

Лабораторная работа №4

Задание для самостоятельного выполнения

Тимофеева Екатерина Николаевна

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
4	Выводы	19

Список иллюстраций

3.1	Схема моделируемой сети при $N=30$	11
3.2	Изменение размера окна ТСР на линке 1-го источника при $N=30$.	12
3.3	Изменение размера окна ТСР на всех источниках при $N=30$	13
3.4	Изменение размера длины очереди на линке ($R1-R2$) при $N=30$. .	13
3.5	Изменение размера средней длины очереди на линке ($R1-R2$) при $N=30$	14
3.6	Изменение размера окна ТСР на линке 1-го источника при $N=30$.	17
3.7	Изменение размера окна ТСР на всех источниках при $N=30$	17
3.8	Изменение размера длины очереди на линке ($R1-R2$) при $N=30$. .	18
3.9	Изменение размера средней длины очереди на линке ($R1-R2$) при $N=30$	18

1 Цель работы

Выполнить задание для самостоятельного выполнения.

2 Задание

1. Для приведённой схемы разработать имитационную модель в пакете NS-2;
2. Построить график изменения размера окна TCP (в Xgraph и в GNUPlot);
3. Построить график изменения длины очереди и средней длины очереди на первом маршрутизаторе;
4. Оформить отчёт о выполненной работе.

3 Выполнение лабораторной работы

Описание моделируемой сети:

- сеть состоит из N TCP-источников, N TCP-приёмников, двух маршрутизаторов $R1$ и $R2$ между источниками и приёмниками (N — не менее 20);
- между TCP-источниками и первым маршрутизатором установлены дуплексные соединения с пропускной способностью 100 Мбит/с и задержкой 20 мс очередью типа DropTail;
- между TCP-приёмниками и вторым маршрутизатором установлены дуплексные соединения с пропускной способностью 100 Мбит/с и задержкой 20 мс очередью типа DropTail;
- между маршрутизаторами установлено симплексное соединение ($R1-R2$) с пропускной способностью 20 Мбит/с и задержкой 15 мс очередью типа RED, размером буфера 300 пакетов; в обратную сторону — симплексное соединение ($R2-R1$) с пропускной способностью 15 Мбит/с и задержкой 20 мс очередью типа DropTail;
- данные передаются по протоколу FTP поверх TCP Reno;
- параметры алгоритма RED: $q_{min} = 75$, $q_{max} = 150$, $q_w = 0,002$, $p_{max} = 0.1$;
- максимальный размер TCP-окна 32; размер передаваемого пакета 500 байт; время моделирования — не менее 20 единиц модельного времени.

Откроем файл .tcl на редактирование, в нем построим сеть. Зададим $N = 30$ TCP-источников, $N = 30$ TCP-приёмников, два маршрутизатора $r1$ и $r2$ между

источниками и приёмниками. Между TCP-источниками и первым маршрутизатором установим дуплексные соединения с пропускной способностью 100 Мбит/с и задержкой 20 мс очередью типа DropTail; между TCP-приёмниками и вторым маршрутизатором установлены дуплексные соединения с пропускной способностью 100 Мбит/с и задержкой 20 мс очередью типа DropTail; между маршрутизаторами установлено симплексное соединение (R1–R2) с пропускной способностью 20 Мбит/с и задержкой 15 мс очередью типа RED, размером буфера 300 пакетов; в обратную сторону - симплексное соединение (R2–R1) с пропускной способностью 15 Мбит/с и задержкой 20 мс очередью типа DropTail. Данные передаются по протоколу FTP поверх TCP Reno. Зададим также параметры алгоритма RED: $q_{min} = 75$, $q_{max} = 150$, $q_w = 0,002$, $p_{max} = 0.1$. Также нам нужно выполнить мониторинг окна TCP и мониторинг очереди. Листинг такой программы выглядит следующим образом:

```
# 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

Agent/TCP set window_ 32
```

```
Agent/TCP set pktSize_ 500
```

```
# 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";  
        }  
    }  
}
```

```
exec rm -f temp.q temp.a
```

```
exec touch temp.a temp.q
```

```
set f [open temp.q w]
```

```
puts $f "0.Color: Purple"
```

```
close $f
```

```
set f [open temp.a w]
```

```
puts $f "0.Color: Purple"
```

```
close $f
```



```
exec awk $awkCode all.q
```

```
# Запуск xgraph с графиками окна TCP и очереди:
```

```
exec xgraph -fg pink -bg purple -bb -tk -x time -t "TCPReoCWND" WindowVsTimeReno
```

```
exec xgraph -fg pink -bg purple -bb -tk -x time -t "TCPReoCWND" WindowVsTimeReno
```

```
exec xgraph -bb -tk -x time -y queue temp.q &
```

```
exec xgraph -bb -tk -x time -y queue temp.a &
```

```
exec nam out.nam &
```

```
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"
```

```
}
```

```
set r1 [$ns node]
```

```
set r2 [$ns node]
```

```
$ns simplex-link $r1 $r2 20Mb 15ms RED
```

```
$ns simplex-link $r2 $r1 15Mb 20ms DropTail
```

```
$ns queue-limit $r1 $r2 300
```

```
set N 30
```

```

for {set i 0} {$i < $N} {incr i} {
    set n1($i) [$ns node]
    $ns duplex-link $n1($i) $r1 100Mb 20ms DropTail
    set n2($i) [$ns node]
    $ns duplex-link $n2($i) $r2 100Mb 20ms DropTail

    set tcp($i) [$ns create-connection TCP/Reno $n1($i) TCPSink $n2($i) $i]
    set ftp($i) [$tcp($i) attach-source FTP]
}

```

Мониторинг размера окна TCP:

```

set windowVsTimeOne [open WindowVsTimeRenoOne w]
puts $windowVsTimeOne "0.Color: White"
set windowVsTimeAll [open WindowVsTimeRenoAll w]
puts $windowVsTimeAll "0.Color: White"

```

```

set qmon [$ns monitor-queue $r1 $r2 [open qm.out w] 0.1];
[$ns link $r1 $r2] queue-sample-timeout;

```

Мониторинг очереди:

```

set redq [[$ns link $r1 $r2] queue]
$redq set thresh_ 75
$redq set maxthresh_ 150
$redq set q_weight_ 0.002
$redq set linterm_ 10

```

```

set tchan_ [open all.q w]
$redq trace curq_
$redq trace ave_

```

```
$redq attach $tchan_
```

```
for {set i 0} {$i < $N} {incr i} {  
    $ns at 0.0 "$ftp($i) start"  
    $ns at 0.0 "plotWindow $tcp($i) $windowVsTimeAll"  
}
```

```
$ns at 0.0 "plotWindow $tcp(1) $windowVsTimeOne"
```

```
# at-событие для планировщика событий, которое запускает
```

```
# процедуру finish через 20s после начала моделирования
```

```
$ns at 20.0 "finish"
```

```
# запуск модели
```

```
$ns run
```

Запустив созданную программу на выполнение получим nam файл со схемой моделируемой сети (рис. 3.1).

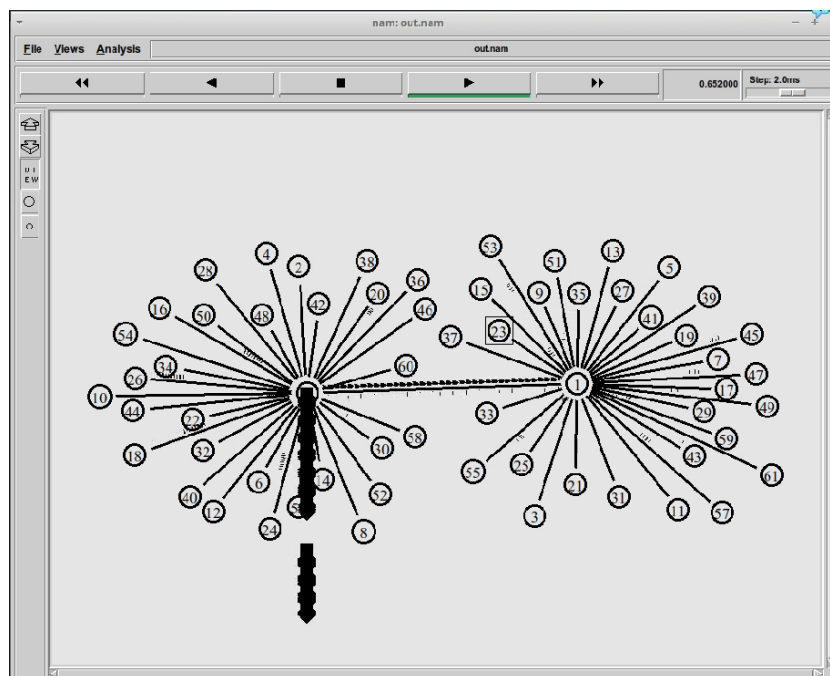


Рис. 3.1: Схема моделируемой сети при N=30

Также получим графики изменения размера окна ТСР на линке 1-го источника (рис. 3.2) и на всех источниках (рис. 3.3). Графики построены с помощью xgraph.

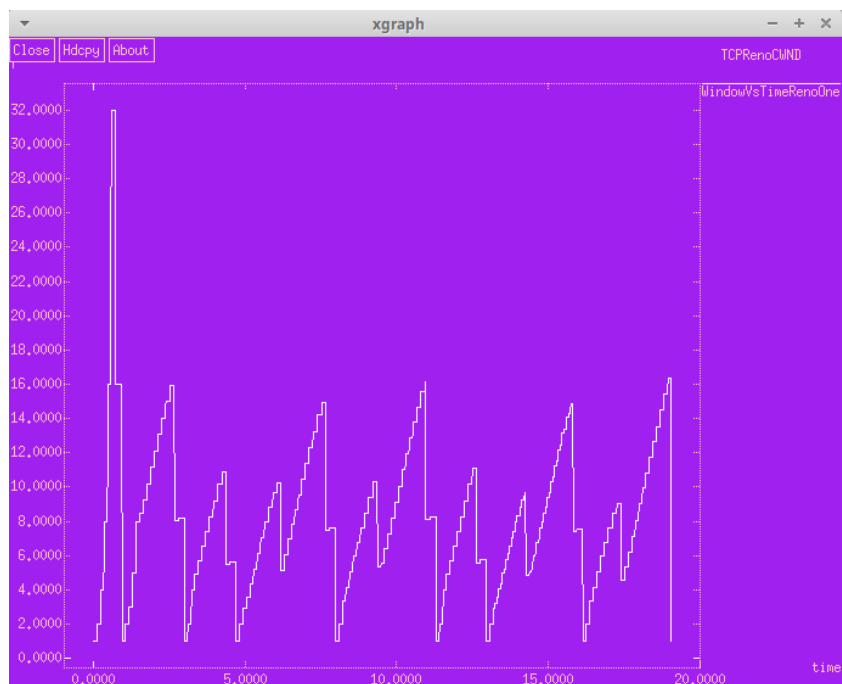


Рис. 3.2: Изменение размера окна ТСР на линке 1-го источника при N=30

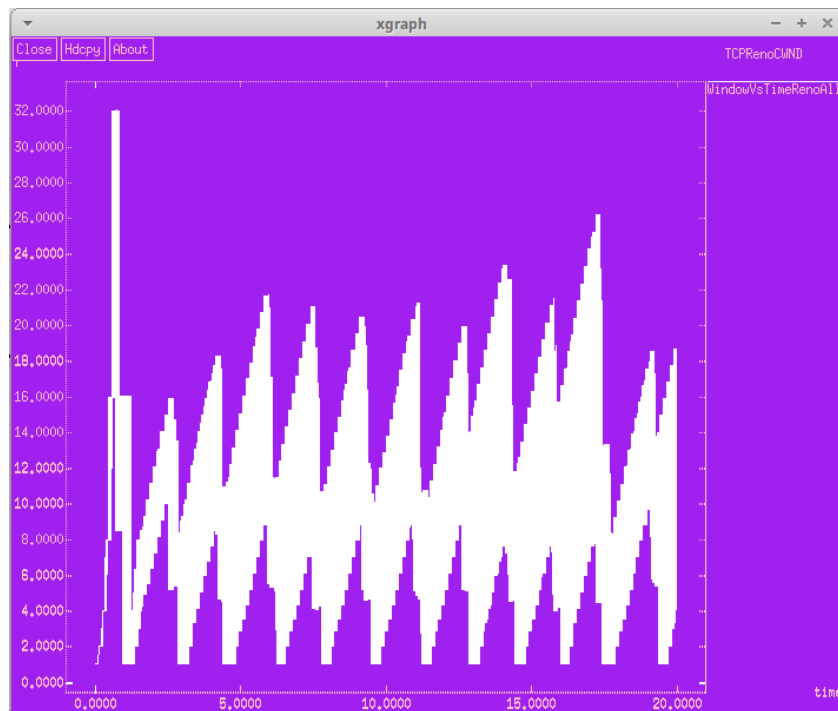


Рис. 3.3: Изменение размера окна TCP на всех источниках при $N=30$

Еще получим графики изменения размера длины очереди (рис. 3.4) и размера средней длины очереди (рис. 3.5). Графики построены с помощью xgraph.

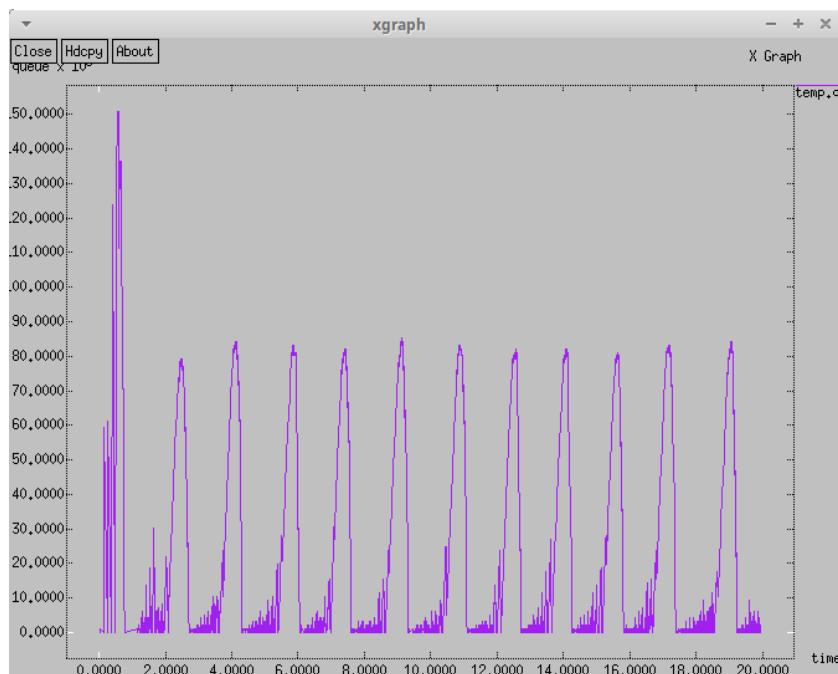


Рис. 3.4: Изменение размера длины очереди на линке (R1–R2) при $N=30$

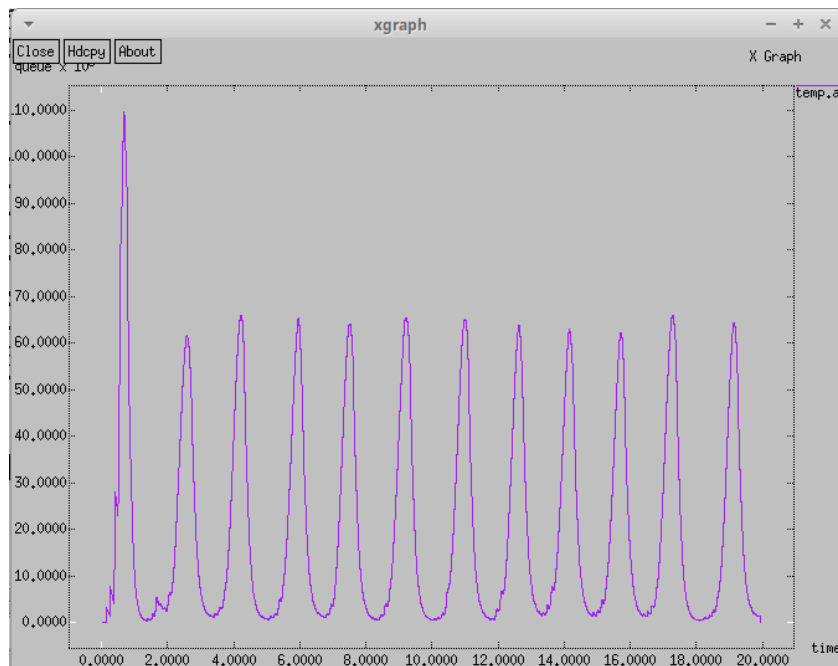


Рис. 3.5: Изменение размера средней длины очереди на линке (R1–R2) при N=30

Напишем программу для построения графиков в GNUPlot:

```
#!/usr/bin/gnuplot -persist
# задаём текстовую кодировку,
# тип терминала, тип и размер шрифта

set encoding utf8
set term pngcairo font "Helvetica,9"

# задаём выходной файл графика
set out 'window_1.png'

# задаём название графика
set title "Изменение размера окна TCP на линке 1-го источника при N=30"

# подписи осей графика
```

```

set xlabel "t[s]" font "Helvetica, 10"
set ylabel "CWND [pkt]" font "Helvetica, 10"

# построение графика, используя значения
# 1-го и 2-го столбцов файла WindowVsTimeRenoOne
plot "WindowVsTimeRenoOne" using ($1):($2) with lines title "Размер окна TCP"

# задаём выходной файл графика
set out 'window_2.png'

# задаём название графика
set title "Изменение размера окна TCP на всех N источниках при N=30"

# построение графика, используя значения
# 1-го и 2-го столбцов файла WindowVsTimeRenoAll
plot "WindowVsTimeRenoAll" using ($1):($2) with lines title "Размер окна TCP"

# задаём выходной файл графика
set out 'queue.png'

# задаём название графика
set title "Изменение размера длины очереди на линке (R1-R2)"

# подписи осей графика
set xlabel "t[s]" font "Helvetica, 10"
set ylabel "Queue Length [pkt]" font "Helvetica, 10"

# построение графика, используя значения
# 1-го и 2-го столбцов файла temp.q

```

```

plot "temp.q" using ($1):($2) with lines title "Текущая длина очереди"

# задаём выходной файл графика
set out 'av_queue.png'

# задаём название графика
set title "Изменение размера средней длины очереди на линке (R1-R2)"

# подписи осей графика
set xlabel "t[s]" font "Helvetica, 10"
set ylabel "Queue Avg Length [pkt]" font "Helvetica, 10"

# построение графика, используя значения
# 1-го и 2-го столбцов файла temp.a
plot "temp.a" using ($1):($2) with lines title "Средняя длина очереди"

```

Сделаем исполняемым и запустим его. Получим 4 графика.

Графики изменения размера окна TCP на линке 1-го источника (рис. 3.6) и на всех источниках (рис. 3.7).

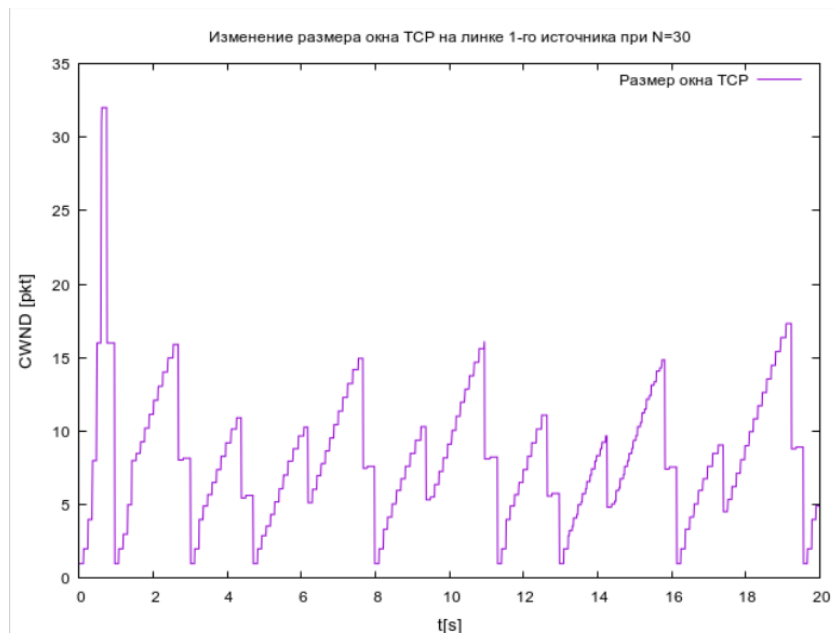


Рис. 3.6: Изменение размера окна TCP на линке 1-го источника при N=30

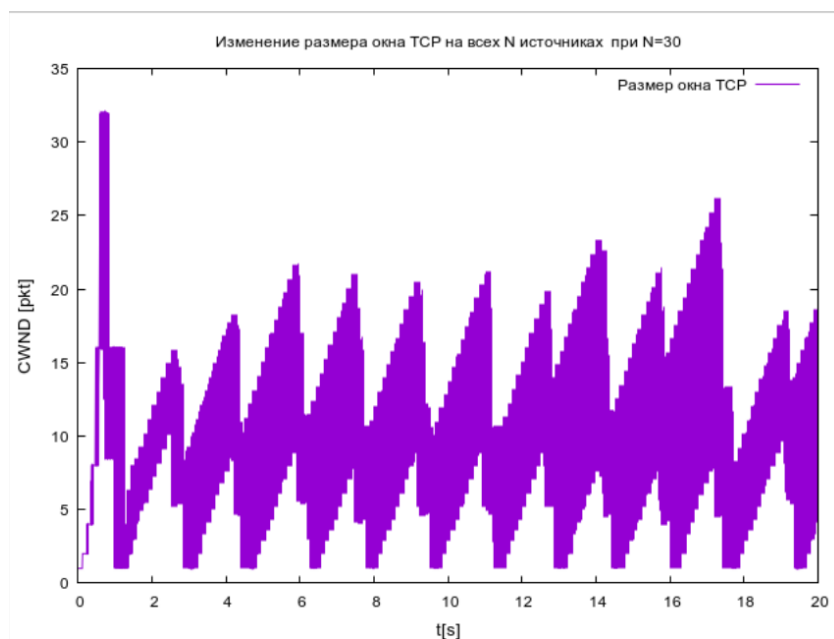


Рис. 3.7: Изменение размера окна TCP на всех источниках при N=30

Графики изменения размера длины очереди (рис. 3.8) и размера средней длины очереди (рис. 3.9).

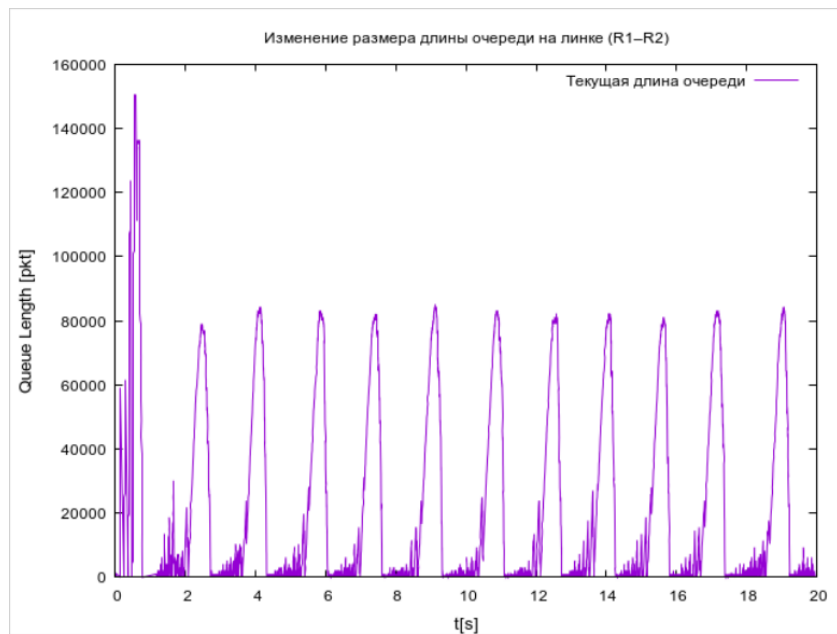


Рис. 3.8: Изменение размера длины очереди на линке (R1–R2) при N=30

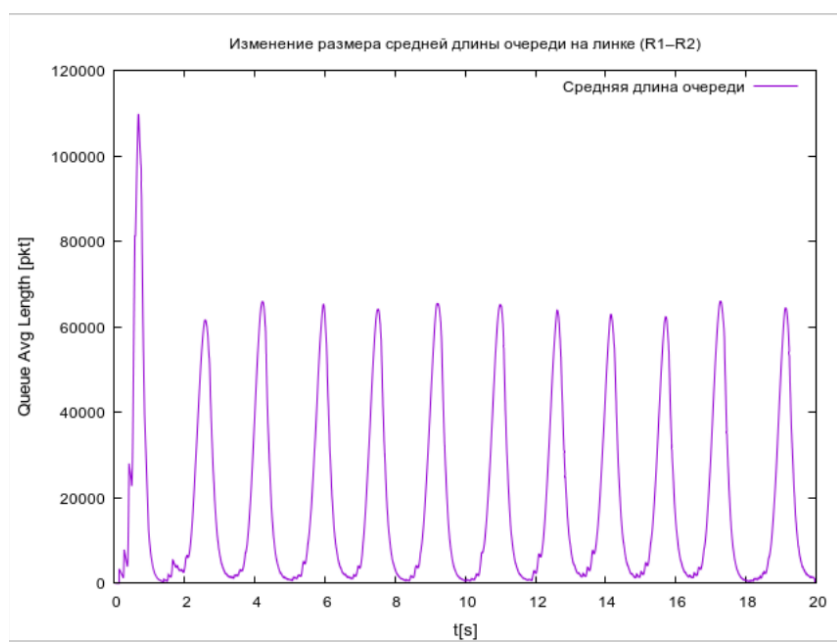


Рис. 3.9: Изменение размера средней длины очереди на линке (R1–R2) при N=30

4 Выводы

В результате выполнения данной лабораторной работы была разработана имитационная модель в пакете NS-2, построены графики изменения размера окна ТСР, изменения длины очереди и средней длины очереди.