**Networking Module: Algorithms for Networking : Year-2017-2018**

**Supervised by :**

**Professor.Chadi Barakat**

M1 International Computer Science,

University of Nice Sophia Antipolis

***Mini Project:*** *TCP Congestion Control performance with NS-2*

-Submitted by

Cyril Naves Samuel

**TCP Simulation:**

**Given:**

1. **Source S and Destination R**
2. **File Size infinity**
3. **S ---🡪 R Link Rate: 10 Mbps ; One Way Delay 10 ms**
4. **Packet Size: 1500 bytes (maximum) + headers**
5. **Given:**

Buffer B at input of link

Advertised Window W by receiver R

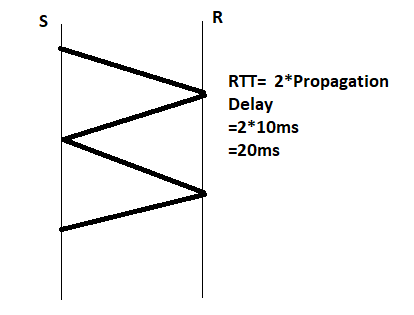
**Determine:**

Size of W (packets and bytes) to avoid packet queuing at Buffer.

Denote Window by W0.

**Solution:**

* Here Window size is advertised by the receiver R which implies that it is a flow control mechanism
* Also it also involves no loss, so there can be no saw tooth behavior
* To avoid packet queuing buffer B and window size W should be equal
* We know that to avoid packet queuing at buffer we tune the window size.
* Since the total Link Rate is given by 1500 bytes of packet size with no separate header size mentioned we assume that the protocol stack efficiency is 100% which is 10 Mbps ( Max Rate at TCP Layer delivered to the application)
* Since
* RTT is calculated by:



* Throughput desired = 10Mbps which avoids the queuing
* Therefore **Window Size= RTT \* Throughput 🡺**
* W =20ms\*10Mbps

=20\*10-3\*10\*106

=200\*103bps

=200Kbps

In Bytes: 200/8=25KBps

In Packets: Window Size/Packets Size

🡺25KBps/1500Bytes

=25000/1500

= Approx 16 packets

* **Therefore Window Size W0:**

**In Bytes:** 25KBps

**In Packets:** 16 packets

1. **Determine:**

Value of W to have buffer Overflow W1:

**Solution:**

* To have a Buffer overflow the Window Size should be greater than the Buffer. In our case Window Size W0 = 16 packets
* But to have a buffer overflow, the new window Size should be greater than the buffer size
* Ideally (in good case) it should be Buffer Size is close to the Window Size
* Let’s assume Buffer size (B) to be 12 packets
* W1=W0+B 🡺W1= 16 packets + 12 packets🡺**28 packets**

**c)Given:**

* Three values **W,W1,W0**,(between W0 and W1)=**Wm**,(above W1)=**Wu**

**Determine: File Transfer between S and R:**

1. **Dynamics of Window as a function of time. Plot the minimum of the congestion window and the receiver window.**

**Solution:**

**Code:**

***set ns [new Simulator]***

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

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

***$ns duplex-link $n0 $n1 10Mb 20ms DropTail***

***$ns queue-limit $n0 $n1 10***

***set tcp1 [new Agent/TCP/Reno]***

***$ns attach-agent $n0 $tcp1***

***$tcp1 set window\_ 32***

***$tcp1 set packet\_size\_ 1500***

***set sink1 [new Agent/TCPSink]***

***$ns attach-agent $n1 $sink1***

***$ns connect $tcp1 $sink1***

***set ftp1 [new Application/FTP]***

***$ftp1 attach-agent $tcp1***

***$ftp1 set type\_ FTP***

***$ns at 0.1 "$ftp1 start"***

***$ns at 100.0 "$ftp1 stop"***

***$ns at 100.0 "finish"***

***proc plotWindow {tcpSource outfile} {***

***set time 1***

***global ns***

***set now [$ns now]***

***set cwnd [$tcpSource set cwnd\_]***

***puts $outfile "$now $cwnd"***

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

***}***

***set outfile [open "dynamics.xg" w]***

***$ns at 0.0 "plotWindow $tcp1 $outfile"***

***proc finish {} {***

***global outfile***

***close $outfile***

***exec xgraph dynamics.xg -geometry 100x100 &;***

***exit 0***

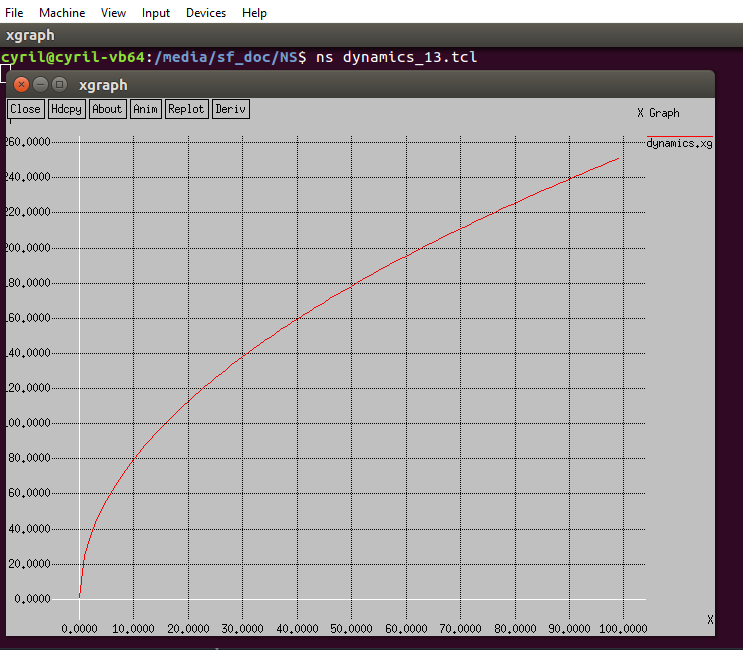
***}***

***$ns run***

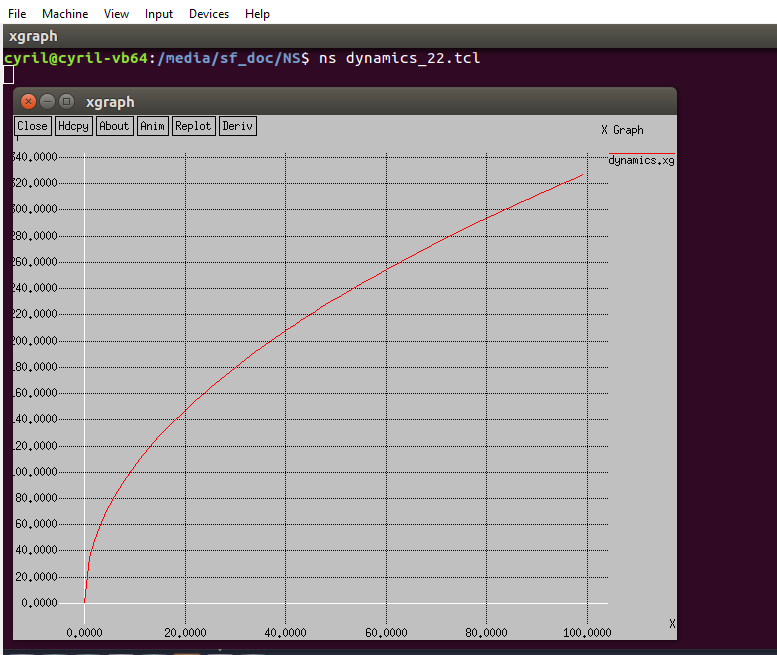
**Derived Values: W0 = 16 W1 = 28**

**Value Change in Code: *set window\_ value is to change for the different values of the window size.***

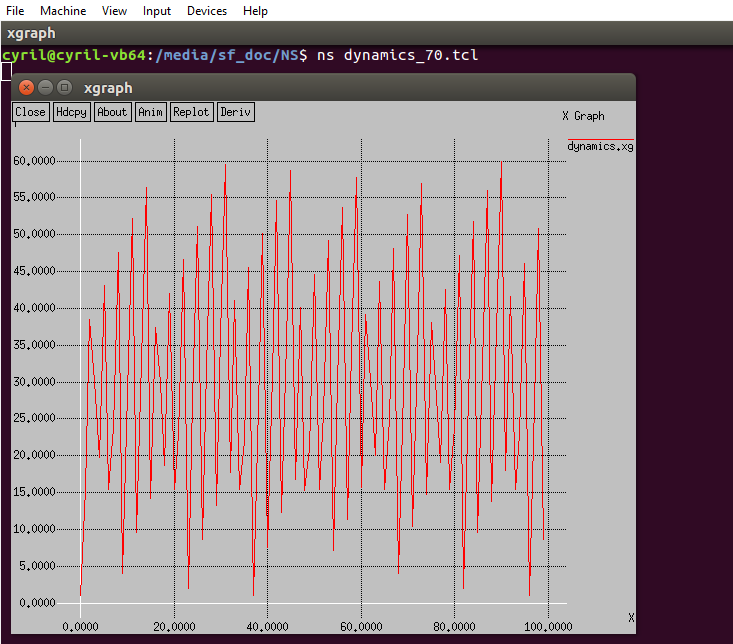
**Case 1: Less than W0 = 13**



**Case2: (between W0 and W1)= 22**



**Case 3: (above W1)= 70**



1. **Round-Trip Time as a function of time.**

**Solution:**

**Code:**

***set ns [new Simulator]***

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

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

***$ns duplex-link $n0 $n1 10Mb 20ms DropTail***

***$ns queue-limit $n0 $n1 10***

***set tcp1 [new Agent/TCP/Reno]***

***$ns attach-agent $n0 $tcp1***

***$tcp1 set window\_ 70***

***$tcp1 set packet\_size\_ 1500***

***set sink1 [new Agent/TCPSink]***

***$ns attach-agent $n1 $sink1***

***$ns connect $tcp1 $sink1***

***set ftp1 [new Application/FTP]***

***$ftp1 attach-agent $tcp1***

***$ftp1 set type\_ FTP***

***$ns at 0.1 "$ftp1 start"***

***$ns at 100.0 "$ftp1 stop"***

***$ns at 100.0 "finish"***

***proc plotWindow {tcpSource outfile} {***

***set time 1***

***global ns***

***set now [$ns now]***

***set cwnd [$tcpSource set rtt\_]***

***puts $outfile "$now $cwnd"***

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

***}***

***set outfile [open "rtt.xg" w]***

***$ns at 0.0 "plotWindow $tcp1 $outfile"***

***proc finish {} {***

***global outfile***

***close $outfile***

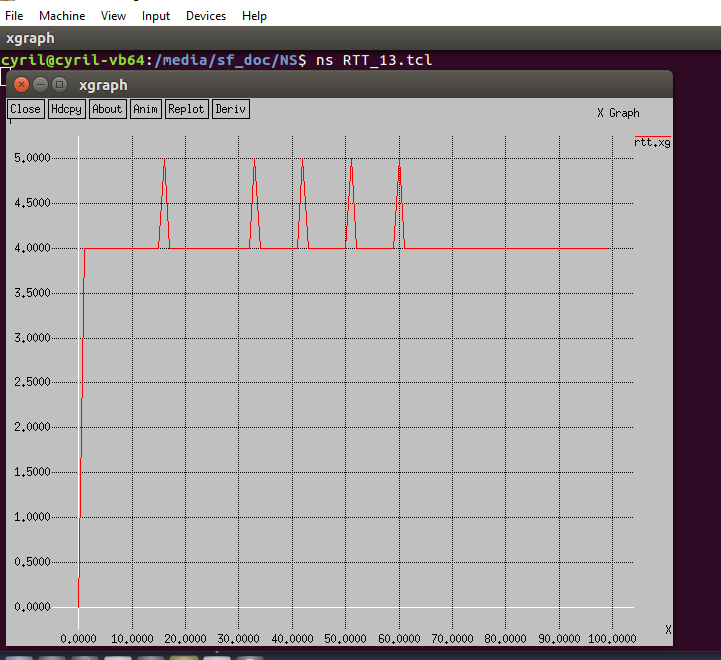
***exec xgraph rtt.xg geometry 300x300 &;***

***exit 0***

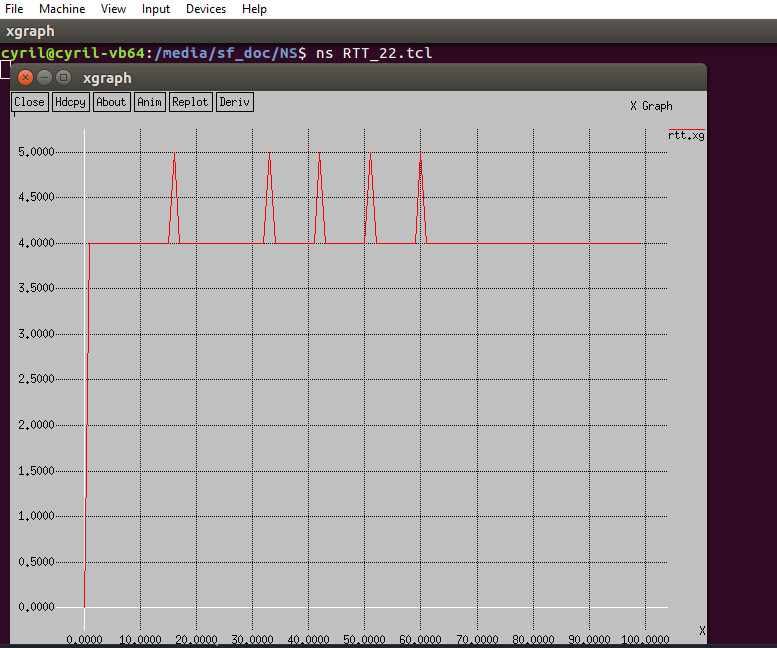
***}***

***$ns run***

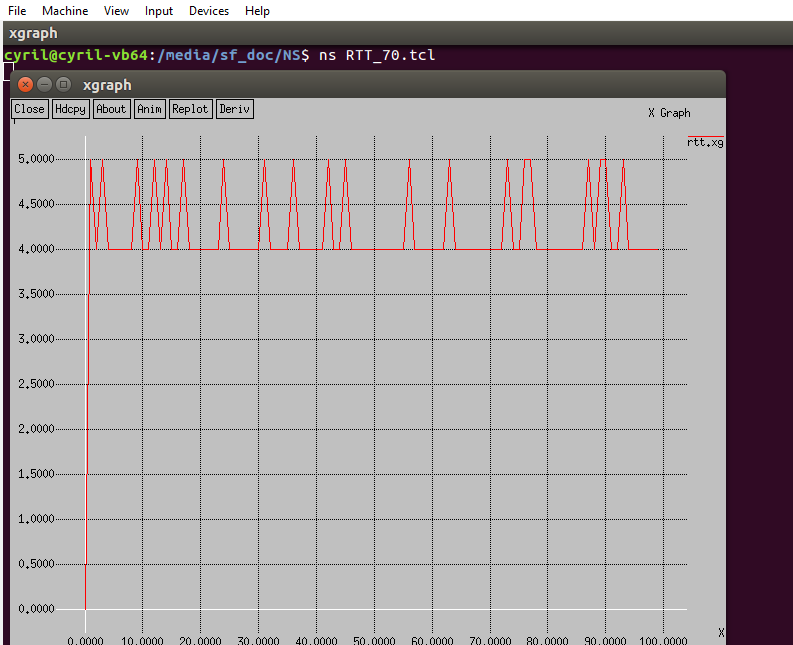
**Case 1: Less than W0 = 13**



**Case2: (between W0 and W1)= 22**



**Case 3: (above W1)= 70**



1. **Queuing Delay in Buffer B as a function of time.**

**Solution:**

**Code:**

***set ns [new Simulator]***

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

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

***$ns duplex-link $n0 $n1 10Mb 20ms DropTail***

***$ns queue-limit $n0 $n1 10***

***set tcp1 [new Agent/TCP/Reno]***

***$ns attach-agent $n0 $tcp1***

***$tcp1 set window\_ 13***

***$tcp1 set packet\_size\_ 1500***

***set sink1 [new Agent/TCPSink]***

***$ns attach-agent $n1 $sink1***

***$ns connect $tcp1 $sink1***

***set ftp1 [new Application/FTP]***

***$ftp1 attach-agent $tcp1***

***$ftp1 set type\_ FTP***

***$ns at 0.1 "$ftp1 start"***

***$ns at 100.0 "$ftp1 stop"***

***$ns at 100.0 "finish"***

***set monitor [$ns monitor-queue $n0 $n1 stdout]***

***proc queueLength {tcpSource file} {***

***global ns monitor***

***set time 0.1***

***set len [$monitor set size\_]***

***set now [$ns now]***

***puts $file "$now $len"***

***$ns at [expr $now+$time] "queueLength $tcpSource $file"}***

***set file [open "queue.xg" w]***

***$ns at 0 "queueLength $tcp1 $file"***

***proc finish {} {***

***global file***

***close $file***

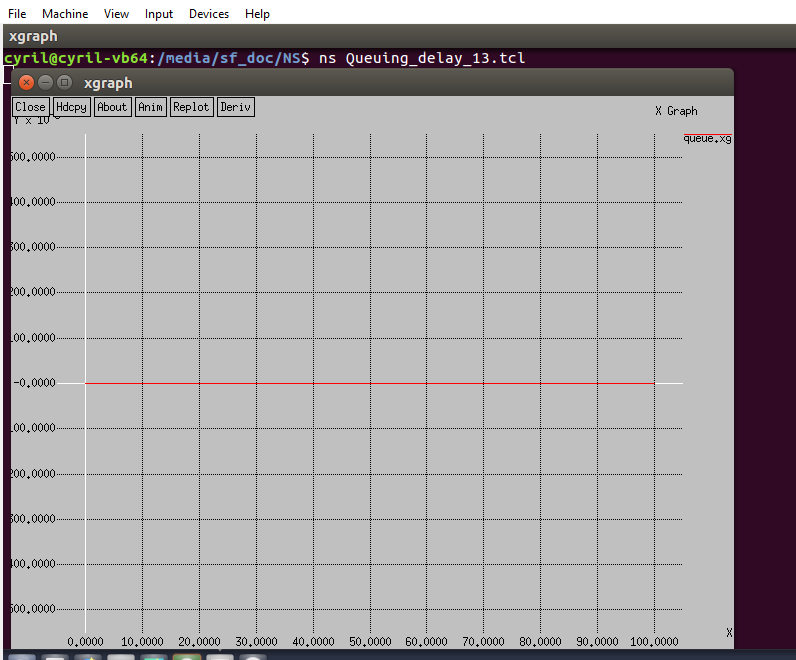
***exec xgraph queue.xg geometry 300x300 &;***

***exit 0***

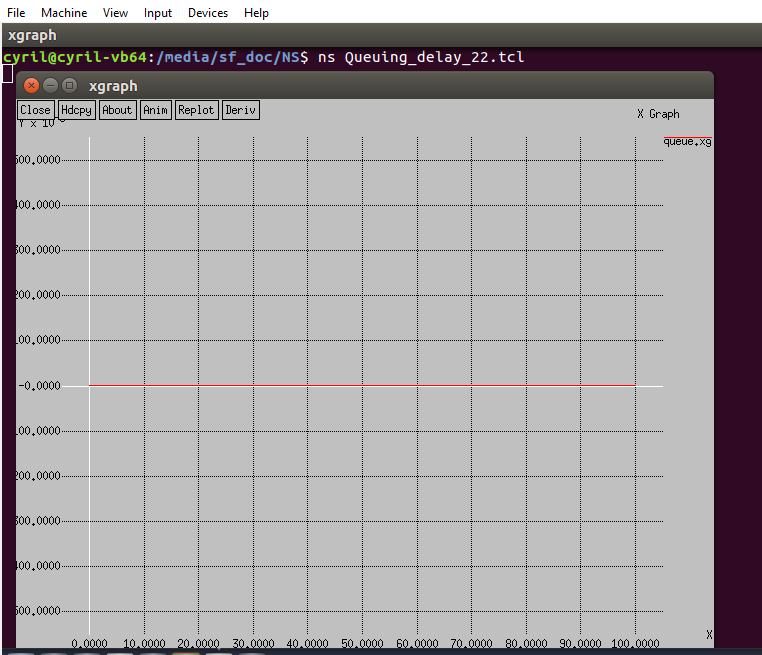
***}***

***$ns run***

**Case 1: Less than W0 = 13**



**Case2: (between W0 and W1)= 22**



**Case 3: (above W1)= 70**



1. **Instantaneous Download rate as a function of time(Window divided by round-trip time RTT.**

**Solution:**

**Code:**

***set ns [new Simulator]***

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

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

***$ns duplex-link $n0 $n1 10Mb 20ms DropTail***

***$ns queue-limit $n0 $n1 10***

***set tcp1 [new Agent/TCP/Reno]***

***$ns attach-agent $n0 $tcp1***

***$tcp1 set window\_ 13***

***$tcp1 set packet\_size\_ 1500***

***set sink1 [new Agent/TCPSink]***

***$ns attach-agent $n1 $sink1***

***$ns connect $tcp1 $sink1***

***set ftp1 [new Application/FTP]***

***$ftp1 attach-agent $tcp1***

***$ftp1 set type\_ FTP***

***$ns at 1. "$ftp1 start"***

***$ns at 100.0 "$ftp1 stop"***

***$ns at 100.0 "finish"***

***proc plotWindow {tcpSource outfile} {***

***set time 1***

***global ns***

***set now [$ns now]***

***set cw [$tcpSource set cwnd\_]***

***set rt [$tcpSource set rtt\_]***

***set r [expr 1.0$rt]***

***set c [expr $cw]***

***set cwnd [expr $c/$r]***

***puts $outfile "$now $cwnd"***

***$ns at [expr $now+$time] "plotWindow $tcpSource $outfile"}***

***set outfile [open "download.xg" w]***

***$ns at 1.0 "plotWindow $tcp1 $outfile"***

***proc finish {} {***

***global outfile***

***close $outfile***

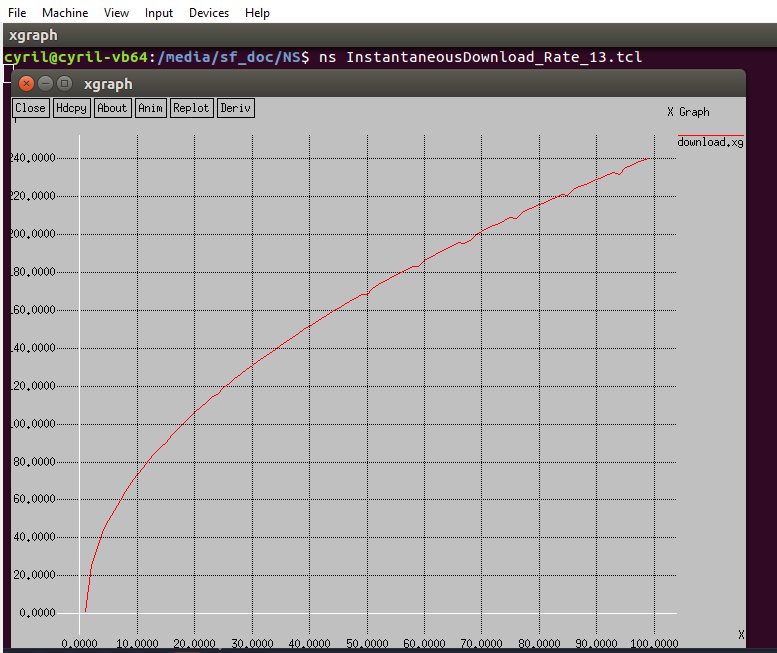
***exec xgraph download.xg geometry 300x300 &;***

***exit 0***

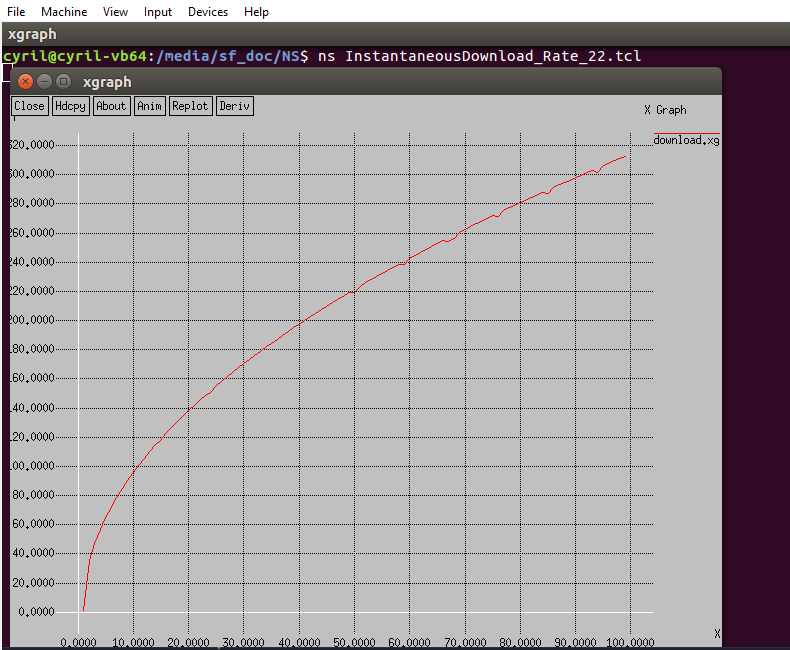
***}***

***$ns run***

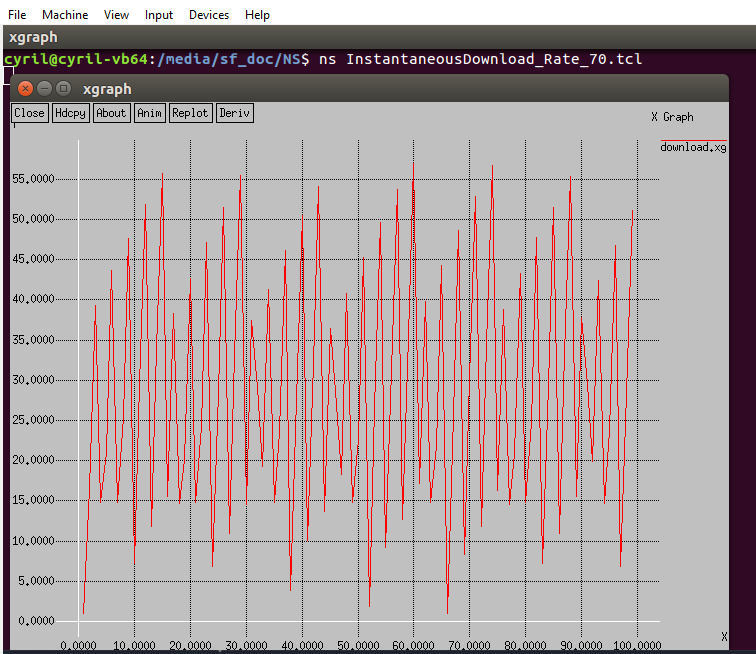
**Case 1: Less than W0 = 13**



**Case2: (between W0 and W1)= 22**



**Case 3: (above W1)= 70**



**d)Determine:** To vary the throughput (the long term average download rate) as a function of W:

**Solution:**

* We know that the Throughput of the TCP connection is given by 🡺

=**WindowSize/RTT**

* During a particular round Trip interval TCP sends data as a function of Congestion Window and RTT
* This explains that the throughput varies with the window size but to explain further let us consider a scenario where we ignore slow start phase which occurs after time out.
* Then we know when there is a loss, window drops to w/2 in the throughput is dropped to: **W/2RTT**
* Then the TCP throughput varies linearly between **W/RTT** and **W/2RTT**, average throughput is given = **0.75 \* W/RTT**

**Square Root Formula for TCP Throughput:**

**Given:**

* **Link Drops packets with probability p**
* **Set Receiver Window W to W0**
* **Increase Loss Rate p from 0 to 1**

**Determine:**

* **Simulate 10 values of p with different seeds**

**Solution**

**Code:**

***set ns [new Simulator]***

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

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

***$ns duplex-link $n0 $n1 10Mb 20ms DropTail***

***$ns queue-limit $n0 $n1 10***

***set tcp1 [new Agent/TCP/Reno]***

***$ns attach-agent $n0 $tcp1***

***$tcp1 set window\_ 16***

***$tcp1 set packet\_size\_ 1500***

***set sink1 [new Agent/TCPSink]***

***$ns attach-agent $n1 $sink1***

***$ns connect $tcp1 $sink1***

***set ftp1 [new Application/FTP]***

***$ftp1 attach-agent $tcp1***

***$ftp1 set type\_ FTP***

***$ns at 1 "$ftp1 start"***

***$ns at 100.0 "$ftp1 stop"***

***$ns at 100.0 "finish"***

***proc plotWindow {tcpSource outfile} {***

***set time 1***

***global ns n0 n1 sink1***

***set lossyLink [$ns link $n0 $n1]***

***set lossModel [new ErrorModel]***

***$lossModel set rate .20***

***$lossModel unit packet***

***$lossModel drop-target [new Agent/Null]***

***$lossyLink install-error $lossModel***

***set now [$ns now]***

***set cwnd [$sink1 set bytes\_]***

***set rtt [$tcpSource set rtt\_]***

***puts $cwnd***

***puts $rtt***

***puts $outfile "$now $cwnd"***

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

***}***

***set outfile [open "congestion.xg" w]***

***$ns at 0.0 "plotWindow $tcp1 $outfile"***

***proc finish {} {***

***global outfile***

***close $outfile***

***exec xgraph congestion.xg geometry 300x300 &;***

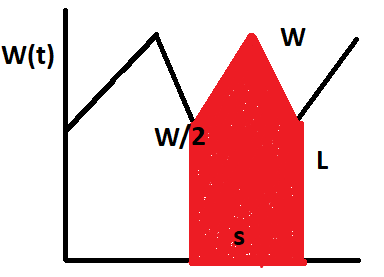
***exit 0***

***}***

***$ns run***

* **Calculate long term average download rate and plot it as a function of P**

**Solution:**



* We know that for each cycle L of congestion window, the window ranges from W/2 when loss occurs every p packets and then increases to W
* Here the sender window gives the no of packets sent, the number of them sent in a cycle is N
* Then we know that the cycle N 🡺 W/2+(W/2+1)+(W+2+2)+..W/2+W/2 =>3/4W(W/2+1)
* The average no of packets dropped is 1/P
* N🡺3/8W2  =1/P
* Average Download Rate is given by : 3/2\*(W/RTT)
* Where W is given by : sqrt((8/3)/P)
* There Average Download Rate: 3/2\*( sqrt((8/3)/P)/RTT) which equates to a sqrt formulae
* Curve compare to square root formulae: packet\_size\*sqrt(3/2p)/RTT

Here in comparison with the curve the send rate when using a square root formulae for low values of P the sending rate increase proportionally

**Fairness of TCP:**

**Given:**

* **Window Size -W0 - 16 packets**
* **Buffer Size B -10 packets**
* **Start download at 0 and then another download at 100 secs**

**Determine:**

**Solution:**

**Code:**

***set ns [new Simulator]***

***set f0 [open download1.tr w]***

***set f1 [open download2.tr w]***

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

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

***set n2 [$ns node]***

***set n3 [$ns node]***

***set n4 [$ns node]***

***$ns duplex-link $n0 $n3 10Mb 20ms DropTail***

***$ns duplex-link $n1 $n3 10Mb 20ms DropTail***

***$ns duplex-link $n2 $n3 10Mb 20ms DropTail***

***$ns duplex-link $n3 $n4 10Mb 20ms DropTail***

***proc finish {} {***

***global f0 f1***

***close $f0***

***close $f1***

***exec xgraph download1.tr download2.tr geometry 800x400 &***

***exit 0***

***}***

***proc attach-expoo-traffic { node sink size burst idle rate } {***

***set ns [Simulator instance]***

***set source [new Agent/UDP]***

***$ns attach-agent $node $source***

***set traffic [new Application/Traffic/Exponential]***

***$traffic set packetSize\_ $size***

***$traffic set burst\_time\_ $burst***

***$traffic set idle\_time\_ $idle***

***$traffic set rate\_ $rate***

***$traffic attach-agent $source***

***$ns connect $source $sink***

***return $traffic***

***}***

***proc record {} {***

***global sink0 sink1 f0 f1***

***set ns [Simulator instance]***

***set time 0.50***

***set bw0 [$sink0 set bytes\_]***

***set bw1 [$sink1 set bytes\_]***

***set now [$ns now]***

***puts $f0 "$now [expr $bw0/$time\*8/1000000]"***

***puts $f1 "$now [expr $bw1/$time\*8/1000000]"***

***$sink0 set bytes\_ 0***

***$sink1 set bytes\_ 0***

***$ns at [expr $now+$time] "record"***

***}***

***set sink0 [new Agent/LossMonitor]***

***set sink1 [new Agent/LossMonitor]***

***$ns attach-agent $n4 $sink0***

***$ns attach-agent $n4 $sink1***

***set source0 [attach-expoo-traffic $n0 $sink0 350 2s 1s 150k]***

***set source1 [attach-expoo-traffic $n1 $sink1 350 2s 1s 250k]***

***$ns at 0.0 "record"***

***$ns at 100.0 "$source0 start"***

***$ns at 0.0 "$source1 start"***

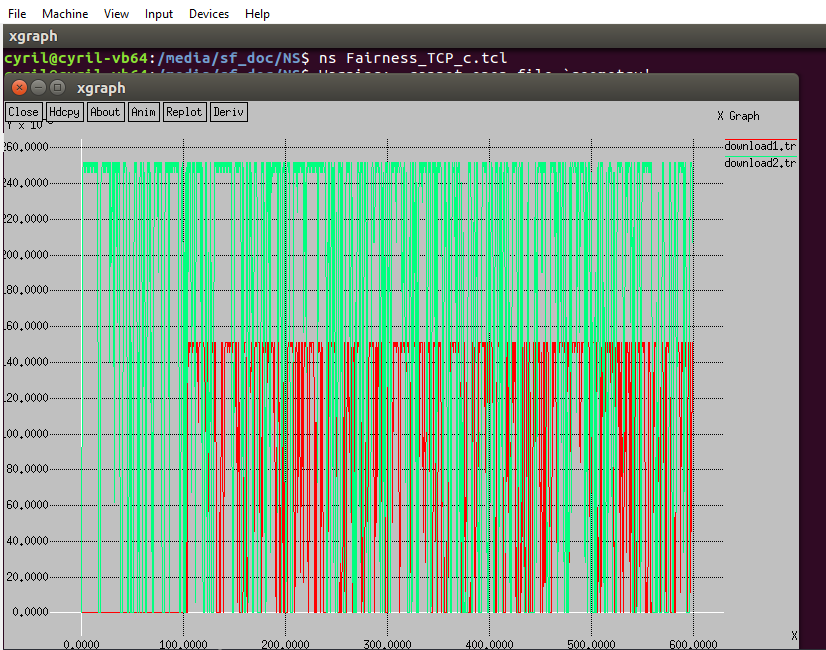
***$ns at 600.0 "$source0 stop"***

***$ns at 600.0 "$source1 stop"***

***$ns at 600.0 "finish"***

***$ns run***

* **Plot Congestion Window**



* **Two downloads to converge**

**Solution:**

In this scenario the downloads try to compete for the available bandwidth but they will not be able to converge to fairness as in Additive Increase Multiplicative decrease (AIMD) also it just converges to the efficiency of the allocation of the resources but not with the fairness.

* **Change of Packet Size or RTT of TCP connection change the rate achieved.**

**Solution:**

Yes, If the RTT of TCP connection is increase it shows a bias against that connection and favors a connection with a smaller RTT although its supposed to show the fairness in the bandwidth allocation

Also when the Packet Size of a connection is smaller then the TCP connection is favored than a connection with a larger packet size.

**Appendix:**

**Synopsis of this Project Code and Simulation of NS2:**

***TCP Algorithms are governed by four important ones:***

1. **Slow-Start**: TCP increase Congestion window each time an ACK is received so it doubles the congestion for every RTT
2. **Congestion Avoidance**: When Congestion window increases above ssthresh it enters congestion avoidance phase.
3. **Fast Retransmit**: When duplicate ACK is received 3 times, then it is conformed that the packets is lost and then retransmitted again as it does not wait for timer to detect the packet loss
4. **Fast Recovery**: When a loss is detected, fast retransmit is performed.

But then it does not enter a slow start phase rather a fast recovery whereby it will reduce the congestion window by half and then it increases by rwnd,cwnd,no of ACK.

***Reason for Choosing TCP Reno Agent in this project:***

Four variants of TCP agent:

* TCP Tahoe – Simplest includes every algorithm except fast recovery
* TCP Reno – All algorithm implemented but has performance issue when in packet loss but better than others.
* TCP New Reno – Better performance when packets are lost and better fast recovery implemented.
* SACK: Here it maintains the information which packet is missed and only retransmits them. (Selective Acknowledgment)

**Generic Setup with eg snippet:**

Nodes are created

*set n0 [$ns node]*

Links, Queue are created between the nodes

$ns duplex-link $n0 $n1 10Mb 20ms DropTail

Sending Node is assigned a TCP Agent- Reno

set tcp1 [new Agent/TCP/Reno]

Then window size and packet size are defined

$tcp1 set window\_ 25

Receiving Node is TCP Agent – Sink

set sink1 [new Agent/TCPSink]

FTP Traffic is setup with the created sending node and the sink

$ns connect $tcp1 $sink1 $ftp1 attach-agent $tcp1

Then traffic is started and stopped

$ns at 0.1 "$ftp1 start"

**Specific Setup for each derivation:**

1. **Dynamics of Congestion Window:**

Here now and cwnd\_ values are plotted in a file and then plotted using Xgraph

1. **Round Trip Time as a function of Time:**

Here now and rtt\_ values are plotted in a file and then plotted using Xgraph

1. **Queuing Delay in Buffer B:**

Here additional monitor is setup to monitor the queue length by

set monitor [$ns monitor-queue $n0 $n1 stdout]

Here now and set size\_values are plotted in a file and then plotted using Xgraph

1. **Instantaneous Download Rate:**

Here congestion window cwnd\_, RTT rtt\_ are divided to obtain the download rate which is then plotted along with now using Xgraph

1. **Square Root Formulae for TCP throughput:**

Here between the link between S and R an object that randomly drops TCP packets with probability p (set rate). This object belongs to the class ErrorModel

set lossyLink [$ns link $n0 $n1]

set lossModel [new ErrorModel]

$lossModel set rate .20

$lossModel unit packet

$lossModel drop-target [new Agent/Null]

$lossyLink install-error $lossModel

1. **Fairness of TCP:**

Here periodically the bandwidth are recorded which are received by the traffic sinks and writes it to the files.

puts $f0 "$now [expr $bw0/$time\*8/1000000]"