Отчет по лабораторной работе 2

Исследование протокола TCP и алгоритма управления очередью RED

Шалыгин Георгий Эдуардович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучение протокола TCP и алгоритма управления очередью RED.

2 Задание

1. Запустить симуляцию для предложенной в примере сети.
2. Изменить в модели на узле s1 тип протокола TCP с Reno на NewReno, затем на Vegas. Сравните и поясните результаты.
3. Внести изменения при отображении окон с графиками (измените цвет фона, цвет траекторий, подписи к осям, подпись траектории в легенде).

3 Выполнение лабораторной работы

1. Опишем первую симуляцию(fig. 3.1).

```
set ns [new Simulator]

set nf [open out.nam w]

$ns namtrace-all $nf

set f [open out.tr w]

$ns trace-all $f

proc finish {} {
    global ns f nf
    $ns flush-trace
    close $f
    close $nf
    exec nam out.nam &
    exit 0
}

set N 5
for {set i 1} {$i < $N} {incr i} {
    set n_(s$i) [$ns node]
}

set n_(r1) [$ns node]
set n_(r2) [$ns node]

$ns duplex-link $n_(s1) $n_(r1) 10Mb 2ms DropTail
$ns duplex-link $n_(s2) $n_(r1) 10Mb 2ms DropTail
$ns duplex-link $n_(r1) $n_(r2) 1.5Mb 20ms RED
$ns queue-limit $n_(r1) $n_(r2) 25
$ns queue-limit $n_(r2) $n_(r1) 25
$ns duplex-link $n_(s3) $n_(r2) 10Mb 4ms DropTail
$ns duplex-link $n_(s4) $n_(r2) 10Mb 5ms DropTail

set tcp1 [$ns create-connection TCP/Reno $n_(s1) TCPSink $n_(s3) 0]
set tcp2 [$ns create-connection TCP/Reno $n_(s2) TCPSink $n_(s3) 1]
set ftp1 [$tcp1 attach-source FTP]
set ftp2 [$tcp2 attach-source FTP]
```

Рис. 3.1: Код для первой модели 1

Здесь:

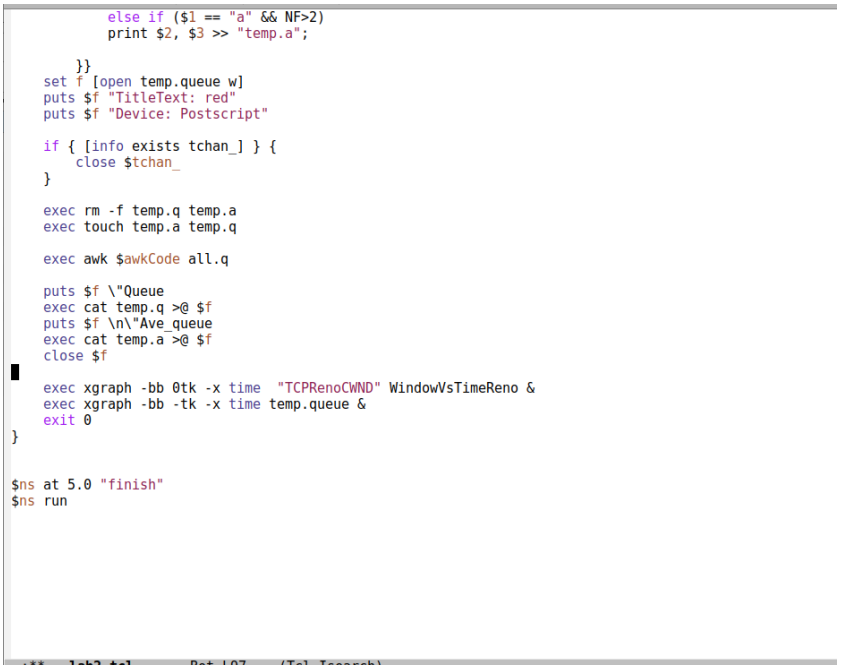
- создаются 6 узлов
- между ними определяются дуплексные соединения
- определяется лимит очереди на соединении r1-r2
- создаются tcp/Reno агенты и ftp поверх

```
set windowVsTime [open WindowVsTimeReno w]
puts $windowVsTime "0.Color: green"
set qmon [$ns monitor-queue $n_(r1) $n_(r2) [open qm.out w] 0.1];
[$ns link $n_(r1) $n_(r2)] queue-sample-timeout;

set redq [[$ns link $n_(r1) $n_(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"
$ns at 3.0 "$ftp2 start"
$ns at 10 "finish"

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"
}

proc finish {} {
    global tchan_
    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"
```

Рис. 3.2: Код для первой модели 2

Здесь:

- описывается запись данных в файл
- задаются события симуляции
- процедуры finish, где создаются два файла для сохранения значений скользящего среднего размера очереди и абсолютных его значений.



```
            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

    exec xgraph -bb 0tk -x time  "TCPRenoCWND" WindowVsTimeReno &
    exec xgraph -bb -tk -x time temp.queue &
    exit 0
}


$ns at 5.0 "finish"
$ns run
```

Рис. 3.3: Код для первой модели 2

Здесь задаются параметры для xghraph в соотв. файле и происходит вызов отрисовки.

2. Получим графики (fig. 3.4, fig. 3.5)

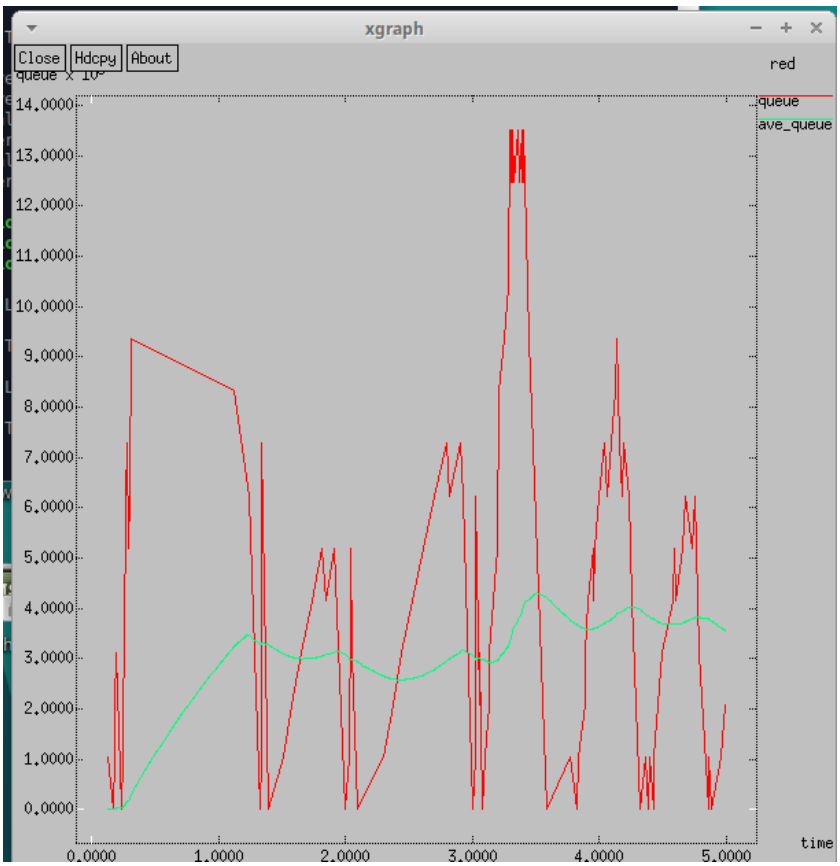


Рис. 3.4: График для размера очереди

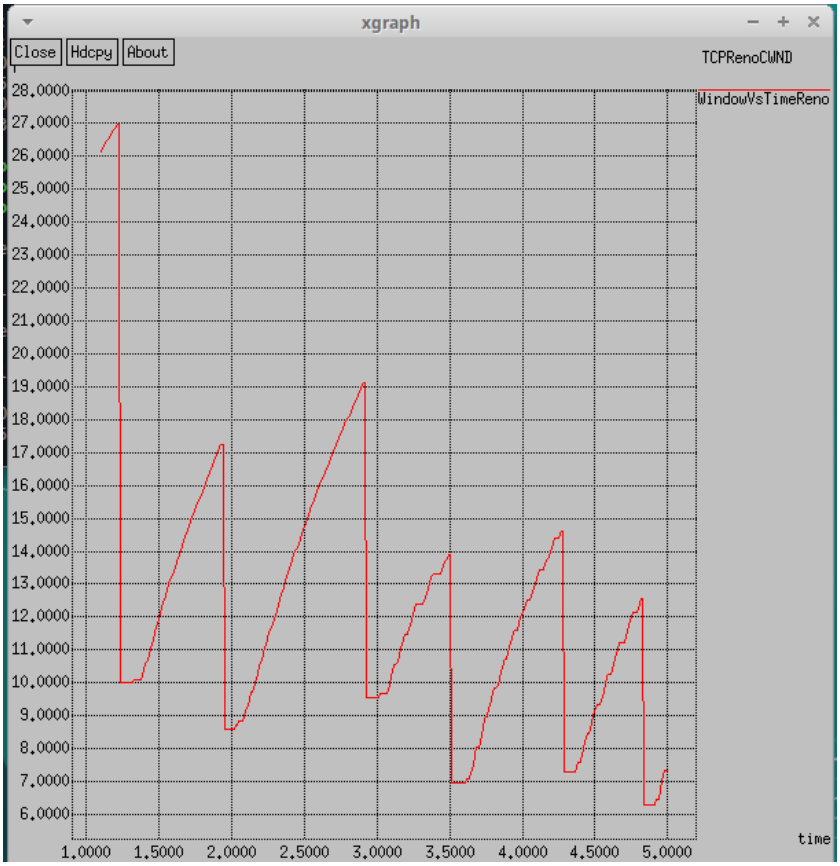


Рис. 3.5: График для размера очереди

Видим мгновенное изменение размера очереди (красным) и скользящее среднее (зеленым) (оно остается примерно на уровне 3). На нижнем графике отображено изменение размера окна, он совершает резкие падения и плавные линейные скачки.

3. Изменим код отображения (fig:006, fig:007):

```
puts $f "0.Color: red"
puts $f "1.Color: black"
```

Рис. 3.6: Изменение цвета траекторий

```
exec xgraph -bg white -bb 0tk -x time -t -y Y "TCPRenoCWND" WindowVsTimeReno &
exec xgraph -bg white -bb -tk -x time -y Queue_cnt -y queue temp.queue &
```

Рис. 3.7: Изменение цвета бэкграунда и подписей

4. Изменим тип протокола на Reno (fig. 3.8).

```
set tcp1 [$ns create-connection TCP/Newreno $n_(s1) TCPSink $n_(s3) 0]
set tcp2 [$ns create-connection TCP/Reno $n_(s2) TCPSink $n_(s3) 1]
```

Рис. 3.8: тип протокола на Reno

5. Построим графики (fig. 3.9).

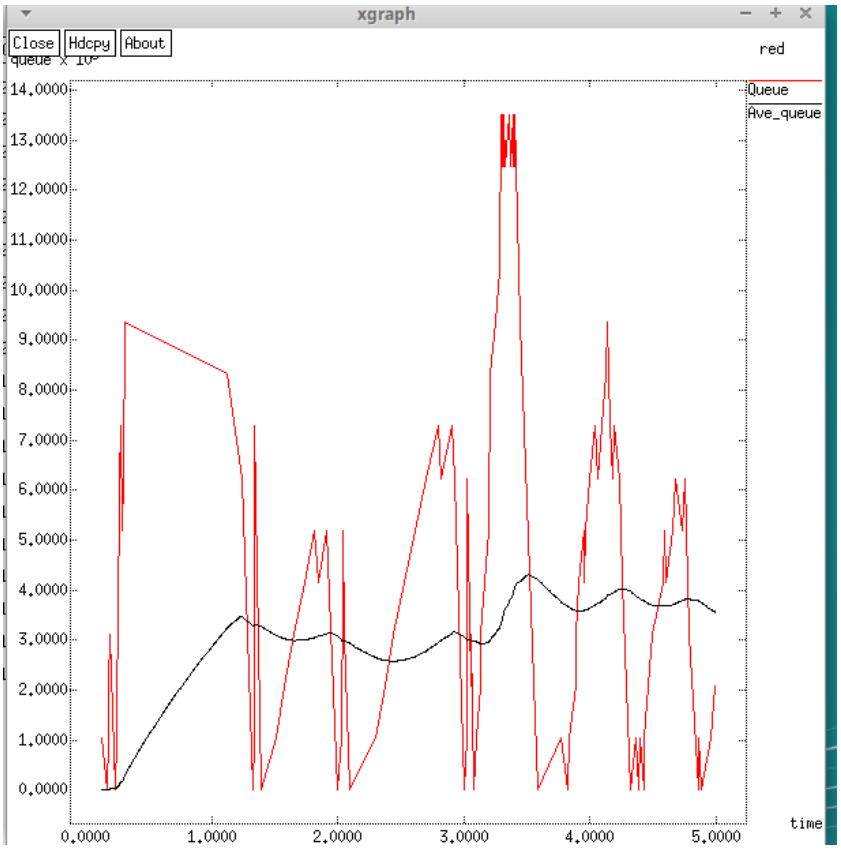


Рис. 3.9: Результаты newreno

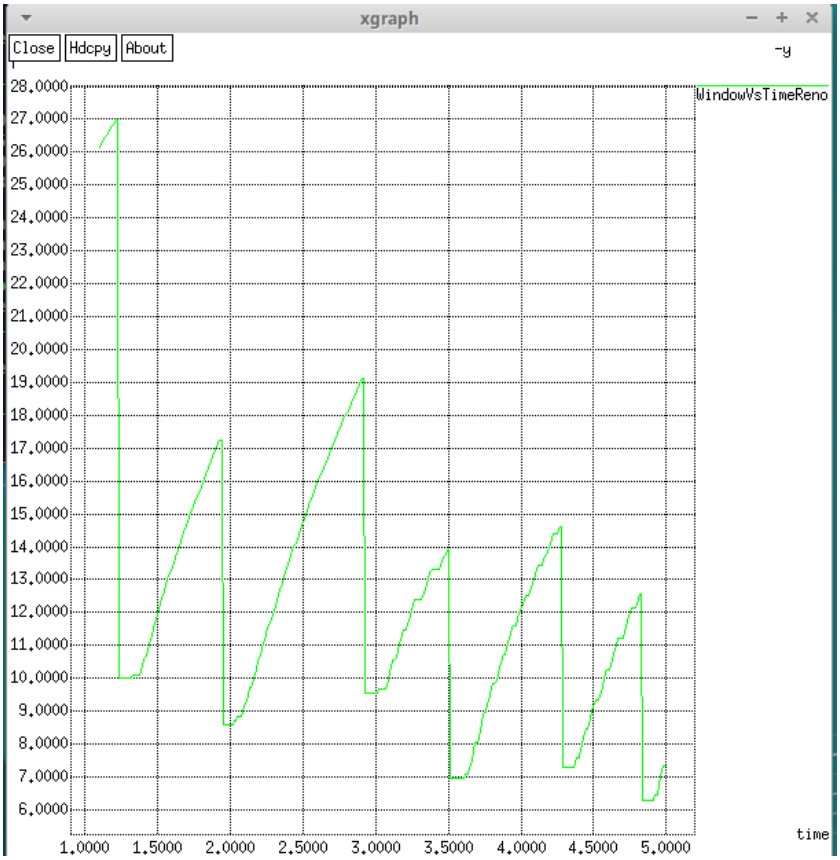


Рис. 3.10: Результаты newreno 2

Как видим, изменений практически нет.

6. Изменим тип протокола на Vegas (fig. 3.11).

```
set tcp1 [$ns create-connection TCP/Vegas $n_(s1) TCPSink $n_(s3) 0]
set tcp2 [$ns create-connection TCP/Reno $n_(s2) TCPSink $n_(s3) 1]
```

Рис. 3.11: тип протокола на Reno

7. Построим графики (fig. 3.12).

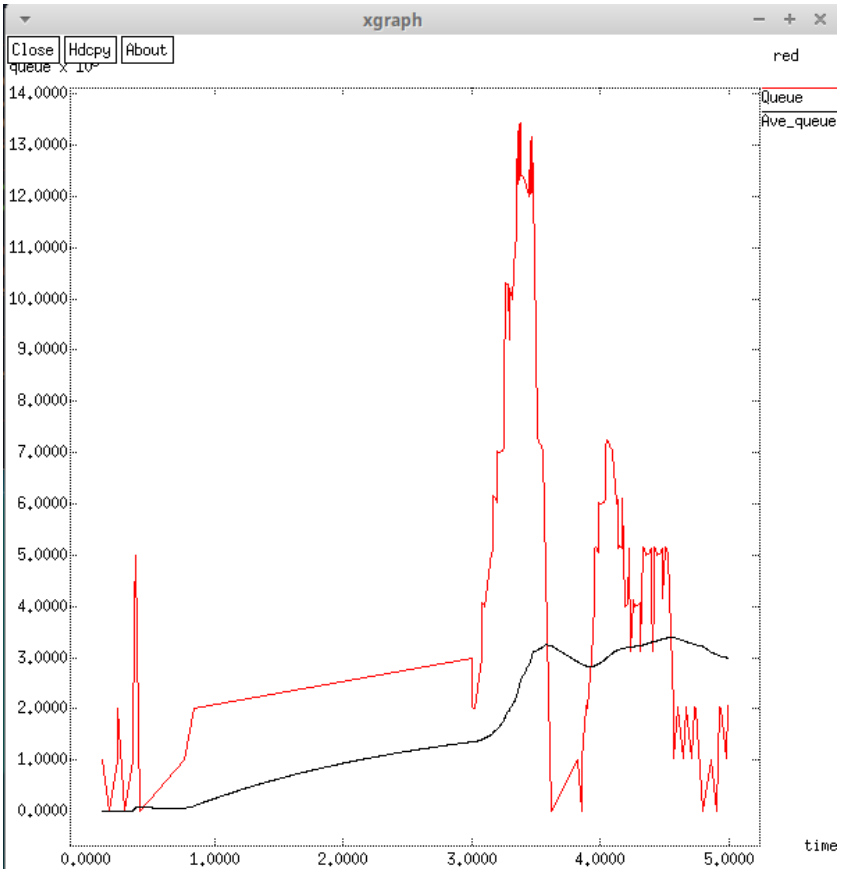


Рис. 3.12: Результаты newreno

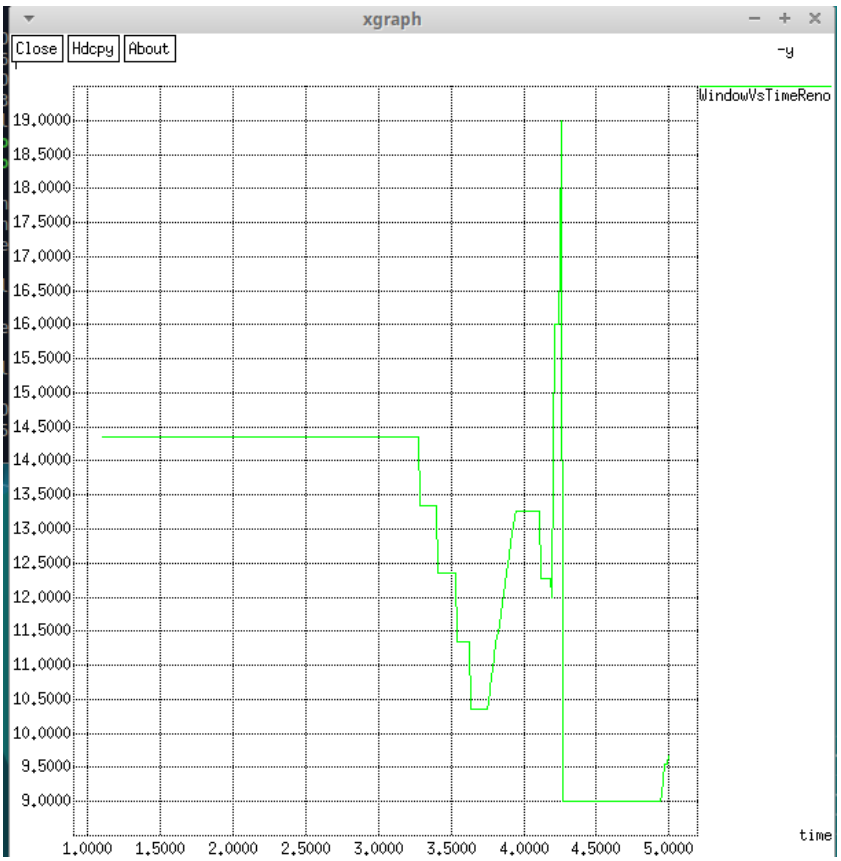


Рис. 3.13: Результаты newreno

Как видим, здесь размер окна сначала остается неизменным , затем совершает резкие колебания. Скользящее среднее изначально меньше чем для Reno, однако со временем возрастает до аналогичного уровня.

4 Выводы

В итоге были рассмотрены протоколы tcp и тип управления очередью red. А также средство Xgraph.