Лабораторная работа №2

Имитационное моделирование

Серёгина Ирина Андреевна

Содержание

Список иллюстраций

Список таблиц

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. 4.1).

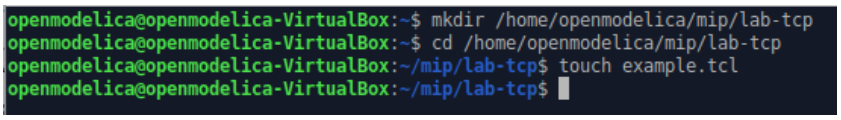

Рис. 4.1: Создание директории

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. 4.2), график изменения длины очереди со средним показателем очереди (рис. 4.3).

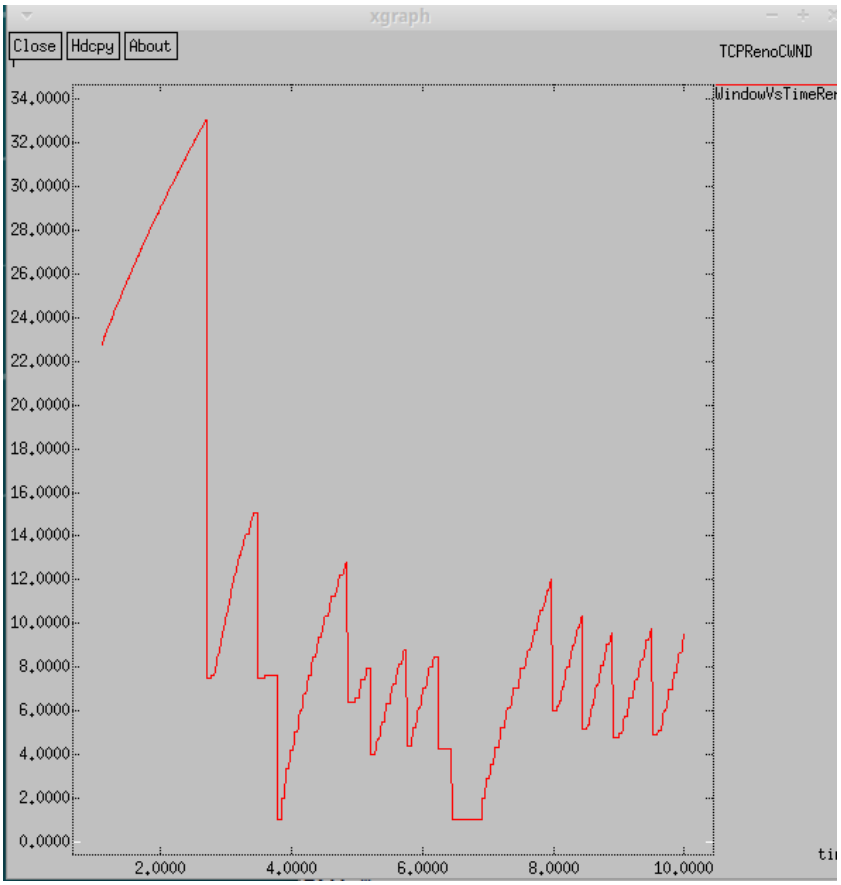

Рис. 4.2: График изменения TCP окна

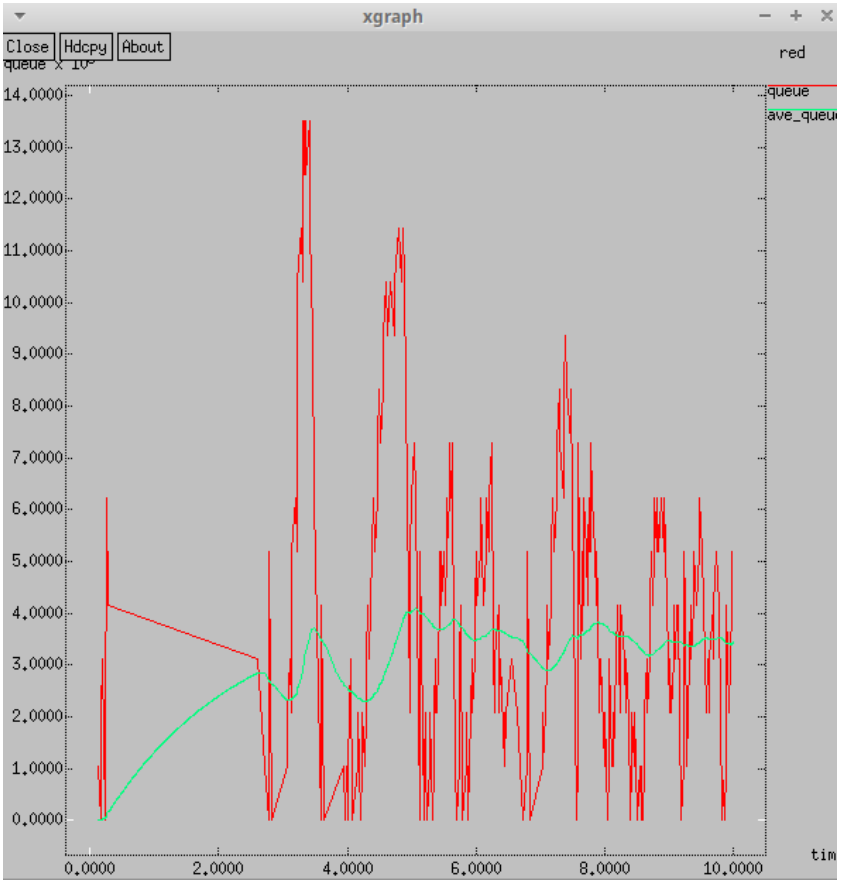

Рис. 4.3: График динамики длины очереди

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. 4.4).

```
# Агенты и приложения:
set tcp1 [$ns create-connection TCP/Newreno $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
```

Рис. 4.4: Вношу изменения в код

Получаю график изменения TCP окна (рис. 4.5) и изменения длины очереди (рис. 4.6).

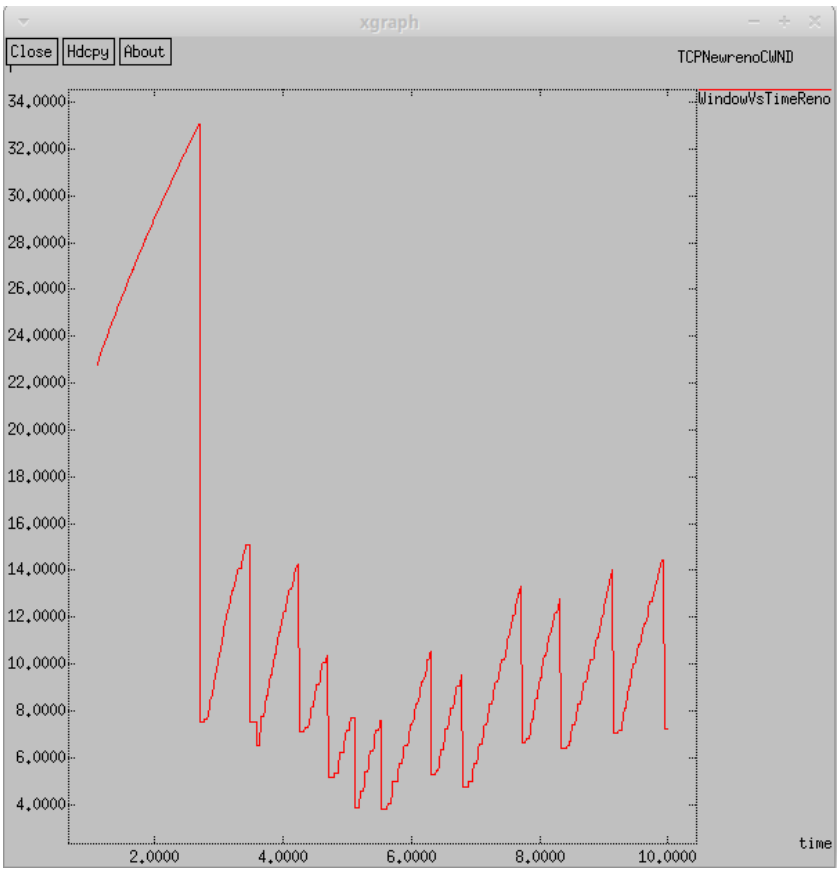

Рис. 4.5: График изменения TCP окна для Newreno

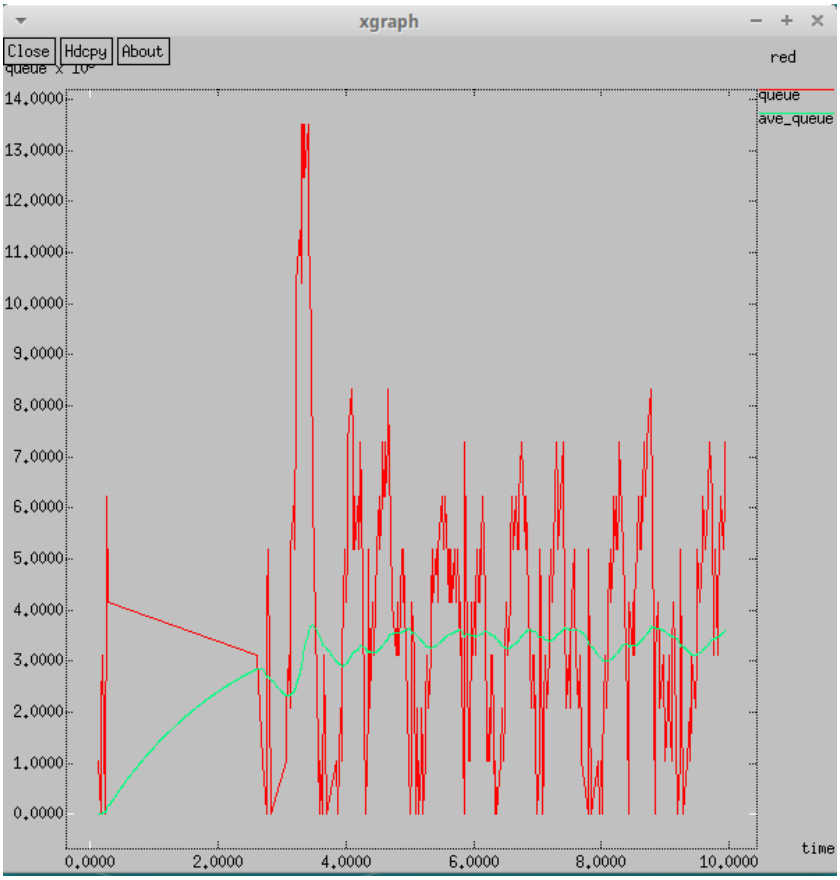

Рис. 4.6: График динамики длины очереди для Reno

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

Теперь поменяю Newreno на Vegas (рис. 4.7).

```
# Агенты и приложения:
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
```

Рис. 4.7: Вношу изменения в код

Получаю график изменения TCP окна (рис. 4.8) и изменения длины очереди (рис. 4.9).

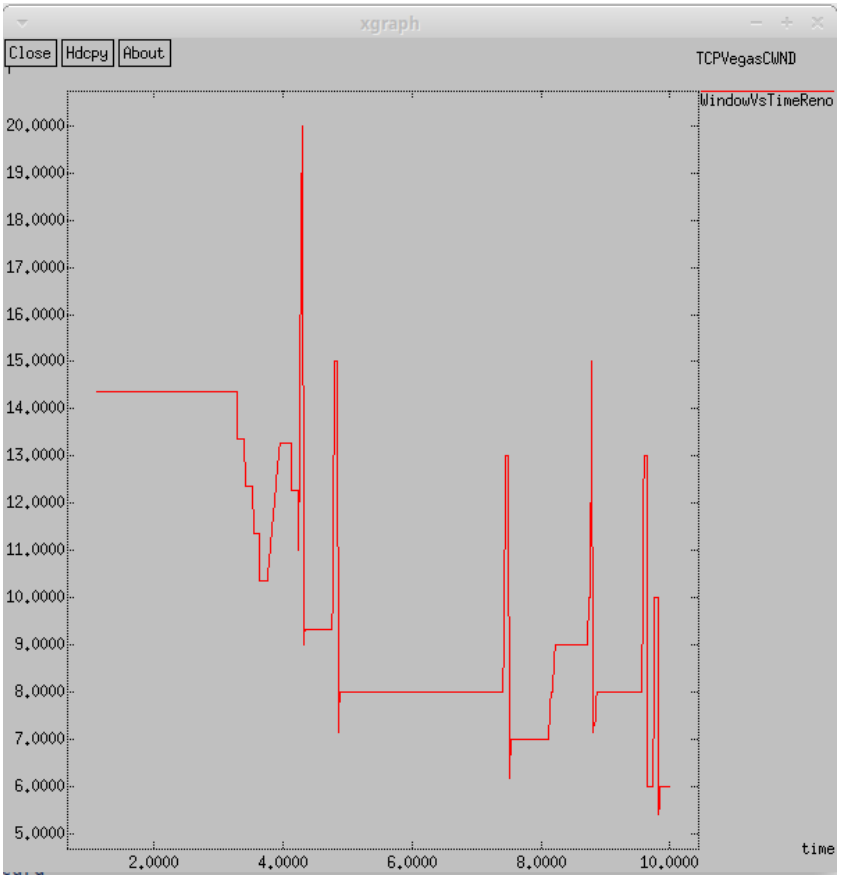

Рис. 4.8: График изменения TCP окна для Vegas

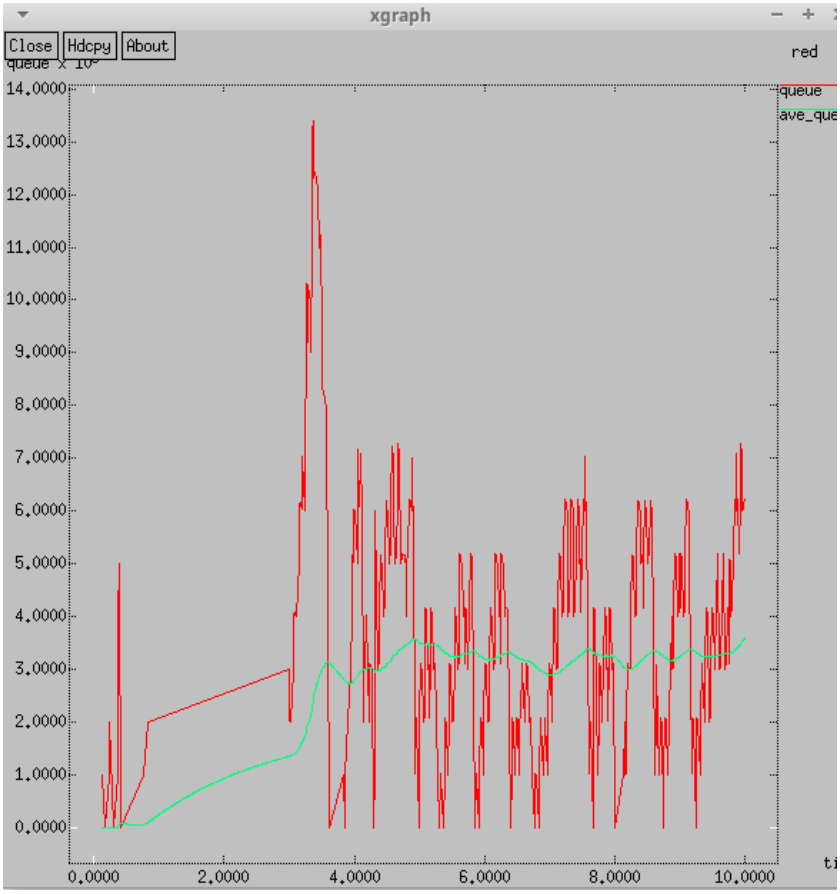

Рис. 4.9: График динамики длины очереди для Vegas

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. 4.10).

```
set f [open temp.queue w]
puts $f "TitleText: red"
puts $f "Device: Postscript"
puts $f "0.Color: Blue"
puts $f "1.Color: Pink"
```

Рис. 4.10: Изменение цвета графиков

Меняю легенды (рис. 4.11).

```
exec rm -f temp.q temp.a
exec touch temp.a temp.q
exec awk $awkCode all.q
puts $f \"Line"
exec cat temp.q >@ $f
puts $f \n\"AVG_Line"
exec cat temp.a >@ $f
close $f
```

Рис. 4.11: Изменение легенд

Затем меняю цвет фона и название оси очереди (рис. 4.12).

```
# Запуск xgraph с графиками окна TCP и очереди:
exec xgraph -bg green -bb -tk -x time -t "TCPRenoCWND" WindowVsTimeReno &
exec xgraph -bg green -bb -tk -x time -y line temp.queue &
```

Рис. 4.12: Изменение фона и названия осей

Так выглядит график, в который были внесены изменения (рис. 4.13).

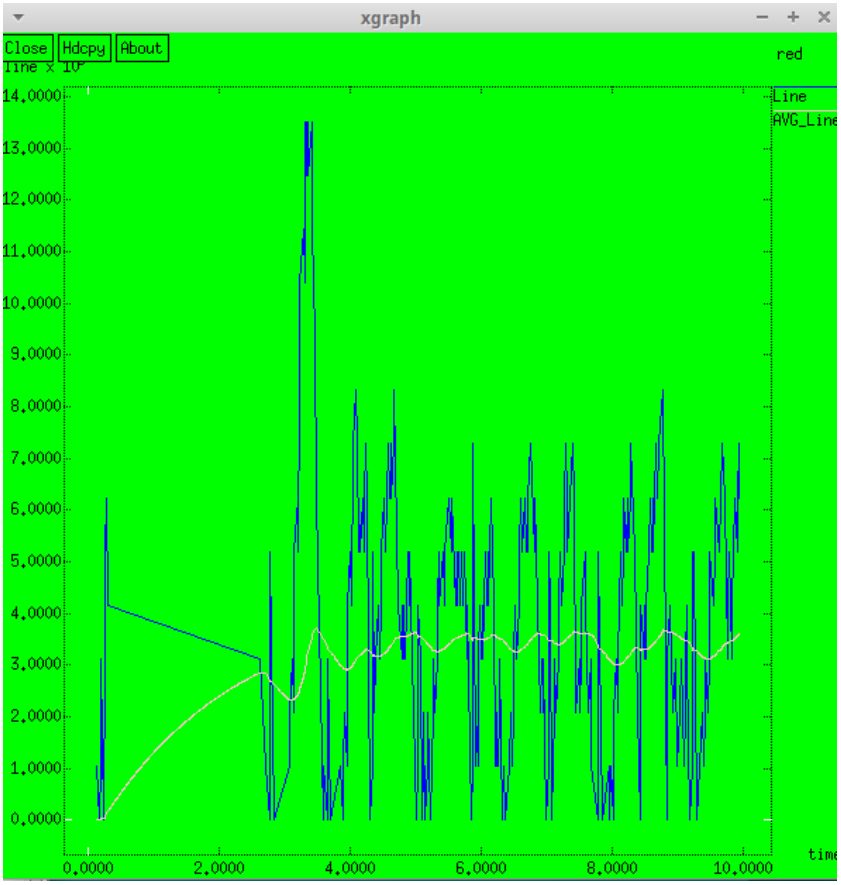

Рис. 4.13: Измененный график

5 Выводы

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