Лабораторная работа 4

Мажитов Магомед Асхабович

Содержание

[1 Цель работы 1](#_Toc164561468)

[2 Задание 1](#_Toc164561469)

[3 Выполнение лабораторной работы 2](#_Toc164561470)

[4 Выводы 3](#_Toc164561471)

[5 Листинг программ 4](#_Toc164561472)

[5.1 nodes.tcl 4](#_Toc164561473)

[5.4 main.tcl 5](#_Toc164561474)

[5.6 queue.tcl 6](#_Toc164561475)

[5.7 plot.sh 6](#_Toc164561476)

# 1 Цель работы

Самостоятельно смоделировать сеть с определенными правилами.

# 2 Задание

Описание моделируемой сети:

* сеть состоит из N TCP-источников, N TCP-приёмников, двух маршрутизаторов R1 и R2 между источниками и приёмниками (N — не менее 20);
* между TCP-источниками и первым маршрутизатором установлены дуплексные соединения с пропускной способностью 100 Мбит/с и задержкой 20 мс очередью типа DropTail;
* между TCP-приёмниками и вторым маршрутизатором установлены дуплексные соединения с пропускной способностью 100 Мбит/с и задержкой 20 мс очередью типа DropTail;
* между маршрутизаторами установлено симплексное соединение (R1–R2) с пропускной способностью 20 Мбит/с и задержкой 15 мс очередью типа RED, размером буфера 300 пакетов; в обратную сторону — симплексное соединение (R2–R1) с пропускной способностью 15 Мбит/с и задержкой 20 мс очередью типа DropTail;
* данные передаются по протоколу FTP поверх TCPReno;
* параметры алгоритма RED: qmin = 75, qmax = 150, qw = 0, 002, pmax = 0.1;
* максимальный размер TCP-окна 32; размер передаваемого пакета 500 байт; время моделирования — не менее 20 единиц модельного времени.

# 3 Выполнение лабораторной работы

1. Начнем с основного файла, в нем мы имеем создание симулятора и добавление внешних файлов. Также тут мы задаем процедуры [finish](file:///C:\Users\magom\Desktop\мип\лаб4\report.docx#finish.tcl) и [plotWindow](file:///C:\Users\magom\Desktop\мип\лаб4\report.docx#plotWindow.tcl), которые отвечают за создание файлов, необходимых для графиков и запуск отрисовки графиков; и создания файла размера окна. Также тут же находится небольшой кусок кода, который отвечает за симулируемое время, то бишь запускает процессы, необходимые нашей симуляции, а именно запуск ftp и запуск процедуры [plotWindow](file:///C:\Users\magom\Desktop\мип\лаб4\report.docx#plotWindow.tcl). [Здесь представлен листинг нашей программы](#ns2-redtcl)
2. [Далее мы задаем наши узлы](#nodes.tcl), создаем два маршрутизатора и соединяем их с нашим узлами.
3. Теперь, [мы задаем нашу очередь](#queue.tcl), в ней мы настраиваем параметры и задаем файл трассировки.
4. Запустив программу, мы увидим запуск xgraph с изменением размера окна и длины очереди и nam, который показывает нам нашу моделируемую сеть.
5. Запустив наш скрипт [plot.sh](#plot.sh) мы получим на выходе три файла с нашими графиками:

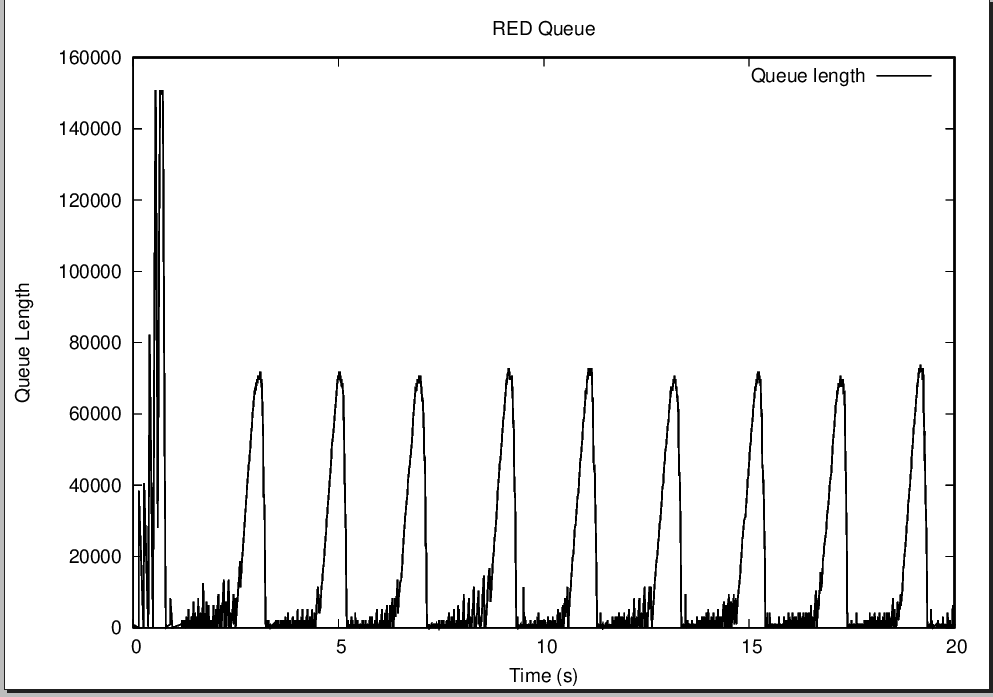


Рисунок 1. Изменение размера длины очереди

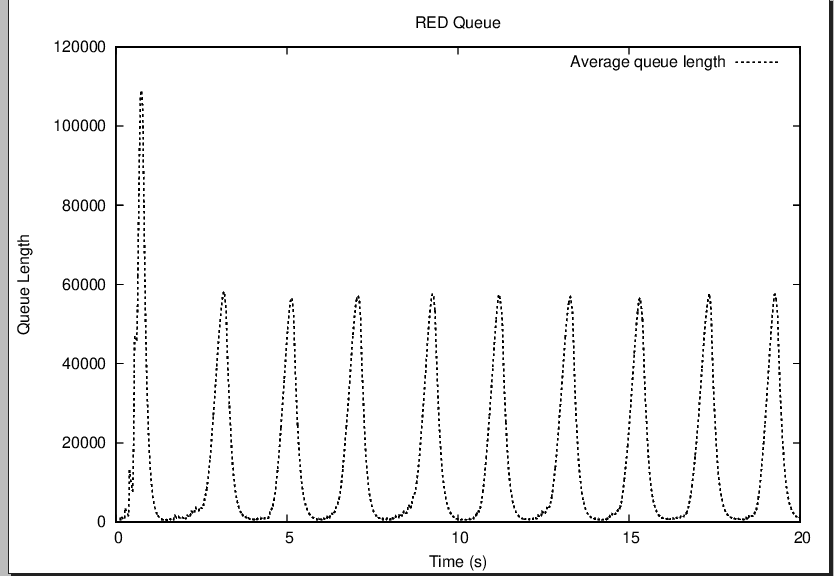


Рисунок 2. Изменение размера средней длины очереди

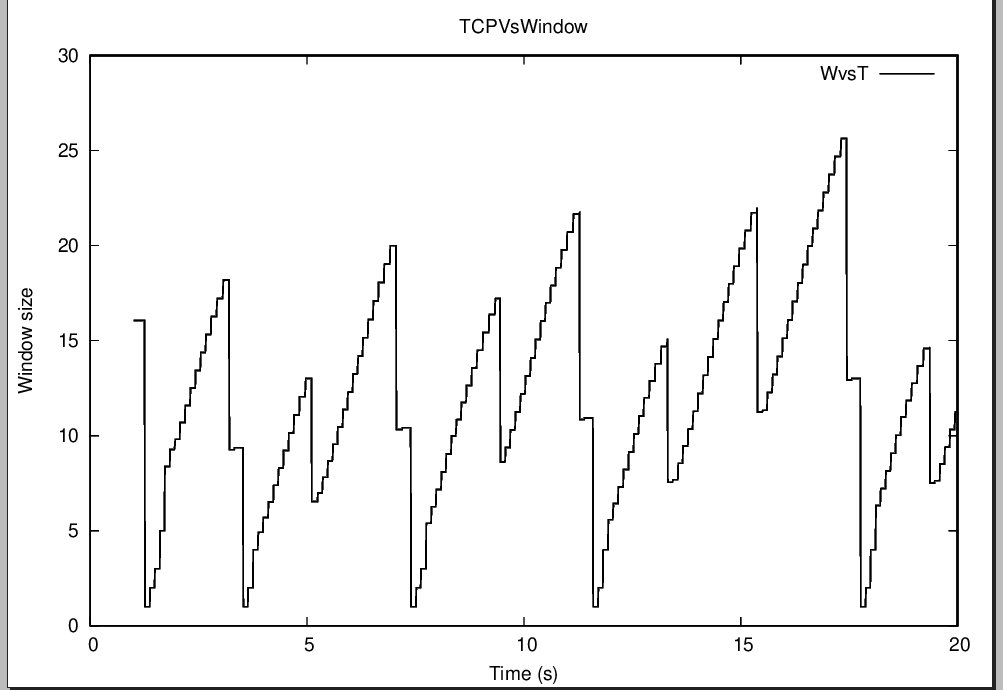


Рисунок 3. Изменение размера окна, так как мы задали потолок окна, то он его не будет превышать

# 4 Выводы

По мере выполнения работы, я приобрел практические навыки по работе с NS2.

# 5 Листинг программ

## 5.1 nodes.tcl

set node\_(r0) [$ns node]  
set node\_(r1) [$ns node]  
$node\_(r0) color "red"  
$node\_(r1) color "red"  
$node\_(r0) label "red"  
  
set n 20  
  
for {set i 0} {$i < $n} {incr i} {  
 set node\_(s$i) [$ns node]  
 $node\_(s$i) color "blue"  
 $node\_(s$i) label "ftp"  
 $ns duplex-link $node\_(s$i) $node\_(r0) 100Mb 20ms DropTail  
  
 set node\_(s[expr $n + $i]) [$ns node]  
 $ns duplex-link $node\_(s[expr $n + $i]) $node\_(r1) 100Mb 20ms DropTail  
}  
  
$ns simplex-link $node\_(r0) $node\_(r1) 20Mb 15ms RED  
$ns simplex-link $node\_(r1) $node\_(r0) 15Mb 20ms DropTail  
  
$ns queue-limit $node\_(r0) $node\_(r1) 300  
$ns queue-limit $node\_(r1) $node\_(r0) 300  
  
  
for {set t 0} {$t < $n} {incr t} {  
 $ns color $t green  
 set tcp($t) [$ns create-connection TCP/Reno $node\_(s$t) TCPSink $node\_(s[expr $n + $t]) $t]  
 $tcp($t) set window\_ 32  
 $tcp($t) set maxcwnd\_ 32  
 $tcp($t) set packetsize\_ 500  
 set ftp($t) [$tcp($t) attach-source FTP]  
}  
  
$ns simplex-link-op $node\_(r0) $node\_(r1) orient right  
$ns simplex-link-op $node\_(r1) $node\_(r0) orient left  
$ns simplex-link-op $node\_(r0) $node\_(r1) queuePos 0  
$ns simplex-link-op $node\_(r1) $node\_(r0) queuePos 0  
  
  
for {set m 0} {$m < $n} {incr m} {  
 $ns duplex-link-op $node\_(s$m) $node\_(r0) orient right  
 $ns duplex-link-op $node\_(s[expr $n + $m]) $node\_(r1) orient left  
}

## 5.4 main.tcl

set ns [new Simulator]  
  
set nf [open out.nam w]  
$ns namtrace-all $nf  
  
source "nodes.tcl"  
source "queue.tcl"

proc plotwindow {tcpsource file} {  
 global ns  
 set time 0.01  
 set now [$ns now]  
 set cwnd [$tcpsource set cwnd\_]  
 puts $file "$now $cwnd"  
 $ns at [expr $now+$time] "plotwindow $tcpsource $file"  
}

for {set r 0} {$r < $n} {incr r} {  
 $ns at 0.0 "$ftp($r) start"  
 $ns at 1.0 "plotwindow $tcp(0) $windowvstime"  
 $ns at 20.0 "$ftp($r) stop"  
}  
  
$ns at 21.0 "finish"

proc finish {} {  
 global ns nf  
 $ns flush-trace  
 close $nf  
 global tchan\_  
 set awkCode {  
 {  
 if ($1 == "Q" && NF>2) {  
 print $2, $3 >> "temp.q";  
 set end $2  
 }  
 else if ($1 == "a" && NF>2)  
 print $2, $3 >> "temp.a";  
 }  
 }  
  
 set f [open temp.queue w]  
 puts $f "TitleText: RED"  
 puts $f "Device: Postscript"  
  
 if { [info exists tchan\_] } {  
 close $tchan\_  
 }  
  
 exec rm -f temp.q temp.a  
 exec touch temp.a temp.q  
  
 exec awk $awkCode all.q  
  
 puts $f \"queue  
 exec cat temp.q >@ $f  
 puts $f \n\"ave\_queue  
 exec cat temp.a >@ $f  
 close $f  
  
 exec xgraph -bb -tk -x time -t "TCPRenoCWND" wvst &  
 exec xgraph -bb -tk -x time -y queue temp.queue &  
 exec nam out.nam &  
 exit 0  
}

$ns run

## 5.6 queue.tcl

set windowvstime [open wvst w]  
set qmon [$ns monitor-queue $node\_(r0) $node\_(r1) [open qm.out w]]  
[$ns link $node\_(r0) $node\_(r1)] queue-sample-timeout  
  
set redq [[$ns link $node\_(r0) $node\_(r1)] queue]  
$redq set qlim\_ 75 150  
$redq set thresh\_ 75  
$redq set maxthresh\_ 150  
$redq set q\_weight\_ 0.002  
$redq set linterm\_ 10  
# $redq set drop-tail\_ true  
  
set tchan\_ [open all.q w]  
$redq trace curq\_  
$redq trace ave\_  
$redq attach $tchan\_

## 5.7 plot.sh

!/usr/bin/gnuplot -persist  
set xrange [0:20]  
  
set terminal postscript eps  
set output "queues.eps"  
set xlabel "Time (s)"  
set ylabel "Queue Length"  
set title "RED Queue"  
plot "temp.q" with lines linestyle 1 lt 1 lw 2 title "Queue length"  
  
set terminal postscript eps  
set output "ave\_queues.eps"  
set xlabel "Time (s)"  
set ylabel "Queue Length"  
set title "RED Queue"  
plot "temp.a" with lines linestyle 2 lt 3 lw 2 title "Average queue length"  
  
set terminal postscript eps  
set output "TCP.eps"  
set xlabel "Time (s)"  
set ylabel "Window size"  
set title "TCPVsWindow"  
plot "wvst" with lines linestyle 1 lt 1 lw 2 title "WvsT"