РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

дисциплина: Моделирование информационных процессов

Студент: Николаев Александр Викторович

Группа: НФИбд-01-17

МОСКВА

2020 г.

Цель работы

При помощи NS-2 смоделировать сеть с большим количеством источников, приемников и парой маршрутизаторов для закрепления навыков работы в этой программе. Также проанализировать информацию о размере окон и очередей, построив соответствующие графики в программах xgrahp и GNUplot, продемонстрировав умение с ними работать.

Выполнение работы

Постановка задачи: реализовать модель сети, согласно следующему описанию:

- Сеть состоит из N TCP-источников, N TCP-приёмников, двух маршрутизаторов R1 и R2 между источниками и приёмниками (N – не менее 20). В своей работе я взял N = 24;
- 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 TCPReno;
- Параметры алгоритма RED: $q_{min} = 75$, $q_{max} = 150$, $q_w = 0{,}002$, $p_{max} = 0.1$;
- Максимальный размер TCP-окна 32; время моделирования – не менее 20 единиц модельного времени (я выбрал ровно 20)

Для выполнения работы я разбил задачу на три этапа. 1) Разработал имитационную модель в пакете NS-2 (написал соответствующий tcl файл). 2) Добавил в него код для построения графиков в xgraph. 3) Написал скрипт для построения графиков в GNUplot (они все строятся по отдельности, но сохраняются в один pdf файл).

Таким образом в результате работы получилось две программы: первая – реализующая модель, построение в xgraph всех необходимых графиков (а их 5 штук) и сохранение информации для дальнейшего построения этих же графиков в gnuplot (файл lab4.tcl). Вторая – скрипт, для gnuplot (файл gnu_plot).

Я постарался снабдить код в листинге подробными комментариями, поэтому здесь не буду подробно останавливаться на кусках кода, а лучше покажу результаты и немного проанализирую графики.

Результат

В результате создания имитационной модели была получена следующая схема:

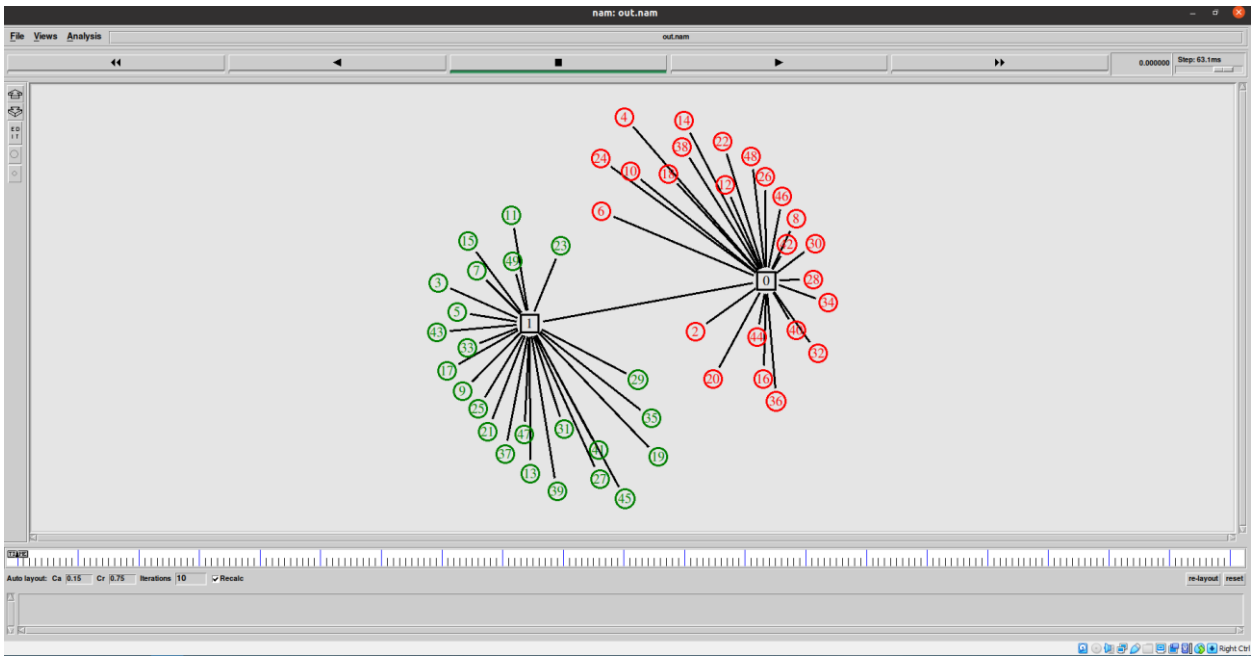

Рисунок 1. Имитационная модель в NS-2. nam интерфейс

Специально для отображения в nam и отличий я сделал маршрутизаторы квадратными, приемники зелеными, а источники красными. (В коде я обозначаю источник буквой s (source), а приемник буквой t (так принято в теории графов)). Видно, что всё соединено по условию задачи.

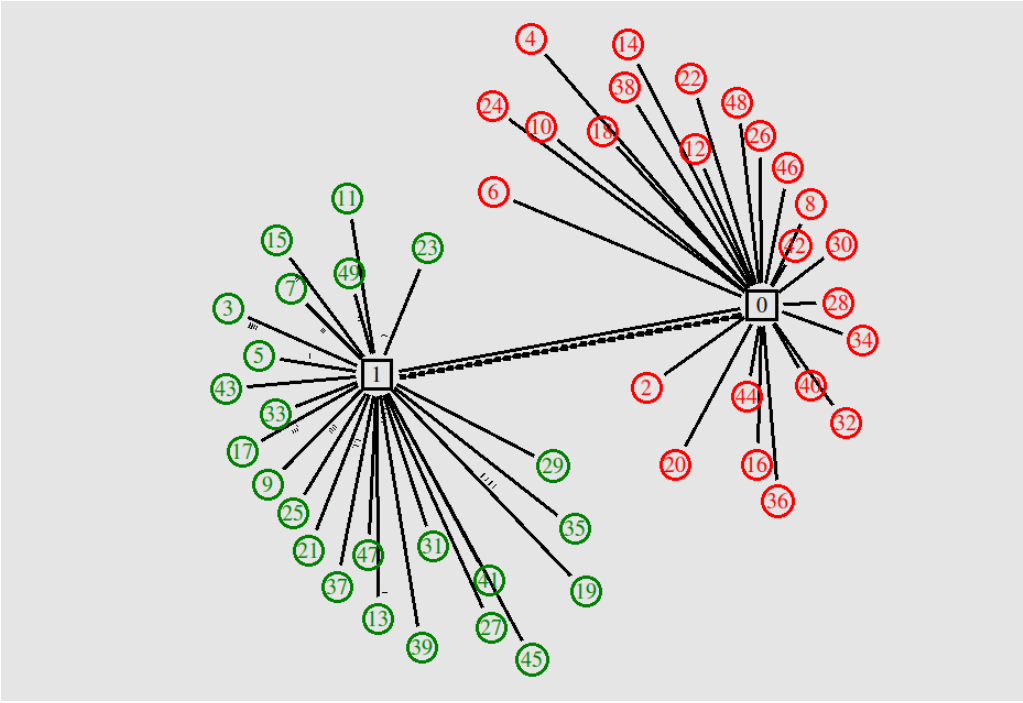

Рисунок 2. Процесс передачи данных

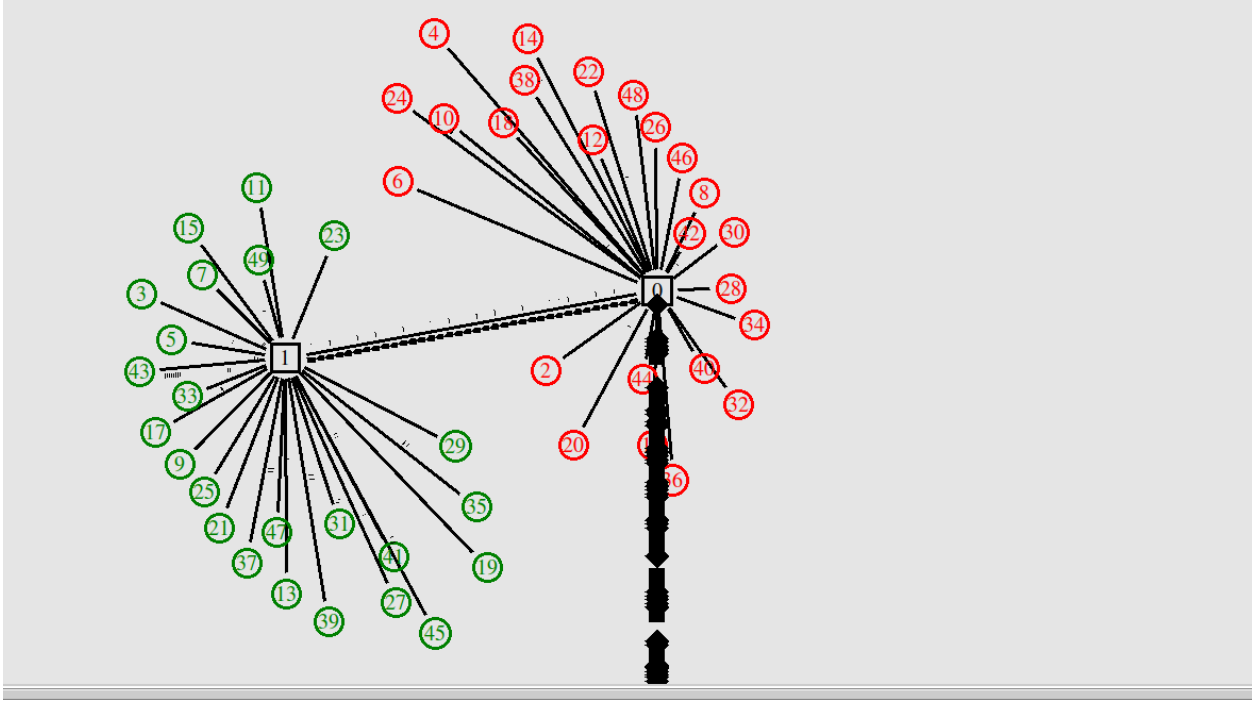

Рисунок 3. Потеря пакетов

На рис. 3. Видна потеря пакетов, и это естественно, т.к. очередь имеет лимит и tcp окно так же не бесконечно.

Графики размера TCP окон.

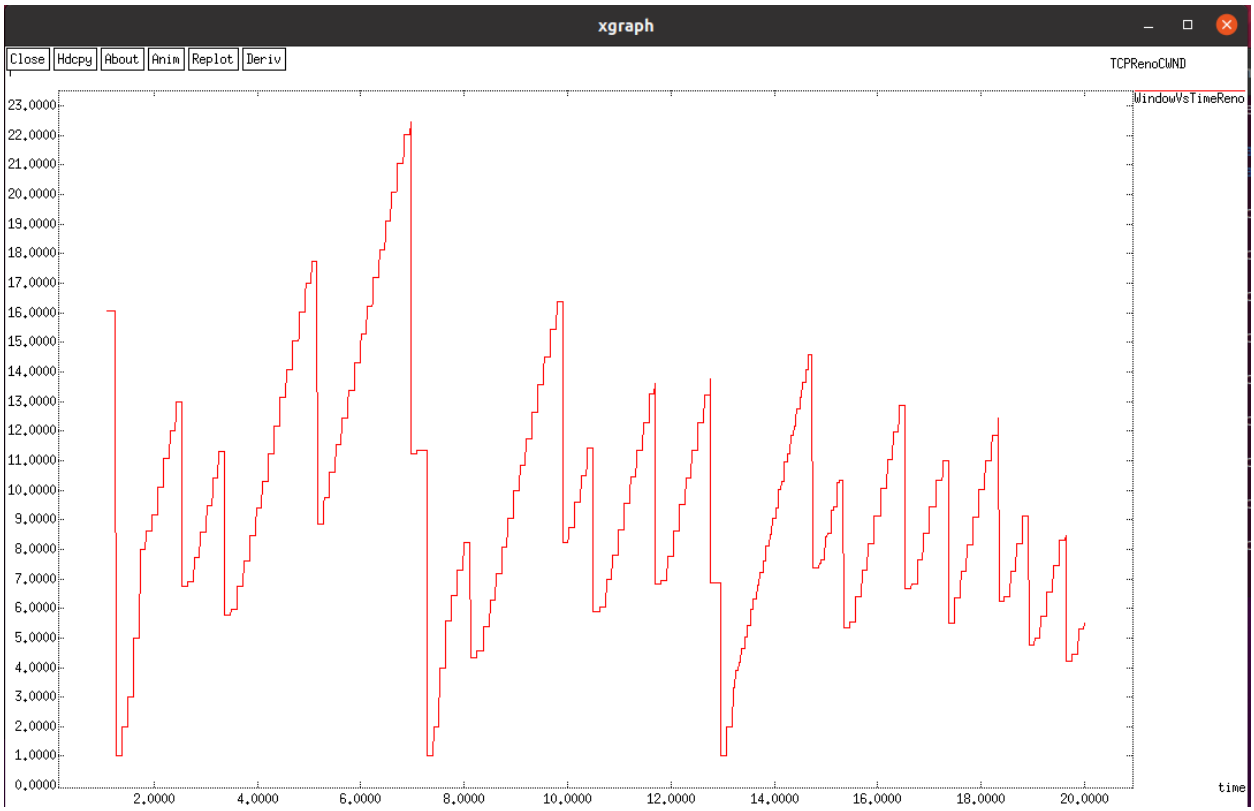

Рисунок 4. График изменения TCP-окна на одном источнике. Xgraph

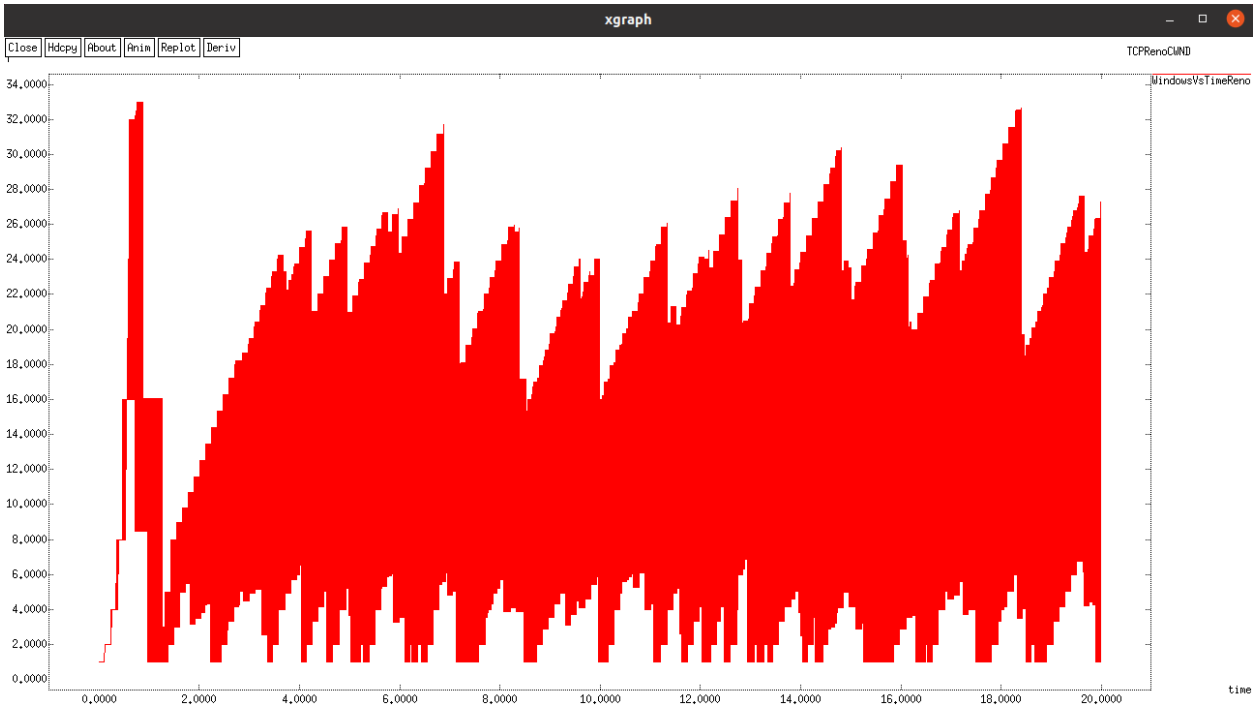

Рисунок 5. График изменения TCP-окон на всех источниках. Xgraph.

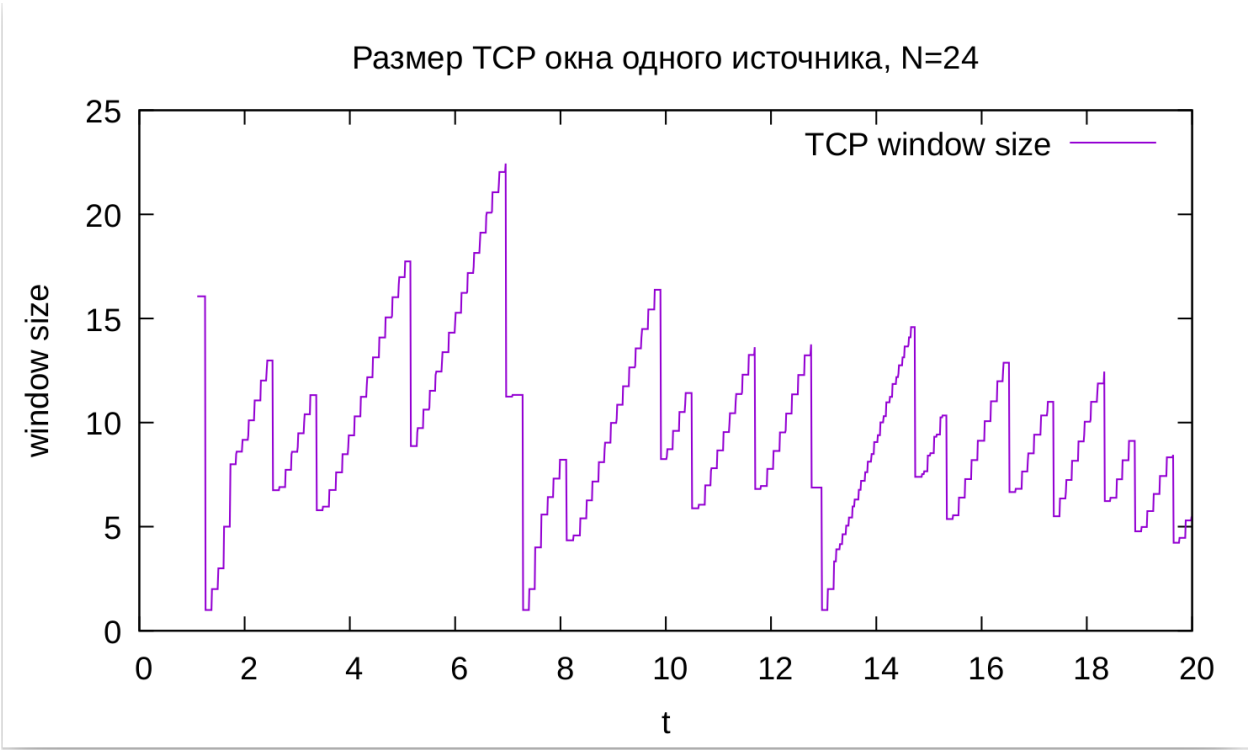

Рисунок 6. Аналогично рис. 4., только GNUplot

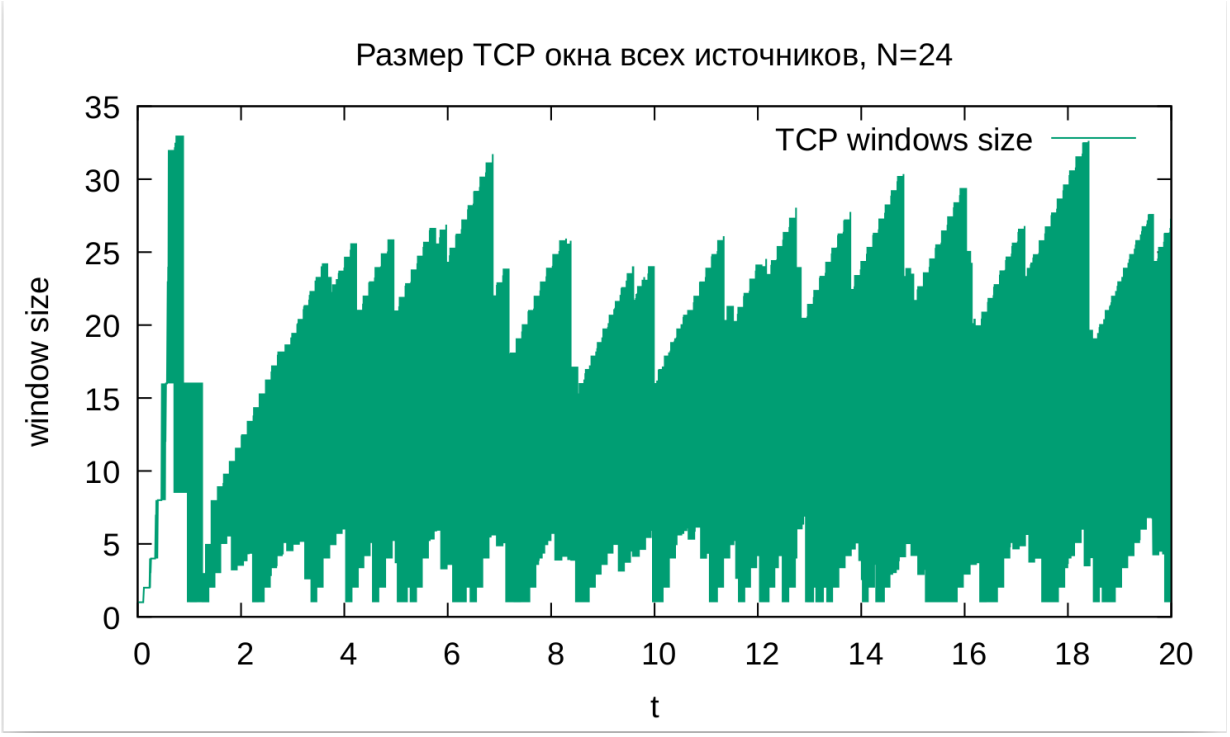

Рисунок 7. Аналогично рис. 5., только GNUplot

Основные выводы по графикам: не превосходят (ни по отдельности, ни все вместе) 32, т.к. это максимальный размер окна. В первые две секунды происходит резкий скачок, что естественно, т.к. передача данных начинается одновременно со всех источников на все приемники. После этого в среднем размер окна стабилизируется.

Графики очереди.

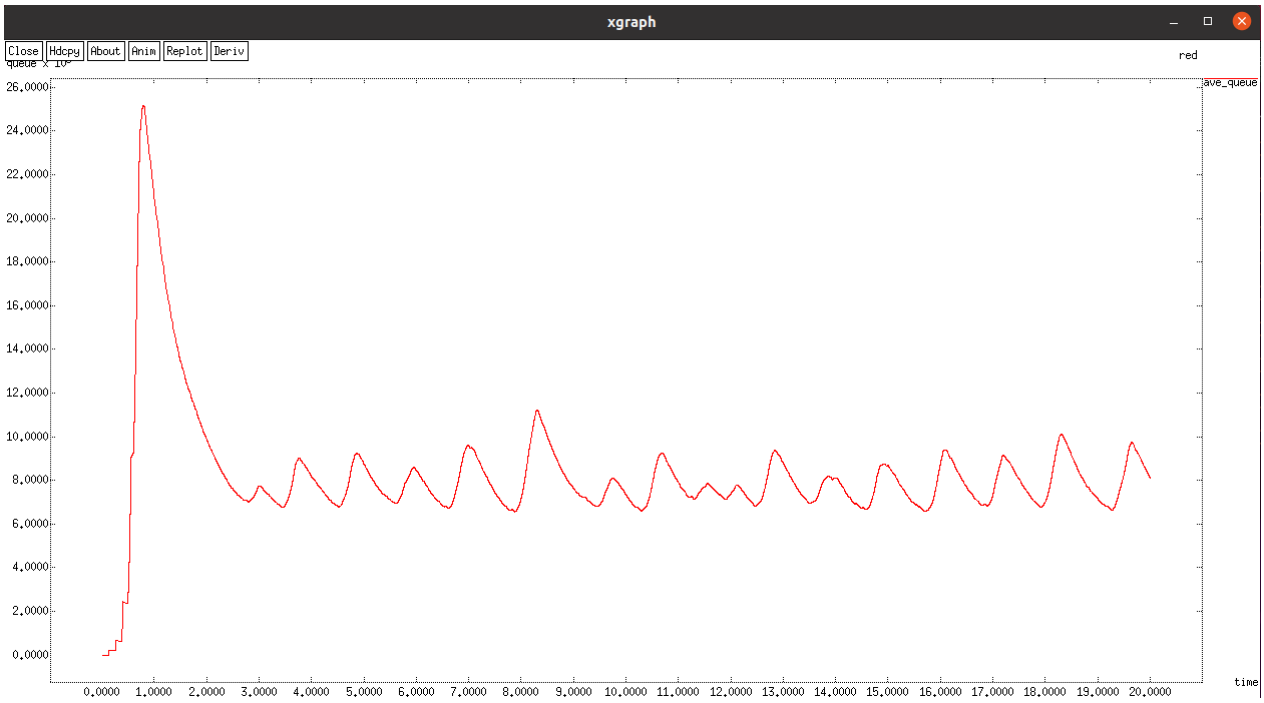

Рисунок 8. График изменения длины очереди на линке (R1-R2). Xgraph

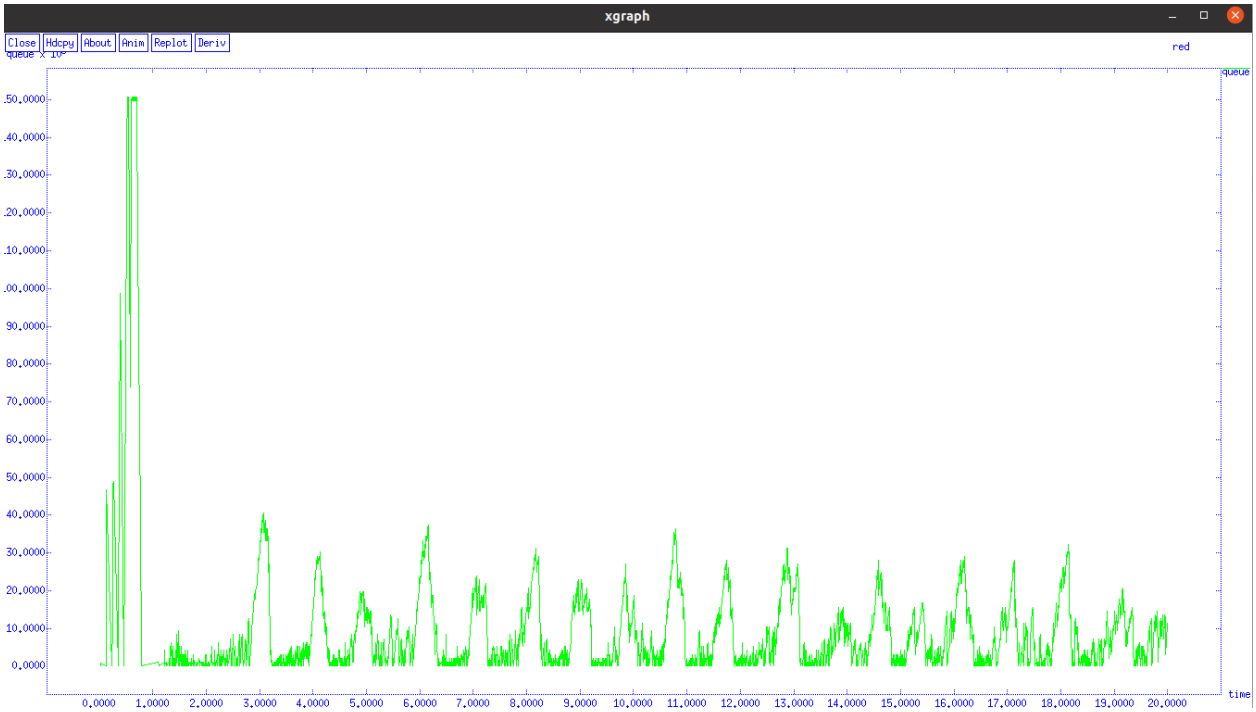

Рисунок 9. График изменения средней длины очереди на линке (R1-R2). Xgraph.

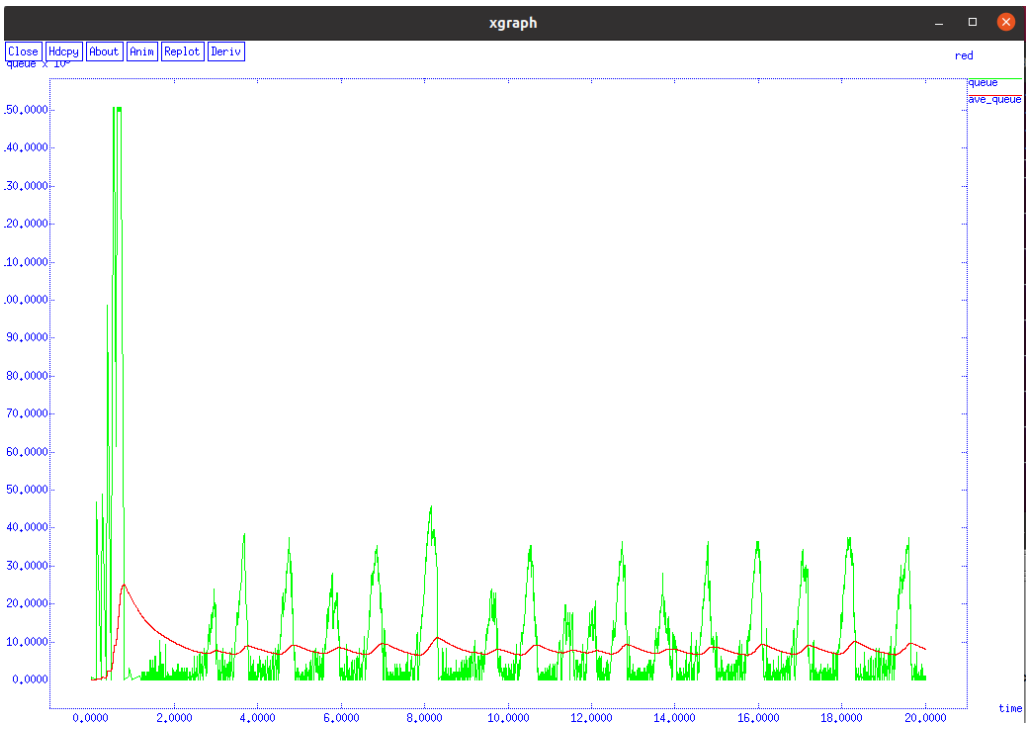

Рисунок 10. График изменения обеих длин. Xgraph

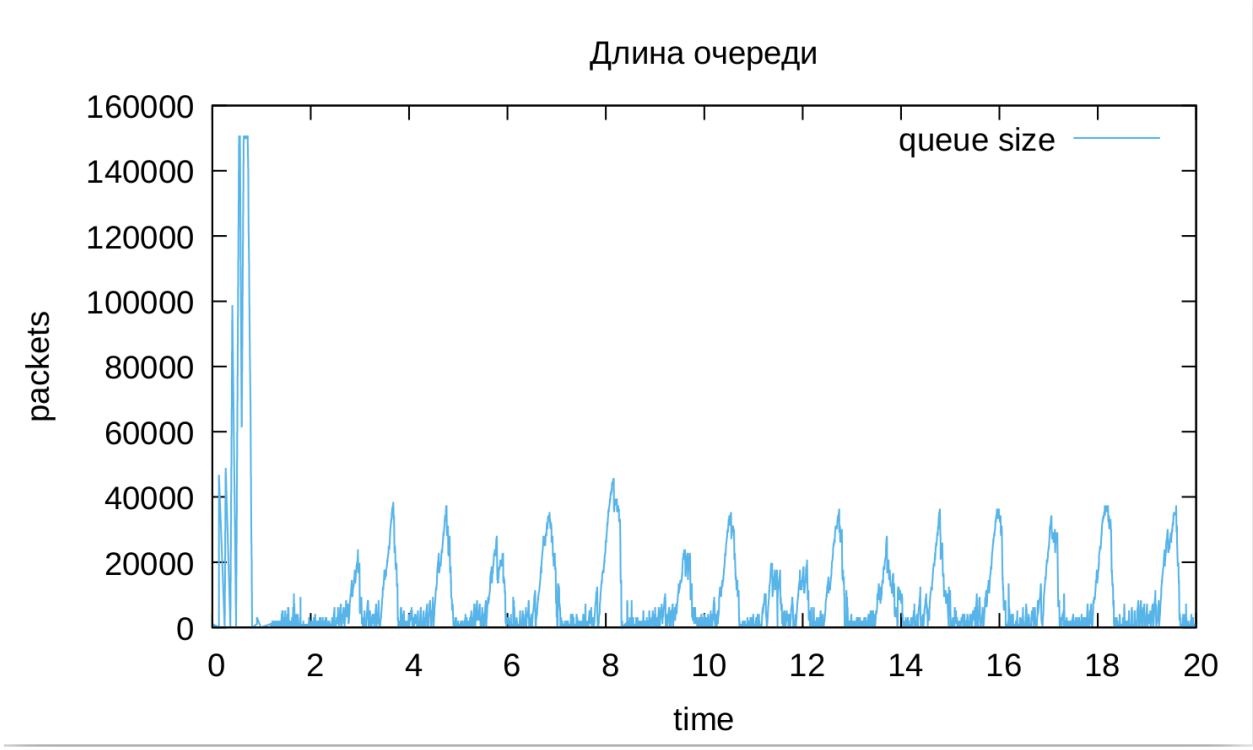

Рисунок 11. Аналогично рис. 8. только GNUplot

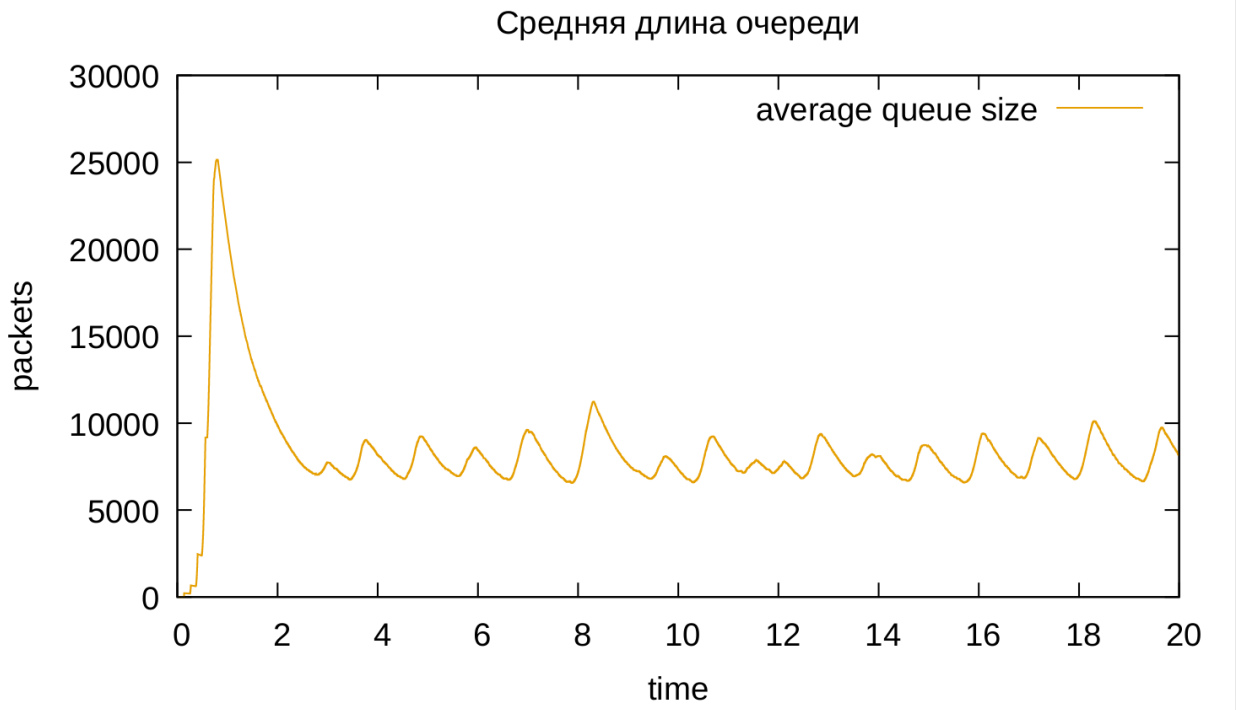

Рисунок 12. Аналогично рис. 9. только GNUplot

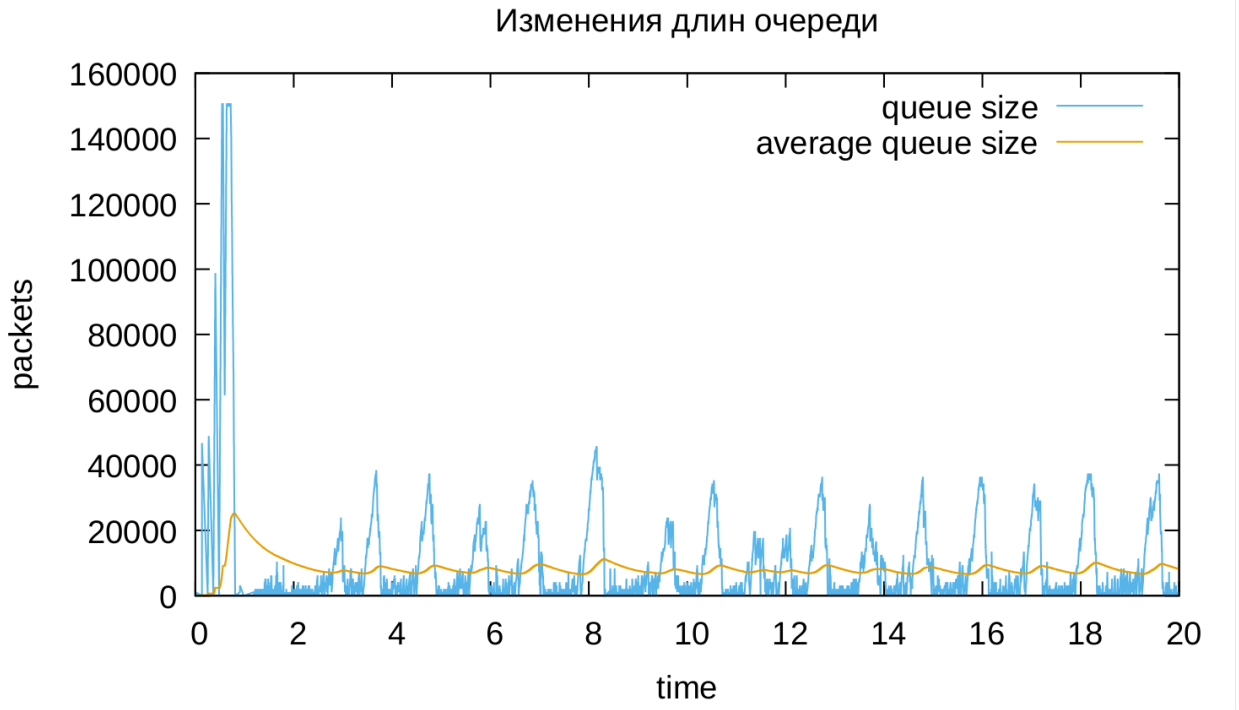

Рисунок 13. Аналогично рис. 10. только GNUplot

Немного выводов, по этим графикам: (по сути это временные ряды) видна периодичность начиная со второй секунды (примерно каждую секунду) модельного времени, для обоих длин очередей. Тренда нет, то есть длина стабильна. Виден заметный скачок длины на первых секундах, опять же это из-за того, что все источники начинают передачу данных.

Вывод

В результате выполнения лабораторной работы мы смогли построить имитационную модель сложной сети с помощью NS-2, а так же проанализировать поведение размера TCP-окна и длины очереди (RED), благодаря мониторингу соответствующих агентов и построению графиков средствами xgraph и GNUplot.

Приложение

Листинг 1. (NS-2 prog)

```
set ns [new Simulator]

# Задаем файл трассировки для нама (логи)
set nam_f [open out.nam w]
$ns namtrace-all $nam_f

# Задаем файл трассировки всего (все логи)
set all_f [open out.tr w]
$ns trace-all $all_f

# Процедура отрисовки размера 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"
}

# Процедура finish, запускающая nam и xgraph
proc finish {} {
        global ns all_f nam_f
        $ns flush-trace
        close $all_f
        close $nam_f
        # запуск nam
        exec nam out.nam &

        # запись мониторинга очереди RED, для её последующей отрисовки (взято
из 2й лр)
        global tchan_
        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";
                }
        }
```

```
        # зададим файл, в который запишем информацию для отрисовки как длины
очереди, так и средней длины
        set f [open temp.queue w]
        puts $f "TitleText: red"
        puts $f "Device: Postscript"

        # файл, только для длины очереди
        set f1 [open temp1.queue w]
        puts $f1 "TitleText: red"
        puts $f1 "Device: Postscript"

        # файл, только для средней длины очереди
        set f2 [open temp2.queue w]
        puts $f2 "TitleText: red"
        puts $f2 "Device: Postscript"

        if {    [info exists tchan_]    } {
                close $tchan_
        }

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

        # записываем соответствующую информацию в необходимые файлы
        exec awk $awkCode all.q
        puts $f \"queue
        exec cat temp.q >@ $f
        puts $f \n\"ave_queue
        exec cat temp.a >@ $f

        # Изменим так же цвет для графика
        puts $f "0.Color: green"
        puts $f "1.Color: red"
        puts $f "Foreground: blue"

        puts $f1 \"queue
        exec cat temp.q >@ $f1

        puts $f1 "0.Color: green"
        puts $f1 "Foreground: blue"

        puts $f2 \n\"ave_queue
        exec cat temp.a >@ $f2

        puts $f1 "0.Color: green"
        puts $f1 "Foreground: blue"

        close $f
        close $f1
        close $f2

        # Переходим к отрисовки, запускаем xgraph
        # сначала отрисуем размер окна для одного TCP источника
        exec xgraph -bg white -bb -tk -x time -t "TCPRenoCWND"
WindowVsTimeReno &
        # теперь для всех
        exec xgraph -bg white -bb -tk -x time -t "TCPRenoCWND"
WindowsVsTimeReno &
```

```
        # теперь отрисуем длины очереди (три графика, на одном обе длины, на
других по отдельности)
        exec xgraph -bg white -bb -tk -x time -y queue temp.queue &
        exec xgraph -bg white -bb -tk -x time -y queue temp1.queue &
        exec xgraph -bg white -bb -tk -x time -y queue temp2.queue &
        exit 0
}

# Создадим два роутера (ноды r1 и r2)
set node_(r1) [$ns node]
set node_(r2) [$ns node]
# Сделаем их квадратными (чисто для nam)
$node_(r1) shape box
$node_(r2) shape box
# Создадим между ними соотвествующее заданию соединение и ограничим очередь
размером буфера 300 пакетов
$ns simplex-link $node_(r1) $node_(r2) 20Mb 15ms RED
$ns queue-limit $node_(r1) $node_(r2) 300
$ns simplex-link $node_(r2) $node_(r1) 15Mb 20ms DropTail


# Мониторинг размера окна TCP:
set allWindowsVsTime [open WindowsVsTimeReno w]
# Еще один мониторинг файл, для отрисовки размера окна только одного
источника
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;

# Максимальный размер TCP окна
Agent/TCP set window_ 32

# Зададим 20 источников и приемников
set N 24
for {set i 1} {$i <= $N} {incr i} {
        # Задаем i-й источник (si), i=1..20
        set node_(s$i) [$ns node]
        # Покрасим его в красный цвет (для nam)
        $node_(s$i) color red
        # Соединим источник с первым роутером
        $ns duplex-link $node_(s$i) $node_(r1) 100Mb 20ms DropTail

        # Задаем i-й приемник (ti), i=1..20
        set node_(t$i) [$ns node]
        # Покрасим приемники в зеленый цвет (для nam)
        $node_(t$i) color green
        # Соединим приемник со вторым роутером
        $ns duplex-link $node_(t$i) $node_(r2) 100Mb 20ms DropTail

        # Создадим передачу данных по протоколу FTP поверх TCP/Reno от i-го
источника к i-му приемнику
        set tcp_($i) [$ns create-connection TCP/Reno $node_(s$i) TCPSink
$node_(t$i) 1]

        # Сам FTP
        set ftp_($i) [$tcp_($i) attach-source FTP]

        # Создадим at-событие, во-первых, старт передачи данных и во-вторых,
отрисовка окна
```

```
        $ns at 0.0 "$ftp_($i) start"
        $ns at 0.0 "plotWindow $tcp_($i) $allWindowsVsTime"
}

# Отдельное событие, для отрисовки графика размера окна только отдного
источника
$ns at 1.1 "plotWindow $tcp_(1) $windowVsTime"

# Задаем параметры алгоритма RED
Queue/RED set tresh_ 75
Queue/RED set maxtresh_ 150
Queue/RED set q_weight 0.002
# обратное к pmax (по умолчанию вообще 10, но на всякий случай поставим)
Queue/RED set linterm 10

# 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_

# Финиш-событие и запуск симуляции
$ns at 20 "finish"
$ns run
```

Листинг 2. (GNUplot bash script)

```
#!/usr/bin/gnuplot -persist

# задаём текстовую кодировку,
# тип терминала, тип и размер шрифта
set encoding utf8
set term pdfcairo font "Arial,14"

# задаём выходной файл графика
set out 'graphics.pdf'

# задаём название графика
set title "Размер TCP окна одного источника, N=24"

# задаём стиль линии
set style line 2

# подписи осей графика
set xlabel "t"
set ylabel "window size"

plot "WindowVsTimeReno" with lines lc 1 title "TCP window size"

#set out 'all_w.pdf'
set title "Размер TCP окна всех источников, N=24"
plot "WindowsVsTimeReno" with lines lc 2 title "TCP windows size"


set title "Длина очереди"
set xlabel "time"
set ylabel "packets"
```

```
plot "temp.q" with lines lc 3 title "queue size"


set title "Средняя длина очереди"
plot "temp.a" with lines lc 4 title "average queue size"

set title "Изменения длин очереди"
plot "temp.q" with lines lc 3 title "queue size",\
       "temp.a" with lines lc 4 title "average queue size"
```