

# **Лабораторная работа 2.**

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

Туем Гислен

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Реализация модели</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Запуск кода</b>	<b>13</b>
<b>6</b>	<b>Изменение тип протокола TCP</b>	<b>14</b>
<b>7</b>	<b>Изменения отображении окон с графиками</b>	<b>16</b>
<b>8</b>	<b>Выводы</b>	<b>18</b>

# Список иллюстраций

4.1	код . . . . .	12
5.1	График динамики размера окна TCP и динамики длины очереди и средней длины очереди . . . . .	13
6.1	вывод рафика с TCP/Newreno на узле s1 . . . . .	14
6.2	вывод рафика с TCP/Vegas на узле s1 . . . . .	15
7.1	вывод рафика с изменением отображением окном с графиками . .	17

## **Список таблиц**

# 1 Цель работы

Исследовать протокол TCP и алгоритм управления очередью RED.

## 2 Задание

1. Реализовать пример модели 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 Реализация модели**

Требуется разработать сценарий, реализующий модель, построить в Xgraph график изменения ТСР-окна, график изменения длины очереди и средней длины очереди.

## 4 Выполнение лабораторной работы

```
#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]
$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";
            }
        }
    }
```



```

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 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"
}

```

Здесь cwnd\_ – текущее значение окна перегрузки

# 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]
```

Здесь window\_ – верхняя граница окна приёмника (Advertisement Window) TCP соединения.

```
# 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_
```

Здесь curq\_ – текущий размер очереди, ave\_ – средний размер очереди.

```
#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"
```

```
# Формирование файла с данными о размере окна TCP:
```

```
proc plotWindow {tcpSource file} {
```

```
global ns
```

```
set time 0.01
```

```
set now [$ns now]
```

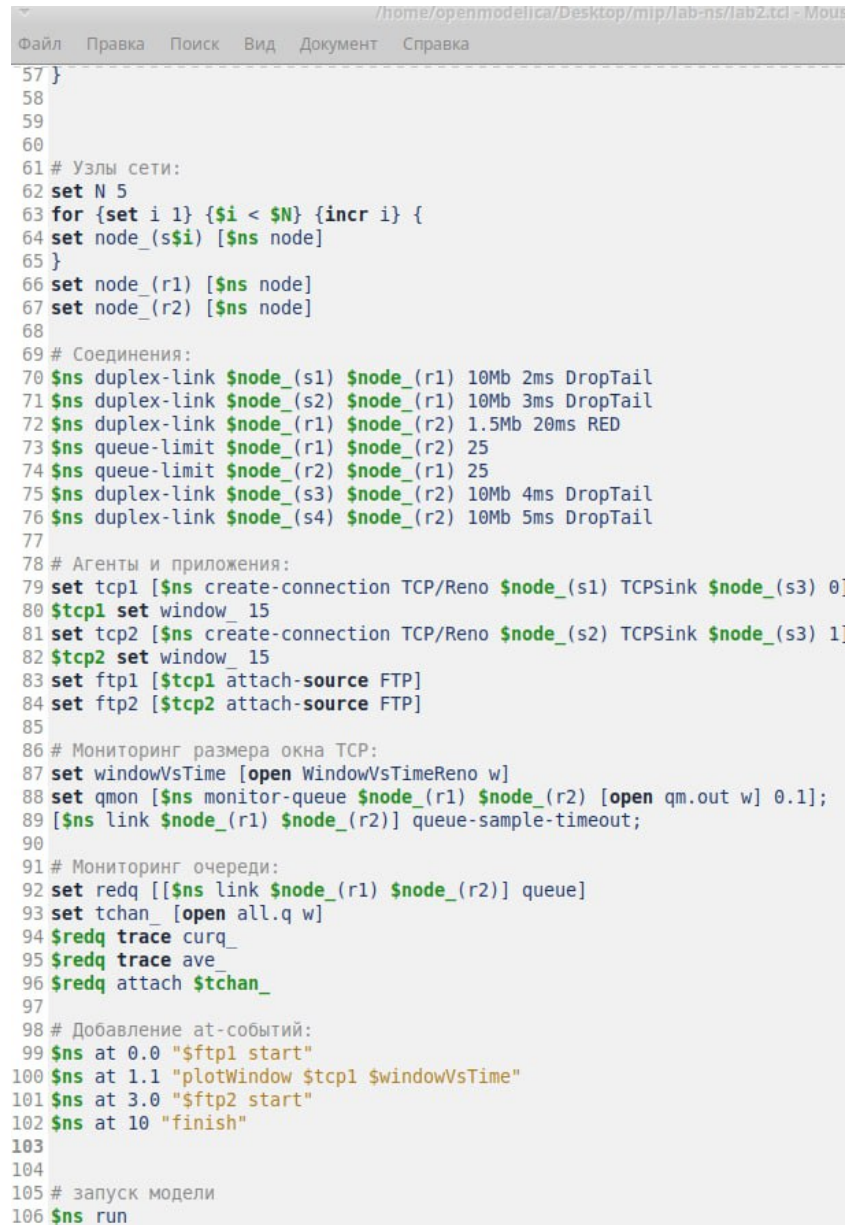
```
set cwnd [$tcpSource set cwnd_]
```

```
puts $file "$now $cwnd"
```

```
$ns at [expr $now+$time] "plotWindow $tcpSource $file"
}
```

Здесь `cwnd_` — текущее значение окна перегрузки.

мы можем посмотреть как его на картинке (рис. 4.1).



```

/home/openmodelica/Desktop/mip/lab-ns/lab2.tcl - Mou
Файл  Правка  Поиск  Вид  Документ  Справка
57 }
58
59
60
61 # Узлы сети:
62 set N 5
63 for {set i 1} {$i < $N} {incr i} {
64   set node_($i) [$ns node]
65 }
66 set node_(r1) [$ns node]
67 set node_(r2) [$ns node]
68
69 # Соединения:
70 $ns duplex-link $node_(s1) $node_(r1) 10Mb 2ms DropTail
71 $ns duplex-link $node_(s2) $node_(r1) 10Mb 3ms DropTail
72 $ns duplex-link $node_(r1) $node_(r2) 1.5Mb 20ms RED
73 $ns queue-limit $node_(r1) $node_(r2) 25
74 $ns queue-limit $node_(r2) $node_(r1) 25
75 $ns duplex-link $node_(s3) $node_(r2) 10Mb 4ms DropTail
76 $ns duplex-link $node_(s4) $node_(r2) 10Mb 5ms DropTail
77
78 # Агенты и приложения:
79 set tcp1 [$ns create-connection TCP/Reno $node_(s1) TCPSink $node_(s3) 0]
80 $tcp1 set window_ 15
81 set tcp2 [$ns create-connection TCP/Reno $node_(s2) TCPSink $node_(s3) 1]
82 $tcp2 set window_ 15
83 set ftp1 [$tcp1 attach-source FTP]
84 set ftp2 [$tcp2 attach-source FTP]
85
86 # Мониторинг размера окна TCP:
87 set windowVsTime [open WindowVsTimeReno w]
88 set qmon [$ns monitor-queue $node_(r1) $node_(r2) [open qm.out w] 0.1];
89 [$ns link $node_(r1) $node_(r2)] queue-sample-timeout;
90
91 # Мониторинг очереди:
92 set redq [[$ns link $node_(r1) $node_(r2)] queue]
93 set tchan_ [open all.q w]
94 $redq trace curq_
95 $redq trace ave_
96 $redq attach $tchan_
97
98 # Добавление at-событий:
99 $ns at 0.0 "$ftp1 start"
100 $ns at 1.1 "plotWindow $tcp1 $windowVsTime"
101 $ns at 3.0 "$ftp2 start"
102 $ns at 10 "finish"
103
104
105 # запуск модели
106 $ns run

```

Рис. 4.1: код

## 5 Запуск кода

После запуска кода мы получим изменения ТСР-окна, график изменения длины очереди и средней длины очереди

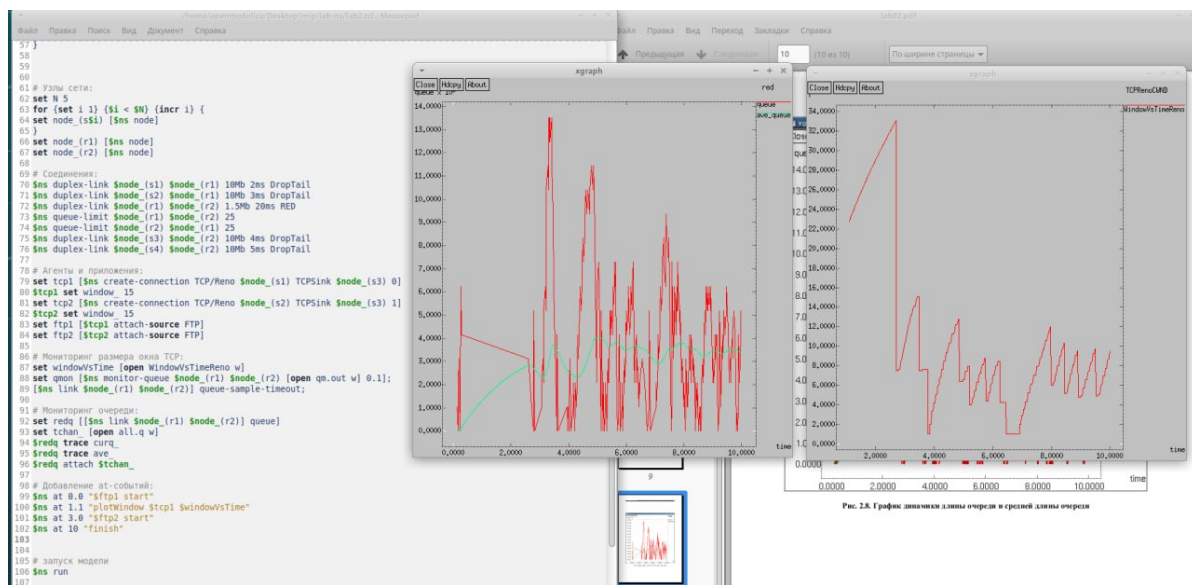


Рис. 5.1: График динамики размера окна ТСР и динамики длины очереди и средней длины очереди

## 6 Изменение тип протокола TCP

Изменение в модели на узле s1 тип протокола TCP с Reno на NewReno

# Агенты и приложения:

```
set tcp1 [$ns create-connection TCP/Newreno $node_(s1) TCPSink $node_(s3) 0]
```

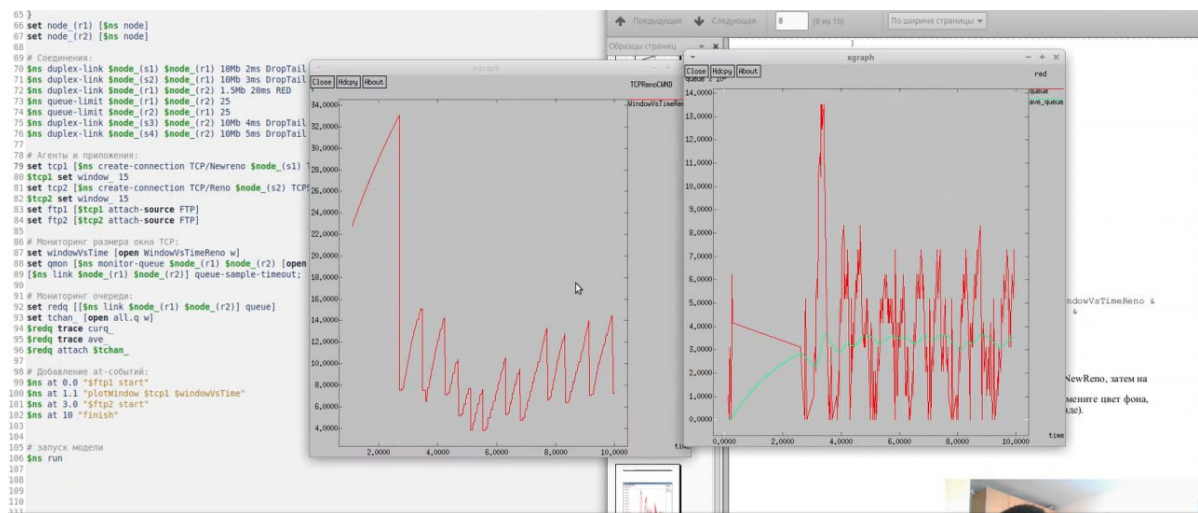


Рис. 6.1: вывод графика с TCP/Newreno на узле s1

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.

Изменение в модели на узле s1 тип протокола TCP с Reno на Vegas

# Агенты и приложения:

```
set tcp1 [$ns create-connection TCP/Vegas $node_(s1) TCPSink $node_(s3) 0]
```

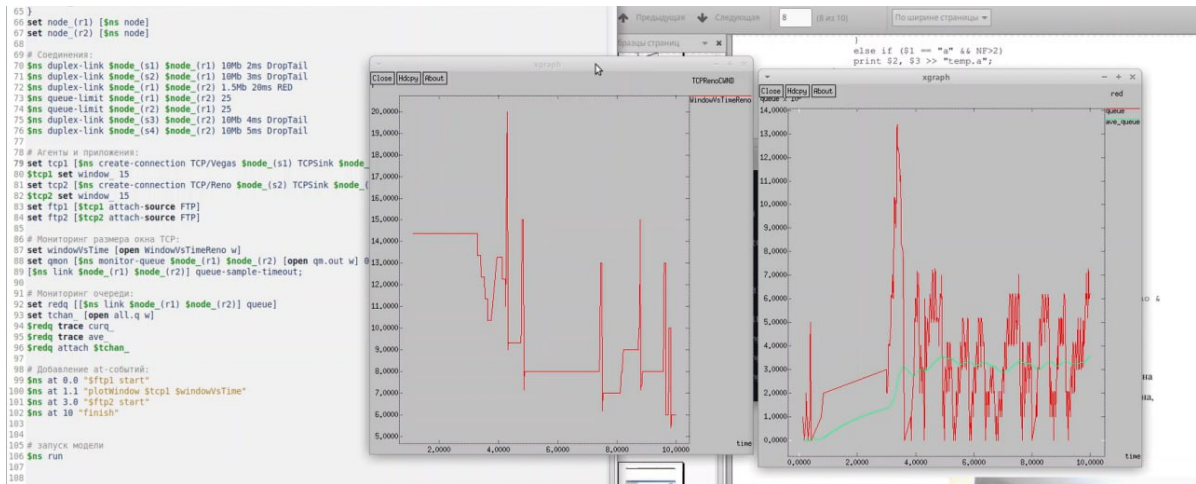


Рис. 6.2: вывод графика с TCP/Vegas на узле s1

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.

## 7 Изменения отображении окон с графиками

Внесем изменения при отображении окон с графиками, изменим цвет фона, цвет траекторий, подписи к осям и подпись траектории в легенде. В процедуре `finish` изменим цвет траекторий, подписи легенд, а также добавив опции `-fg` и `-bg` изменим цвет текста и фона в `xgraph`.

```
set f [open temp.queue w]
puts $f "TitleText: RED"
puts $f "Device: Postscript"
puts $f "0.color: white"
puts $f "1.color: red"
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 \"petit
exec cat temp.q >@ $f
puts $f \"ave_petit
exec cat temp.a >@ $f
close $f
```



```
exit 0
```

```
}
```

```
exec xgraph -fg yellow -bg black -bb -tk -x time -t "TCPrenoCWND" WindowVsTimeReno  
& exec xgraph -fg green -bg black -bb -tk -x time -y queue temp.queue &
```

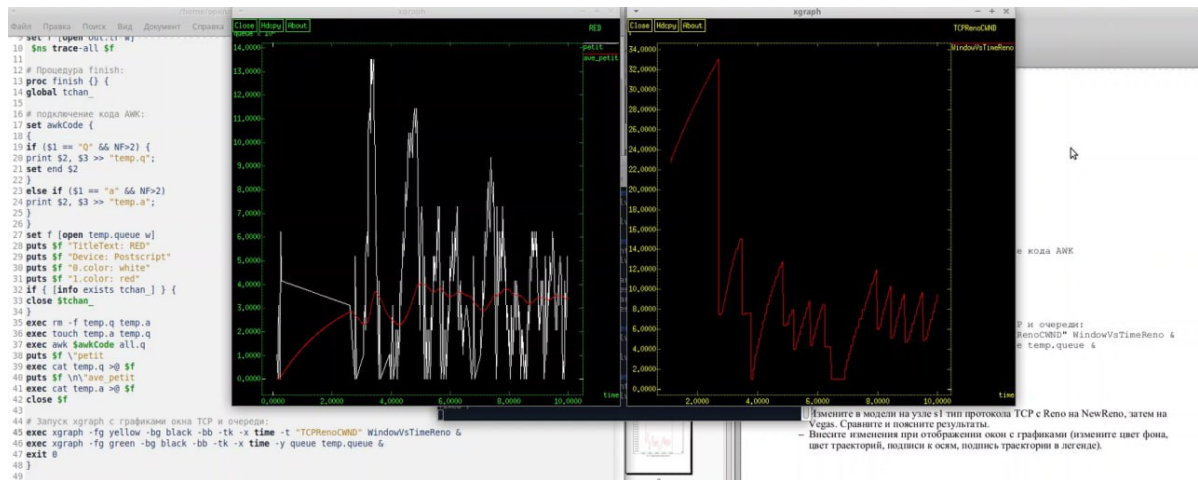


Рис. 7.1: вывод графика с изменением отображением окном с графиками

## **8 Выводы**

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