http://www.jgyan.com/ns2/link%20command.php

Viva Questions

1. What is UDP?
2. What is CBR?