

Лабораторная работа №2

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

Кадров Виктор Максимович

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Пример с дисциплиной RED.	6
3.2	Изменения в модели на узле s1 типа протокола TCP с Reno на NewReno, затем на Vegas.	8
3.3	Изменения при отображении окон с графиками (изменить цвет фона, цвет траекторий, подписи к осям, подпись траектории в легенде).	10
4	Выводы	13
	Список литературы	14

Список иллюстраций

3.1	Скрипт модели с дисциплиной RED	7
3.2	Скрипт модели с дисциплиной RED	7
3.3	График динамики размера окна TCP(сверху) и график динамики длины очереди и средней длины очереди(снизу) при типе протокола TCP Reno на узле s1	8
3.4	Скрипт изменений на узле s1 типа протокола TCP с Reno на Newreno	8
3.5	График динамики размера окна TCP(сверху) и график динамики длины очереди и средней длины очереди(снизу) при типе протокола TCP NewReno на узле s1	9
3.6	Скрипт изменений на узле s1 типа протокола TCP с Reno на Vegas .	9
3.7	График динамики размера окна TCP(сверху) и график динамики длины очереди и средней длины очереди(снизу) при типе протокола TCP Vegas на узле s1	10
3.8	Изменение процедуры finish	11
3.9	Изменение мониторинга размера окна TCP	11
3.10	Результаты изменений отображения окон с графиками	12

1 Цель работы

Исследовать протокол TCP и алгоритм управления очередью RED[1].

2 Задание

1. Рассмотреть 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

3.1 Пример с дисциплиной RED.

Постановка задачи. Описание моделируемой сети: – 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.

Первая часть скрипта модели с дисциплиной RED. (рис. 3.1)

```

Терминал - openmodelica@openmodelica-VirtualBox: ~/lab2
GNU nano 2.9.3 lab2.tcl

et ns [new Simulator]

et nf [open out.nam w]

ns namtrace-all $nf

et f [open out.tr w]

ns trace-all $f

proc finish {} {
    global tchan
    # подключение кода AWK
    set awkCode {
        {
            if ($1 == "Q" %6 NF+2) {
                print $2, $3 >> "temp.q";
                set end $2
            }
            else if ($1 == "a" %6 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 "\nqueue"
    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 размерах 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"
}

```

Рис. 3.1: Скрипт модели с дисциплиной RED

Вторая часть скрипта модели с дисциплиной RED. (рис. 3.2)

```

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]

$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]

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_

$ns at 0.0 "$ftp1 start"
$ns at 1.1 "plotWindow $tcp1 $windowVsTime"
$ns at 2.0 "$ftp2 start"
$ns at 10 "finish"

```

Рис. 3.2: Скрипт модели с дисциплиной RED

График динамики размера окна TCP(сверху) и график динамики длины очереди и средней длины очереди(снизу) при типе протокола TCP Reno на узле s1. (рис. 3.3).

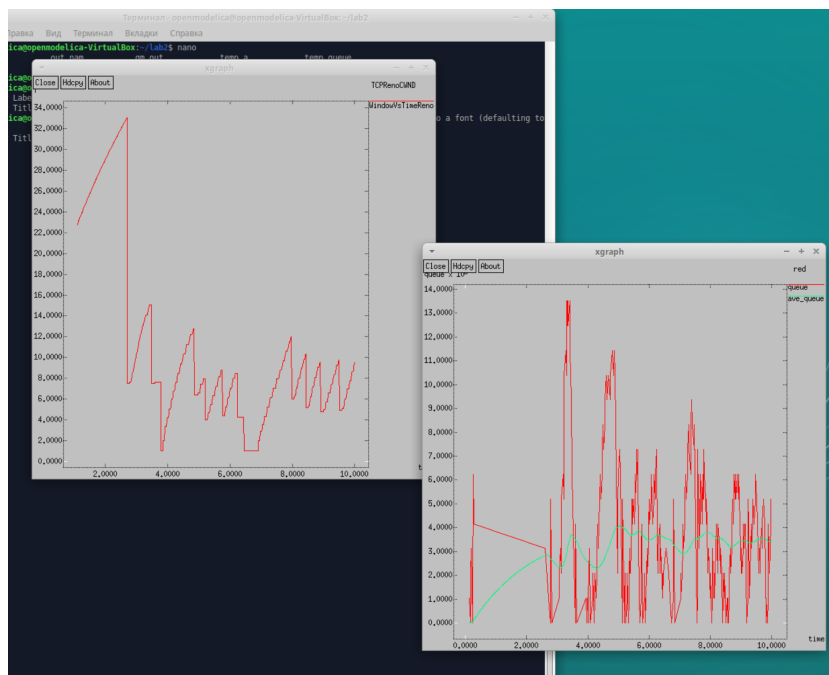


Рис. 3.3: График динамики размера окна TCP(сверху) и график динамики длины очереди и средней длины очереди(снизу) при типе протокола TCP Reno на узле s1

3.2 Изменения в модели на узле s1 типа протокола TCP с Reno на NewReno, затем на Vegas.

Скрипт изменений на узле s1 типа протокола TCP с Reno на Newreno. (рис. 3.4).

```
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. 3.4: Скрипт изменений на узле s1 типа протокола TCP с Reno на Newreno

График динамики размера окна TCP(сверху) и график динамики длины

очереди и средней длины очереди(снизу) при типе протокола TCP NewReno на узле s1. (рис. 3.5).

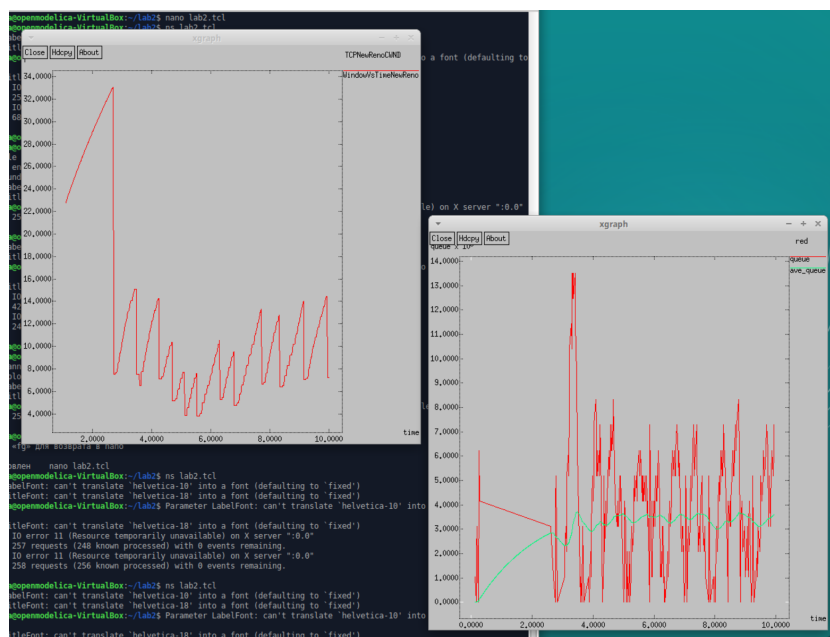


Рис. 3.5: График динамики размера окна TCP(сверху) и график динамики длины очереди и средней длины очереди(снизу) при типе протокола TCP NewReno на узле s1

TCP NewReno: Улучшенная версия Reno, устраняющая его недостатки:

- Улучшенный Fast Recovery. TCP NewReno остается в режиме восстановления после первой потери пакета и корректно обрабатывает несколько потерянных пакетов за один цикл передачи.
- Более точный механизм обнаружения потерь и адаптации скорости.

Скрипт изменений на узле s1 типа протокола TCP с Reno на Vegas. (рис. 3.6).

```
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]

set windowVsTime [open WindowVsTimeVegas w]
set proc [$ns register-queue $node_s1 $node_s2] [open proc-out-1 0 1]
```

Рис. 3.6: Скрипт изменений на узле s1 типа протокола TCP с Reno на Vegas

График динамики размера окна TCP(сверху) и график динамики длины

очереди и средней длины очереди(снизу) при типе протокола TCP Vegas на узле s1. (рис. 3.7).

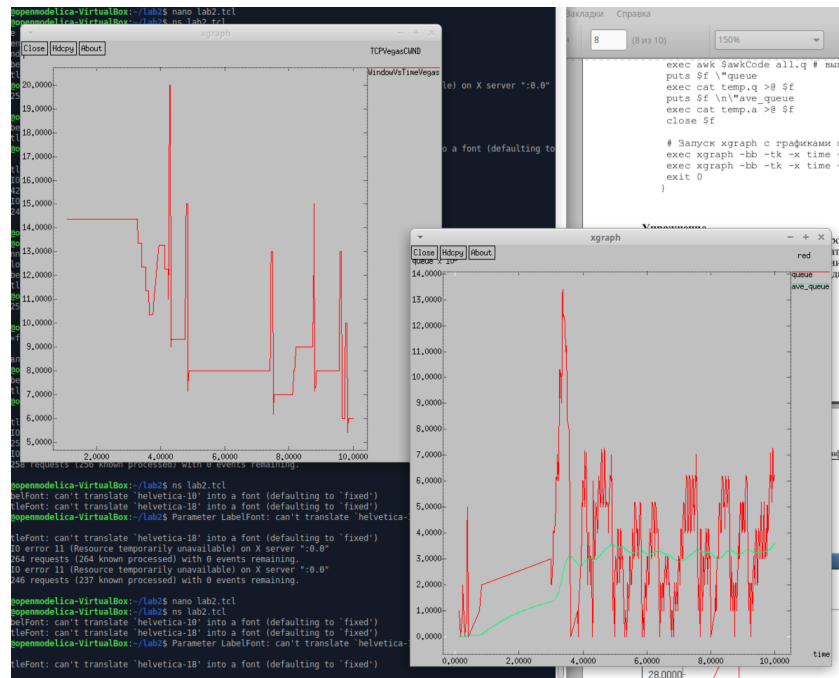


Рис. 3.7: График динамики размера окна TCP(сверху) и график динамики длины очереди и средней длины очереди(снизу) при типе протокола TCP Vegas на узле s1

TCP Vegas: TCP Vegas использует другой подход:

- Оценивает задержку пакетов вместо того, чтобы просто реагировать на потери.
- Контролирует перегрузку до ее возникновения, измеряя разницу между ожидаемой и реальной скоростью передачи.
- Более гладкая регулировка CWND, без резких изменений, как в Reno/NewReno.

3.3 Изменения при отображении окон с графиками (изменить цвет фона, цвет траекторий, подписи к осям, подпись траектории в легенде).

Изменение процедуры finish. (рис. 3.8).

```

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 Algorithm"
    puts $f "Device: Postscript"
    puts $f "0.Color: Blue"
    puts $f "1.Color: Orange"
    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 -bg white -bb -tk -x time -y "Window Size" -t "TCPVegasCwnd" WindowVsTimeVegas &
    exec xgraph -bg white -bb -tk -x time -y "Queue" temp.queue &
    exit 0
}

```

Рис. 3.8: Изменение процедуры finish

Изменение мониторинга размера окна TCP. (рис. 3.9).

```

set tcp [tcp -bg -tk -x time -y "Window Size" -t "TCPVegasCwnd" WindowVsTimeVegas &]

set windowVsTime [open WindowVsTimeVegas w]
puts $windowVsTime "0.Color: Blue"
set qmon [$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 code [ifns link $node_(r1) $node_(r2)] queue

```

Рис. 3.9: Изменение мониторинга размера окна TCP

Результаты изменений отображения окон с графиками. (рис. 3.10).

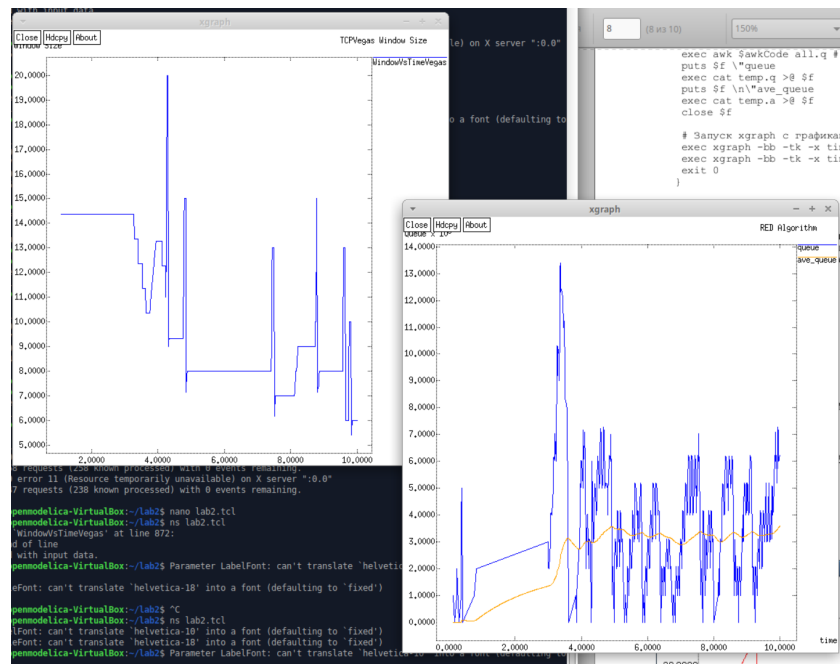


Рис. 3.10: Результаты изменений отображения окон с графиками

4 Выводы

Мы исследовали протокол TCP и алгоритм управления очередью RED.

Список литературы

1. Королькова А.В., Кулябов Д.С. Лабораторная работа 2. Исследование протокола ТСР и алгоритма управления очередью RED [Электронный ресурс].