РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

ОТЧЕТ

по лабораторной работе № 2

дисциплина: Моделирование информационных процессов

Студент: Николаев Александр Викторович

Группа: НФИбд-01-17

**МОСКВА**

2020 г.

**Цель работы**

Приобретение навыков работы с TCP. Изучение мониторинга очередей. Обучение работе с дисциплиной RED. Построение графиков и заключение соответствующих выводов с помощью xgraph.

**Постановка задачи:** описание моделируемой сети:

* сеть состоит из 6 узлов;
* между всеми узлами установлено дуплексное соединение с различными пропускной способностью и задержкой 10 мс (см. рисунок 1);
* узел r1 использует очередь с дисциплиной RED для накопления пакетов, максимальный размер которой составляет 25;
* TCP-источники на узлах s1 и s2 подключаются к TCP-приёмнику на узле s3;
* генераторы трафика FTP прикреплены к TCP-агентам.

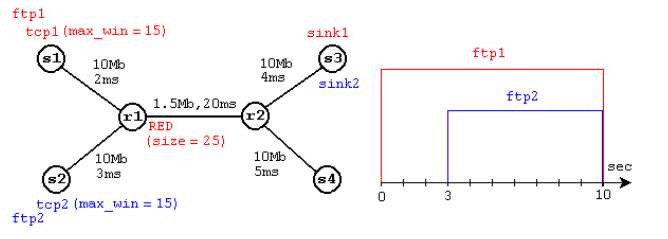


Рисунок . Схема сети

Требуется разработать сценарий, реализующий модель согласно рис. 1, построить в Xgraph график изменения TCP-окна, график изменения длины очереди и средней длины очереди, а также выполнить упражнение.

**Выполнение работы**

См. листинг кода 1 в приложении. Результаты см. рисунки дальше.

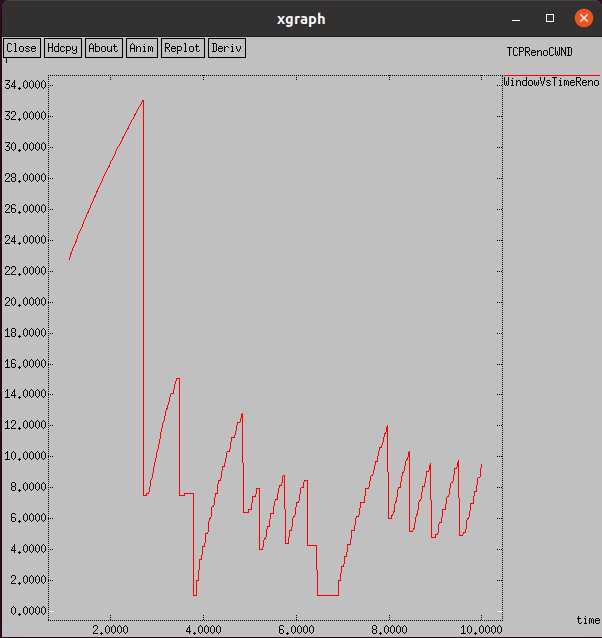


Рисунок . График динамики размера окна TCP (Reno)

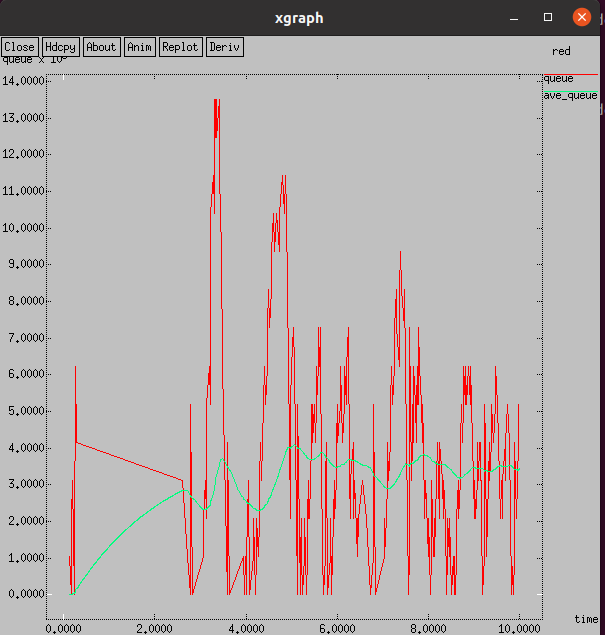


Рисунок . График динамики длины очереди и средней длины очереди (Reno)

**Упражнение**

* Изменить в модели на узле s1 тип протокола TCP с Reno на NewReno, затем на Vegas. Сравнить и пояснить результаты.
* – Внести изменения при отображении окон с графиками (изменить цвет фона, цвет траекторий, подписи к осям, подпись траектории в легенде).

**Выполнение упражнения**

Для изменения в модели тип протокола необходимо изменить всего одну строчку кода, не будем на этом заострять внимание и сделаем выводы о результатах. Так же на рис. 7. Продемонстрированы изменения при отображении окон (изменен background, foreground, curve color и axis labels). Они делаются несложно, достаточно правильно внести ключи и значения в соответствующий файл, т.е. подать xgraph правильные опции.

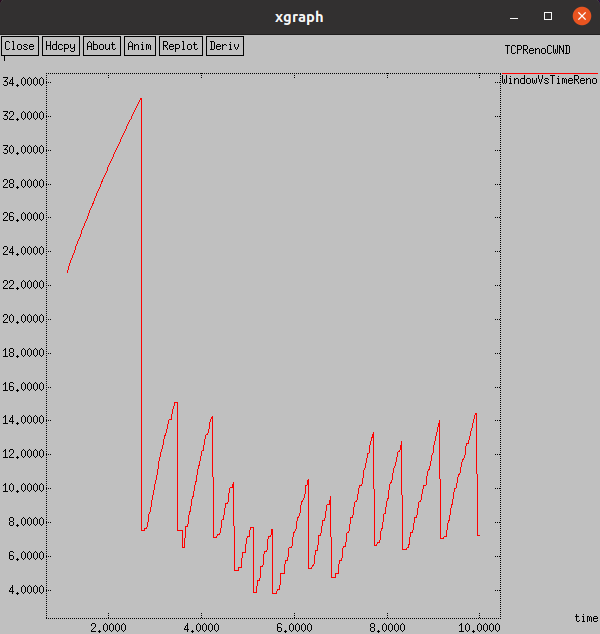
****

Рисунок . График динамики размера окна TCP (Newreno)

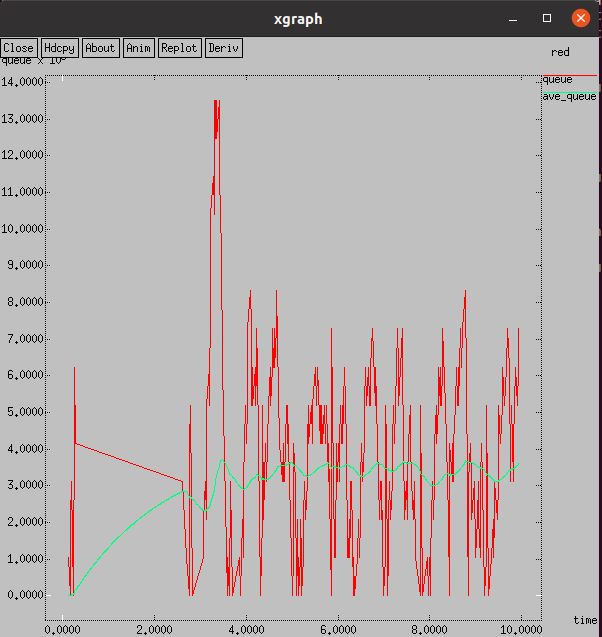
****

Рисунок . График динамики длины очереди и средней длины очереди (Newreno)

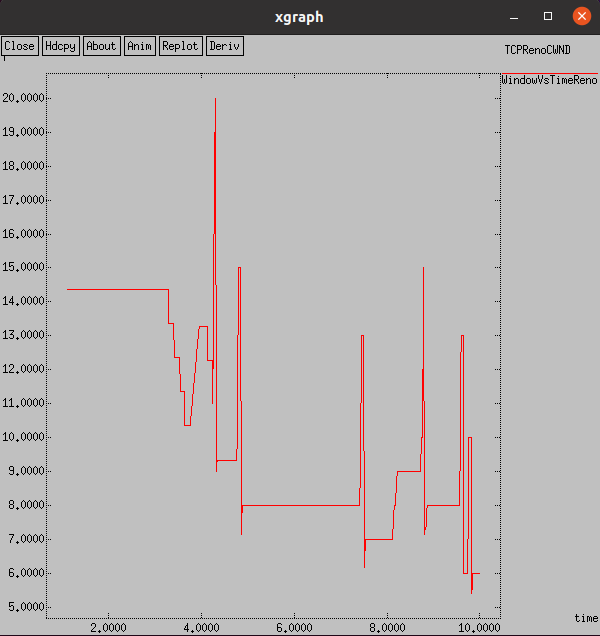
****

Рисунок . График динамики размера окна TCP (Vegas)

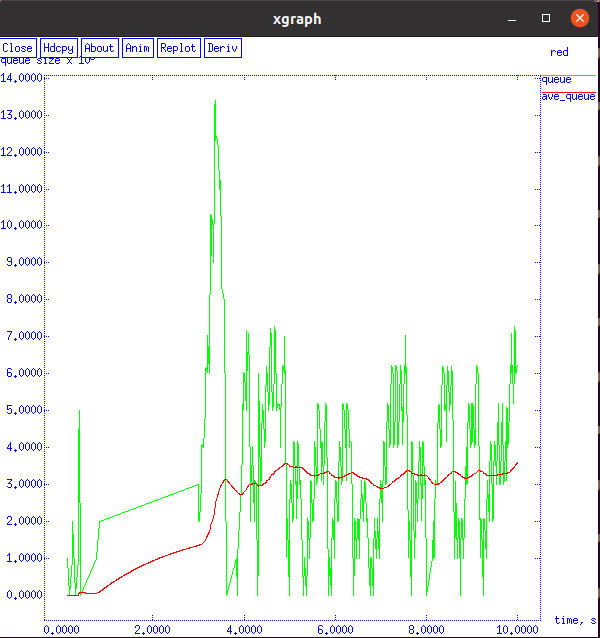
****

Рисунок . График динамики длины очереди и средней длины очереди (Vegas)

Сделаем некоторые выводы о различиях. Для начало, Newreno мало отличается от Reno, и является просто более улучшенной версией. Поэтому очень сложно найти различия между ними. Тем не менее, Newreno быстрее восстанавливает подачу пакетов, за счет чего размер окна не находится в стагнации. Vegas уже существенно отличается от Reno, поскольку основной акцент делает на задержку в передачи пакетов, а не в их потери. Соответственно и очередь меньше, и окно меньше.

**Вывод**

В ходе выполнения лабораторный работы мы познакомились с дисциплиной RED, научились пользоваться Xgraph. Посмотрели на различные типы TCP агентов и разницу в их поведении при мониторинге очереди.

**Приложение.**

**Листинг 1. (стандартный TCP/Reno агент)**

set ns [new Simulator]

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"

}

proc finish {} {

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" WindowVsTimeReno &

#exec xgraph -bb -tk -x time -y queue temp.queue &

exit 0

}

#Model

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]

$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

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"

$ns run

Поскольку листинги остальных программ отличаются мало, я позволю себе предоставить ещё один с изменением интерфейса вывода графика и типа TCP с Reno на Vegas. Остальные листинги можно найти в приложенном архиве программ и убедиться, что всё работает.

**Листинг 2. (Vegas, change xgraph settings)**

set ns [new Simulator]

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"

}

proc finish {} {

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

puts $f "0.Color: green"

puts $f "1.Color: red"

puts $f "Foreground: blue"

puts $f "XUnitText: time, s"

puts $f "YUnitText: queue size"

close $f

#exec xgraph -bb -tk -x time -t "TCPRenoCWND" WindowVsTimeReno &

exec xgraph -bg white -bb -tk temp.queue &

exit 0

}

#Model

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]

$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

set tcp1 [$ns create-connection TCP/Vegas $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"

$ns run