

Лабораторная работа № 2

Исследование протокола TCP и алгоритма управления очередью RED

Джахангиров Илгар Залид оглы

Содержание

| | | |
|----------|--|-----------|
| 0.1 | Цель работы | 3 |
| 0.2 | Задание | 3 |
| 0.3 | Выполнение лабораторной работы | 3 |
| 1 | Изменение протокола TCP | 10 |
| 2 | Выводы | 14 |

Список иллюстраций

| | | |
|-----|--|----|
| 1 | График динамики размера окна TCP | 8 |
| 2 | График динамики длины очереди и средней длины очереди . . . | 8 |
| 1.1 | График динамики размера окна TCP. Тип NewReno | 11 |
| 1.2 | График динамики длины очереди и средней длины очереди. Тип NewReno | 12 |
| 1.3 | График динамики длины очереди и средней длины очереди. Тип Vegas | 13 |

0.1 Цель работы

Исследовать протокол TCP и алгоритм управления очередью RED.

0.2 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).

0.3 Выполнение лабораторной работы

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.

```
# создание объекта Simulator
set ns [new Simulator]

# открытие на запись файла out.nam для визуализатора nam
set nf [open out.nam w]

# все результаты моделирования будут записаны в переменную nf
$ns namtrace-all $nf

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

# 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
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
}

# 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"
}

# Узлы сети:
set N 5
for {set i 1} {$i < $N} {incr i} {
    set node_(s$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/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_

# Добавление at-событий:
$ns at 0.0 "$ftp1 start"
$ns at 1.1 "plotWindow $tcp1 $windowVsTime"
$ns at 3.0 "$ftp2 start"
$ns at 10 "finish"

# запуск модели
$ns run

```

После запуска кода получаем график изменения TCP-окна (рис. 1), а также график изменения длины очереди и средней длины очереди (рис. 2).

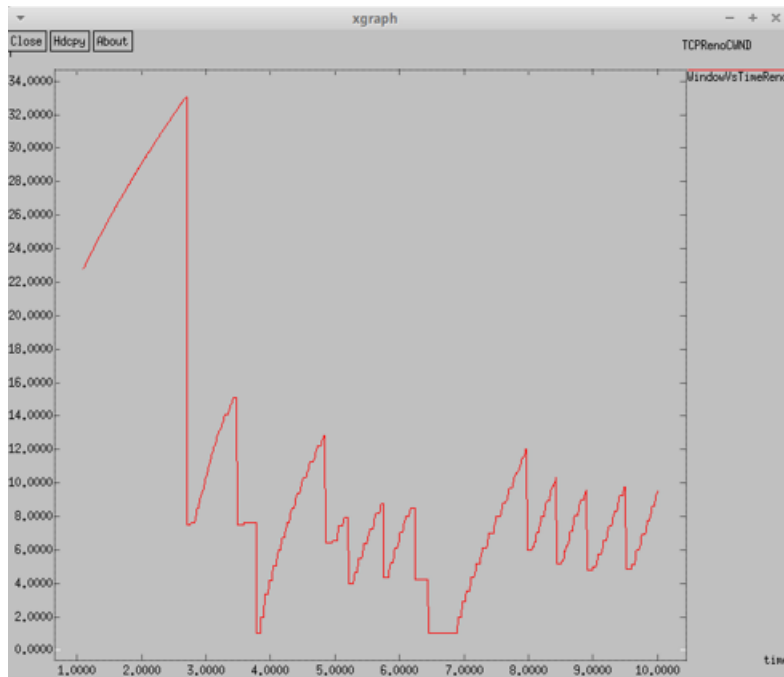


Рис. 1: График динамики размера окна TCP

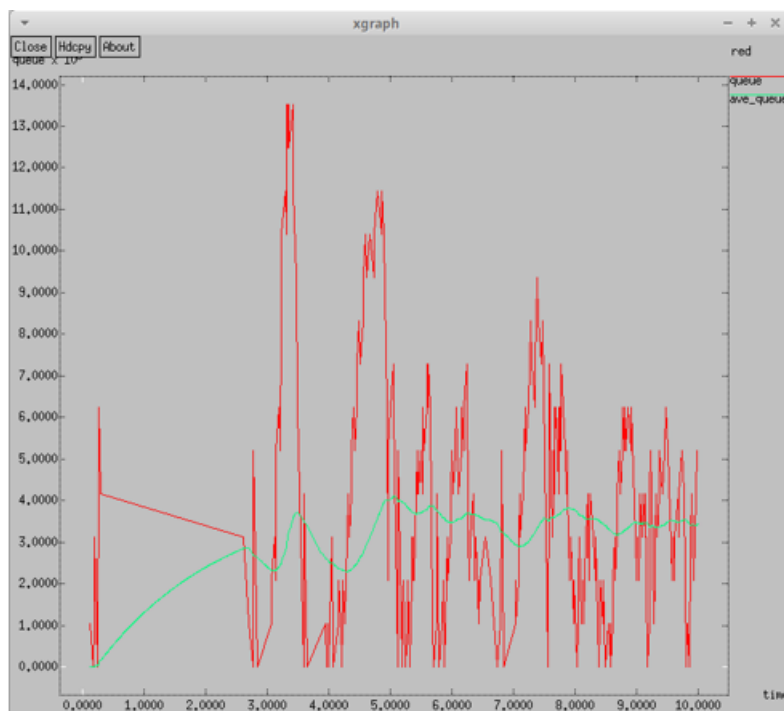


Рис. 2: График динамики длины очереди и средней длины очереди

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

Максимальная длина достигает значения 14.

1 Изменение протокола ТСР

Сначала требуется изменить тип Reno на NewReno. Для этого изменим код:

```
# 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
```

В результате получим следующие график изменения ТСР-окна , а также график изменения длины очереди

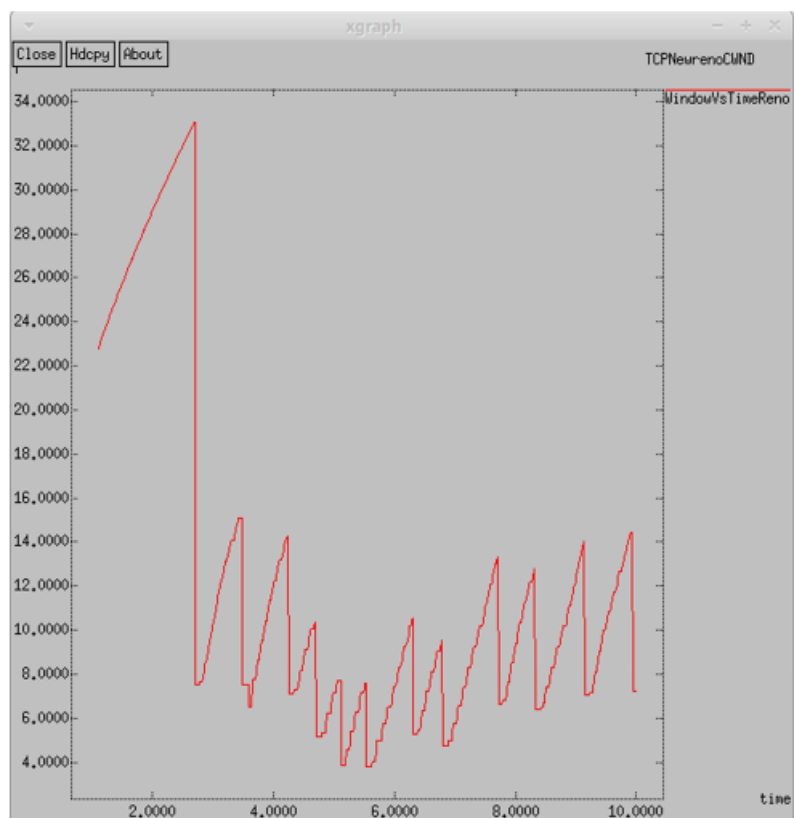


Рис. 1.1: График динамики размера окна TCP. Тип NewReno

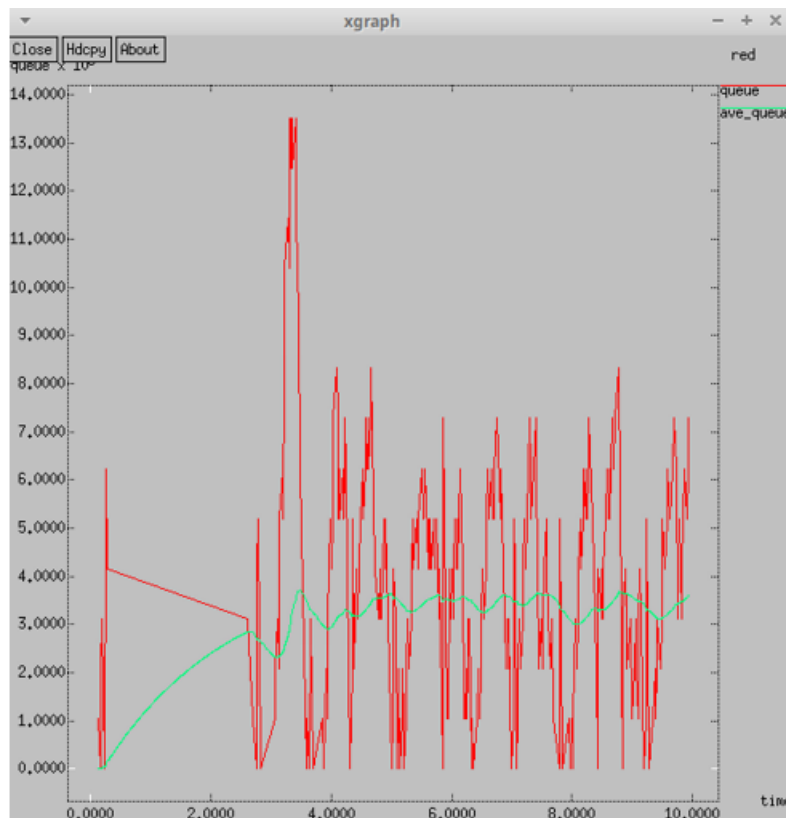


Рис. 1.2: График динамики длины очереди и средней длины очереди. Тип NewReno

Так же, как было в графике с типом Reno значение средней длины очереди находится в пределах от 2 до 4, а максимальное значение длины равно 14. Графики достаточно похожи. В обоих алгоритмах размер окна увеличивается до тех пор, пока не произойдёт потеря сегмента.

Теперь изменим тип Reno на Vegas. Для этого изменим код:

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
```

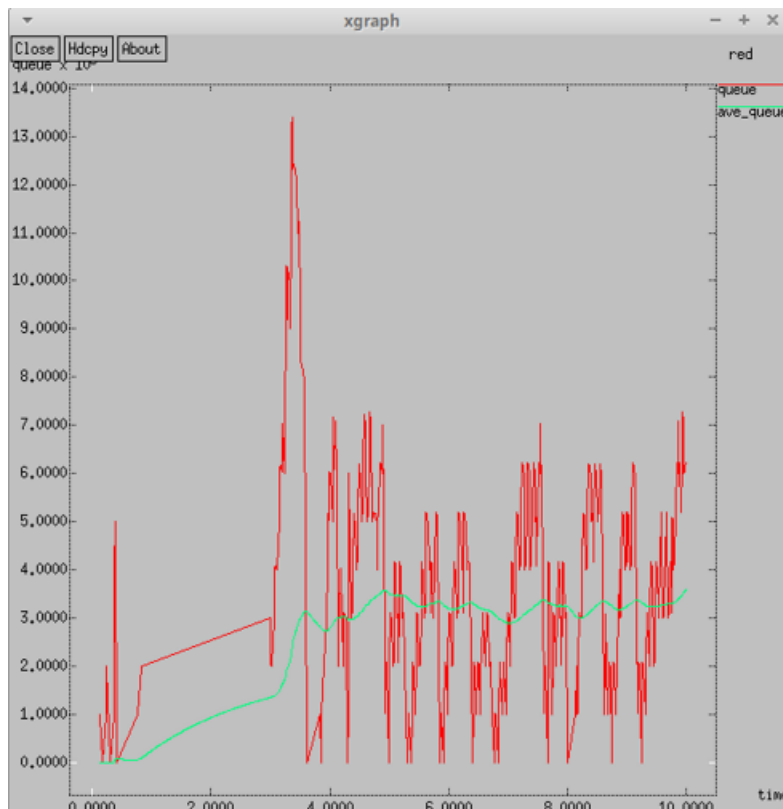


Рис. 1.3: График динамики длины очереди и средней длины очереди. Тип Vegas

По графику видно, что средняя длина очереди опять находится в диапазоне от 2 до 4 (но можно заметить, что значение длины чаще бывает меньшим, чем при типе Reno/NeReno). Максимальная длина достигает значения 14. Сильные отличия можно заметить по графикам динамики размера окна. При Vegas максимальный размер окна составляет 20, а не 34, как в NewReno. TCP Vegas обнаруживает перегрузку в сети до того, как случайно теряется пакет, и мгновенно уменьшается размер окна. Таким образом, TCP Vegas обрабатывает перегрузку без каких-либо потерь пакета.

2 Выводы

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