**Experiment 7**

**Aim:** Study and installation of NS2

**Install NS-2 using this command :**

*sudo apt-get install ns2*

Nam is also needed to install. Nam (Network Animator) is an animation tool to graphically represent the network and packet traces.

**Use this command :**

*sudo apt-get install nam*

**Basic Commands : -***set a 8 set b [expr $a/8]*

**Define new procedures with proc command**

*proc factorial fact {*

*if {$fact <= 1}*

*{ return 1 } expr*

*$fact \* [factorial [expr $fact-1]]*

*}*

**• To open a file for reading :**

*set testfile [open hello.dat r]*

Similarly, put command is used to write data into the file

*set testfile [open hello.dat w] puts $testfile “hello1”*

• **To call subprocesses within another process, exec is used, which creates a subprocess and waits for it to complete.**

*exec rm $testfile*

• To be able to run a simulation scenario, a network topology must first be created. In ns2, the topology consists of a collection of nodes and links.

*set ns [new Simulator]*

In the network topology nodes can be added in the following manner :

*set n0 [$ns node] set n1 [$ns node]*

• Traffic agents (TCP, UDP etc.) and traffic sources (FTP, CBR etc.) must be set up if the node is not a router. It enables to create CBR traffic source using UDP as transport protocol or an FTP traffic source using TCP as a transport protocol.

To start the implementation of a .tcl file in ns2 you need to type the following command in the terminal

*ns filename.tcl*

After this the nam (Network Animator) will open.

**Program:**

#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

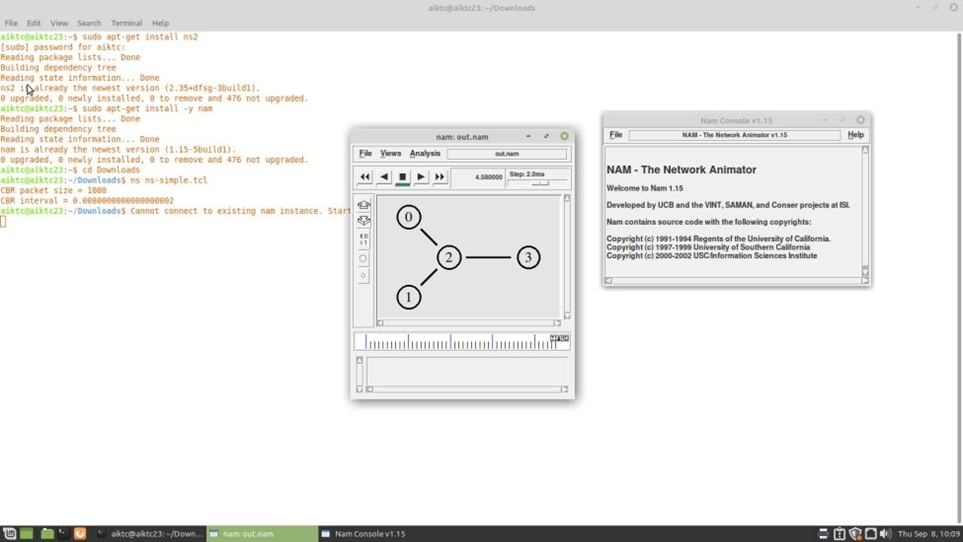
$ns at 5.0 "finish"

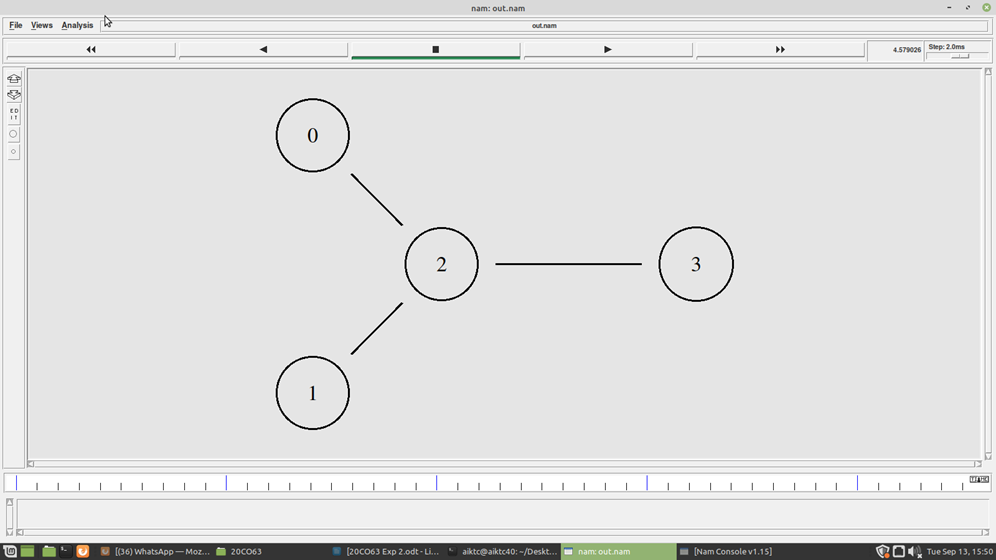
#Print CBR packet size and interval

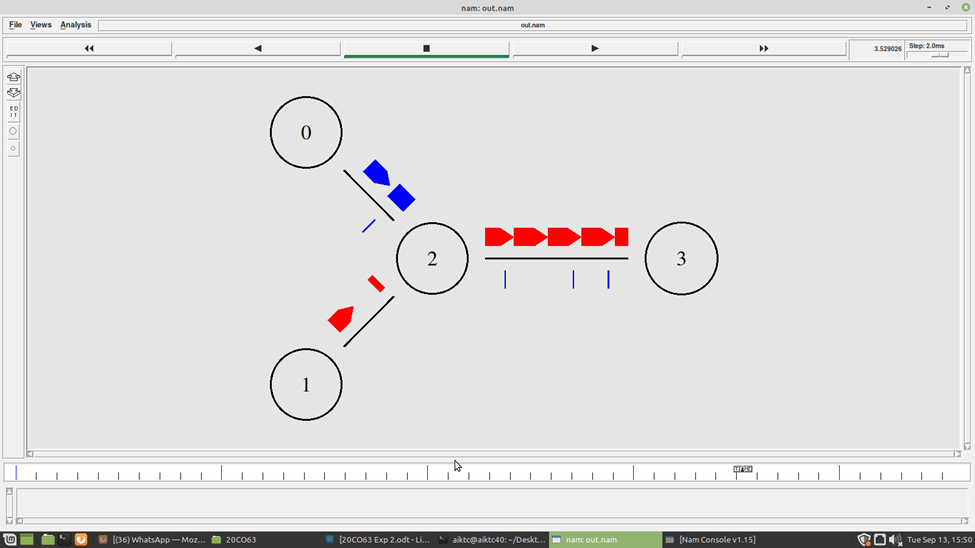
puts "CBR packet size = [$cbr set packet\_size\_]" puts "CBR interval = [$cbr set interval\_]"

#Run the simulation

$ns run



****

****