

Отчёт по лабораторной работе №2

**Исследование протокола TCP и алгоритма управления очередью
RED**

Надежда Александровна Рогожина

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	21
	Список литературы	22

Список иллюстраций

4.1	Создание нового файла	9
4.2	Реализация модели	10
4.3	Реализация модели	11
4.4	Реализация модели	12
4.5	График изменения размера окна Reno	13
4.6	График фактической и средней длины очереди Reno	14
4.7	Изменение визуализации графика фактической и средней длины очереди Reno	15
4.8	Изменение визуализации графика изменения размера окна Reno	16
4.9	График фактической и средней длины очереди NewReno	17
4.10	График изменения размера окна NewReno	18
4.11	График фактической и средней длины очереди Vegas	19
4.12	График изменения размера окна Vegas	20

Список таблиц

3.1	Описание некоторых протоколов передачи данных	8
-----	---	---

1 Цель работы

Проанализировать разницу между 3-мя протоколами передачи данных: - TCP Reno - TCP NewReno - TCP Vegas

2 Задание

На основе приведенной в лабораторной работе модели с 6 узлами, изменить типы протоколов и проанализировать результаты. Поменять визуализацию графиков.

3 Теоретическое введение

Протокол управления передачей (Transmission Control Protocol, TCP) имеет средства управления потоком и коррекции ошибок, ориентирован на установление соединения.

- Флаг Указатель срочности (Urgent Pointer, URG) устанавливается в 1 в случае использования поля Указатель на срочные данные.
- Флаг Подтверждение (Acknowledgment, ACK) устанавливается в 1 в случае, если поле Номер подтверждения (Acknowledgement Number) содержит данные. В противном случае это поле игнорируется.
- Флаг Выталкивание (Push, PSH) означает, что принимающий стек TCP должен немедленно информировать приложение о поступивших данных, а не ждать, пока буфер заполнится.
- Флаг Сброс (Reset, RST) используется для отмены соединения из-за ошибки приложения, отказа от неверного сегмента, попытки создать соединение при отсутствии затребованного сервиса.
- Флаг Синхронизация (Synchronize, SYN) устанавливается при инициировании соединения и синхронизации порядкового номера.
- Флаг Завершение (Finished, FIN) используется для разрыва соединения. Он указывает, что отправитель закончил передачу данных.

Управление потоком в протоколе TCP осуществляется при помощи скользящего окна переменного размера: - поле Размер окна (Window) (длина 16 бит) содержит количество байт, которое может быть послано после байта, получение

которого уже подтверждено; - если значение этого поля равно нулю, это означает, что все байты, вплоть до байта с номером Номер подтверждения - 1, получены, но получатель отказывается принимать дальнейшие данные; - разрешение на дальнейшую передачу может быть выдано отправкой сегмента с таким же значением поля Номер подтверждения и ненулевым значением поля Размер окна.

Регулирование трафика в TCP: - контроль доставки — отслеживает заполнение входного буфера получателя с помощью параметра Размер окна (Window); - контроль перегрузки — регистрирует перегрузку канала и связанные с этим потери, а также понижает интенсивность трафика с помощью Окна перегрузки (Congestion Window, CWnd) и Порога медленного старта (Slow Start Threshold, SSThresh).

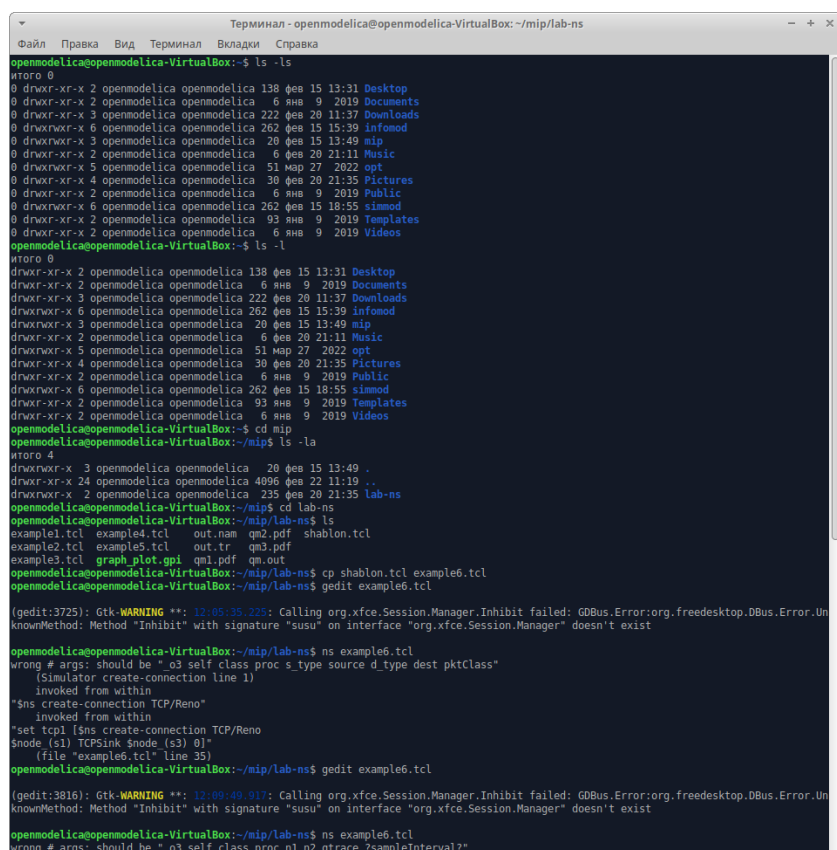
Например, в табл. 3.1 приведено краткое описание анализируемых протоколов передачи данных.

Таблица 3.1: Описание некоторых протоколов передачи данных

Характеристика	TCP Reno	TCP NewReno	TCP Vegas
Основной подход	Реакция на потери пакетов	Реакция на потери пакетов	Управление на основе задержек
Множественные потери	Неэффективно	Эффективно	Нет потерь (проактивно)
Агрессивность	Высокая	Высокая	Низкая
Стабильность	Средняя	Средняя	Высокая
Сложность реализации	Простая	Средняя	Сложная

4 Выполнение лабораторной работы

Первым делом мы копируем шаблон в новый файл и открываем его на редактирование (рис. 4.1).



```
Терминал - openmodelica@openmodelica-VirtualBox: ~/mip/lab-ns
Файл  Правка  Вид  Терминал  Вкладки  Справка
openmodelica@openmodelica-VirtualBox:~$ ls -ls
итого 0
0 drwxr-xr-x 2 openmodelica openmodelica 138 фев 15 13:31 Desktop
0 drwxr-xr-x 2 openmodelica openmodelica 6 янв 9 2019 Documents
0 drwxr-xr-x 3 openmodelica openmodelica 222 фев 20 11:37 Downloads
0 drwxr-xr-x 6 openmodelica openmodelica 262 фев 15 15:39 infomod
0 drwxr-xr-x 3 openmodelica openmodelica 20 фев 15 13:49 mip
0 drwxr-xr-x 2 openmodelica openmodelica 6 фев 20 21:11 Music
0 drwxr-xr-x 5 openmodelica openmodelica 51 мар 27 2022 opt
0 drwxr-xr-x 4 openmodelica openmodelica 30 фев 20 21:35 Pictures
0 drwxr-xr-x 2 openmodelica openmodelica 6 янв 9 2019 Public
0 drwxr-xr-x 6 openmodelica openmodelica 262 фев 15 18:55 simmod
0 drwxr-xr-x 2 openmodelica openmodelica 93 янв 9 2019 Templates
0 drwxr-xr-x 2 openmodelica openmodelica 6 янв 9 2019 Videos
openmodelica@openmodelica-VirtualBox:~$ ls -lt
итого 0
drwxr-xr-x 2 openmodelica openmodelica 138 фев 15 13:31 Desktop
drwxr-xr-x 2 openmodelica openmodelica 6 янв 9 2019 Documents
drwxr-xr-x 3 openmodelica openmodelica 222 фев 20 11:37 Downloads
drwxr-xr-x 6 openmodelica openmodelica 262 фев 15 15:39 infomod
drwxr-xr-x 3 openmodelica openmodelica 20 фев 15 13:49 mip
drwxr-xr-x 2 openmodelica openmodelica 6 фев 20 21:11 Music
drwxr-xr-x 5 openmodelica openmodelica 51 мар 27 2022 opt
drwxr-xr-x 4 openmodelica openmodelica 30 фев 20 21:35 Pictures
drwxr-xr-x 2 openmodelica openmodelica 6 янв 9 2019 Public
drwxr-xr-x 6 openmodelica openmodelica 262 фев 15 18:55 simmod
drwxr-xr-x 2 openmodelica openmodelica 93 янв 9 2019 Templates
drwxr-xr-x 2 openmodelica openmodelica 6 янв 9 2019 Videos
openmodelica@openmodelica-VirtualBox:~$ cd mip
openmodelica@openmodelica-VirtualBox:~/mip$ ls -la
итого 4
drwxr-xr-x 3 openmodelica openmodelica 20 фев 15 13:49 .
drwxr-xr-x 24 openmodelica openmodelica 4096 фев 22 11:19 ..
drwxr-xr-x 2 openmodelica openmodelica 235 фев 20 21:35 lab-ns
openmodelica@openmodelica-VirtualBox:~/mip$ cd lab-ns
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ ls
example1.tcl example4.tcl out.nam qm2.pdf shablon.tcl
example2.tcl example5.tcl out.tr qm3.pdf
example3.tcl graph_plot.gpi qm1.pdf qm.out
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ cp shablon.tcl example6.tcl
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ gedit example6.tcl

(gedit:3725): Gtk-WARNING **: 12:05:35.225: Calling org.xfce.Session.Manager.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: Method "Inhibit" with signature "ssus" on interface "org.xfce.Session.Manager" doesn't exist

openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ ns example6.tcl
wrong # args: should be "o3 self class proc s_type source d_type dest pktClass"
(simulator create-connection line 1)
invoked from within
"$ns create-connection TCP/Reno"
invoked from within
"set tcp1 [$ns create-connection TCP/Reno
$node ($s1) TCPSink $node ($s3) 0]"
(file "example6.tcl" line 35)
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ gedit example6.tcl

(gedit:3816): Gtk-WARNING **: 12:09:49.917: Calling org.xfce.Session.Manager.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: Method "Inhibit" with signature "ssus" on interface "org.xfce.Session.Manager" doesn't exist

openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ ns example6.tcl
wrong # args: should be "o3 self class proc n1 n2 qtrace ?sampleInterval?"
```

Рис. 4.1: Создание нового файла

В тексте лабораторной работы был дан код алгоритма, который было необходимо реализовать. Мы его переписали в новый созданный файл (рис. 4.2, рис. 4.3, рис. 4.4).

```

/home/openmodelica/mip/lab-ns/example6.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка

# 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

# Узлы сети:
set N 5
for {set i 1} {$i < $N} {incr i} {
  set node_($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_

```

Рис. 4.2: Реализация модели

```

/home/openmodelica/mip/lab-ns/example6.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка

$redq trace ave
$redq attach $tchan_

# процедура finish закрывает файлы трассировки
# и запускает визуализатор nam
proc finish {} {
    global ns f nf
    $ns flush-trace
    close $f
    close $nf
    exec nam out.nam &
    exit 0
}

# at-событие для планировщика событий, которое запускает
# процедуру finish через 5 с после начала моделирования
# Добавление at-событий:
$ns at 0.0 "$ftpl start"
$ns at 1.1 "plotWindow $tcp1 $windowVstime"
$ns at 3.0 "$ftpl2 start"
$ns at 10 "finish"

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"
}

# 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_
}

```

Рис. 4.3: Реализация модели

```

/home/openmodelica/mip/lab-ns/example6.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка

$ns at 10 "finish"

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"
}

# 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

# выполнение кода AWK
exec awk $awkCode all.q
puts $f "\"queue"
exec cat temp.q >@ $f
puts $f "\\ave_queue"
exec cat temp.a >@ $f
close $f

# Запуск xgraph с графиками окна TCP и очереди:
exec xgraph -bb -tk -x time -t "TCP Reno Cwnd" WindowVsTimeReno &
exec xgraph -bb -tk -x time -y queue temp.queue &
exit 0
}

# запуск модели
$ns run

```

Рис. 4.4: Реализация модели

Запустив в терминале команду `ns example6.tcl`, мы получили первый результат (рис. 4.5, рис. 4.6).

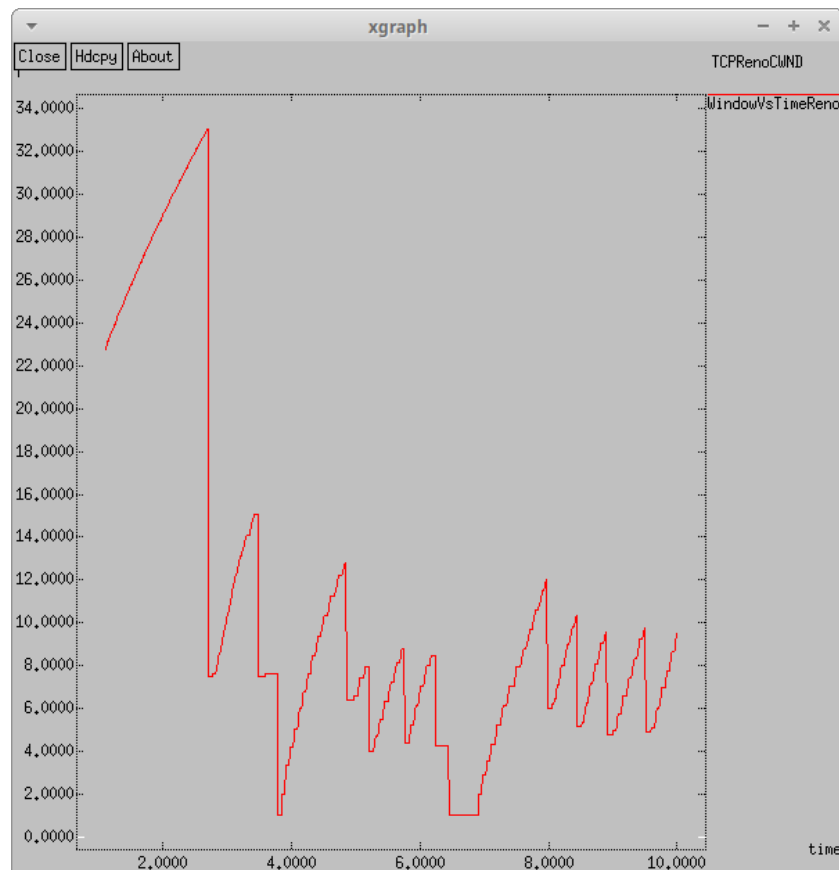


Рис. 4.5: График изменения размера окна Reno

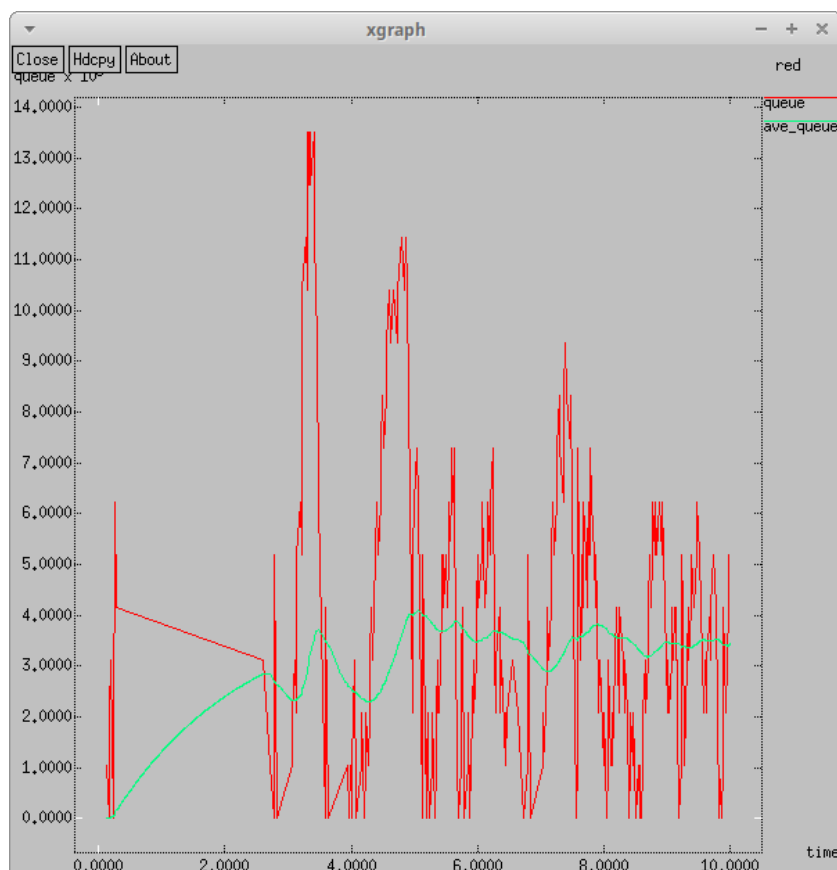


Рис. 4.6: График фактической и средней длины очереди Reno

Для более приятной и понятной визуализации, я изменила график: - `-bg white` для белого фона - `0.Color=purple` для цвета первой рисуемой линии - `1.Color=orange` для цвета второй рисуемой линии - третью линию оставила как есть, красной (рис. 4.7, рис. 4.8).

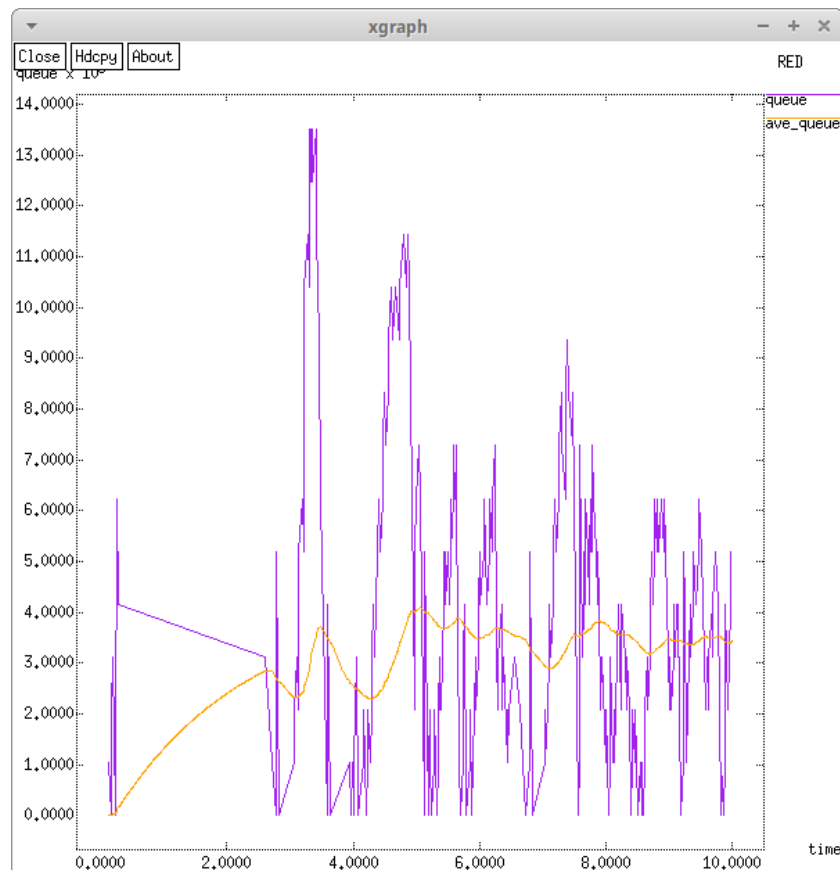


Рис. 4.7: Изменение визуализации графика фактической и средней длины очереди Reno

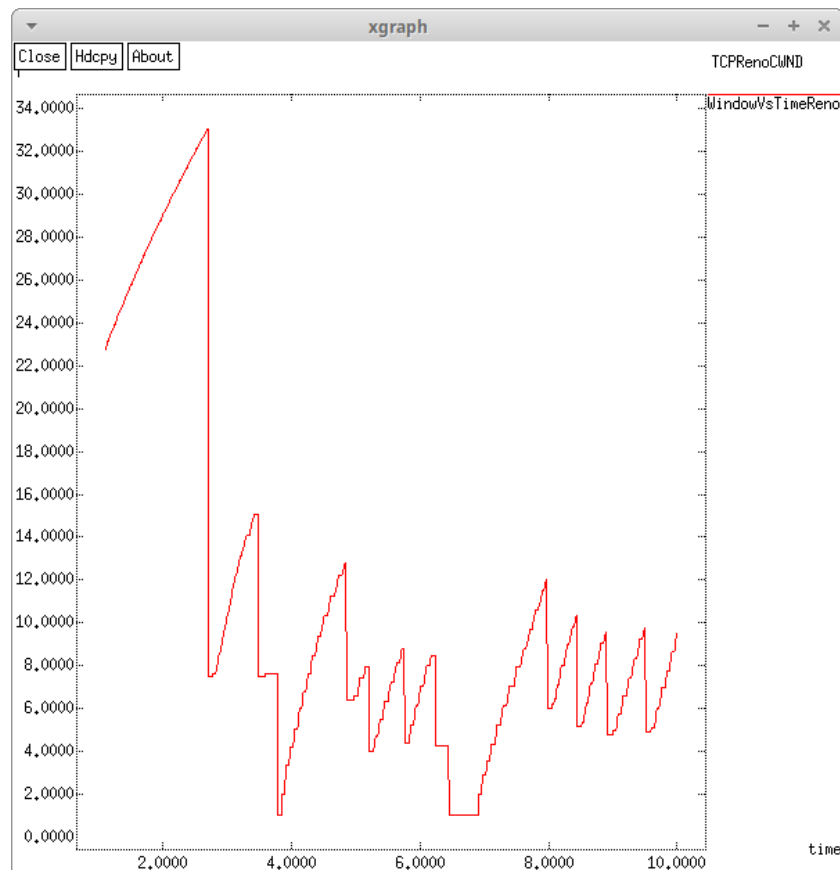


Рис. 4.8: Изменение визуализации графика изменения размера окна Reno

Далее, мы поменяли тип протокола TCP с Reno на NewReno (рис. 4.9, рис. 4.10).

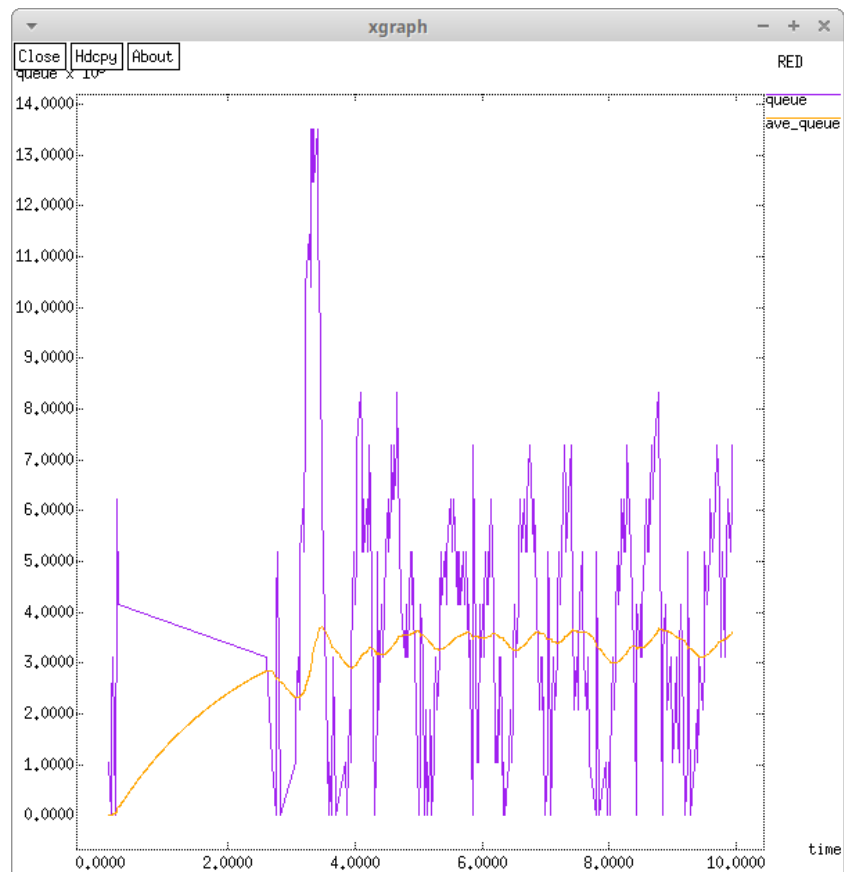


Рис. 4.9: График фактической и средней длины очереди NewReno

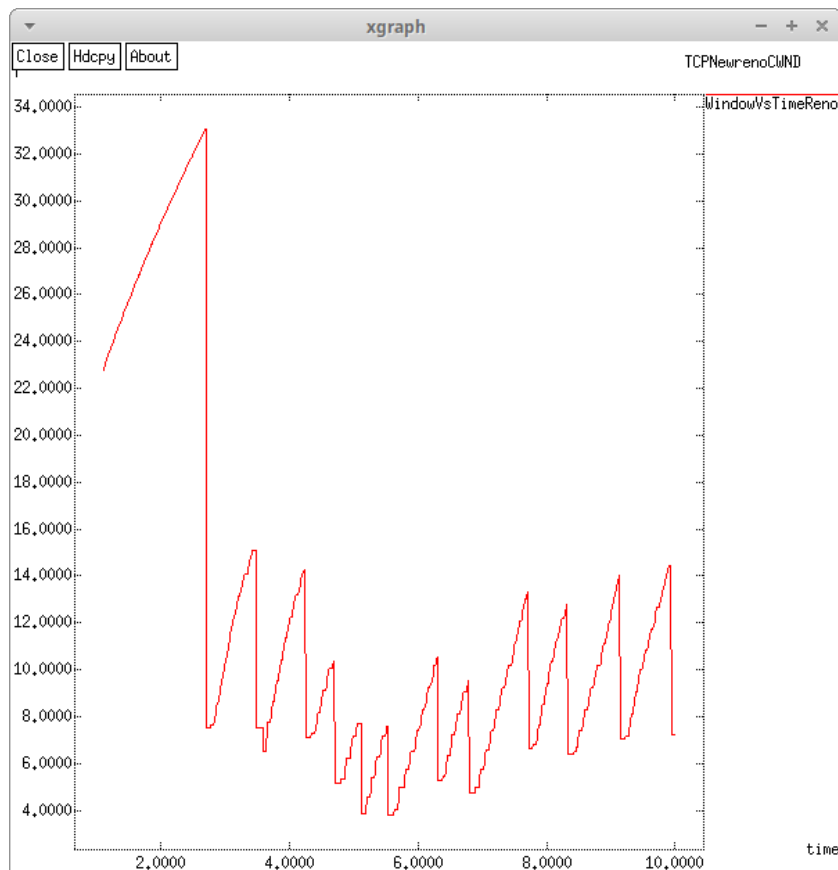


Рис. 4.10: График изменения размера окна NewReno

После, мы поменяли тип протокола TCP с NewReno на Vegas (рис. 4.11, рис. 4.12).

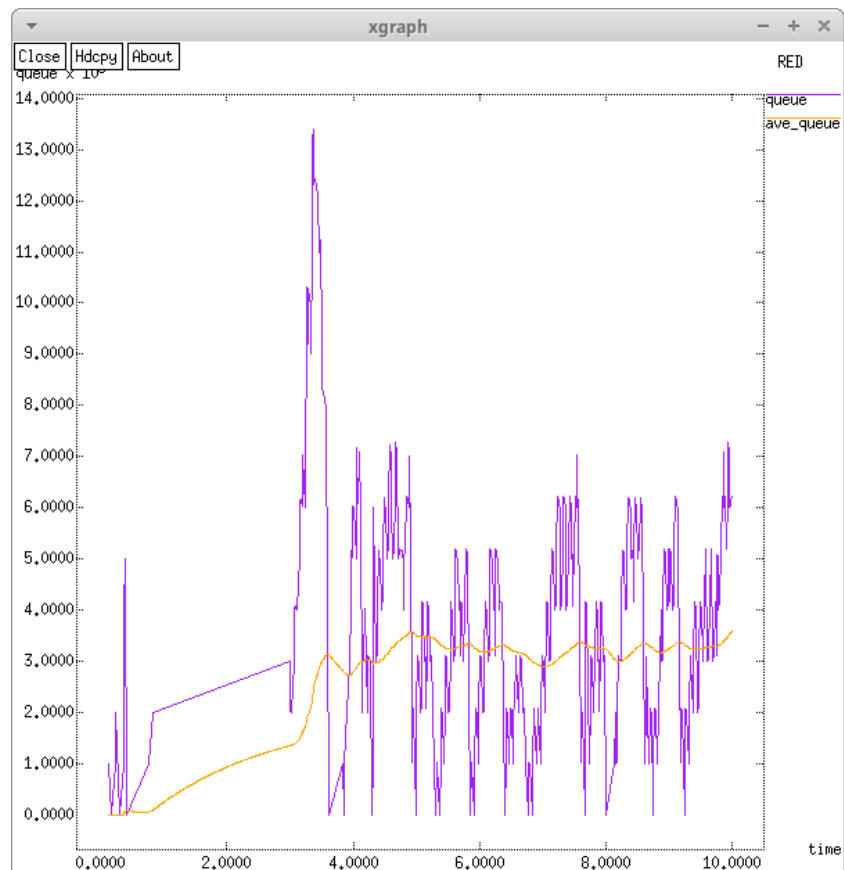


Рис. 4.11: График фактической и средней длины очереди Vegas

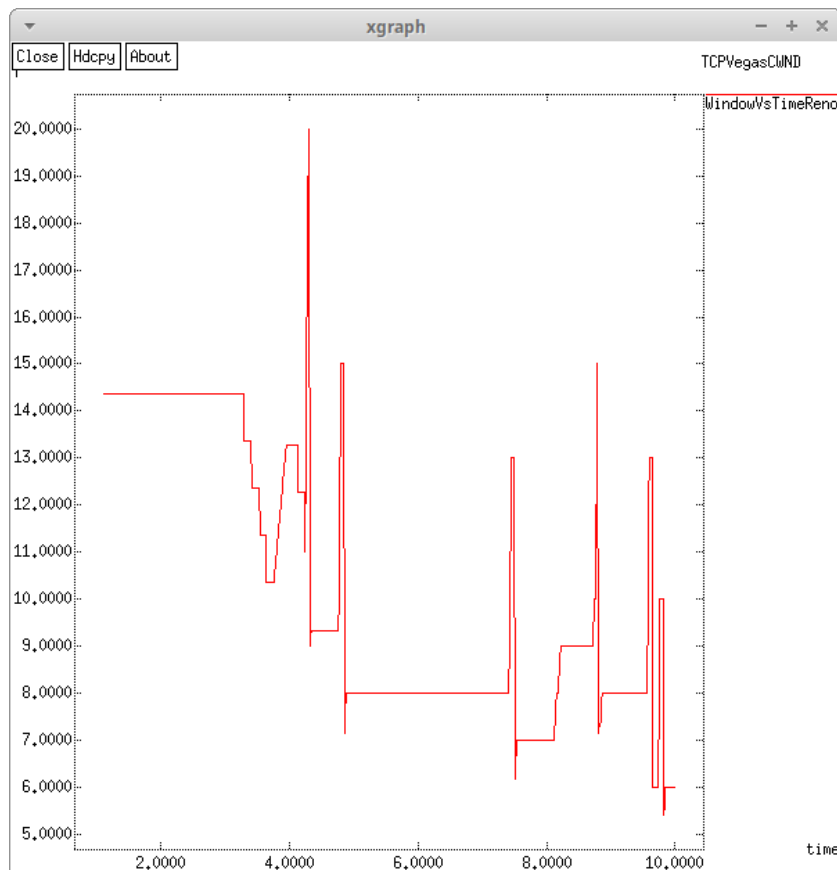


Рис. 4.12: График изменения размера окна Vegas

Т.к. у нас Reno и NewReno похожи по подходу, графики тоже похожи, но у NewReno стабильнее график и меньше разброс значений, т.к. у Reno увеличение окна происходит линейно и регулируется только при потере пакетов, а у NewReno есть регуляризация принятия пакетов после потери (увеличенный быстрый старт), а у TCP Vegas совершенно иной подход к регуляризации трафика, в следствие чего среднее число пакетов в очереди выглядит еще более стабильно. А также пик размера окна у него сильно меньше, чем у двух предыдущих алгоритмов.

5 Выводы

В ходе лабораторной работы мы определили различия между 3-мя протоколами ТСП и приобрели базовые навыки работы со средством визуализации `xgraph`.

Список литературы