

Experiment No. 10

Aim: Simulate congestion control (leaky bucket / token bucket). **code/Netsim**

Theory:

Network simulation (NS) is one of the types of simulation, which is used to simulate the networks such as in MANETs, VANETs etc. It provides simulation for routing and multicast protocols for both wired and wireless networks. NS is licensed for use under version 2 of the GNU (General Public License) and is popularly known as NS2. It is an object-oriented, discrete event-driven simulator written in C++ and Otcl/tcl.

What is NS2: NS2 stands for Network Simulator Version 2. It is an open-source event-driven simulator designed specifically for research in computer communication networks.

NS-2 can be used to implement network protocols such as TCP and UDP, traffic source behavior such as FTP, Telnet, Web, CBR and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms and many more. In ns2, C++ is used for detailed protocol implementation and Otcl is used for the setup. The compiled C++ objects are made available to the Otcl interpreter and in this way, the ready-made C++ objects can be controlled from the OTcl level.

Features of NS2

1. It is a discrete event simulator for networking research.
2. It provides substantial support to simulate bunch of protocols like TCP, FTP, UDP, https and DSR.
3. It simulates wired and wireless network.
4. It is primarily Unix based.
5. Uses TCL as its scripting language.
6. Otcl: Object oriented support
7. Tclcl: C++ and otcl linkage
8. Discrete event scheduler

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
}
```

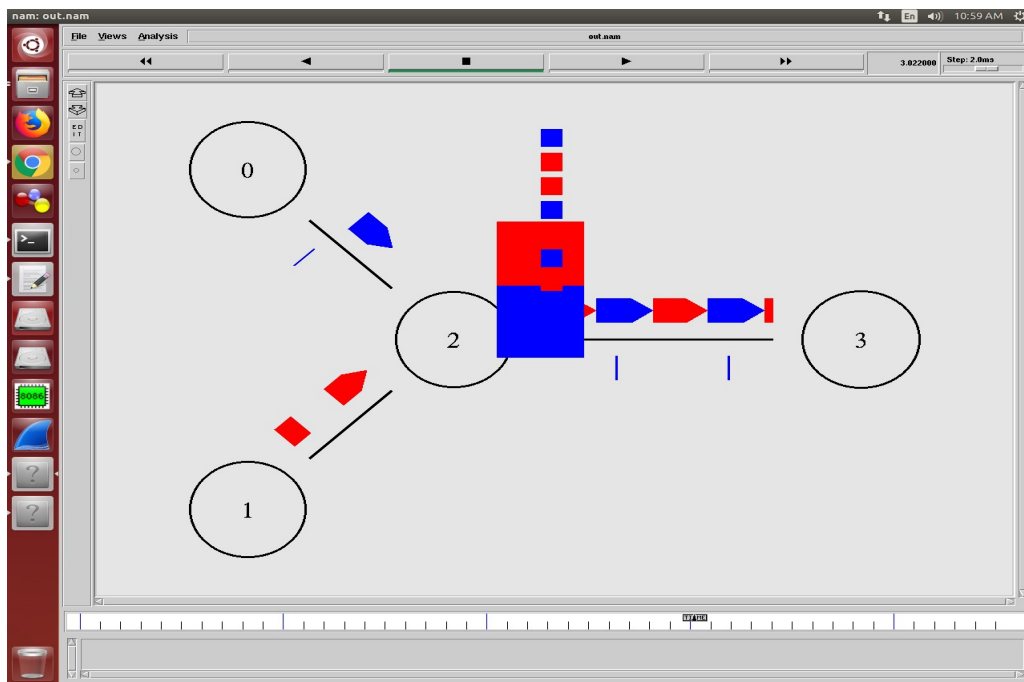
```

# 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:

Hence simulation of congestion control (leaky bucket / token bucket) has been performed successfully.