Лабораторная работа №2

Имитационное моделирование

Серёгина Ирина Андреевна

Содержание

# 1 Цель работы

Ознакомление с протоколом TCP и алгоритмом управления очередью RED, приобретение практических навыков использования.

# 2 Задание

1. Выполнить пример с дисциплиной RED
2. Выполнить упражнение

# 3 Теоретическое введение

Протокол управления передачей (Transmission Control Protocol, TCP) имеет средства управления потоком и коррекции ошибок, ориентирован на установление соединения. Объект мониторинга очереди оповещает диспетчера очереди о поступлении пакета. Диспетчер очереди осуществляет мониторинг очереди.

# 4 Выполнение лабораторной работы

Для начала создаю новую директорию, где буду выполнять лабораторную работу, а также файл, в котором буду прописывать сценарий (рис. 1).

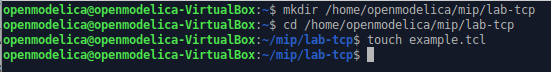


Рис. 1: Создание директории

Требуется приписываю сценарий, реализующий модель согласно установленным требованиям, построить в Xgraph график изменения TCP-окна, график изменения длины очереди и средней длины очереди.

# создание объекта Simulator  
set ns [new Simulator]  
  
# открытие на запись файла out.nam для визуализатора nam  
set nf [open out.nam w]  
  
# все результаты моделирования будут записаны в переменную nf  
$ns namtrace-all $nf  
  
# открытие на запись файла трассировки out.tr  
# для регистрации всех событий  
set f [open out.tr w]  
# все регистрируемые события будут записаны в переменную f  
$ns trace-all $f  
  
# Процедура finish:  
proc finish {} {  
 global tchan\_  
 # подключение кода AWK:  
 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  
 # Запуск xgraph с графиками окна TCP и очереди:  
 exec xgraph -bb -tk -x time -t "TCPRenoCWND" WindowVsTimeReno &  
 exec xgraph -bb -tk -x time -y queue temp.queue &  
 exit 0  
}  
  
# Формирование файла с данными о размере окна TCP:  
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"  
}  
  
# Узлы сети:  
set N 5  
for {set i 1} {$i < $N} {incr i} {  
 set node\_(s$i) [$ns node]  
}  
set node\_(r1) [$ns node]  
set node\_(r2) [$ns node]  
# Соединения:  
$ns duplex-link $node\_(s1) $node\_(r1) 10Mb 2ms DropTail  
$ns duplex-link $node\_(s2) $node\_(r1) 10Mb 3ms DropTail  
$ns duplex-link $node\_(r1) $node\_(r2) 1.5Mb 20ms RED  
$ns queue-limit $node\_(r1) $node\_(r2) 25  
$ns queue-limit $node\_(r2) $node\_(r1) 25  
$ns duplex-link $node\_(s3) $node\_(r2) 10Mb 4ms DropTail  
$ns duplex-link $node\_(s4) $node\_(r2) 10Mb 5ms DropTail  
  
# Агенты и приложения:  
set tcp1 [$ns create-connection TCP/Reno $node\_(s1) TCPSink $node\_(s3) 0]  
$tcp1 set window\_ 15  
set tcp2 [$ns create-connection TCP/Reno $node\_(s2) TCPSink $node\_(s3) 1]  
$tcp2 set window\_ 15  
set ftp1 [$tcp1 attach-source FTP]  
set ftp2 [$tcp2 attach-source FTP]  
  
# Мониторинг размера окна TCP:  
set windowVsTime [open WindowVsTimeReno w]  
set qmon [$ns monitor-queue $node\_(r1) $node\_(r2) [open qm.out w] 0.1];  
[$ns link $node\_(r1) $node\_(r2)] queue-sample-timeout;  
# Мониторинг очереди:  
set redq [[$ns link $node\_(r1) $node\_(r2)] queue]  
set tchan\_ [open all.q w]  
$redq trace curq\_  
$redq trace ave\_  
$redq attach $tchan\_  
  
  
# Добавление at-событий:  
$ns at 0.0 "$ftp1 start"  
$ns at 1.1 "plotWindow $tcp1 $windowVsTime"  
$ns at 3.0 "$ftp2 start"  
$ns at 10 "finish"  
  
# запуск модели  
$ns run

Я запустила код и получила график изменения TCP окна (рис. 2), график изменения длины очереди со средним показателем очереди (рис. 3).

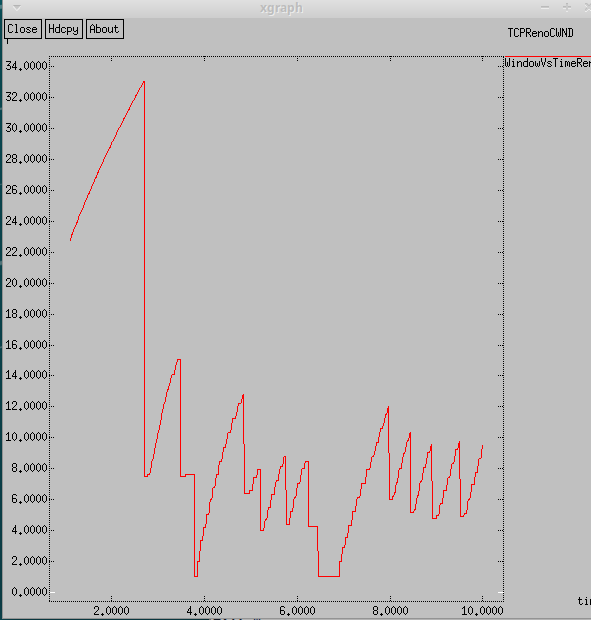


Рис. 2: График изменения TCP окна

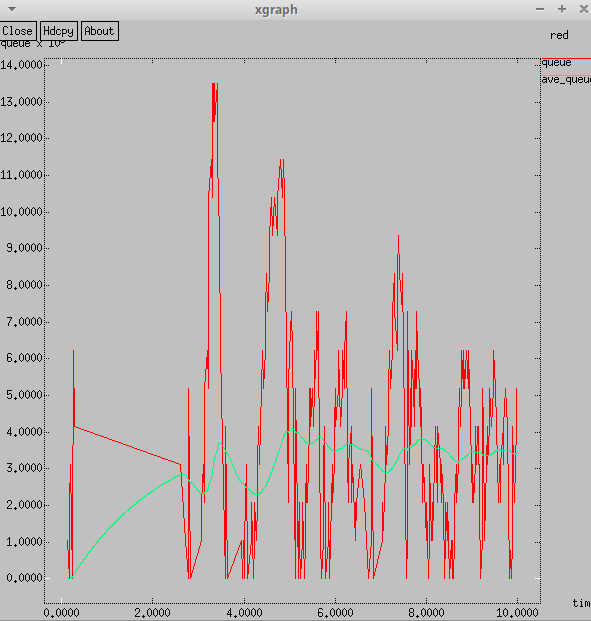


Рис. 3: График динамики длины очереди

Как видно на графике, средняя длины очереди колеблется в промежутке между 2 и 4, ее максимальная длина примерно равна 13,5.

При выполнении упражнения требовалось сменить тип Reno на Newreno (рис. 4).

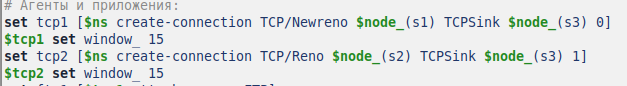


Рис. 4: Вношу изменения в код

Получаю график изменения TCP окна (рис. 5) и изменения длины очереди (рис. 6).

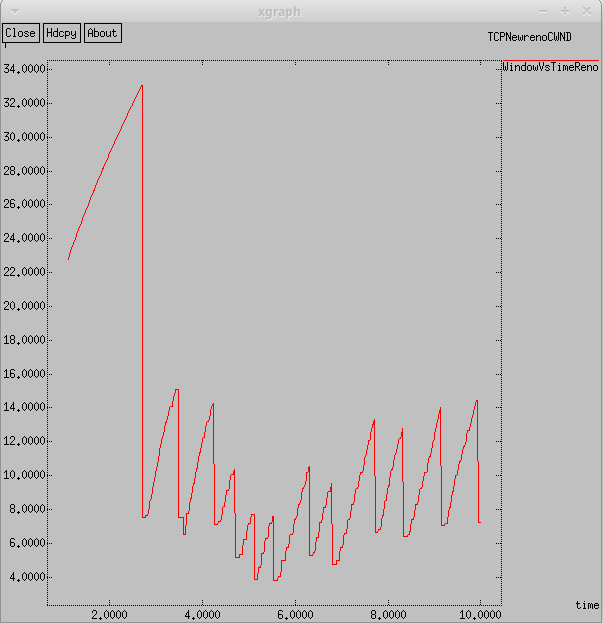


Рис. 5: График изменения TCP окна для Newreno

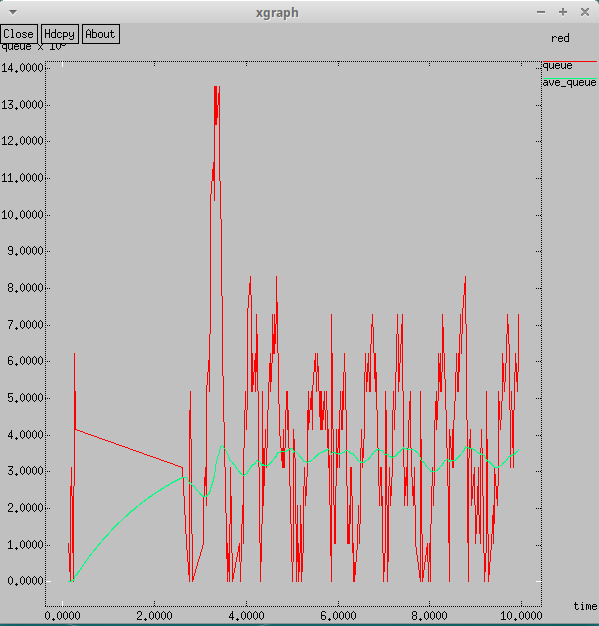


Рис. 6: График динамики длины очереди для Reno

Графики сильно схожи с теми, что я получила раньше, окно увеличивается вплоть до момента потери пакетов.

Теперь поменяю Newreno на Vegas (рис. 7).



Рис. 7: Вношу изменения в код

Получаю график изменения TCP окна (рис. 8) и изменения длины очереди (рис. 9).

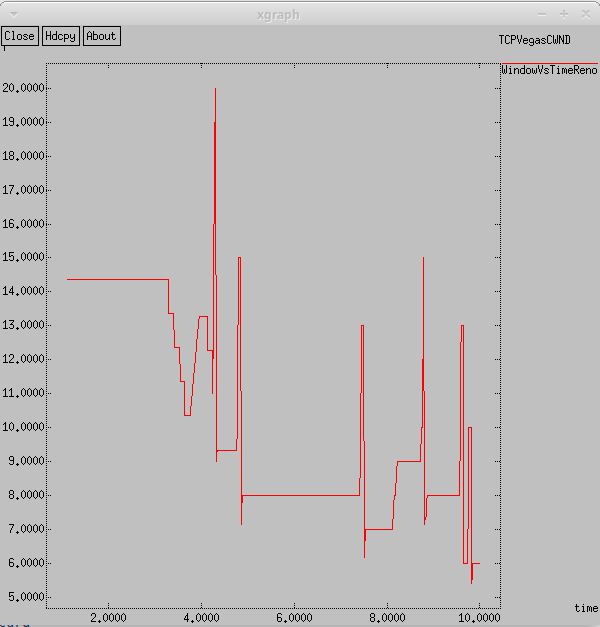


Рис. 8: График изменения TCP окна для Vegas

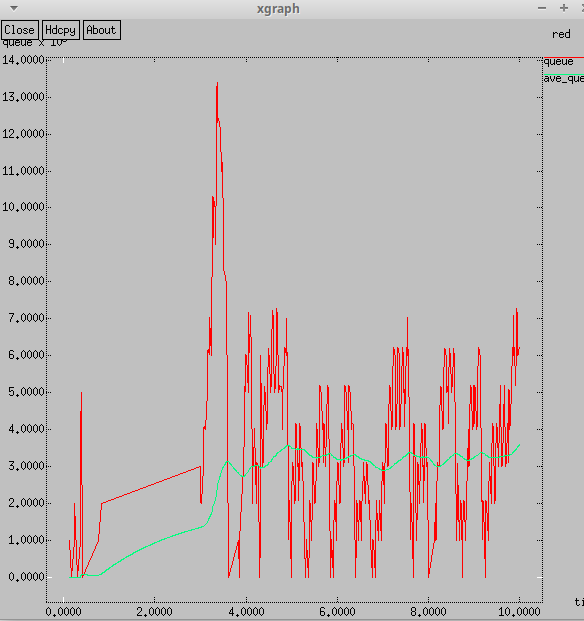


Рис. 9: График динамики длины очереди для Vegas

Здесь можно заметить, что средняя длина очереди находится все ближе к значению 3, все еще не выходя за границы 2 и 4, максимальная длина очереди все та же. На графике размера окна видно, что его максимальный размер меньше, чем у предыдущих, 20, а не 34, то есть Vegas обнаруживает перегрузку до потери пакета и уменьшает размер окна.

Теперь поработаю с изменением отображения окон. Внесу изменения при отображении окон с графиками (изменю цвет фона, цвет траекторий, подписи к осям, подпись траектории в легенде).

Меняю цвет графиков (рис. 10).

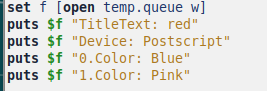


Рис. 10: Изменение цвета графиков

Меняю легенды (рис. 11).

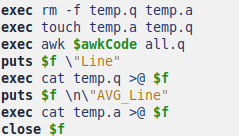


Рис. 11: Изменение легенд

Затем меняю цвет фона и название оси очереди (рис. 12).

Изменение фона и названия осей

Рис. 12: Изменение фона и названия осей

Так выглядит график, в который были внесены изменения (рис. 13).

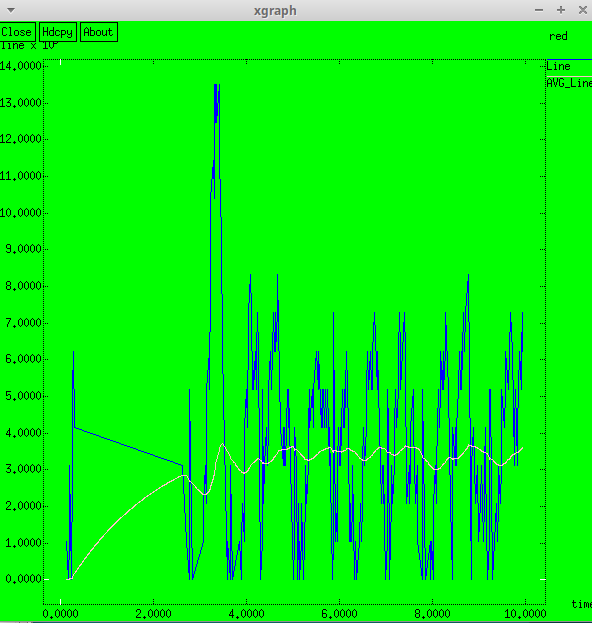


Рис. 13: Измененный график

# 5 Выводы

Я ознакомилась с протоколом TCP и алгоритмом управления очередью RED, приобрела практические навыки использования.