

# **Лабораторная работа №2**

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

Городянский Фёдор Николаевич

# Содержание

|  |           |
|--|-----------|
| <b>Цель работы</b>                               | <b>4</b>  |
| <b>Задание</b>                                   | <b>5</b>  |
| <b>Выполнение лабораторной работы</b>            | <b>6</b>  |
| Изменение протокола ТСР . . . . .                | 11        |
| Изменение отображения окон с графиками . . . . . | 15        |
| <b>Выводы</b>                                    | <b>18</b> |

## Список иллюстраций

|   |  |    |
|---|--|----|
| 1 | Графики изменения ТСР-окна, динамики длины очереди и средней длины очереди . . . . .                                 | 11 |
| 2 | График динамики длины очереди и средней длины очереди. Тип NewReno . . . . .   | 13 |
| 3 | Графики динамики размера окна ТСР, динамики длины очереди и средней длины очереди. Тип Vegas . . . . .               | 14 |
| 4 | Графики динамики размера окна ТСР, динамики длины очереди и средней длины очереди с изменением отображения . . . . . | 17 |

## **Цель работы**

Исследовать протокол TCP и алгоритм управления очередью RED.

# 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).

# 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;
- ТСП-источники на узлах s1 и s2 подключаются к ТСП-приёмнику на узле s3;
- генераторы трафика FTP прикреплены к ТСП-агентам.

Теперь разработаем сценарий, реализующий модель согласно описанию, чтобы построить в Xgraph график изменения ТСП-окна, график изменения длины очереди и средней длины очереди.

```

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_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

# at-событие для планировщика событий, которое запускает
# процедуру finish через 10 с после начала моделирования
# Добавление at-событий:
$ns at 0.0 "sftp1 start"
$ns at 1.1 "plotwindow $tcp1 $windowVstime"
$ns at 3.0 "sftp2 start"
$ns at 10 "finish"

# запуск модели
$ns run

```

(рис. [-@fig:001])

# 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 "\\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"
}

# 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 ТСР-окна, а также график изменения длины очереди и средней длины очереди (рис. [-@fig:002]).

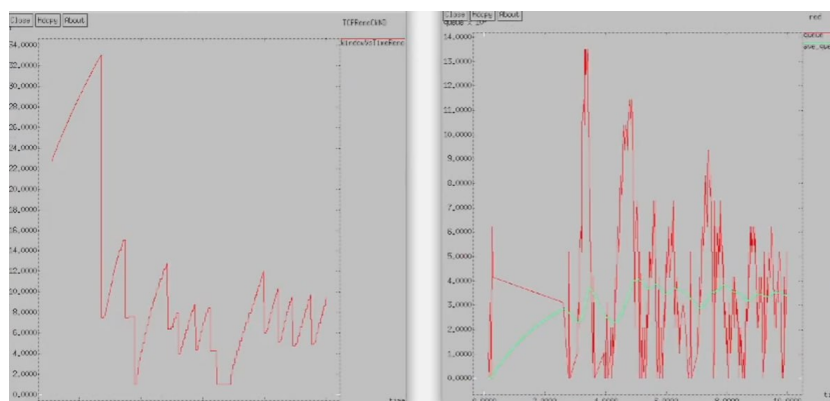


Рис. 1: Графики изменения ТСР-окна, динамики длины очереди и средней длины очереди

По графику видно, что средняя длина очереди находится в диапазоне от 2 до 4. Максимальная длина достигает значения 14.

## 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
```

```

)
# 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 < $N} {incr i} {
  set node_$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/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]

# 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

```

(рис. [-@fig:003])

В результате получим график изменения длины очереди и средней длины очереди (рис. [-@fig:004]).

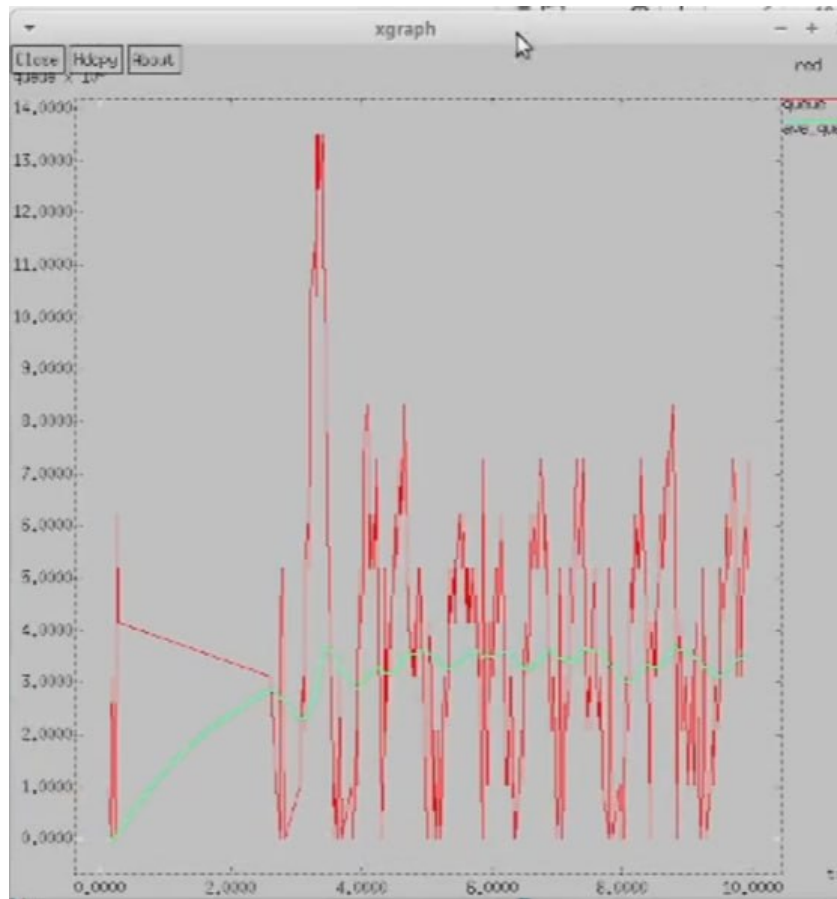


Рис. 2: График динамики длины очереди и средней длины очереди. Тип NewReno

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
```

```

~/home/openmodelica/mipifabns/example6.tcl - Mousepad
Файл Правка Поиск Вид Документ Справка
set tcp1 [tcpSource set tcp1]
puts $file "$now $cmd"
$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_(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/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]

# 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

# at-событие для планировщика событий, которое запускает
# процедуру finish через 10 с после начала моделирования
# Добавление at-событий:
$ns at 0.0 "$ftp1 start"
$ns at 1.1 "plotWindow tcp1 $windowVsTime"
$ns at 3.0 "$ftp2 start"

```

(рис. [-@fig:005])

В результате получим следующие график изменения TCP-окна, а также график изменения длины очереди и средней длины очереди (рис. [-@fig:006]).

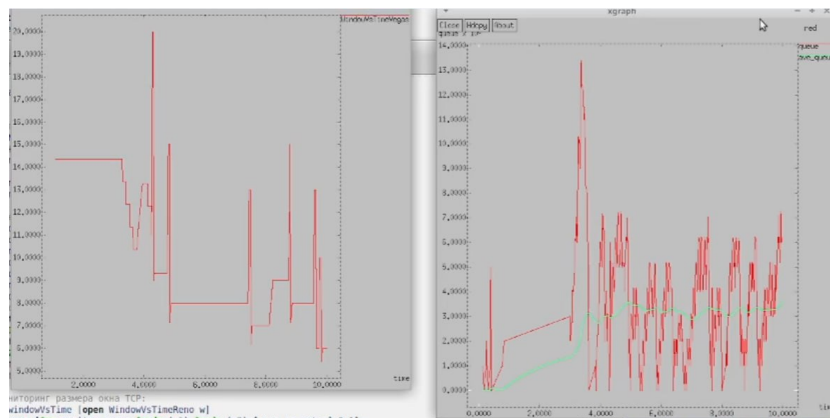


Рис. 3: Графики динамики размера окна TCP, динамики длины очереди и средней длины очереди. Тип Vegas

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.

## Изменение отображения окон с графиками

Внесем изменения при отображении окон с графиками, изменим цвет фона, цвет траекторий, подписи к осям и подпись траектории в легенде. Для этого изменим наш код:

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: Blue"
puts $f "1.Color: Purple"
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_Vegas"
exec cat temp.q >@ $f
puts $f "\n\"ave_queue_Vegas"
exec cat temp.a >@ $f
```

```
close $f
```

# Запуск xgraph с графиками окна TCP и очереди:

```
exec xgraph -fg blue -bg purple -bb -tk -x time -t "TCPVegasCWND" WindowVsTimeVegas &
```

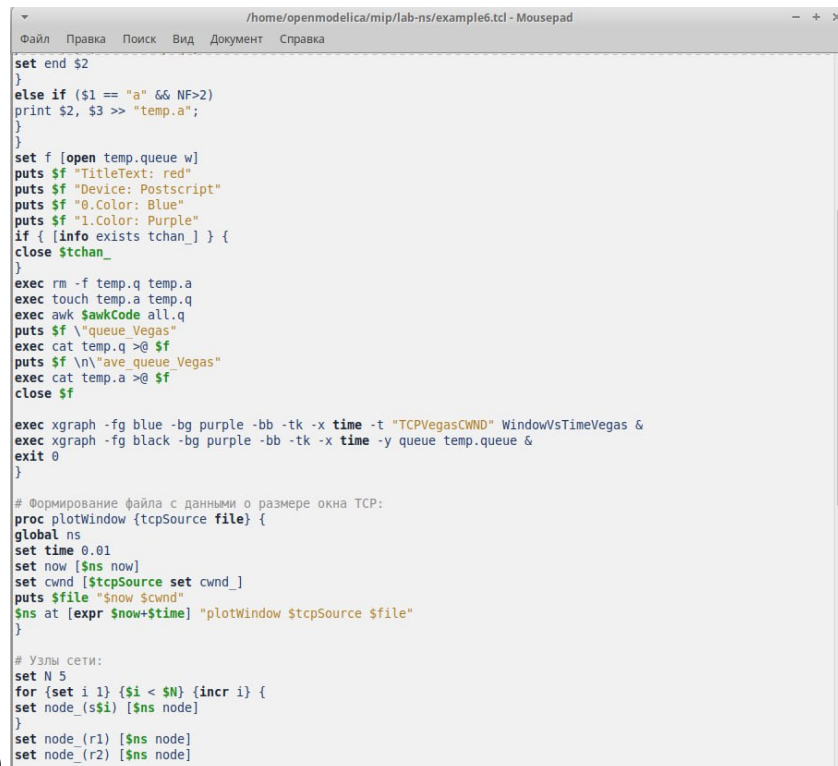
```
exec xgraph -fg black -bg purple -bb -tk -x time -y ochered temp.queue &
```

В разделе мониторинга размера окна TCP также изменим цвет траектории и подпись легенды.

```
set windowVsTime [open WindowVsTimeReno w]
```

```
puts $windowVsTime "0.Color: White"
```

```
puts $windowVsTime \"Razmer_Okna\"
```



```
set end $2
}
else if {$1 == "a" 66 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: Blue"
puts $f "1.Color: Purple"
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_Vegas\""
exec cat temp.q >@ $f
puts $f "\\n\\\"ave queue Vegas\""
exec cat temp.a >@ $f
close $f

exec xgraph -fg blue -bg purple -bb -tk -x time -t "TCPVegasCWND" WindowVsTimeVegas &
exec xgraph -fg black -bg purple -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 < $N} {incr i} {
set node_($i) [$ns node]
}
set node_(r1) [$ns node]
set node_(r2) [$ns node]
```

(рис. [-@fig:007])

В результате получим следующие график изменения TCP-окна, а также график изменения длины очереди и средней длины очереди (рис. [-@fig:008]).



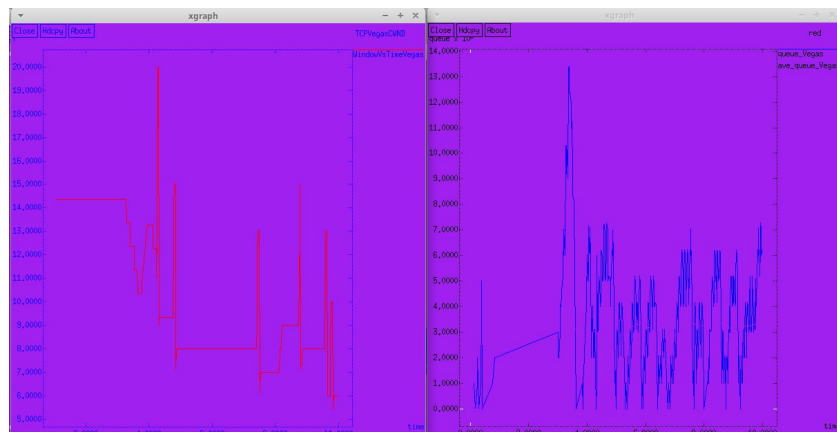


Рис. 4: Графики динамики размера окна ТСР, динамики длины очереди и средней длины очереди с изменением отображения

## **Выводы**

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