

# **Лабораторная работа 3**

**Моделирование стохастических процессов**

Алади Принц Чисом

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>12</b>

## Список иллюстраций

3.1	Результат выполнения программы . . . . .	10
3.2	Листинг программы для отрисовки графика поведения длины очереди в пакетах . . . . .	10
3.3	Запуск программы отрисовки графика . . . . .	11
3.4	График поведения длины очереди . . . . .	11

## **Список таблиц**

# 1 Цель работы

Провести моделирование системы массового обслуживания (СМО).

## 2 Задание

1. Реализовать модель  $M|M|1$ ;
2. Посчитать загрузку системы и вероятность потери пакетов;
3. Построить график изменения размера очереди.

### 3 Выполнение лабораторной работы

$M|M|1$  – это однолинейная СМО с накопителем бесконечной ёмкости. Поступающий поток заявок — пуассоновский с интенсивностью  $\lambda$ . Времена обслуживания заявок — независимые в совокупности случайные величины, распределённые по экспоненциальному закону с параметром  $\mu$ .

Реализуем эту систему. Зададим параметры системы  $\lambda = 30$ ,  $\mu = 33$ , размер очереди 100000, длительность эксперимента 100000. Далее задаем узлы, между которыми будут идти пакеты, и соединяем их симплексным соединением с полосой пропускания 100 Кб/с и задержкой 0 мс, очередью с обслуживанием типа DropTail. Наложим ограничения на размер очереди. Источником трафика ставим UDP-агент, приемником Null-агент. Также осуществим мониторинг очереди. Процедура finish закрывает файлы трассировки. Процедура sendpack – случайно генерирует пакеты по экспоненциальному распределению. Также в данной сценарии рассчитывается по формулам загрузка система и вероятность потери пакетов.

```
# создание объекта Simulator
set ns [new Simulator]

# открытие на запись файла out.tr для регистрации событий
set tf [open out.tr w]
$ns trace-all $tf

# задаём значения параметров системы
set lambda 30.0
set mu 33.0
```

```

# размер очереди для M|M|1 (для M|M|1|R: set qsize R)
set qsize 100000

# устанавливаем длительность эксперимента
set duration 1000.0

# задаём узлы и соединяем их симплексным соединением
# с полосой пропускания 100 Кб/с и задержкой 0 мс,
# очередь с обслуживанием типа DropTail
set n1 [$ns node]
set n2 [$ns node]

set link [$ns simplex-link $n1 $n2 100kb 0ms DropTail]

# наложение ограничения на размер очереди:
$ns queue-limit $n1 $n2 $qsize

# задаём распределения интервалов времени
# поступления пакетов и размера пакетов
set InterArrivalTime [new RandomVariable/Exponential]
$InterArrivalTime set avg_ [expr 1/$lambda]
set pktSize [new RandomVariable/Exponential]
$pktSize set avg_ [expr 100000.0/(8*$mu)]

# задаём агент UDP и присоединяем его к источнику,
# задаём размер пакета
set src [new Agent/UDP]
$src set packetSize_ 100000
$ns attach-agent $n1 $src

# задаём агент-приёмник и присоединяем его
set sink [new Agent/Null]
$ns attach-agent $n2 $sink
$ns connect $src $sink

# мониторинг очереди

```



```

set qmon [$ns monitor-queue $n1 $n2 [open qm.out w] 0.1]
$link queue-sample-timeout
# процедура finish закрывает файлы трассировки
proc finish {} {
    global ns tf
    $ns flush-trace
    close $tf
    exit 0
}
# процедура случайного генерирования пакетов
proc sendpacket {} {
    global ns src InterArrivalTime pktSize
    set time [$ns now]
    $ns at [expr $time +[$InterArrivalTime value]] "sendpacket"
    set bytes [expr round ([$pktSize value])]
    $src send $bytes
}
# планировщик событий
$ns at 0.0001 "sendpacket"
$ns at $duration "finish"
# расчет загрузки системы и вероятности потери пакетов
set rho [expr $lambda/$mu]
set ploss [expr (1-$rho)*pow($rho,$qsize)/(1-pow($rho,($qsize+1)))]
puts "Теоретическая вероятность потери = $ploss"

set aveq [expr $rho*$rho/(1-$rho)]
puts "Теоретическая средняя длина очереди = $aveq"
# запуск модели
$ns run

```

Запустив эту программу, получим значения загрузки системы и вероятности потери пакетов (рис. 3.1).

```
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ ns lab3.tcl
Теоретическая вероятность потери = 0.0
Теоретическая средняя длина очереди = 9.0909090909090864
```

Рис. 3.1: Результат выполнения программы

В каталоге с проектом создадим отдельный файл, например, graph\_plot touch graph\_plot. Откроем его на редактирование и добавим следующий код, обращая внимание на синтаксис Gnuplot (рис. 3.2).

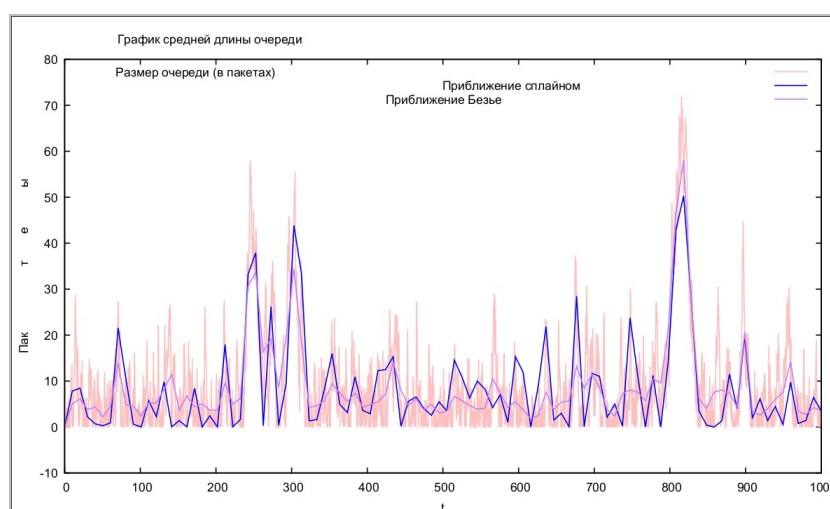


Рис. 3.2: Листинг программы для отрисовки графика поведения длины очереди в пакетах

Сделаем файл исполняемым. После компиляции файла с проектом, запустим скрипт в созданном файле graph\_plot (рис. 3.3), который создаст файл qm.png с результатами моделирования (рис. 3.4).

```
1 #!/usr/bin/gnuplot -persist
2 # задаём текстовую кодировку,
3 # тип терминала, тип и размер шрифта
4
5 set encoding utf8
6 set term pdfcairo font "Arial,9"
7
8 # задаём выходной файл графика
9 set out 'qm.pdf'
10
11 # задаём название графика
12 set title "График средней длины очереди"
13
14 # задаём стиль линии
15 set style line 2
16
17 # подписи осей графика
18 set xlabel "t"
19 set ylabel "Пакеты"
20
21 # построение графика, используя значения
22 # 1-го и 5-го столбцов файла qm.out
23 plot "qm.out" using ($1):($5) with lines lt rgb "pink" title "Размер очереди (в пакетах)", \
24      "qm.out" using ($1):($5) smooth csplines lt rgb "blue" title "Приближение сплайном ", \
25      "qm.out" using ($1):($5) smooth bezier lt rgb "purple" title "Приближение Безье "
26
```

Рис. 3.3: Запуск программы отрисовки графика

```
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ touch grapg_plot
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ chmod +x grapg_plot
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ ./grapg_plot
```

Рис. 3.4: График поведения длины очереди

На данном графике изображен размер очереди в пакетах, а также его приближение сплайном и Безье.

## **4 Выводы**

В процессе выполнения данной лабораторной работы я провел моделирование системы массового обслуживания (СМО).