1. This script defines a simple topology of four nodes, and two agents, a UDP agent with a CBR traffic generator, and a TCP

agent. The simulation runs for . The output is two trace files, out.tr and out.nam. When the simulation completes at

the end of , it will attempt to run a nam visualisation of the simulation on your screen.

Code-

# The preamble

set ns [new Simulator] ;# initialise the simulation

# Predefine tracing

set f [open out.tr w]

$ns trace-all $f

set nf [open out.nam w]

$ns namtrace-all $nf

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

$ns duplex-link $n0 $n2 5Mb 2ms DropTail

$ns duplex-link $n1 $n2 5Mb 2ms DropTail

$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail

# Some agents.

set udp0 [new Agent/UDP] ;# A UDP agent

$ns attach-agent $n0 $udp0 ;# on node $n0

set cbr0 [new Application/Traffic/CBR] ;# A CBR traffic generator agent

$cbr0 attach-agent $udp0 ;# attached to the UDP agent

$udp0 set class\_ 0 ;# actually, the default, but. . .

set null0 [new Agent/Null] ;# Its sink

$ns attach-agent $n3 $null0 ;# on node $n3

$ns connect $udp0 $null0

$ns at 1.0 "$cbr0 start"

puts [$cbr0 set packetSize\_]

puts [$cbr0 set interval\_]

# A FTP over TCP/Tahoe from $n1 to $n3, flowid 2

set tcp [new Agent/TCP]

$tcp set class\_ 1

$ns attach-agent $n1 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n3 $sink

set ftp [new Application/FTP] ;# TCP does not generate its own traffic

$ftp attach-agent $tcp

$ns at 1.2 "$ftp start"

$ns connect $tcp $sink

$ns at 1.35 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

# The simulation runs for .

# The simulation comes to an end when the scheduler invokes the finish{} procedure below.

# This procedure closes all trace files, and invokes nam visualization on one of the trace files.

$ns at 3.0 "finish"

proc finish {} {

global ns f nf

$ns flush-trace

close $f

close $nf

puts "running nam..."

exec nam out.nam &

exit 0

}

# Finally, start the simulation.

$ns run

Q-2 Simulation of Ethernet Lan

Code-

#Lan simulation

set ns [new Simulator]

#define color for data flows

$ns color 1 Blue

$ns color 2 Red

#open tracefiles

set tracefile1 [open out.tr w]

set winfile [open winfile w]

$ns trace-all $tracefile1

#open nam file

set namfile [open out.nam w]

$ns namtrace-all $namfile

proc finish {} {

global ns tracefile1 namfile

$ns flush-trace

close $tracefile1

close $namfile

exec nam out.nam &

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

$n1 color Red

$n1 shape box

#create links between the nodes

$ns duplex-link $n0 $n2 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 2Mb 10ms DropTail

$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail

$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail

set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail Channel]

#Give node position

$ns duplex-link-op $n0 $n2 orient right-down

$ns duplex-link-op $n1 $n2 orient right-up

$ns simplex-link-op $n2 $n3 orient right

$ns simplex-link-op $n3 $n2 orient left

#set queue size of link(n2-n3) to 20

$ns queue-limit $n2 $n3 20

#setup TCP connection

set tcp [new Agent/TCP/Newreno]

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink/DelAck]

$ns attach-agent $n4 $sink

$ns connect $tcp $sink

$tcp set fid\_ 1

$tcp set packet\_size\_ 552

#set ftp over tcp connection

set ftp [new Application/FTP]

$ftp attach-agent $tcp

#setup a UDP connection

set udp [new Agent/UDP]

$ns attach-agent $n1 $udp

set null [new Agent/Null]

$ns attach-agent $n5 $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\_ 0.01Mb

$cbr set random\_ false

#scheduling the events

$ns at 0.1 "$cbr start"

$ns at 1.0 "$ftp start"

$ns at 124.0 "$ftp stop"

$ns at 125.5 "$cbr stop"

proc plotWindow {tcpSource file} {

global ns

set time 0.1

set now [$ns now]

set cwnd [$tcpSource set cwnd\_]

puts $file "$now $cwnd"

$ns at [expr $now+$time] "plotWindow $tcpSource $file"

}

$ns at 0.1 "plotWindow $tcp $winfile"

$ns at 125.0 "finish"

$ns run

Experiment-3.1

Question: To create scenario and study the performance of token ring protocols through simulation.

Token ring is a LAN protocol operating in the MAC layer. Token ring is standardized as per IEEE 802.5. Token ring can operate at

speeds of 4mbps and 16 mbps. The operation of token ring is as follows: When there is no traffic on the network a simple 3-byte token

circulates the ring. If the token is free (no reserved by a station of higher priority as explained later) then the station may seize the token

and start sending the data frame. As the frame travels around the ring ach station examines the destination address and is either

forwarded (if the recipient is another node) or copied. After copying4 bits of the last byte is changed. This packet then continues around

the ring till it reaches the originating station. After the frame makes a round trip the sender receives the frame and releases a new token

onto the ring.

Code -

#Create a simulator object

set ns [new Simulator]

#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 trace file

close $nf

#Executenam on the trace file

exec nam out.nam &

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

#Create links between the nodes

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

$ns duplex-link $n1 $n2 1Mb 10ms DropTail

$ns duplex-link $n2 $n3 1Mb 10ms DropTail

$ns duplex-link $n3 $n4 1Mb 10ms DropTail

$ns duplex-link $n4 $n5 1Mb 10ms DropTail

$ns duplex-link $n5 $n0 1Mb 10ms DropTail

#Create a TCP agent and attach it to node n0

set tcp0 [new Agent/TCP]

$tcp0 set class\_ 1

$ns attach-agent $n1 $tcp0

#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3

set sink0 [new Agent/TCPSink]

$ns attach-agent $n3 $sink0

#Connect the traffic sources with the traffic sink

$ns connect $tcp0 $sink0

# Create a CBR traffic source and attach it to tcp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.01

$cbr0 attach-agent $tcp0

#Schedule events for the CBR agents

$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Run the simulation

$ns run

Experiment - 3.2

AIM:

Question 2: To create scenario and study the performance of token ring protocols through simulation.

THEORY:

Star networks are one of the most common computer network topologies. In its simplest form, a star network consists of one central

switch, hub or computer, which acts as a conduit to transmit messages. This consists of a central node, to which all other nodes are

connected; this central node provides a common connection point for all nodes through a hub. In star topology, every node (computer

workstation or any other peripheral) is connected to a central node called a hub or switch. The switch is the server and the peripherals are

the clients. Thus, the hub and leaf nodes, and the transmission lines between them, form a graph with the topology of a star. If the central

node is passive, the originating node must be able to tolerate the reception of an echo of its own transmission, delayed by the two-way

transmission time (i.e. to and from the central node) plus any delay generated in the central node. An active star network has an active

central node that usually has the means to prevent echo-related problems.

The star topology reduces the damage caused by line failure by connecting all of the systems to a central node. When applied to a

bus-based network, this central hub rebroadcasts all transmissions received from any peripheral node to all peripheral nodes on the

network, sometimes including the originating node. All peripheral nodes may thus communicate with all others by transmitting to, and

receiving from, the central node only. The failure of a transmission line linking any peripheral node to the central node will result in the

isolation of that peripheral node from all others, but the rest of the systems will be unaffected.

Code -

#Create a simulator object

set ns [new Simulator]

#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 trace file

close $nf

#Executenam on the trace file

exec nam out.nam &

exit 0

}

#Create six nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

#Change the shape of center node in a star topology

$n0 shape square

#Create links between the nodes

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

$ns duplex-link $n0 $n2 1Mb 10ms DropTail

$ns duplex-link $n0 $n3 1Mb 10ms DropTail

$ns duplex-link $n0 $n4 1Mb 10ms DropTail

$ns duplex-link $n0 $n5 1Mb 10ms DropTail

#Create a TCP agent and attach it to node n0

set tcp0 [new Agent/TCP]

$tcp0 set class\_ 1

$ns attach-agent $n1 $tcp0

#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3

set sink0 [new Agent/TCPSink]

$ns attach-agent $n3 $sink0

#Connect the traffic sources with the traffic sink

$ns connect $tcp0 $sink0

# Create a CBR traffic source and attach it to tcp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.01

$cbr0 attach-agent $tcp0

#Schedule events for the CBR agents

$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Run the simulation

$ns run