

Лабораторная работа 2

Исследование протокола TCP и алгоритма управления очередью RED

Ланцова Яна Игоревна

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
4	Выводы	15

Список иллюстраций

3.1	создание файла	6
3.2	написание кода	7
3.3	написание кода	8
3.4	График динамики размера окна ТСР	9
3.5	График динамики длины очереди и средней длины очереди . . .	10
3.6	редактирование кода	10
3.7	График динамики длины очереди и средней длины очереди . . .	11
3.8	редактирование кода	11
3.9	График динамики длины очереди и средней длины очереди . . .	12
3.10	редактирование кода	13
3.11	редактирование кода	13
3.12	График динамики длины очереди и средней длины очереди с из- менением отображения	14

1 Цель работы

Исследовать протокол TCP и алгоритм управления очередью RED.

2 Задание

1. Выполнить пример с дисциплиной RED;
2. Изменить в модели на узле s1 тип протокола TCP с Reno на NewReno, затем на Vegas. Сравнить и пояснить результаты;
3. Внести изменения при отображении окон с графиками (изменить цвет фона, цвет траекторий, подписи к осям, подпись траектории в легенде).

3 Выполнение лабораторной работы

Перейдем в рабочую директорию и скопируем содержимое созданного шаблона в новый файл: `cp shablon.tcl example4.tcl` (рис. [fig:001?]).

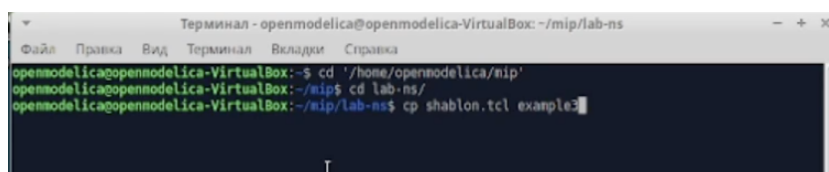


Рис. 3.1: создание файла

Выполним построение сети в соответствии с описанием:

- сеть состоит из 6 узлов;
- между всеми узлами установлено дуплексное соединение с различными пропускной способностью и задержкой 10 мс;
- 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.

Теперь разработаем сценарий, реализующий модель согласно описанию, чтобы построить в Xgraph график изменения TCP-окна, график изменения длины очереди и средней длины очереди (рис. [fig:002?]).

```

/home/openmodelica/mip/lab-ns/example4.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка

set ns [new Simulator]

set nf [open out.nam w]
$ns namtrace-all $nf

set f [open out.tr w]
$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. 3.2: написание кода

Изменим процедуру finish (рис. [fig:003?]):

```

/home/openmodelica/mip/lab-ns/example4.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка
$ns at 0.0 "$tcp1 start"
$ns at 1.1 "plotWindow $tcp1 $windowVsTime"
$ns at 3.0 "$ftp2 start"
$ns at 10 "finish"

# 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"
}

# 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 # выполнение кода AWK
    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. 3.3: написание кода

После запуска кода получаем график изменения TCP-окна (рис. [fig:004?]).

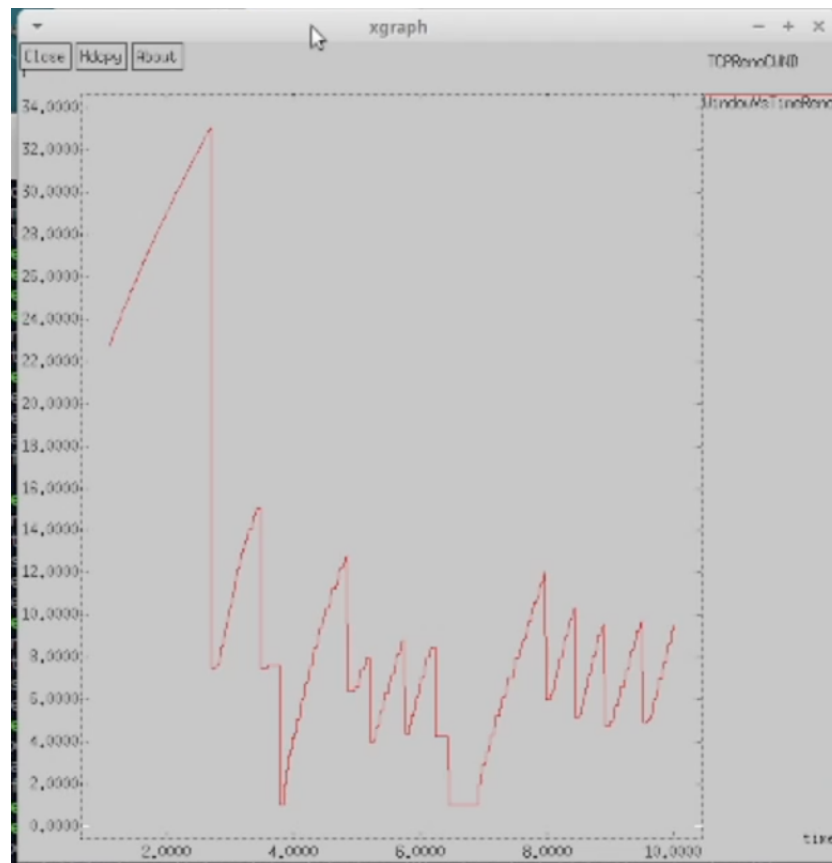


Рис. 3.4: График динамики размера окна ТСР

А также график изменения длины очереди и средней длины очереди (рис. [fig:005?]).

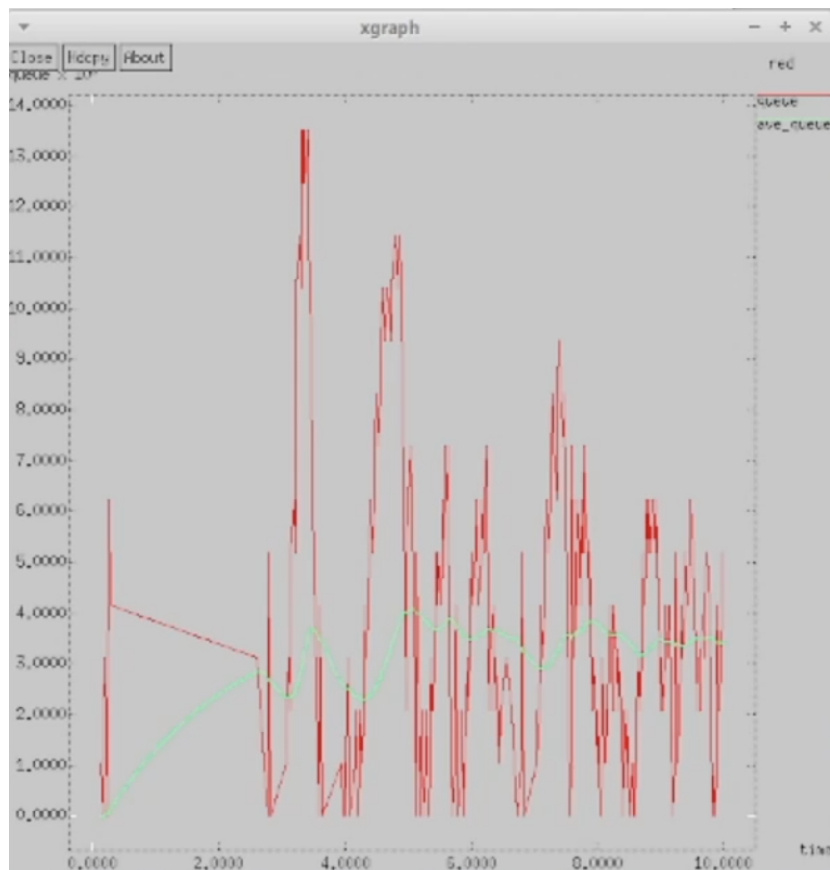


Рис. 3.5: График динамики длины очереди и средней длины очереди

По графику видно, что средняя длина очереди находится в диапазоне от 2 до 4, максимальная достигает значения 14.

Сначала требуется изменить тип протокола TCP Reno на NewReno. Для этого изменим код: (рис. [fig:006?]).

```
# 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]
$tcp2 set window_ 15
set ftp1 [$tcp1 attach-source FTP]
set ftp2 [$tcp2 attach-source FTP]
```

Рис. 3.6: редактирование кода

В результате получим следующие график изменения TCP-окна, а также график изменения длины очереди и средней длины очереди (почему-то у меня пока-

зывается только последний, пыталась это исправить, но все безуспешно) (рис. [fig:007?]).

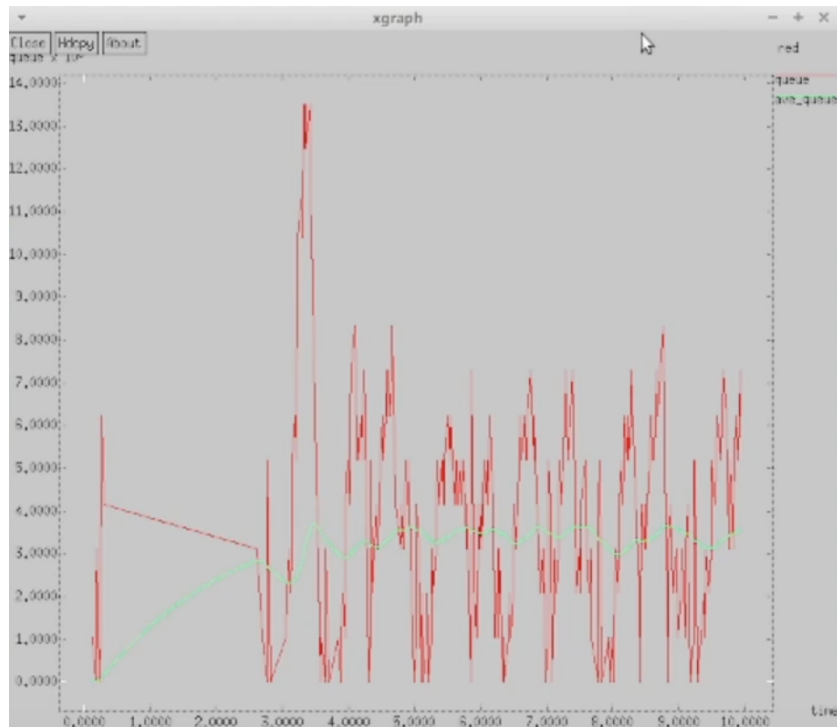


Рис. 3.7: График динамики длины очереди и средней длины очереди

Значение средней длины очереди также находится в пределах от 2 до 4, а максимальное значение так и осталось 14. Этим график с типом NewReno похож на Reno.

Теперь требуется изменить тип протокола TCP NewReno на Vegas. Для этого изменим код: (рис. [fig:008?]).

```
# 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]
```

Рис. 3.8: редактирование кода

В результате получим следующие график изменения TCP-окна, а также график

изменения длины очереди и средней длины очереди (почему-то у меня показывается только последний, пыталась это исправить, но все безуспешно) (рис. [fig:009?]).

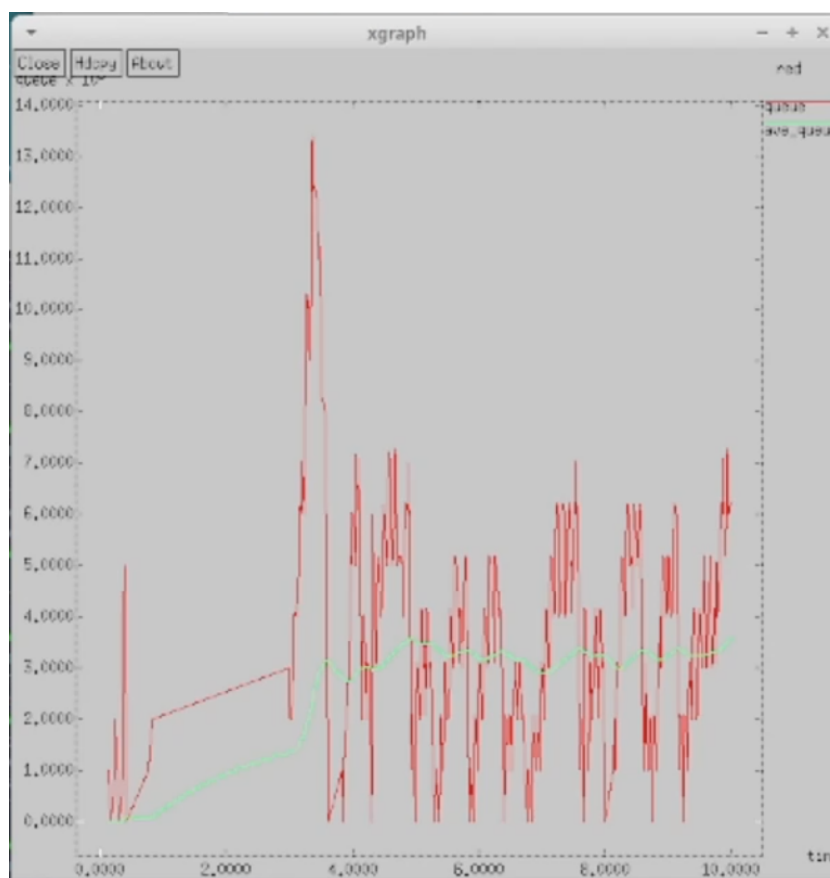


Рис. 3.9: График динамики длины очереди и средней длины очереди

Так же, как было в графике с типом Reno средняя длина очереди опять находится в диапазоне от 2 до 4 (но можно заметить, что значение длины чаще бывает меньшим, чем при типе Reno/NewReno). Максимальная длина достигает значения 14.

Внесем изменения при отображении окон с графиками, изменим цвет фона, цвет траекторий, подписи к осям и подпись траектории в легенде. Для этого изменим наш код: (рис. [fig:010?]).

```

# Процедура finish:
proc finish () {
    global tchan_
    # подключение кода AWK:
    set awkCode {
        {
            if ($1 == "0" && 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: Pink"
    puts $f "Device: Postscript"
    puts $f "0.Color: Pink"
    puts $f "1.Color: Blue"
    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 "\nAAA"
    exec cat temp.q >@ $f
    puts $f "\nBBB"
    exec cat temp.a >@ $f
    close $f

    # Запуск xgraph с графиками окна TCP и очереди:
    exec xgraph -fg white -bg black -bb -tk -x time -t "TCPRenoCWND" WindowVsTimeReno &
    exec xgraph -fg white -bg black -bb -tk -x time -y queue temp.queue &
    exit 0
}

```

Рис. 3.10: редактирование кода

В разделе мониторинга размера окна TCP также изменим цвет траектории и подпись легенды. (рис. [fig:011?]).

```

# Мониторинг размера окна TCP:
set windowVsTime [open WindowVsTimeReno w]
puts $windowVsTime "0.Color: Pink"
puts $windowVsTime \ "Window Size"
set qmon [$ns monitor-queue $node_(r1) $node_(r2) [open qm.out w] 0.1];
[$ns link $node_(r1) $node_(r2)] queue-sample-timeout;

```

Рис. 3.11: редактирование кода

В результате получим следующие график изменения TCP-окна, а также график изменения длины очереди и средней длины очереди (рис. [fig:012?]).

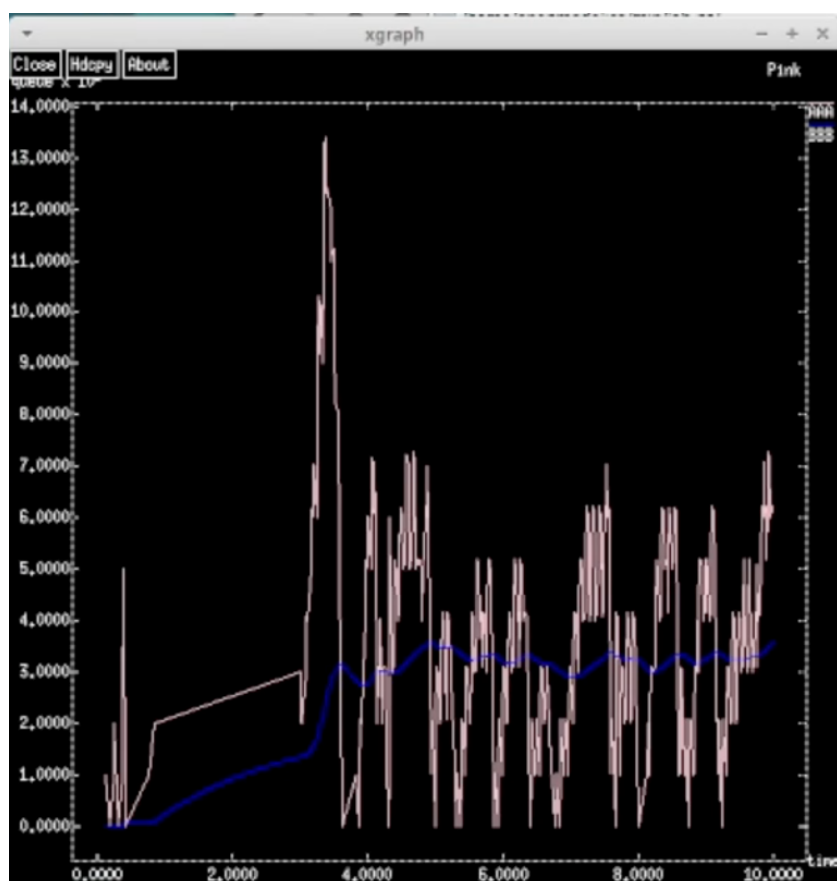


Рис. 3.12: График динамики длины очереди и средней длины очереди с изменением отображения

4 Выводы

В процессе выполнения данной лабораторной работы я исследовала протокол TCP и алгоритм управления очередью RED.