Лабораторная работа №2

Исследование протокола TCP и алгоритма управления очередью RED

Кадров Виктор Максимович

Содержание

Список иллюстраций

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)

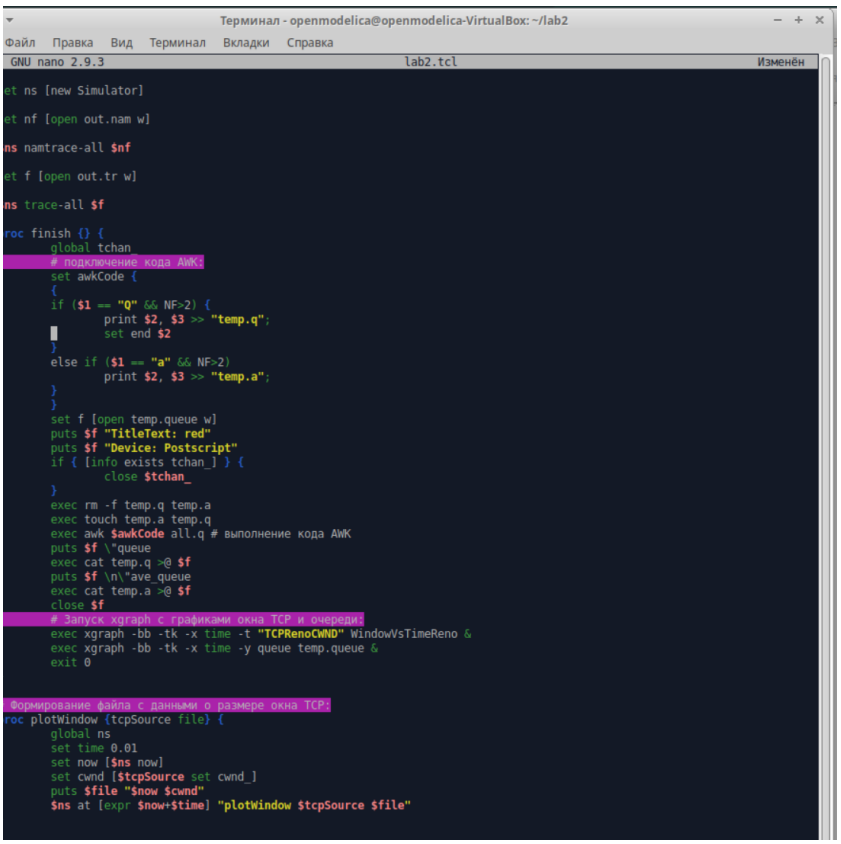

Рис. 3.1: Скрипт модели с дисциплиной RED

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

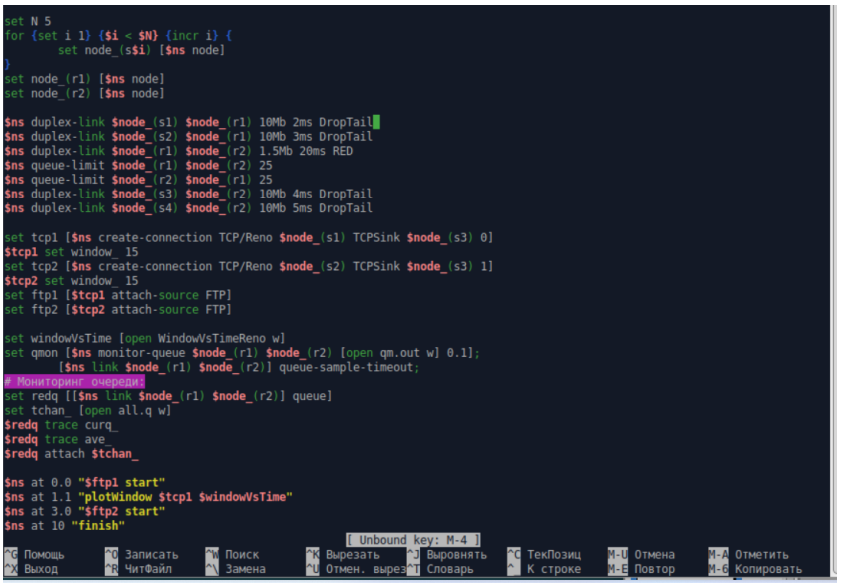

Рис. 3.2: Скрипт модели с дисциплиной RED

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

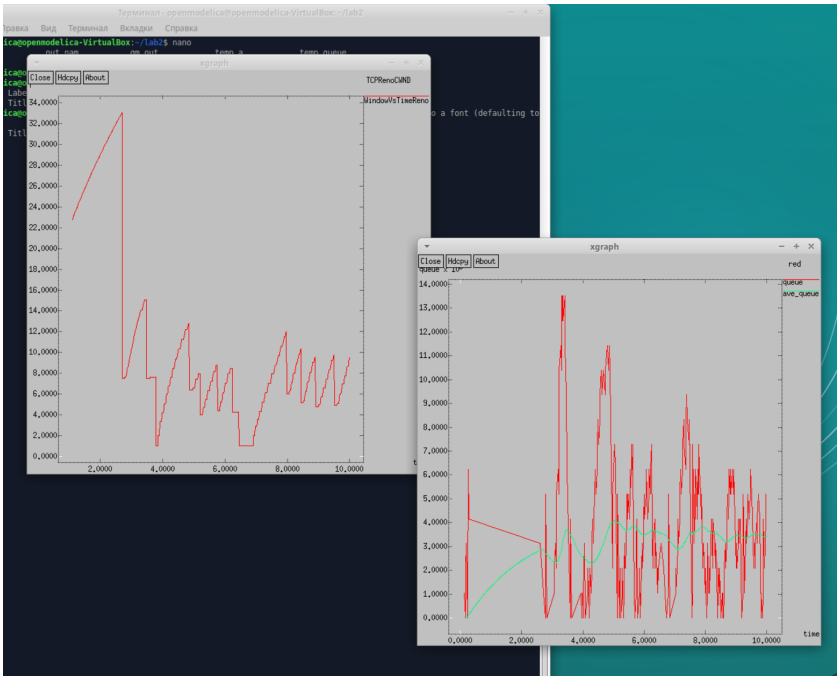

Рис. 3.3: График динамики размера окна TCP(сверху) и график динамики длины очереди и среденей длины очереди(снизу) при типе протокола TCP Reno на узле s1

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

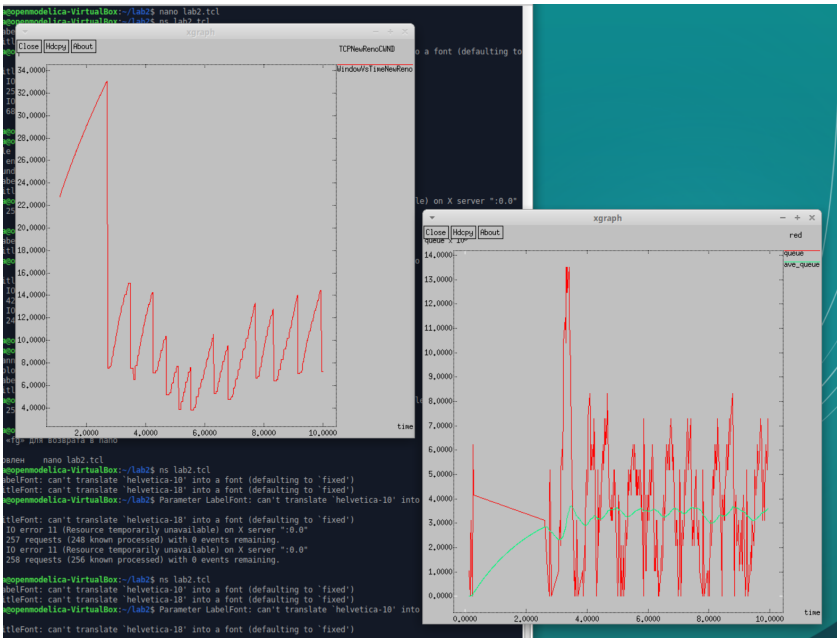

Рис. 3.5: График динамики размера окна TCP(сверху) и график динамики длины очереди и среденей длины очереди(снизу) при типе протокола TCP NewReno на узле s1

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]
```

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

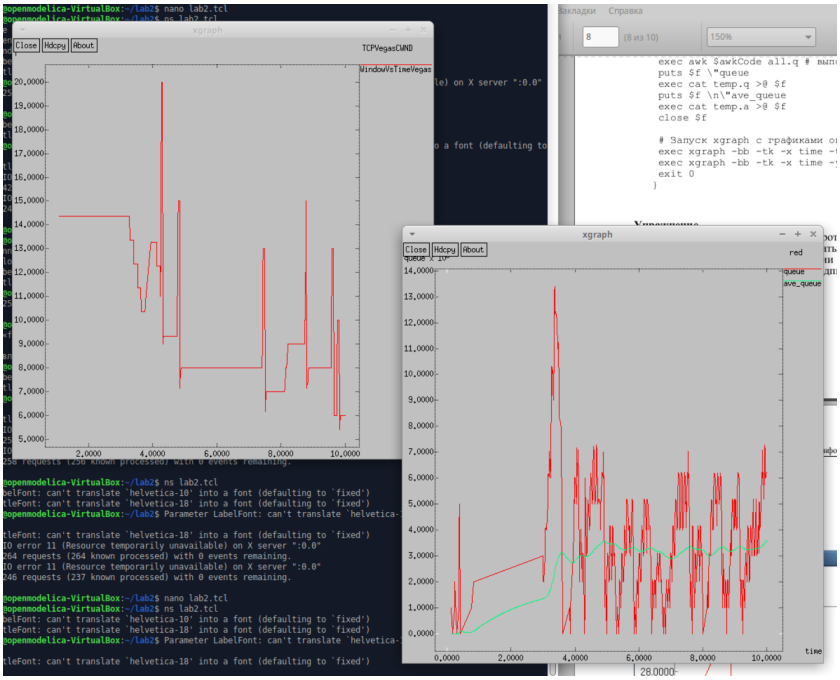

Рис. 3.7: График динамики размера окна TCP(сверху) и график динамики длины очереди и среденей длины очереди(снизу) при типе протокола TCP Vegas на узле s1

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 \"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 -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 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:
```

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 TCP и алгоритма управления очередью RED [Электронный ресурс].