

Отчёт по лабораторной работе 2

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

Наталья Андреевна Сидорова

Содержание

1	Выполнение лабораторной работы	5
2	Выводы	10
	Список литературы	11

Список иллюстраций

1.1	1 часть кода	5
1.2	2 часть кода	6
1.3	Графики	6
1.4	NewReno	7
1.5	NewReno график	7
1.6	Vegas	7
1.7	Vegas график	8
1.8	Код изменения	8
1.9	2 часть	9
1.10	Новый график	9

Список таблиц

1 Выполнение лабораторной работы

Я построила пример с дисциплиной RED. Сеть состоит из 6 узлов, между всеми узлами установлено дуплексное соединение с различной пропускной способностью. Узел r1 использует очередь с дисциплиной RED максимальный размер очереди 25 (рис. 1.1).



```

# 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 закрывает файлы трассировки
# и запускает визуализатор nam
# 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 $f \n\ave.queue
    exec cat temp.a >$f
    close $f
    # Запуск xgraph с графиком окна TCP и очереди:
    exec xgraph -bb -tk -x time -t "TCP RenoCWND" WindowVstTimeReno &
    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
    while { $time < 10 } {
        $ns tcpdump -i eth0 -s 1500 -w $file &
        $ns flush
        $time + 0.01
    }
}

```

Рис. 1.1: 1 часть кода

За основу взяла код шаблона с 1 лабораторной (рис. 1.2).

```

* /home/openmodelica/mip/lab-ms/lab2/0.tcl - Mousepad
Файл Правка Поиск Вид Документ Справка

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 < $N} {incr i} {
  set node ($i) [$ns node]
}
set node_r1 [$ns node]
set node_r2 [$ns node]

# Соединения:
$ns duplex-link $node ($i) $node_r1 10Mb 2ms DropTail
$ns duplex-link $node ($i) $node_r2 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 $node ($i) $node_r2 10Mb 4ms DropTail
$ns duplex-link $node ($i) $node_r2 10Mb 5ms DropTail

# Агенты и приложения:
set tcp1 [$ns create-connection TCP/Reno]
$node ($i) TCPsink $node ($i) 0
$tcp1 set window 15
set tcp2 [$ns create-connection TCP/Reno]
$node ($i) TCPsink $node ($i) 1
$tcp2 set window 15
set ftp1 [$tcp1 attach-source FTP]
set ftp2 [$tcp2 attach-source FTP]

# Мониторинг размера окна TCP:
set windowVTime [open WindowVTimeReno w]
set qmon [$ns monitor-queue $node_r1 $node_r2]
$ns at [expr $now+$time] "plotWindow $tcp1 $windowVTime"
$ns at [expr $now+$time] "plotWindow $tcp2 $windowVTime"
$ns at [expr $now+$time] "finish"

# Мониторинг очереди:
set redq [$ns link $node_r1 $node_r2] queue-sample-timeout;
set redq trace cur;
$redq trace cur;
$redq trace ave;
$redq attach $tchan_

# Добавление ат-событий:
$ns at 0.0 "sftp1 start"
$ns at 1.1 "plotWindow $tcp1 $windowVTime"
$ns at 3.0 "sftp2 start"
$ns at 10 "finish"

# Запуск модели
$ns run

```

Рис. 1.2: 2 часть кода

Запустила Xgraph и посмотрела 2 графика : график изменения размера окна и график изменения размера очереди (рис. 1.3).

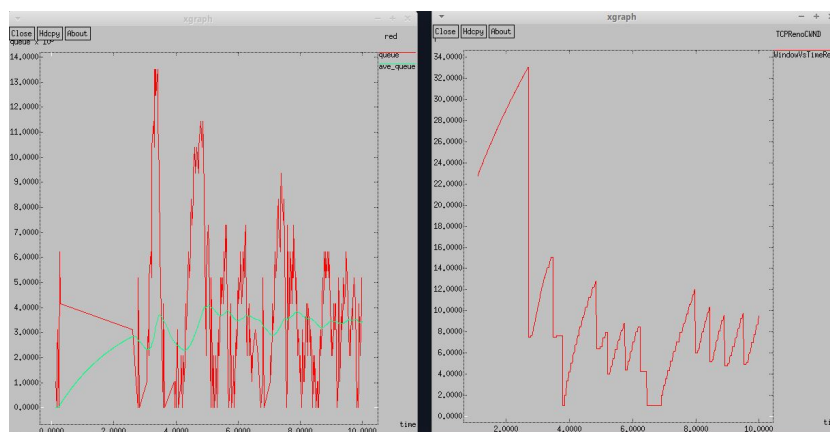


Рис. 1.3: Графики

Для выполнения упражнения я изменила на узле s1 тип протокола TCP с Reno на NewReno (рис. 1.4).

```
# 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. 1.4: NewReno

Получившийся график почти идентичен начальному (рис. 1.5).

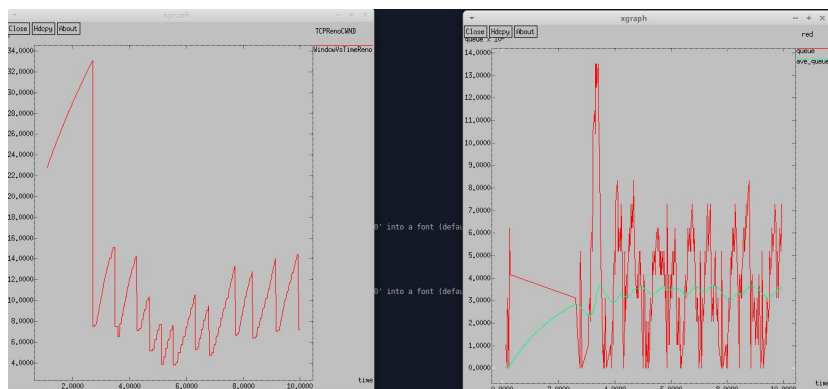


Рис. 1.5: NewReno график

Для выполнения упражнения я изменила на узле s1 тип протокола TCP с NewReno на Vegas (рис. 1.6).

```
# 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. 1.6: Vegas

Получившийся график изменения размеров окна отличается, максимальный размер окна снизился с 30+ до 20, максимальный размер очереди также снизился (рис. 1.7).

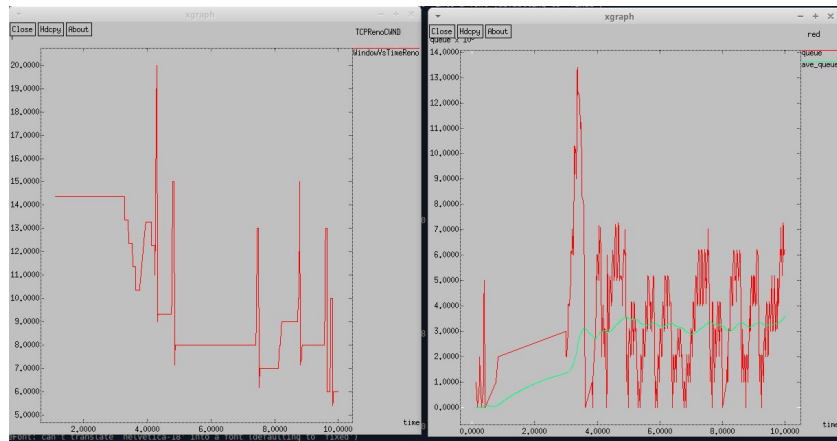


Рис. 1.7: Vegas график

Изменила цвет фона, цвет траекторий, подписи к осям, подпись траектории в легенде (рис. 1.8).

```
# 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"
    puts $f "0.Color: White"
    puts $f "1.Color: Pink"

    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 "\nLine"
    exec cat temp.q >@ $f
    puts $f "\nM_Line"
    exec cat temp.a >@ $f
    close $f
    # Запуск xgraph с графиками окна TCP и очереди:
    exec xgraph -fg pink -bg purple -bb -tk -x time -t "TCP Reno Cwnd" WindowVsTimeReno &
    exec xgraph -fg green -bg purple -bb -tk -x time -y line temp.queue &
    exit 0
}
```

Рис. 1.8: Код изменения

(рис. 1.9).


```

set ftp2 [$tcp2 attach-source FTP]

# Мониторинг размера окна TCP:
set windowVsTime [open WindowVsTimeReno w]
puts $windowVsTime "0.Color: White"
puts $windowVsTime "\"Razmer_Okna\""

set smon [$s monitor queue $node ($s) $node ($

```

Рис. 1.9: 2 часть

получившийся график (рис. 1.10).

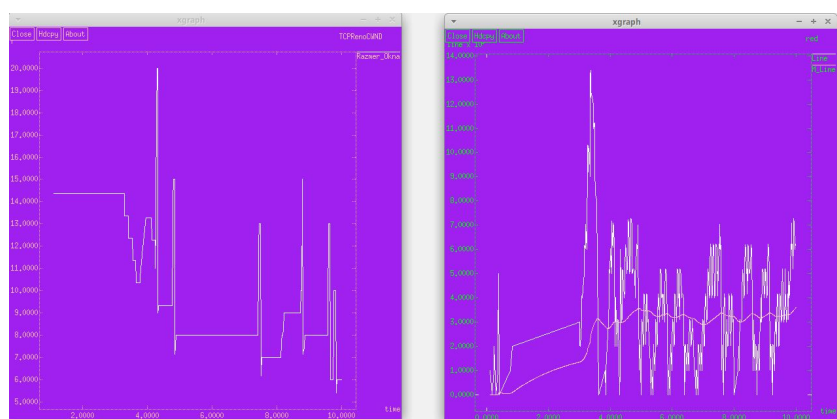


Рис. 1.10: Новый график

2 Выводы

TCP Vegas обнаруживает перегрузку в сети до того, как случайно теряется пакет, и мгновенно уменьшается размер окна. TCP Vegas обрабатывает перегрузку без потерь пакета.

Список литературы