

Лабораторная работа № 3

Моделирование стохастических процессов

Демидова Екатерина Алексеевна

Содержание

1	Цель работы	4
2	Задачи	5
3	Выполнение лабораторной работы	6
4	Выводы	11

Список иллюстраций

3.1	Результаты расчета информации о модели	9
3.2	Запуск скрипта отрисовки графика	10
3.3	График поведения длины очереди	10

1 Цель работы

Смоделировать систему массового обслуживания (СМО).

2 Задачи

- Реализовать модель $M|M|1$
- Посчитать теоретические вероятность потери и среднюю длину очереди
- Нарисовать график поведения длины очереди

3 Выполнение лабораторной работы

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

Реализуем этот вид СМО. Для этого зададим параметры системы, размер очереди равный 100000(относительно наших параметров такой размер можно считать бесконечным), также задаем длительность эксперимента. Пакеты будут идти между двумя узлами, которые соединены симплексным соединением(так как пакеты идут только в одну сторону) с полосой пропускания 100 Кб/с и задержкой 0 мс. Интервалы времени поступления и размера пакетов распределены по экспоненциальному закону(с разными параметрами). Источником трафика является UDP-агент, приемником Null-агент(так как нам не нужно сохранять полученные пакеты). Также осуществляется мониторинг очереди. Процедура finish закрывает файлы трассировки. Процедура sendpack – случайно генерирует пакеты по экспоненциальному распределению(используя случайные значения интервалов времени поступления и размеров пакетов). Также в программе рассчитывается по формулам загрузка система и вероятность потери пакетов. Ниже приведён листинг реализации описанной модели:

```
# 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 $tf
# задаём значения параметров системы
set lambda 30.0
set mu 33.0
# размер очереди для M|M|1 (для M|M|1|R: set qsize R)
set qsize 100000
# устанавливаем длительность эксперимента
set duration 1000.0
# задаём узлы и соединяем их симплексным соединением
# с полосой пропускания 100 Кб/с и задержкой 0 мс,
# очередью с обслуживанием типа DropTail
set n1 [$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]

```

```

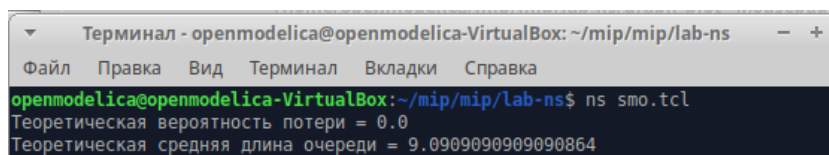
$ns attach-agent $n2 $sink
$ns connect $src $sink
# мониторинг очереди
set qmon [$ns monitor-queue $n1 $n2 [open qm.out w] 0.1]
$link queue-sample-timeout
# процедура finish закрывает файлы трассировки
proc finish {} {
    global ns tf
    $ns flush-trace
    close $tf
    exit 0
}
# процедура случайного генерирования пакетов
proc sendpacket {} {
    global ns src InterArrivalTime pktSize
    set time [$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
```

В результате получим, что вероятность потери пакета нулевая, это объясняется тем, что длина очереди у нас “бесконечна”. А теоретическая средняя длина очереди равна 9(рис. [3.1]).



```
Терминал - openmodelica@openmodelica-VirtualBox: ~/mip/mip/lab-ns  
Файл Правка Вид Терминал Вкладки Справка  
openmodelica@openmodelica-VirtualBox:~/mip/mip/lab-ns$ ns smo.tcl  
Теоретическая вероятность потери = 0.0  
Теоретическая средняя длина очереди = 9.0909090909090864
```

Рис. 3.1: Результаты расчета информации о модели

Теперь нарисуем график поведения длины очереди с помощью graph_plot. Ниже приведен листинг для отрисовки:

```
#!/usr/bin/gnuplot -persist  
# задаём текстовую кодировку,  
# тип терминала, тип и размер шрифта  
set encoding utf8  
set term pdfcairo font "Arial,9"  
# задаём выходной файл графика  
set out 'qm.pdf'  
# задаём название графика  
set title "График средней длины очереди"  
# задаём стиль линии  
set style line 2  
# подписи осей графика  
set xlabel "t"  
set ylabel "Пакеты"  
# построение графика, используя значения  
# 1-го и 5-го столбцов файла qm.out
```

```

plot "qm.out" using ($1):($5) with lines title "Размер очереди (в пакетах)",\
      "qm.out" using ($1):($5) smooth csplines title "Приближение сплайном",\
      "qm.out" using ($1):($5) smooth bezier title "Приближение Безье"

```

Сделаем этот файл исполняемым и запустим скрипт, который создаст qm.pdf с результатами моделирования(рис. [??], [3.3]).

```

openmodelica@openmodelica-VirtualBox:~/mip/mip/lab-ns$ ns smo.tcl
Теоретическая вероятность потери = 0.0
Теоретическая средняя длина очереди = 9.0909090909090864
openmodelica@openmodelica-VirtualBox:~/mip/mip/lab-ns$ chmod +x graph_plot
openmodelica@openmodelica-VirtualBox:~/mip/mip/lab-ns$ ./graph_plot
openmodelica@openmodelica-VirtualBox:~/mip/mip/lab-ns$

```

Рис. 3.2: Запуск скрипта отрисовки графика

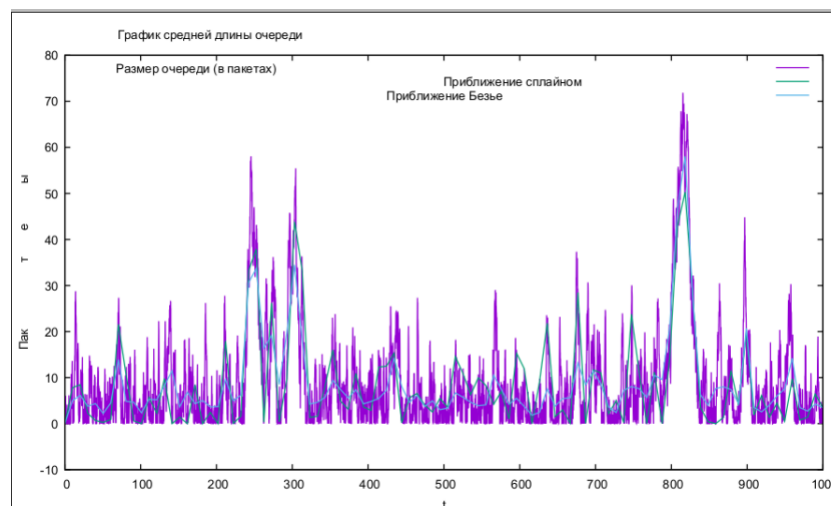


Рис. 3.3: График поведения длины очереди

На данном графике отображено поведение очереди(в пакетах) и его приближения сплайном и безье.

4 Выводы

В результате выполнения работы была смоделирована СМО вида $M|M|1$ и нарисован график поведения длины очереди в этой модели.