

Лабораторная работа №2

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

Акопян Сатеник

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	20
	Список литературы	21

Список таблиц

Список иллюстраций

4.1	рисунок 1	8
4.2	рисунок 2	9
4.3	рисунок 3	9
4.4	рисунок 4	10
4.5	рисунок 5	11
4.6	рисунок 6	12
4.7	рисунок 7	12
4.8	рисунок 8	13
4.9	рисунок 9	14
4.10	рисунок 10	15
4.11	рисунок 11	16
4.12	рисунок 12	17
4.13	рисунок 13	18
4.14	рисунок 14	19

1 Цель работы

Исследование протокола TCP и алгоритма управления очередью RED.

2 Задание

Описание моделируемой сети:

- сеть состоит из 6 узлов;
- между всеми узлами установлено дуплексное соединение с различными пропуск- ной способностью и задержкой 10 мс;
- узел r1 использует очередь с дисциплиной RED для накопления пакетов, макси- мальный размер которой составляет 25;
- ТСП-источники на узлах s1 и s2 подключаются к ТСП-приёмнику на узле s3;
- генераторы трафика FTP прикреплены к ТСП-агентам.

Требуется разработать сценарий, реализующий модель согласно, по- строить в Xgraph график изменения ТСП-окна, график изменения длины очереди и средней длины очереди

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

1. Создается файл tcp.tcl на основе шаблона shablon.tcl с реализацией модели (рис. 4.1, 4.2, 4.3).

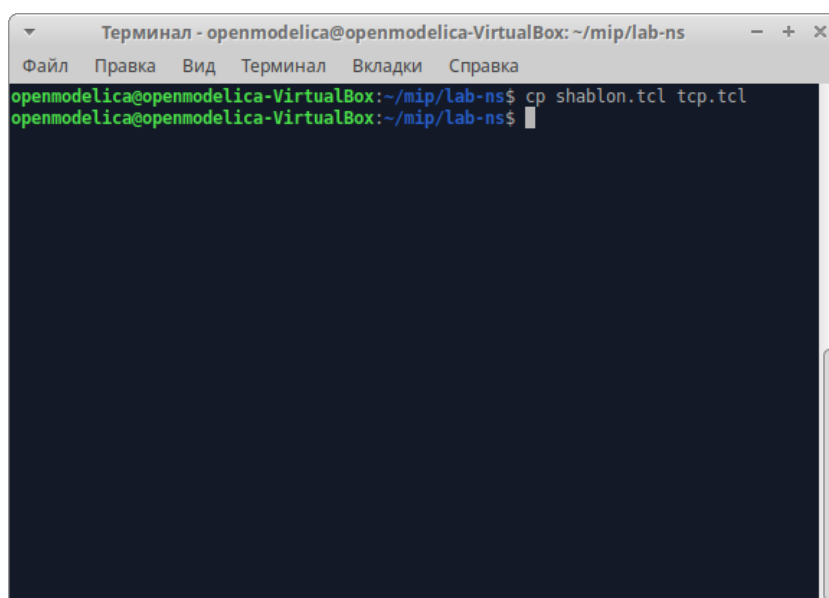


Рис. 4.1: рисунок 1


```
File Edit View Debug Window
# Описание объекта Simulator
set ns [new Simulator]

# открытие на запись файла out.log для визуализатора nam
set of [open out.log w]
# для события-011 вычисления будут выписаны в переменную of
set counter 011

# открытие на запись файла трассировки out.tr
# для регистрации всех событий
set f [open out.tr w]
# все зарегистрированные события будут записаны в переменную f
set trace all $f

# процедура finish закрывает файлы трассировки
# и запускает визуализатор nam
proc finish {} {
    global trace
    # закрываем код AMM:
    set subcode {
        {
            if ($1 == "0" 66 MP-2) {
                print $2, $3 >> "temp.q";
                set end $2
            }
            else if ($1 == "0" 66 MP-2) {
                print $2, $3 >> "temp.n";
            }
        }
    }
    set f [open temp.queue w]
    puts $f "Simulation: end"
    puts $f "Provider Postscript"
    if { [info exists tchan_1] } {
        close tchan_1
    }
    exec ns -f temp.q temp.n
    exec trace temp.n temp.q
    exec set subcode all.q и выполнение кода AMM
    puts $f "Vname"
    exec cat temp.q > $f
    puts $f "Vn queue"
    exec cat temp.n > $f
    close $f
    # вывод в экран - (добавим окно TCP и очередь)
    exec ns/rtr -bb -tk -x time -t "TCPMenuCMD" WindowVTimeInfo &
    exec ns/rtr -bb -tk -x time -y queue temp.queue &
    exit 0
}

# Time setup
set n 5
for {set i 1} {$i < $n} {incr i} {
    set node ($i) [ns node]
}
set node ($1) [ns node]
set node ($2) [ns node]

# Соединение:
ns duplex-link $node ($1) $node ($2) 10Mb 2ms DropTail
ns duplex-link $node ($2) $node ($1) 10Mb 2ms DropTail
ns duplex-link $node ($1) $node ($2) 1Mb 2ms RED
ns queue-limit $node ($1) $node ($2) 25
ns queue-limit $node ($2) $node ($1) 25
ns duplex-link $node ($3) $node ($2) 10Mb 4ms DropTail
ns duplex-link $node ($4) $node ($2) 10Mb 5ms DropTail

# Агенты и приложения:
set tcp1 [ns create-connection TCP/ Reno
    $node ($1) TCPLink $node ($3) 0]
stop set window 15
set tcp2 [ns create-connection TCP/ Reno
    $node ($2) TCPLink $node ($3) 1]
stop set window 15 attach-source PFI
set tcp2 stop attach-source PFI

# Мониторинг очереди окна TCP:
set windowTime [open WindowVTimeInfo w]
set when [ns monitor-queue $node ($1) $node ($2)
    [open out.q 0 1]]
[ns link $node ($1) $node ($2)] queue-sample-timeout:
# Мониторинг очереди:
set req [ns link $node ($1) $node ($2)] queue
set when [open all.q w]
nsred trace on
nsred trace on
nsred trace on
nsred trace on

# Добавление st-command:
ns at 0.0 "stop1 start"
ns at 1.1 "plotWindow [stop WindowVTime]"
ns at 3.0 "stop2 start"
ns at 10 "finish"

proc plotWindow [tcpSource file] {
    global ns
    set time 0.01
    set rtr [ns rtr]
    set cmd [stopSource set cmd]
    puts $file "ns red"
    ns at [expr $time-$time] "plotWindow stopSource $file"
}

# Запуск модели
ns run
```

Рис. 4.2: рисунок 2

```
File Edit View Debug Window
# Описание объекта Simulator
set ns [new Simulator]

# открытие на запись файла out.log для визуализатора nam
set of [open out.log w]
# для события-011 вычисления будут выписаны в переменную of
set counter 011

# открытие на запись файла трассировки out.tr
# для регистрации всех событий
set f [open out.tr w]
# все зарегистрированные события будут записаны в переменную f
set trace all $f

# процедура finish закрывает файлы трассировки
# и запускает визуализатор nam
proc finish {} {
    global trace
    # закрываем код AMM:
    set subcode {
        {
            if ($1 == "0" 66 MP-2) {
                print $2, $3 >> "temp.q";
                set end $2
            }
            else if ($1 == "0" 66 MP-2) {
                print $2, $3 >> "temp.n";
            }
        }
    }
    set f [open temp.queue w]
    puts $f "Simulation: end"
    puts $f "Provider Postscript"
    if { [info exists tchan_1] } {
        close tchan_1
    }
    exec ns -f temp.q temp.n
    exec trace temp.n temp.q
    exec set subcode all.q и выполнение кода AMM
    puts $f "Vname"
    exec cat temp.q > $f
    puts $f "Vn queue"
    exec cat temp.n > $f
    close $f
    # вывод в экран - (добавим окно TCP и очередь)
    exec ns/rtr -bb -tk -x time -t "TCPMenuCMD" WindowVTimeInfo &
    exec ns/rtr -bb -tk -x time -y queue temp.queue &
    exit 0
}

# Time setup
set n 5
for {set i 1} {$i < $n} {incr i} {
    set node ($i) [ns node]
}
set node ($1) [ns node]
set node ($2) [ns node]

# Соединение:
ns duplex-link $node ($1) $node ($2) 10Mb 2ms DropTail
ns duplex-link $node ($2) $node ($1) 10Mb 2ms DropTail
ns duplex-link $node ($1) $node ($2) 1Mb 2ms RED
ns queue-limit $node ($1) $node ($2) 25
ns queue-limit $node ($2) $node ($1) 25
ns duplex-link $node ($3) $node ($2) 10Mb 4ms DropTail
ns duplex-link $node ($4) $node ($2) 10Mb 5ms DropTail

# Агенты и приложения:
set tcp1 [ns create-connection TCP/ Reno
    $node ($1) TCPLink $node ($3) 0]
stop set window 15
set tcp2 [ns create-connection TCP/ Reno
    $node ($2) TCPLink $node ($3) 1]
stop set window 15 attach-source PFI
set tcp2 stop attach-source PFI

# Мониторинг очереди окна TCP:
set windowTime [open WindowVTimeInfo w]
set when [ns monitor-queue $node ($1) $node ($2)
    [open out.q 0 1]]
[ns link $node ($1) $node ($2)] queue-sample-timeout:
# Мониторинг очереди:
set req [ns link $node ($1) $node ($2)] queue
set when [open all.q w]
nsred trace on
nsred trace on
nsred trace on
nsred trace on

# Добавление st-command:
ns at 0.0 "stop1 start"
ns at 1.1 "plotWindow [stop WindowVTime]"
ns at 3.0 "stop2 start"
ns at 10 "finish"

proc plotWindow [tcpSource file] {
    global ns
    set time 0.01
    set rtr [ns rtr]
    set cmd [stopSource set cmd]
    puts $file "ns red"
    ns at [expr $time-$time] "plotWindow stopSource $file"
}

# Запуск модели
ns run
```

Рис. 4.3: рисунок 3

Графики окна TCP и очереди (рис. 4.4, 4.5)

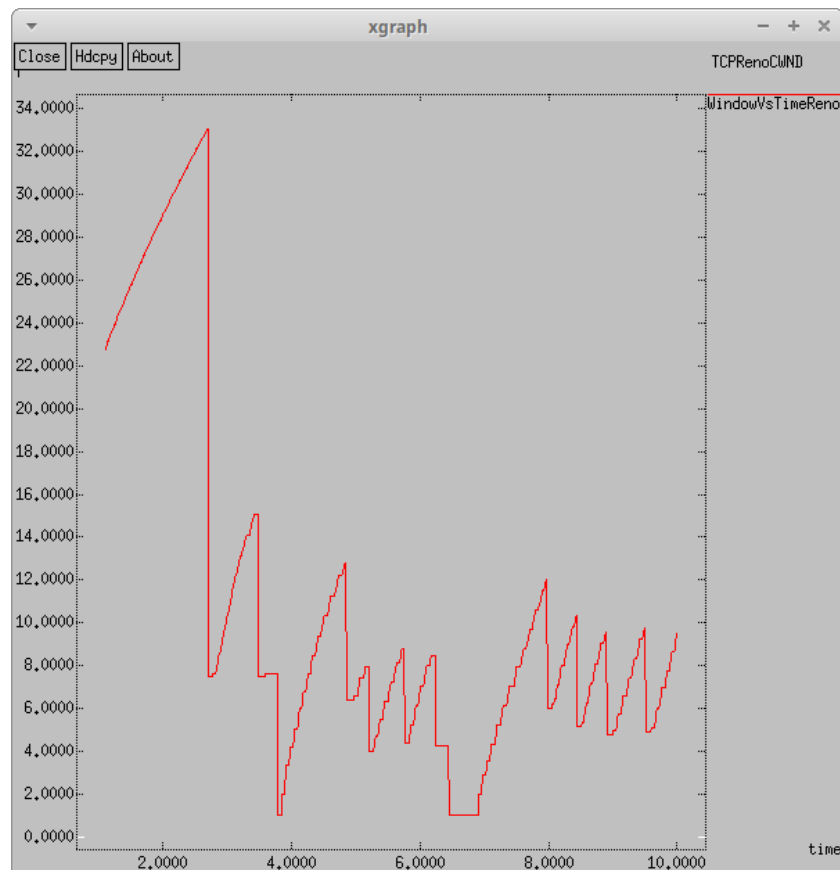


Рис. 4.4: рисунок 4

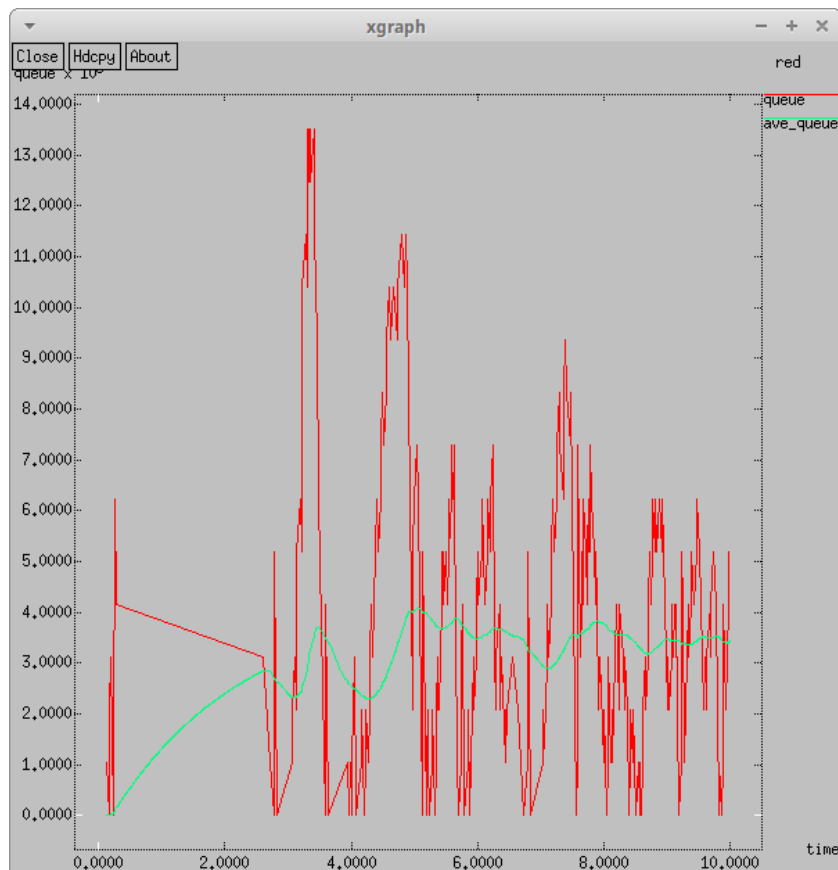


Рис. 4.5: рисунок 5

2. Измените в модели на узле s1 тип протокола TCP с Reno на NewReno, затем на Vegas. (рис. 4.6)

Меняем TCP/Reno -> TCP/Newreno

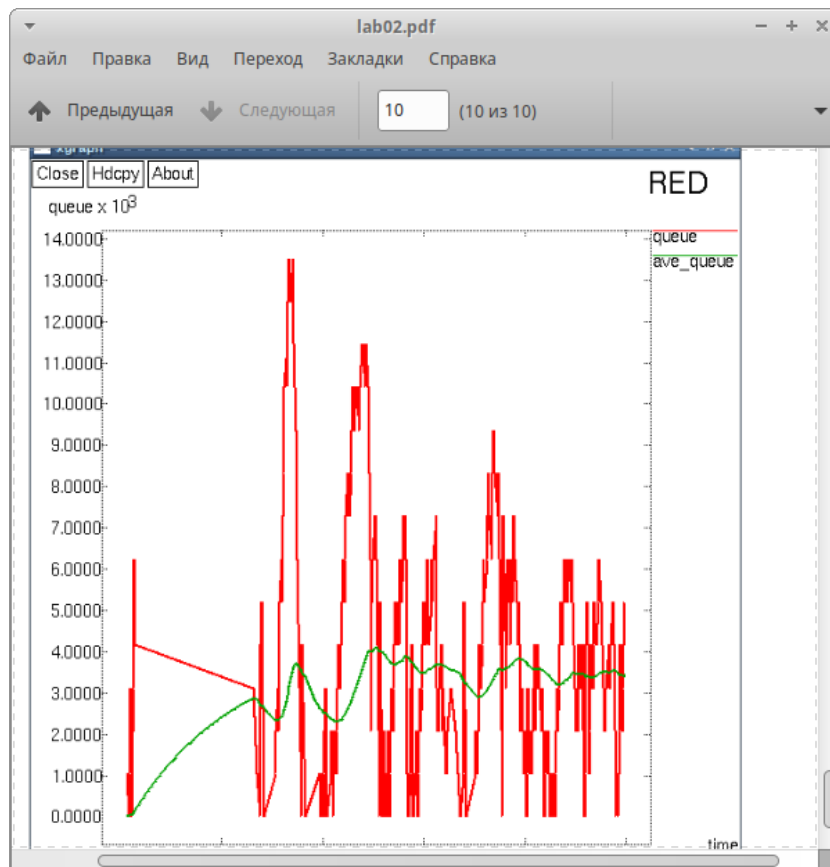


Рис. 4.6: рисунок 6

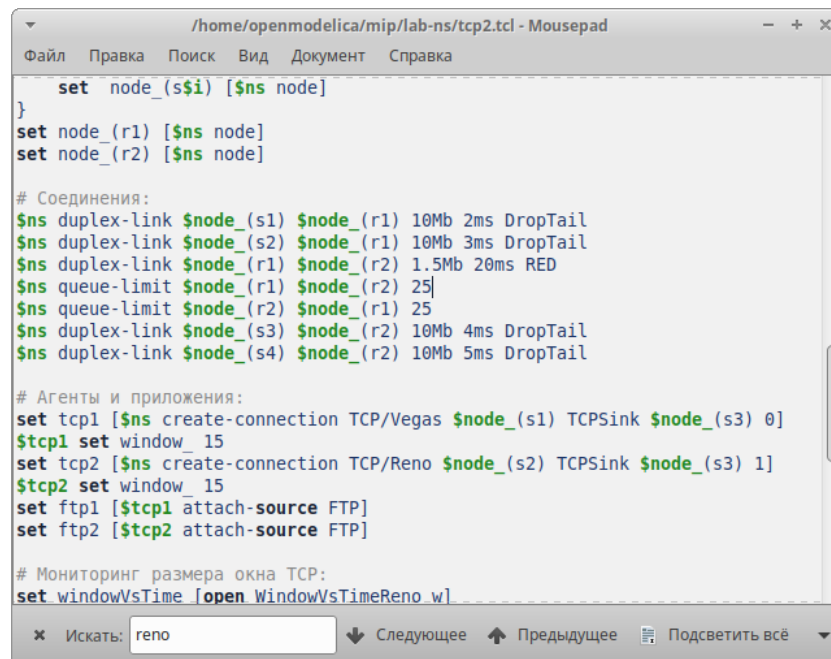
```

Терминал - openmodelica@openmodelica-VirtualBox: ~/mip/lab-ns
Файл  Правка  Вид  Терминал  Вкладки  Справка
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ ns tcp.tcl
ns: finish: awk: командная строка:8: критическая ошибка: невозможно открыть файл
'#' для чтения (Нет такого файла или каталога)
while executing
"exec awk $awkCode all.q # выполнение кода AWK"
(procedure "finish" line 22)
invoked from within
"finish"
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ ns tcp.tcl
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ Parameter LabelFont: can't tr
anslate 'helvetica-10' into a font (defaulting to 'fixed')
Parameter LabelFont: can't translate 'helvetica-10' into a font (defaulting to '
fixed')
Parameter TitleFont: can't translate 'helvetica-18' into a font (defaulting to '
fixed')
Parameter TitleFont: can't translate 'helvetica-18' into a font (defaulting to '
fixed')
XIO: fatal IO error 11 (Resource temporarily unavailable) on X server ":0.0"
after 272 requests (270 known processed) with 0 events remaining.
XIO: fatal IO error 11 (Resource temporarily unavailable) on X server ":0.0"
after 264 requests (264 known processed) with 0 events remaining.
^C
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ cp tcp.tcl tcp2.tcl
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ ns tcp2.tcl

```

Рис. 4.7: рисунок 7

Меняем TCP/Reno -> TCP/Vegas (рис. 4.8)



```

/home/openmodelica/mip/lab-ns/tcp2.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка

    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/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]

```

Искать: reno ↓ Следующее ↑ Предыдущее Подсветить всё

Рис. 4.8: рисунок 8

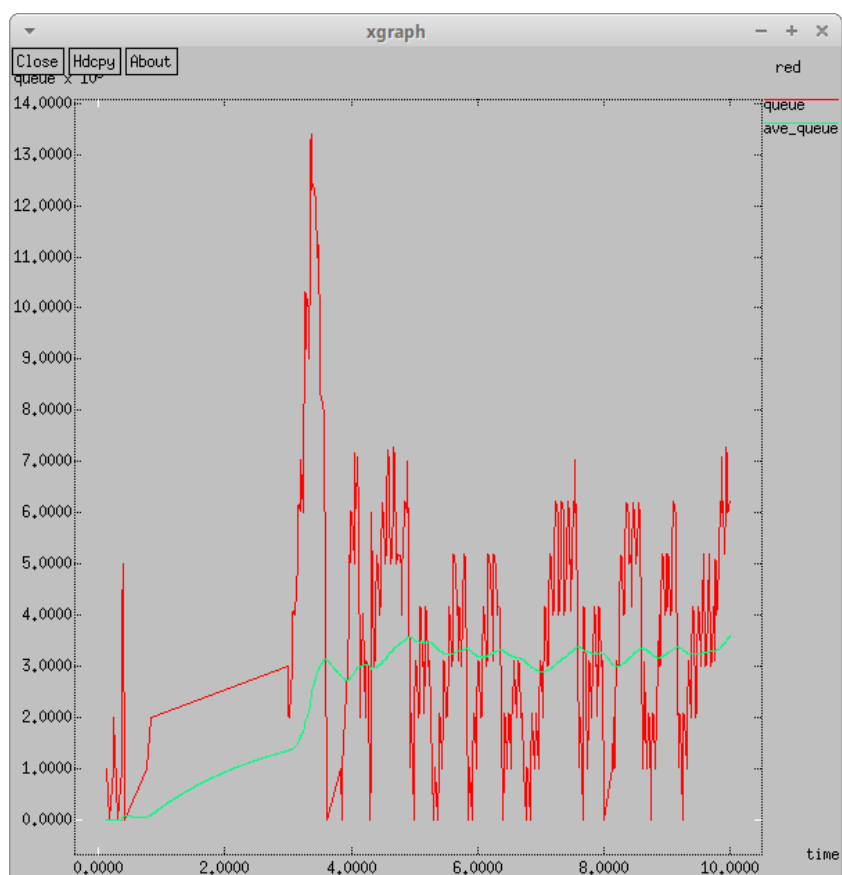


Рис. 4.9: рисунок 9

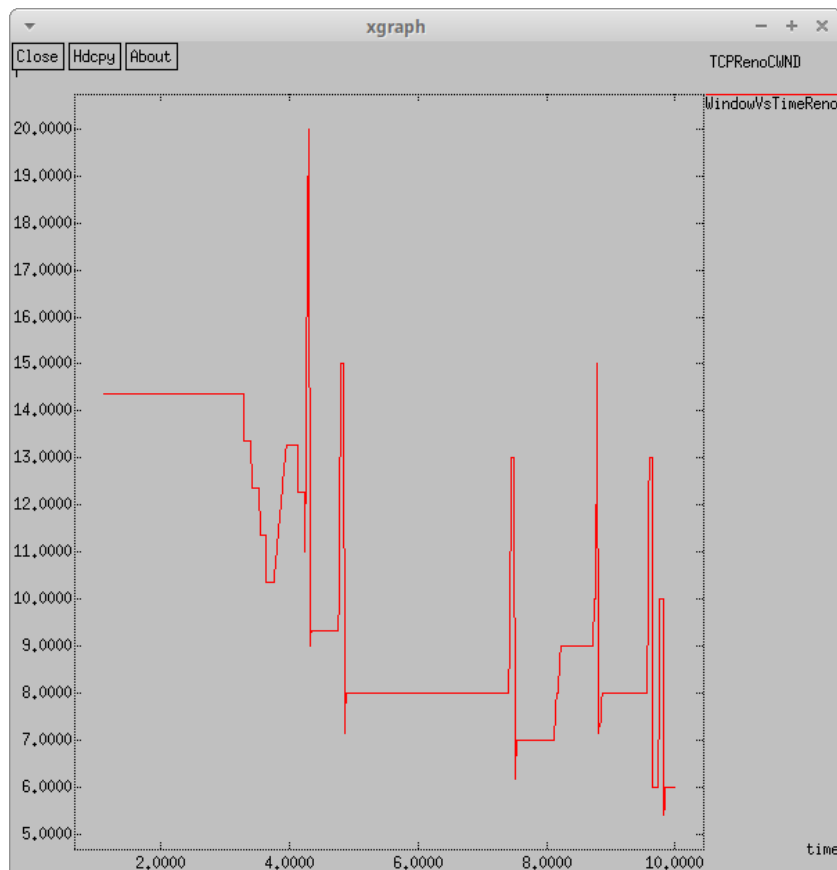


Рис. 4.10: рисунок 10

3. Внесите изменения при отображении окон с графиками (измените цвет фона, цвет траекторий, подписи к осям, подпись траектории в легенде)

Поменяла red -> red!!!! (рис. 4.11)



```

/home/openmodelica/mip/lab-ns/tcp2.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка

    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 \
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

```

Рис. 4.11: рисунок 11

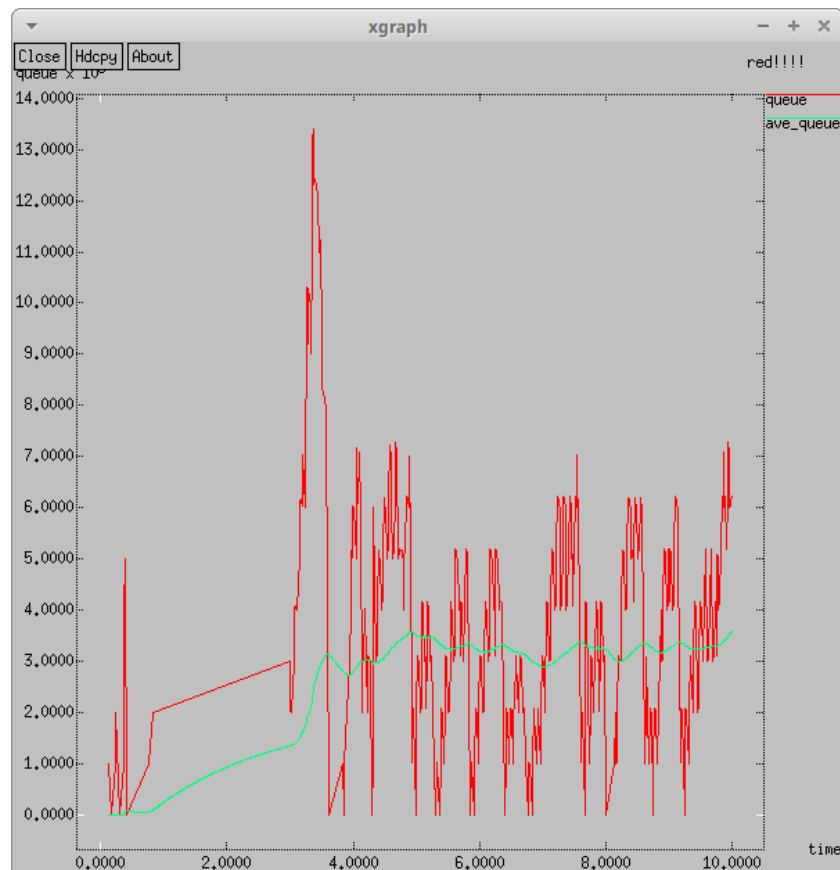


Рис. 4.12: рисунок 12

Поменяла цвета траекторий: (рис. 4.13)

```
/home/openmodelica/mip/lab-ns/tcp2.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка

}
set f [open temp.queue w]
puts $f "TitleText: red!!!"
puts $f "Device: Postscript"
puts $f "0.Color: Blue"
puts $f "1.Color: Black"
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 "\nqueue"
exec cat temp.q >@ $f
puts $f "\nave_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
}

# Узлы сети:
set N 5
for {set i 1} {$i < $N} {incr i} {
    set node ($i) [$ns node]
```

Рис. 4.13: рисунок 13

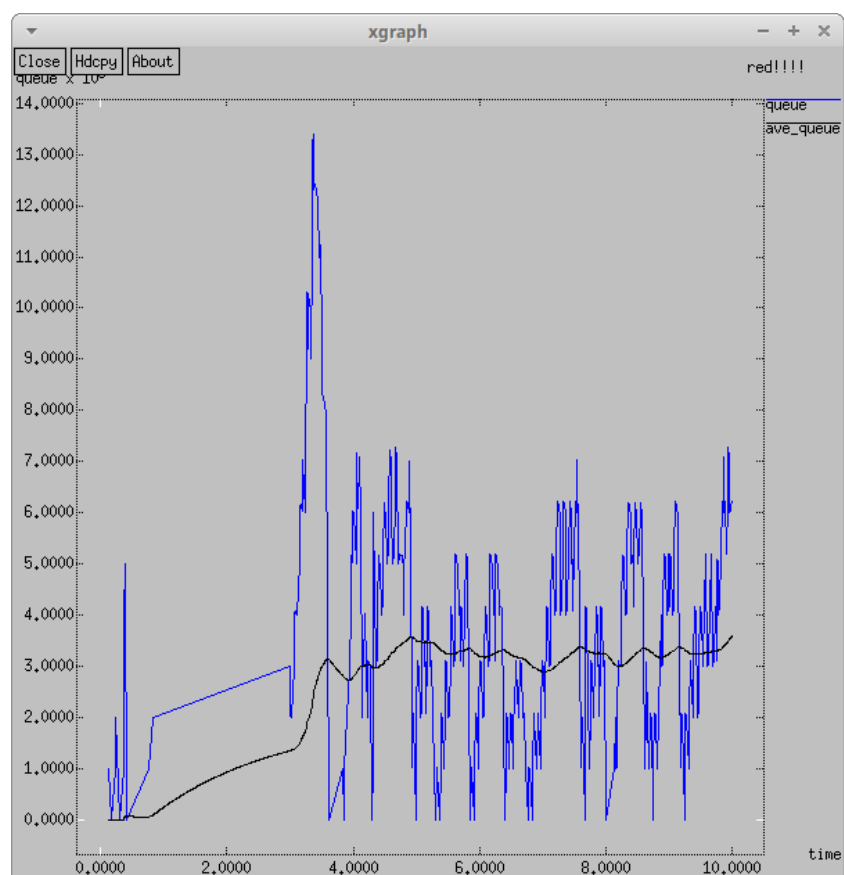


Рис. 4.14: рисунок 14

5 Выводы

В результате данной лабораторной работы я исследовала протокол ТСП и алгоритм управления очередью RED.

Список литературы