SSN COLLEGE OF ENGINEERING, KALAVAKKAM

(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1511 – COMPUTER NETWORKS LAB

Lab Exercise 9: Simulation of congestion control algorithms

Aim:

Write tcl script to simulate the different congestion control algorithms

Algorithm:

1. Write tcl script to simulate:
    a. Create 3 nodes and the links between the nodes as
        i. 0 → 1 10Mb 10 ms duplex link
        ii. 1 → 2 2Mb 10 ms duplex link
    b. Align the nodes properly
    c. Setup a TCP/Tahoe connection over 0 and 2 and its flow id, window size, packet
    d. Show the simulation in network animator and in trace file.
2. Write tcl script to simulate:
    a. Create 3 nodes and the links between the nodes as
        i. 0 → 1 10Mb 10 ms duplex link
        ii. 1 → 2 2Mb 10 ms duplex link
    b. Align the nodes properly
    c. Setup a TCP/Reno connection over 0 and 2 and mention the same flow id, window size, packet used for TCP/Tahoe
    d. Show the simulation in network animator and in trace file.

Definition:

TCP Tahoe: if three duplicate ACKs are received (i.e. four ACKs acknowledging the same packet, which are not piggybacked on data and do not change the receiver's advertised window), Tahoe performs a fast retransmit, sets the slow start threshold to half of the current congestion window, reduces the congestion window to 1 MSS, and resets to slow start state.

TCP Reno: if three duplicate ACKs are received, Reno will perform a fast retransmit and skip the slow start phase by instead halving the congestion window (instead of setting it to 1 MSS like Tahoe), setting the slow start threshold equal to the new congestion window, and enter a phase called fast recovery.

Code:

tahoe.tcl

```
#Create a simulator object
set ns [new Simulator]

#Define a color for data flow (for NAM)
$ns color 1 Blue

#Open the NAM trace file
set nf [open outtahoe.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 outtahoe.nam &
 exit 0
}

#Create three nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n1 10Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail

#Set Queue Size of link (n1-n2) to 10
$ns queue-limit $n1 $n2 10

#Give node position (for NAM)
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right

#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 1
$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]
$ns attach-agent $n2 $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

#used to plot the graph using gnuplot
proc plotWindow {tcpSource outfile} {
 global ns
```

```
 set now [$ns now]
 set cwnd [$tcpSource set cwnd_]
 puts $outfile "$now $cwnd"
 $ns at [expr $now+0.1] "plotWindow $tcpSource $outfile"
}

set outfile [open "congestion2.txt" w]
$ns at 0.0 "plotWindow $tcp $outfile"
$ns at 0.2 "$ftp start"
$ns at 1.0 "$ftp stop"
$ns at 1.0 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n2 $sink"
$ns at 1.2 "finish"

#Run the simulation
$ns run
```

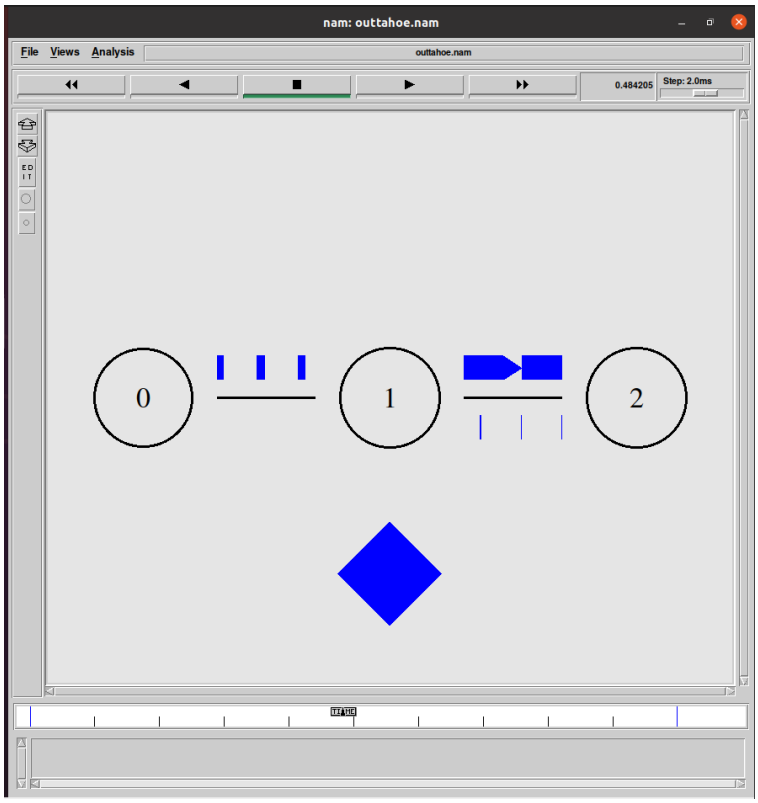
reno.tcl

```
#Create a simulator object
set ns [new Simulator]

#Define a color for data flow (for NAM)
$ns color 1 Blue

#Open the NAM trace file
set nf [open outreno.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 outreno.nam &
 exit 0
}

#Create three nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n1 10Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail

#Set Queue Size of link (n1-n2) to 10
$ns queue-limit $n1 $n2 10

#Give node position (for NAM)
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right

#Setup a TCP connection
set tcp [new Agent/TCP/Reno]
$tcp set class_ 1
$ns attach-agent $n0 $tcp
```

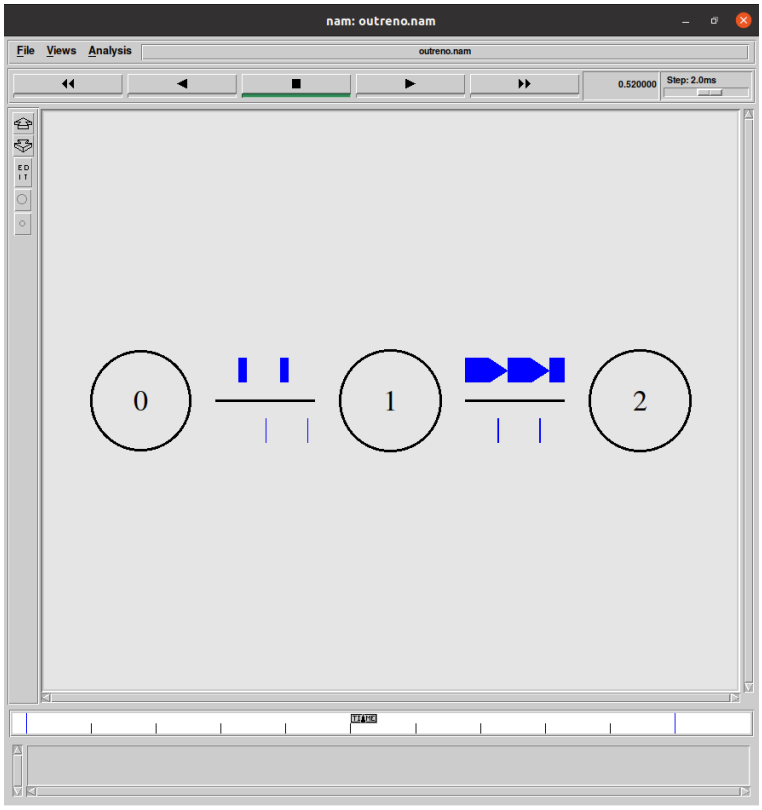
```
set sink [new Agent/TCPSink]
$ns attach-agent $n2 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

#used to plot the graph using gnuplot
proc plotWindow {tcpSource outfile} {
 global ns
 set now [$ns now]
 set cwnd [$tcpSource set cwnd_]
 puts $outfile "$now $cwnd"
 $ns at [expr $now+0.1] "plotWindow $tcpSource $outfile"
}

set outfile [open "congestion2.txt" w]
$ns at 0.0 "plotWindow $tcp $outfile"
$ns at 0.2 "$ftp start"
$ns at 1.0 "$ftp stop"
$ns at 1.0 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n2 $sink"
$ns at 1.2 "finish"

#Run the simulation
$ns run
```

Output:

tahoe.tcl

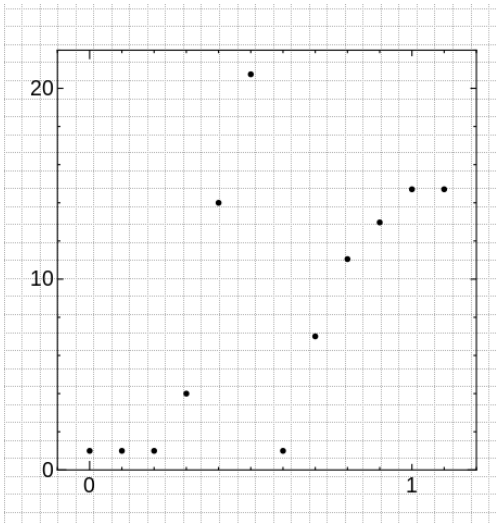


reno.tcl



Comparison (Congestion Window [y-axis] vs Time [x-axis]):

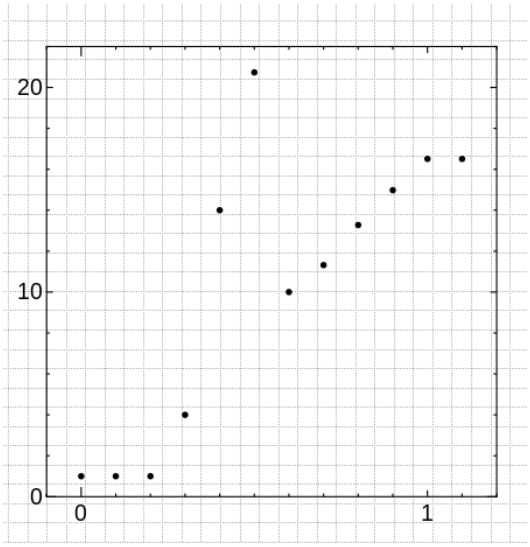
TCP Tahoe:



Tahoe = Slow Start + Additive Increase Multiplicative Decrease + Fast Retransmit

Packet loss is observed during the transmission, following which there is a ‘fast transmit’. After which, the congestion window starts from the initial state known as ‘slow start’.

TCP Reno:



Reno = Slow Start + Additive Increase Multiplicative Decrease + Fast Retransmit + Fast Recovery

Packet loss is observed during the transmission, following which the congestion window is reduced to half rather than initial state called ‘fast recovery’. This is an enhancement over TCP/Tahoe.

Learning outcomes:

1. Reexplored how to run a simple (.tcl) program.
2. Visualized using network animator (nam).
3. Compared TCP/Tahoe and TCP/Reno.