**Experiment 11**

**Aim:** Installation of NS-2 in linux. Simulation of anyone routing protocol in linux

**Theory:**

**STEPS and Explanation**

Now we are going to write a 'template' that you can use for all of the first Tcl scripts. You can write your Tcl scripts in any text editor.

First of all, you need to create a simulator object. This is done with the command

**set ns [new Simulator]**

Now we open a file for writing that is going to be used for the nam trace data.

**set nf [open out.nam w]**

**$ns namtrace-all $nf**

The first line opens the file 'out.nam' for writing and gives it the file handle 'nf'. In the second line we tell the simulator object that we created above to write all simulation data that is going to be relevant for nam into this file.

The next step is to add a 'finish' procedure that closes the trace file and starts nam.

**proc finish {}{**

**global ns nf**

**$ns flush-trace**

**close $nf**

**exec nam out.nam &**

**exit 0**

**}**

This is called after this simulation is over by the command

**$ns at 5.0 “finish”**

The last line finally starts the simulation.

**$ns run**

We can actually save the file now and try to run it with 'ns example1.tcl'. we are going to get an error message like 'nam: empty trace file out.nam' though, because until now we haven't defined any objects (nodes, links, etc.) or events.

**Two nodes, one link**

In this section we are going to define a very simple topology with two nodes that are connected by a link. The following two lines define the two nodes. (Note: You have to insert the code in this section before the line '$ns run', or even better, before the line '$ns at 5.0 "finish"').

**set n0 [$ns node]**

**set n1 [$ns node]**

A new node object is created with the command '$ns node'. The above code creates two nodes and assigns them to the handles 'n0' and 'n1'. The next line connects the two nodes.

**$ns duplex-link $n0 $n1 1Mb**

**10ms DropTail**

This line tells the simulator object to connect the nodes n0 and n1 with a duplex link with the bandwidth 1Megabit, a delay of 10ms and a DropTail queue. Now you can save your file and start the script with 'ns example1.tcl'. nam will be started automatically and you should see an output that resembles the picture below. Sending data the next step is to send some data from node n0 to node n1. In ns, data is always being sent from one 'agent' to another. So the next step is to create an agent object that sends data from node n0, and another agent object that receives the data on node n1.

**#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**

These lines create a UDP agent and attach it to the node n0, then attach a CBR traffic generatot to the UDP agent. CBR stands for 'constant bit rate'. Line 7 and 8 should be self-explaining. The packetSize is being set to 500 bytes and a packet will be sent every 0.005 seconds (i.e. 200 packets per second). The next lines create a Null agent which acts as traffic sink and attach it to node n1.

**set null0 [new Agent/Null]**

**$ns attach-agent $n1 $null0**

Now the two agents have to be connected with each other.

**$ns connect $udp0 $null0**

And now we have to tell the CBR agent when to send data and when to stop sending. Note: It's probably best to put the following lines just before the line '$ns at 5.0 "finish"'.

**$ns at 0.5 "$cbr0 start“**

**$ns at 4.5 "$cbr0 stop"**

Now you can save the file and start the simulation again. When you click on the 'play' button in the nam window, you will see that after 0.5 simulation seconds, node 0 starts sending data packets to node 1. You might want to slow nam down then with the 'Step' slider. Now insert the following lines into the code to create four nodes. (topology)

**set n0 [$ns node]**

**set n1 [$ns node]**

**set n2 [$ns node]**

**set n3 [$ns node]**

The following piece of Tcl code creates three duplex 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 DropTail**

Add the next three lines to your Tcl script and start it again.

**$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**

Note that the autolayout related parts of nam are gone, since now you have taken the layout into your own hands. The options for the orientation of a link are right, left, up, down and combinations of these orientations

**CODE:**

#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

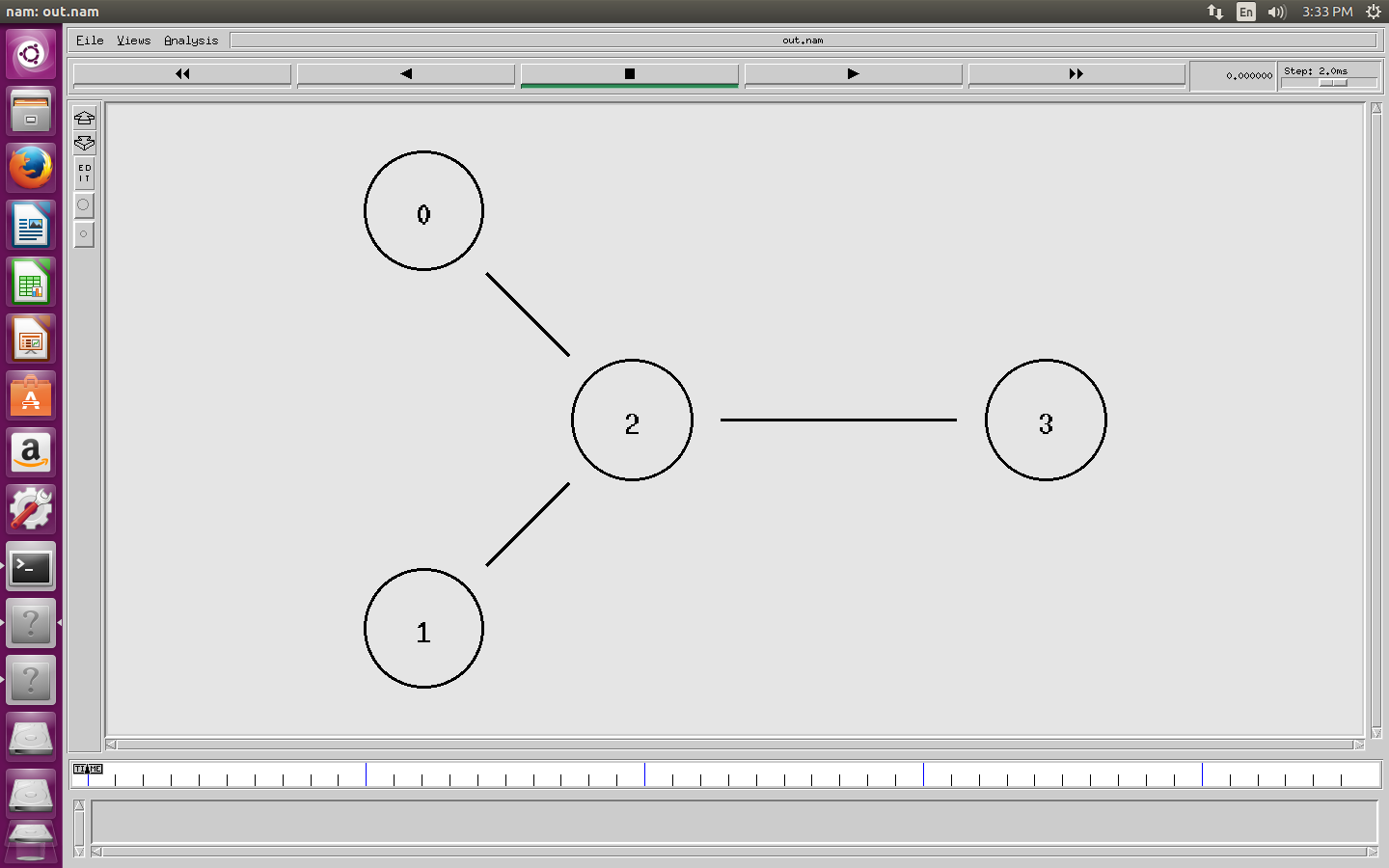
puts "CBR packet size = [$cbr set packet\_size\_]"

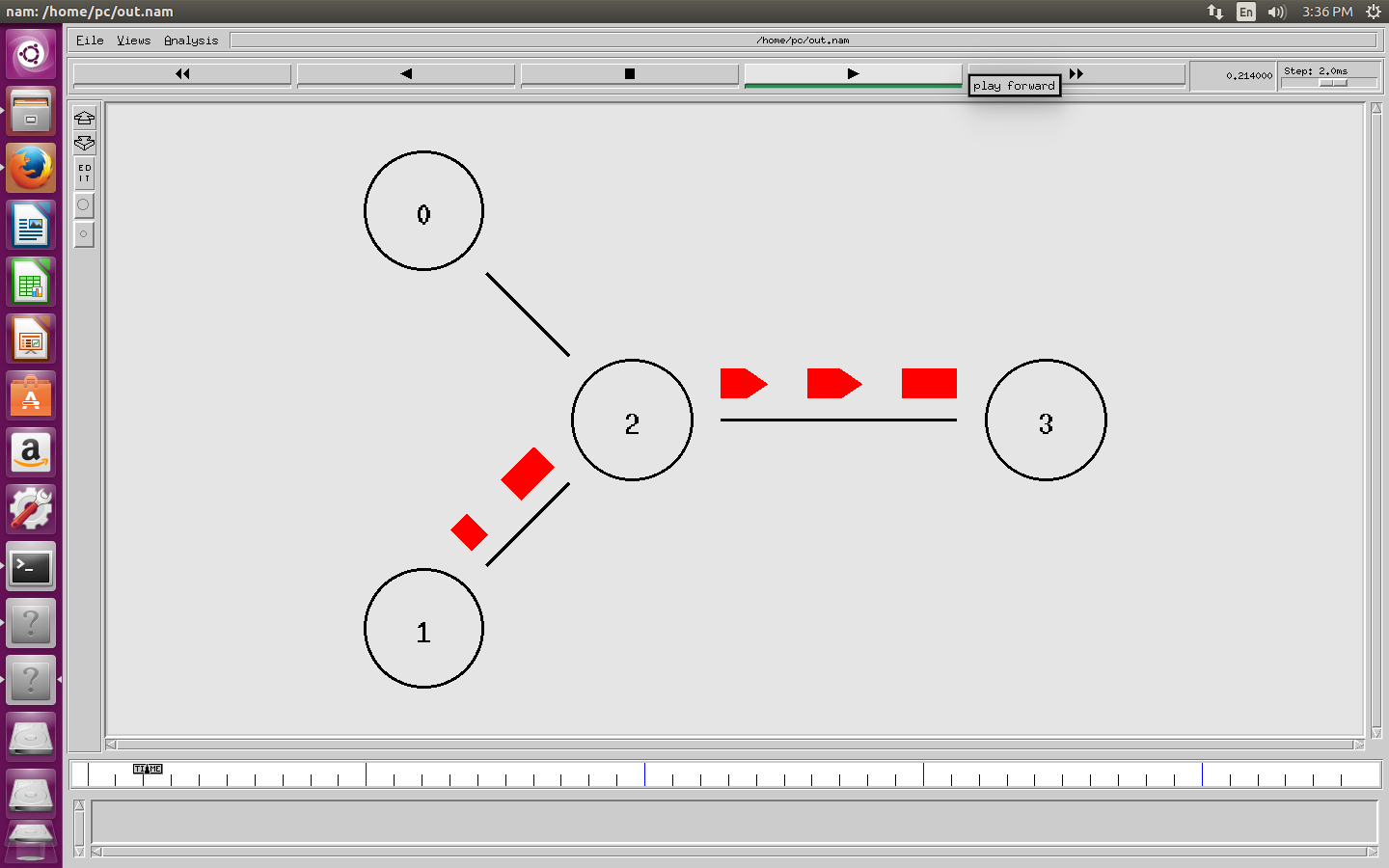
puts "CBR interval = [$cbr set interval\_]"

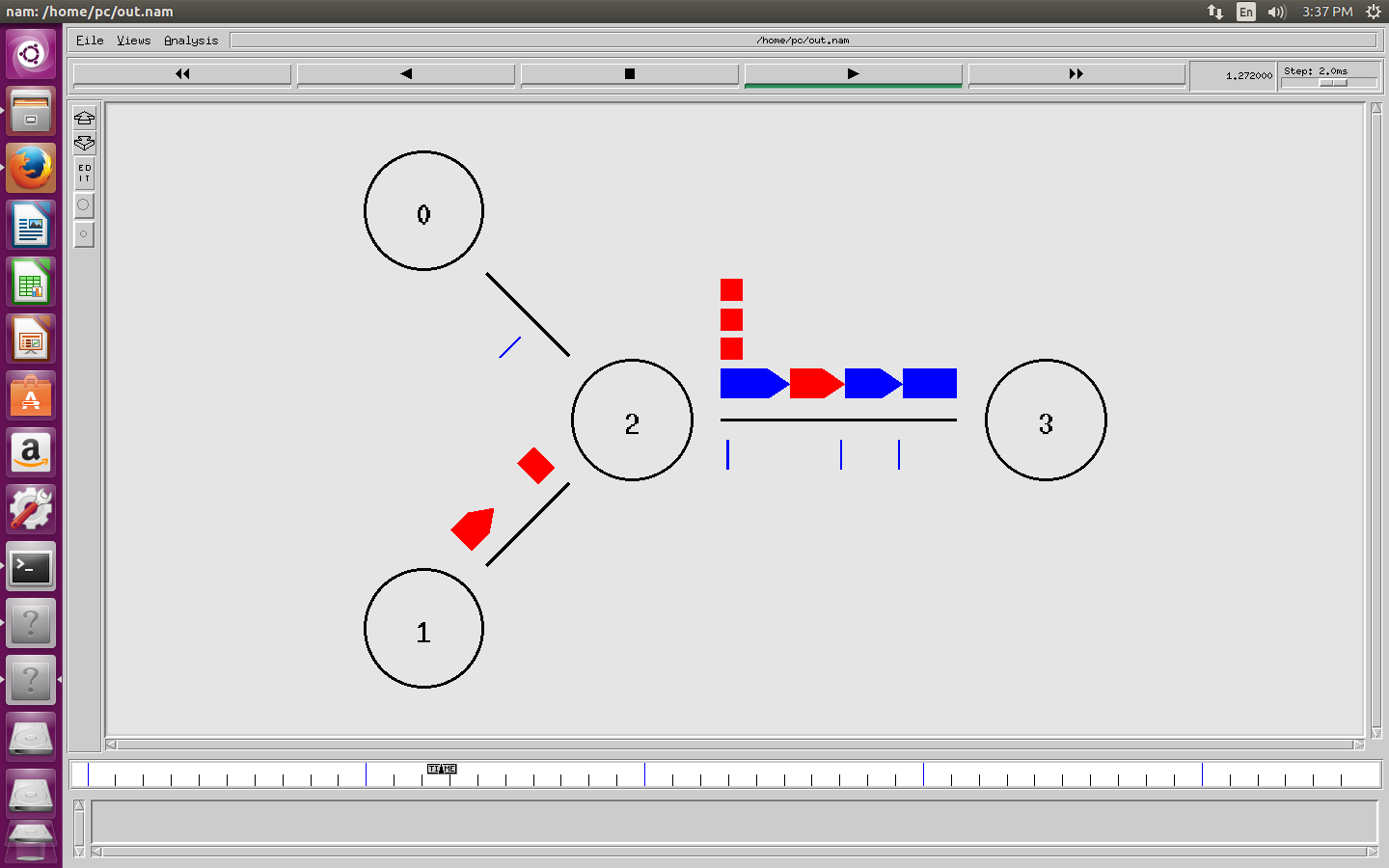
#Run the simulation

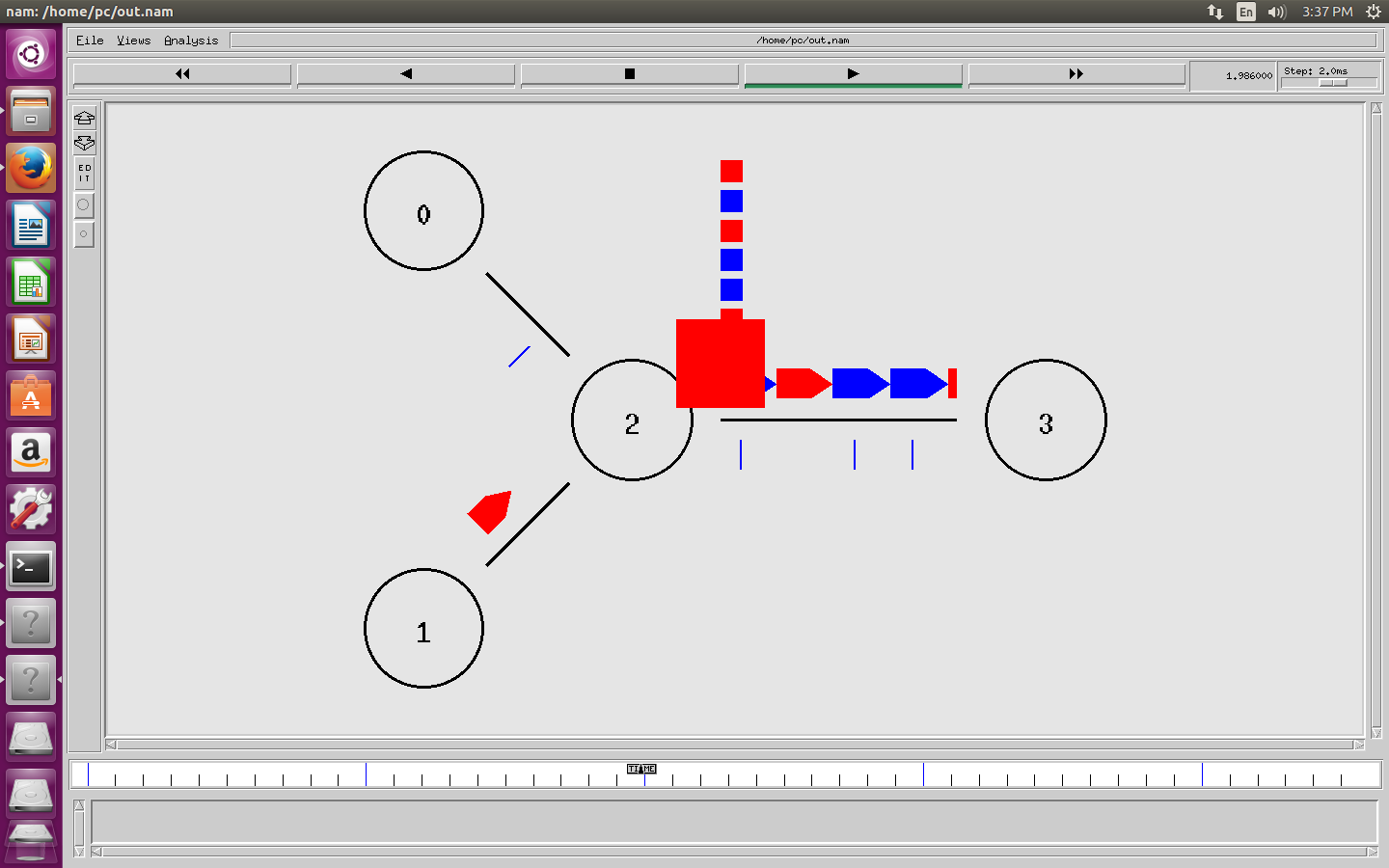
$ns run

**Output:**









**Conclusion:**

NS-2 was installed in linux. Routing protocol was implemented in linux successfully. The aim of the experiment was achieved.