РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 11

дисциплина: Моделирование информационных процессов

Студент: Николаев Александр Викторович

Группа: НФИбд-01-17

МОСКВА

2020 г.

Цель работы

Реализовать модель системы массового обслуживания M|M|1 с помощью CPN Tools и осуществить мониторинг параметров, построенной модели.

Выполнение работы

Создадим два состояния: Queue (заявка в очереди) и Completed (завершенные заявки). Создадим две транзакции со сложной иерархической структурой, которые зададим на отдельных листах: Arrivals и Server. Первая служит для генерации заявки, а вторая для передачи на обработку. Соединим дугами транзакции и состояния.

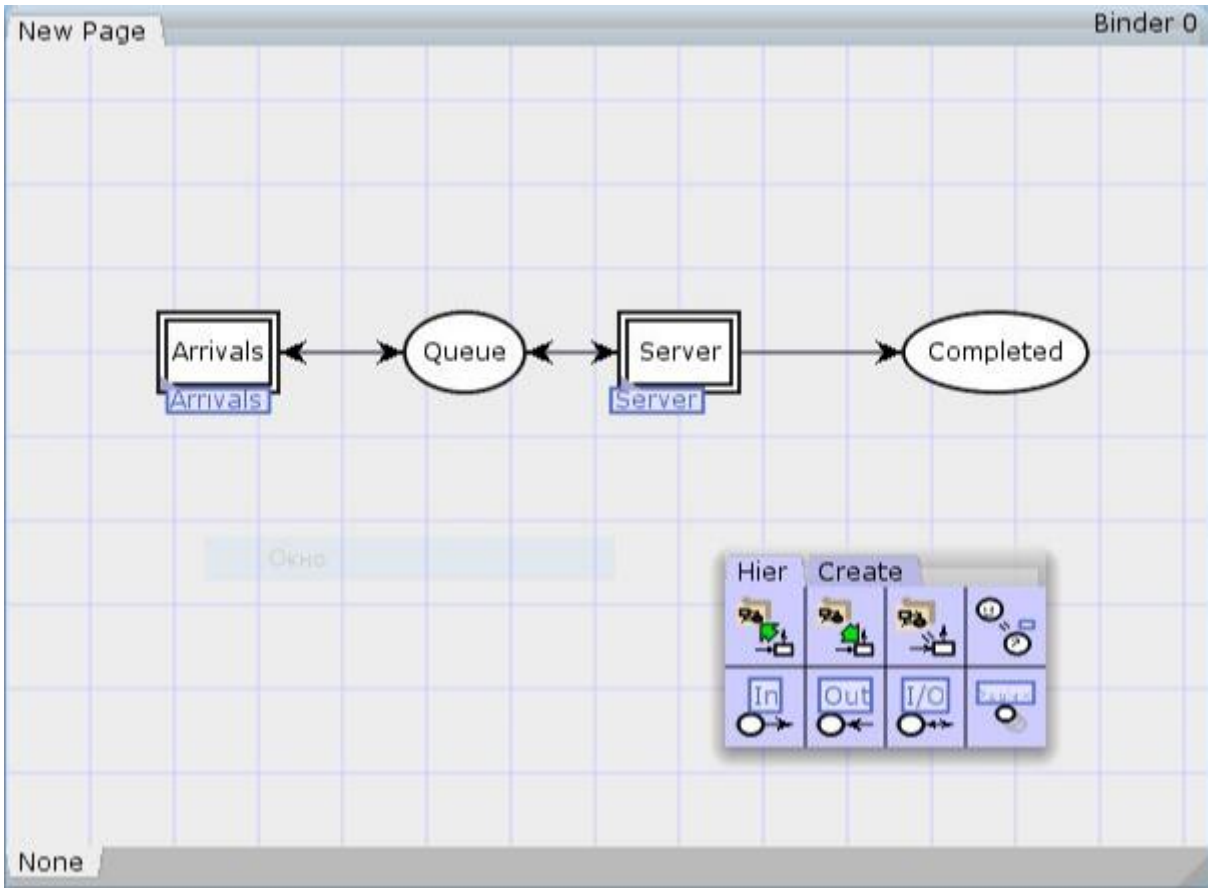


Рисунок 1. Создание системы.

Зададим декларацию.

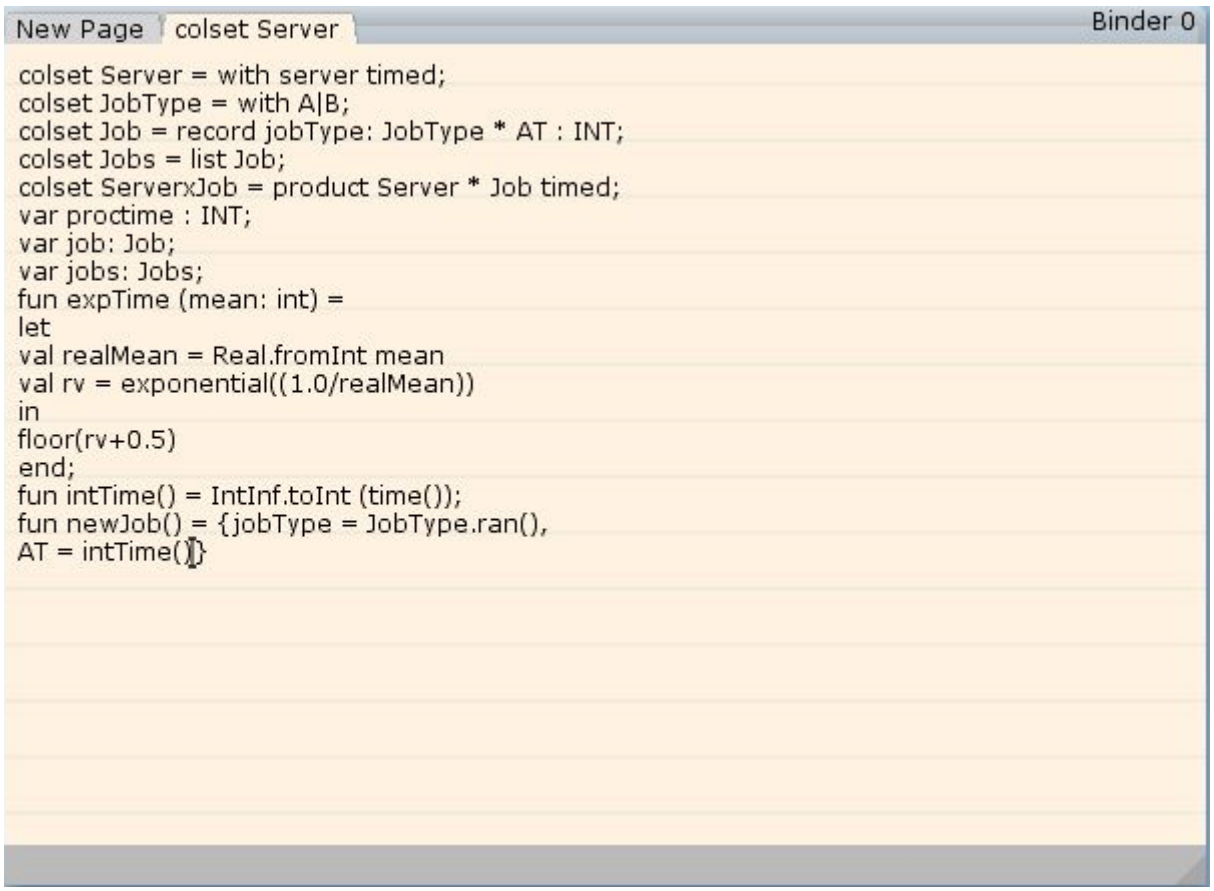


Рисунок 2. Создание деклараций.

В Arrivals создадим три состояния: Init (текущая заявка), Next (следующая) и Queue (очередь). Создадим две транзакции: Init и Arrive. Инициализируем параметры состояний. Создадим правильные связи и получим следующий результат:

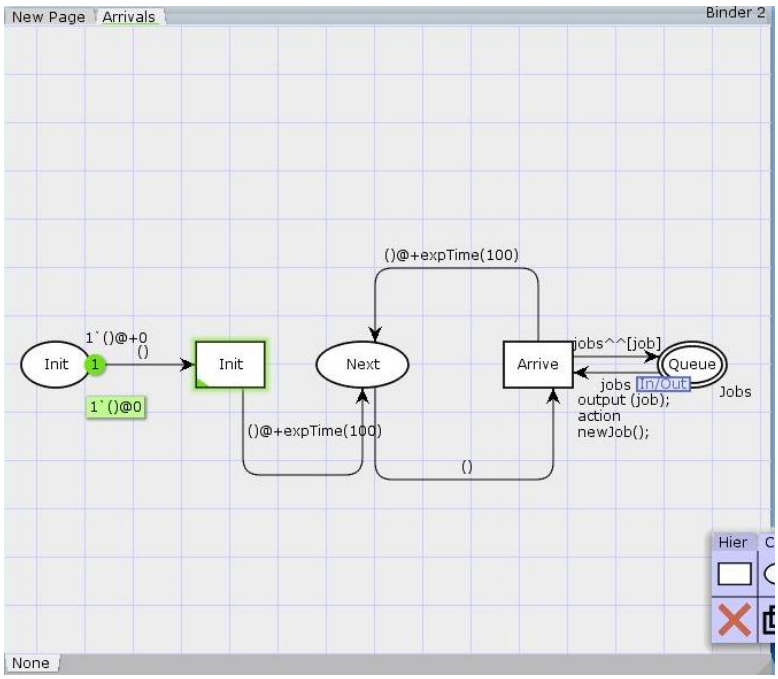


Рисунок 3. Лист Arrivals

На листе Server создадим четыре состояния: Queue, Idle, Busy и Completed. Создадим две транзакции: Start и Stop. На дуге от Queue к Start, укажем, что сервер может начинать обработку заявки, только если очередь не пуста (в обратную сторону даем фидбек). При переходе от Start к Busy будем считать время обработки заявки. При переходе Busy -> Stop сообщаем о завершении обработки заявки. При переходе Stop->Completed, считаем заявку обслуженной, а переходы с server определяют текущее состояние сервера.

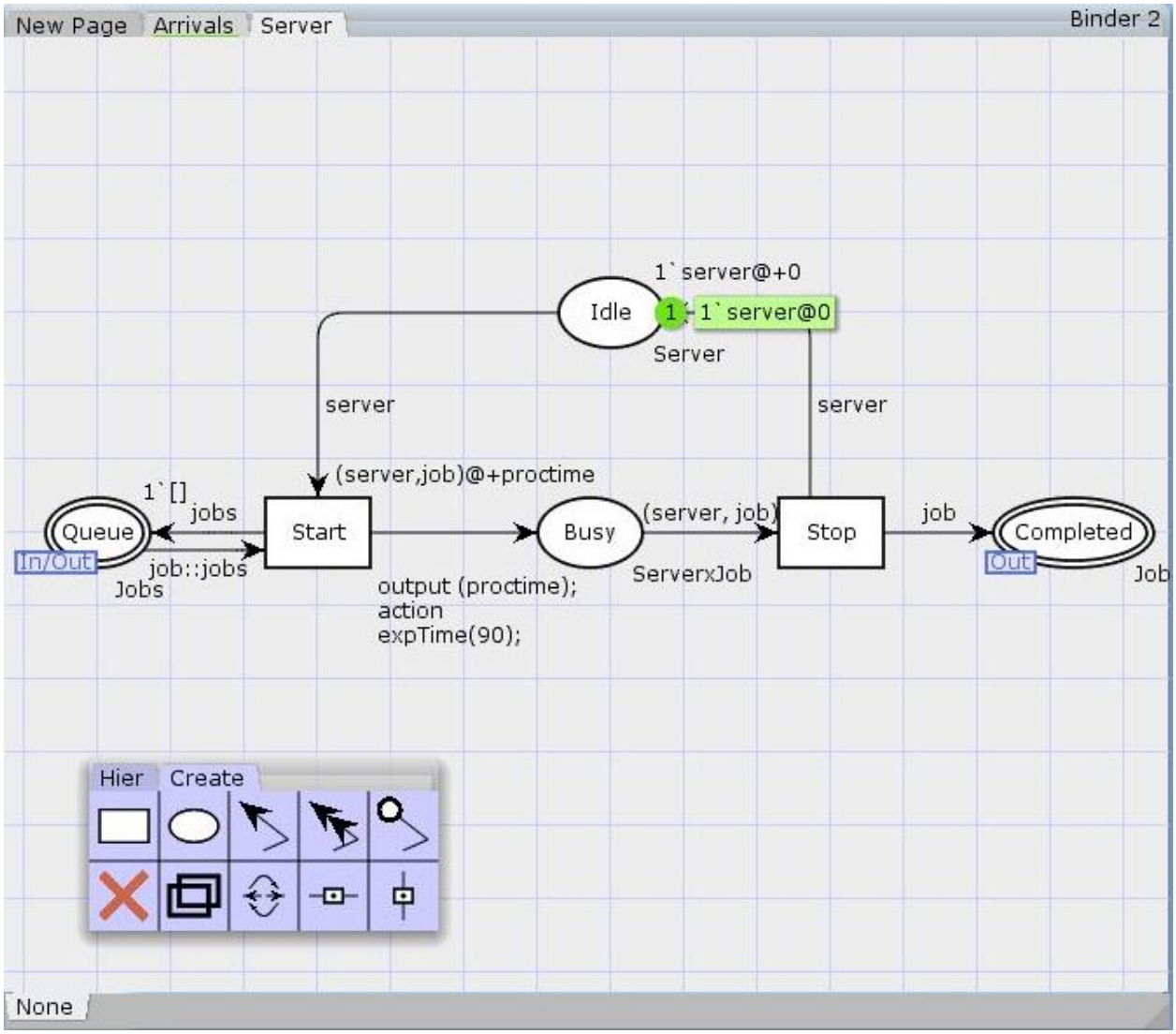


Рисунок 4. Лист Server

Получили работающую систему:

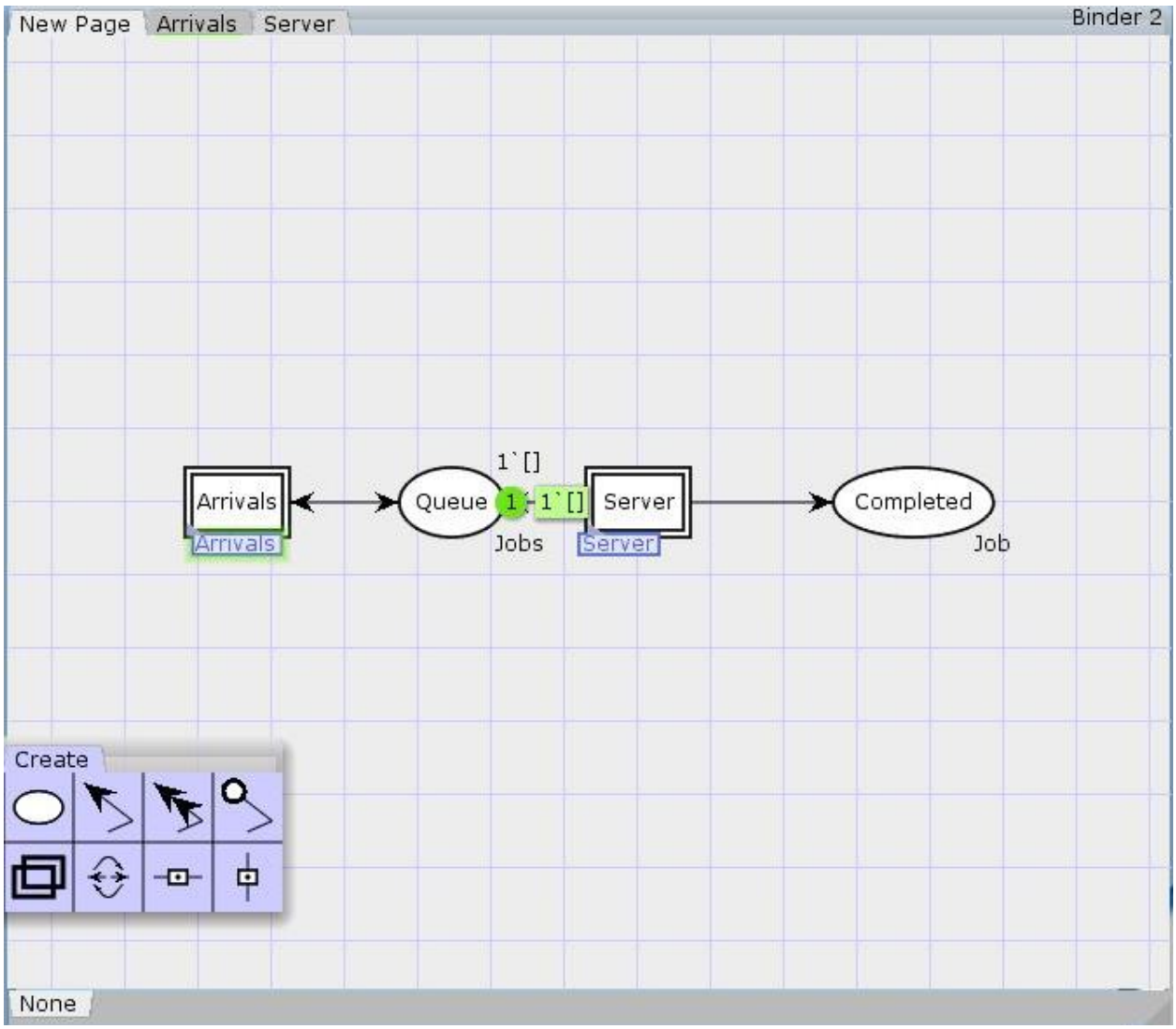


Рисунок 5. Итоговая система

Теперь настроим мониторинг параметров моделируемой системы. Для этого создадим четыре монитора: Ostanovka, Queue Delay, Queue Delay Real и Long Delay Time. Queue Delay будет фиксировать задержку, но в целых числах, в отличии от Queue Delay Real, который фиксирует её в действительных числах. Long Delay Time будет показывать преодолела ли задержка наш лимит (в нашем случае 200).

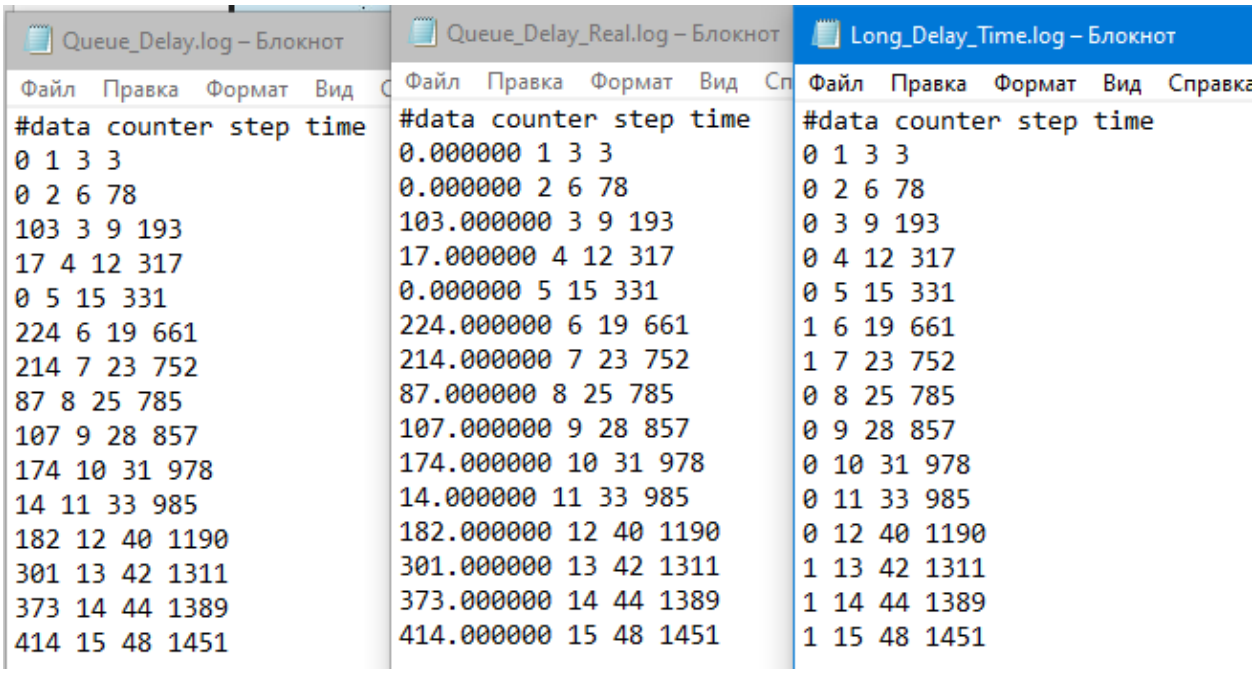
```
▼Monitors
  ▼Queue Delay
    ▶Type: Data collection
    ▶Nodes ordered by pages
    ▶Predicate
    ▼Observer
        fun obs (bindelem) =
        let
          fun obsBindElem (Server'Start (1, {job,jobs,proctime})) = (intTime()-(#AT job))
            | obsBindElem _ = ~1
        in
          obsBindElem bindelem
        end
    ▶Init function
    ▶Stop
  ▼Ostanovka
    ▶Type: Data collection
    ▶Nodes ordered by pages
    ▼Predicate
        fun pred (bindelem) =
        let
          fun predBindElem (Server'Start (1,
                                {job,jobs,proctime})) = Queue_Delay.count()=200
            | predBindElem _ = false
        in
          predBindElem bindelem
        end
    ▶Observer
    ▶Init function
```

Рисунок 6. Декларация мониторов Queue Delay и Ostanovka

```
▼Monitors
  ▶Queue Delay
  ▶Ostanovka
  ▼Queue Delay Real
    ▶Type: Data collection
    ▶Nodes ordered by pages
    ▶Predicate
    ▼Observer
        fun obs (bindelem) =
        let
          fun obsBindElem (Server'Start (1, {job,jobs,proctime})) = Real.fromInt(intTime()-(#AT job))
            | obsBindElem _ = ~1.0
        in
          obsBindElem bindelem
        end
    ▶Init function
    ▶Stop
  ▼Long Delay Time
    ▶Type: Data collection
    ▶Nodes ordered by pages
    ▶Predicate
    ▼Observer
        fun obs (bindelem) =
        if IntInf.toInt(Queue_Delay.last())>=(!longdelaytime)
        then 1
        else 0
    ▶Init function
    ▶Stop
```

Рисунок 7. Декларация мониторов Queue Delay Real и Long Delay Time

Проведем недолгую симуляцию. После завершения симуляции получаем лог файлы.



Queue_Delay.log – Блокнот

Файл Правка Формат Вид

```
#data counter step time
0 1 3 3
0 2 6 78
103 3 9 193
17 4 12 317
0 5 15 331
224 6 19 661
214 7 23 752
87 8 25 785
107 9 28 857
174 10 31 978
14 11 33 985
182 12 40 1190
301 13 42 1311
373 14 44 1389
414 15 48 1451
```

Queue_Delay_Real.log – Блокнот

Файл Правка Формат Вид Сп

```
#data counter step time
0.000000 1 3 3
0.000000 2 6 78
103.000000 3 9 193
17.000000 4 12 317
0.000000 5 15 331
224.000000 6 19 661
214.000000 7 23 752
87.000000 8 25 785
107.000000 9 28 857
174.000000 10 31 978
14.000000 11 33 985
182.000000 12 40 1190
301.000000 13 42 1311
373.000000 14 44 1389
414.000000 15 48 1451
```

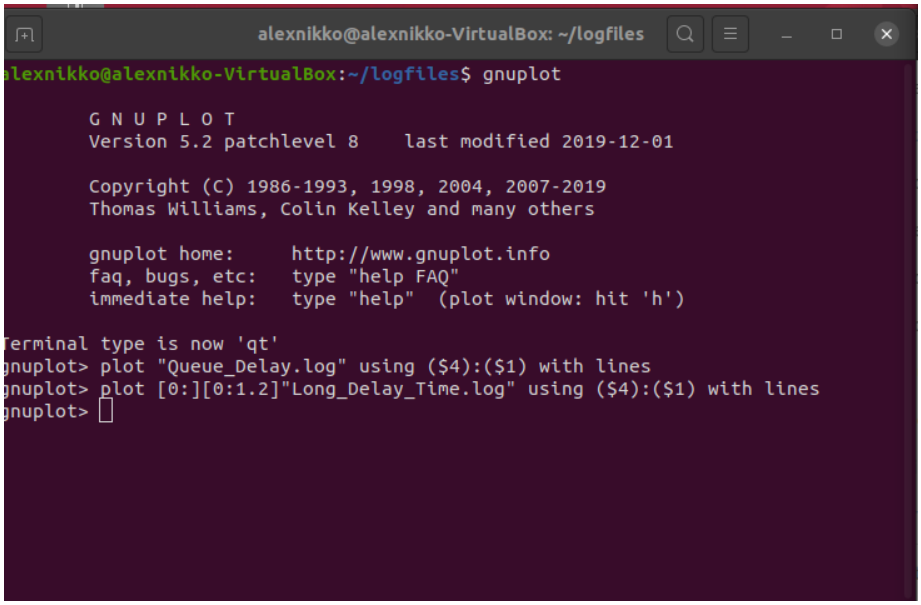
Long_Delay_Time.log – Блокнот

Файл Правка Формат Вид Справка

```
#data counter step time
0 1 3 3
0 2 6 78
0 3 9 193
0 4 12 317
0 5 15 331
1 6 19 661
1 7 23 752
0 8 25 785
0 9 28 857
0 10 31 978
0 11 33 985
0 12 40 1190
1 13 42 1311
1 14 44 1389
1 15 48 1451
```

Рисунок 8. Логи симуляции.

Теперь с помощью gnuplot построим графики изменения задержки и измерения перехода через лимит.



alexnikko@alexnikko-VirtualBox: ~/logfiles

```
alexnikko@alexnikko-VirtualBox:~/logfiles$ gnuplot

        G N U P L O T
        Version 5.2 patchlevel 8    last modified 2019-12-01

        Copyright (C) 1986-1993, 1998, 2004, 2007-2019
        Thomas Williams, Colin Kelley and many others

        gnuplot home:     http://www.gnuplot.info
        faq, bugs, etc:   type "help FAQ"
        immediate help:   type "help"  (plot window: hit 'h')

Terminal type is now 'qt'
gnuplot> plot "Queue_Delay.log" using ($4):($1) with lines
gnuplot> plot [0:][0:1.2]"Long_Delay_Time.log" using ($4):($1) with lines
gnuplot>
```

Рисунок 9. Команды для построения необходимых графиков

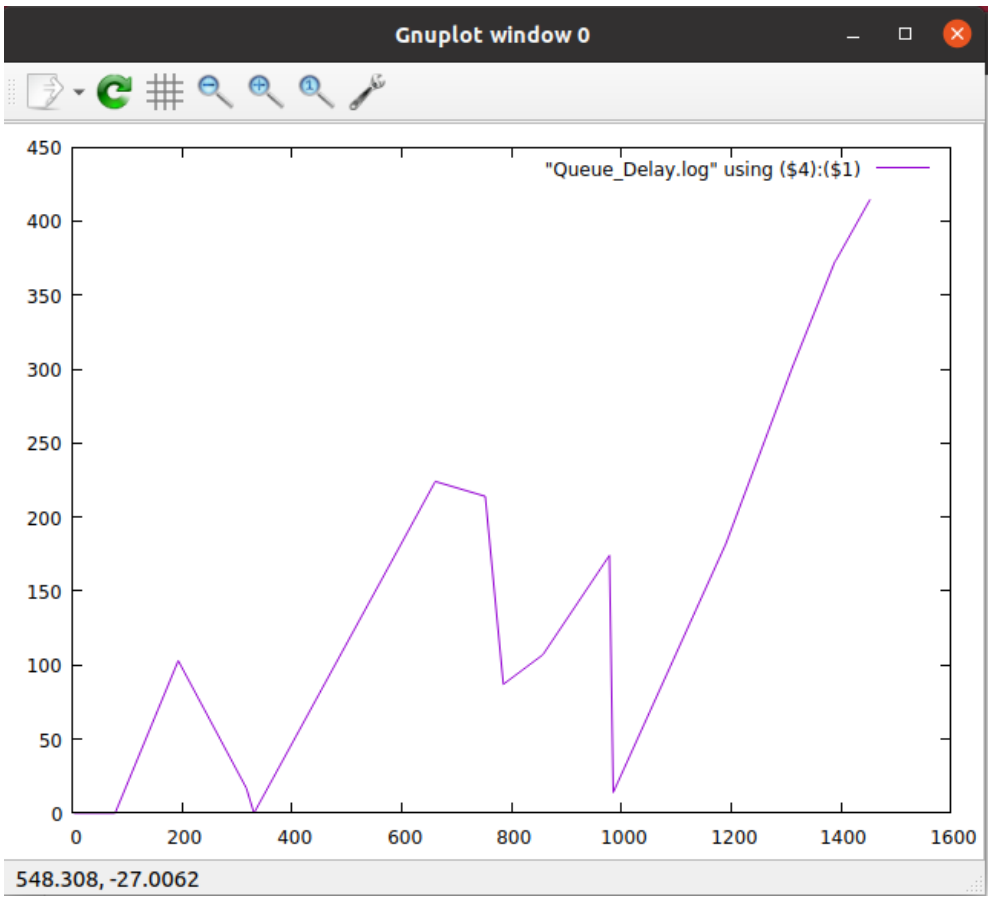


Рисунок 10. График изменения задержки в очереди.

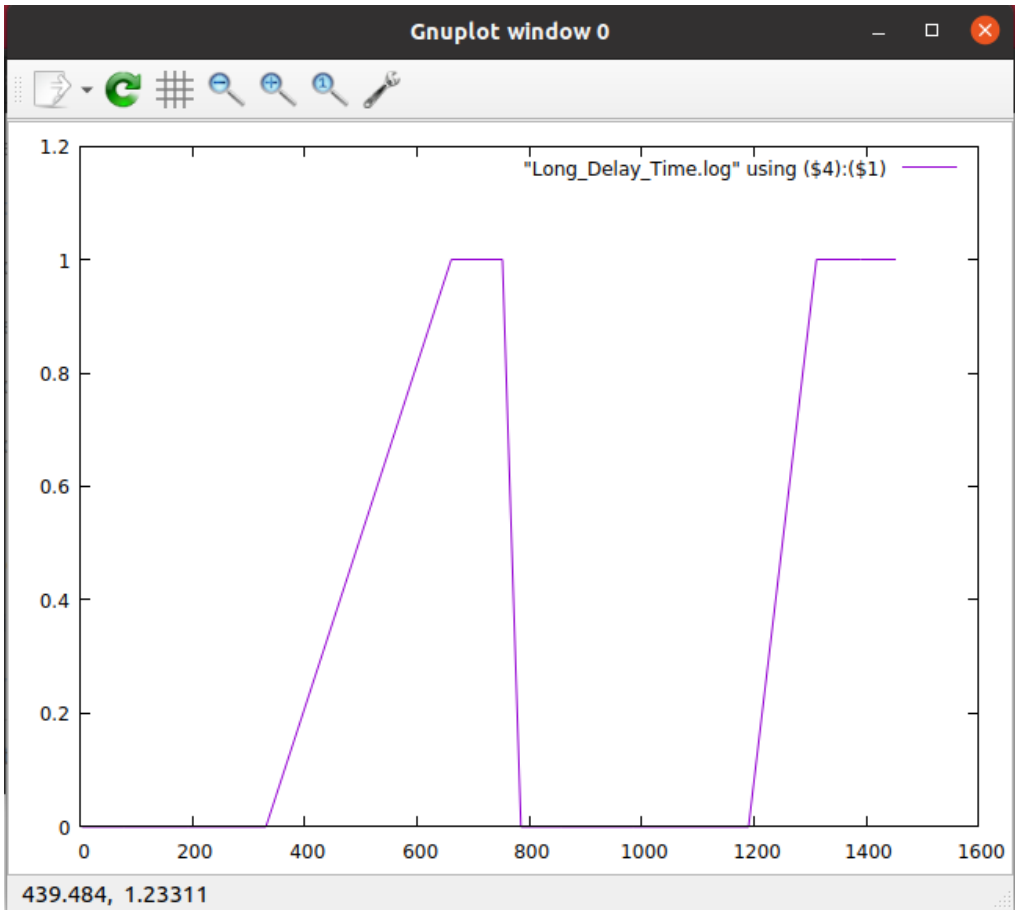


Рисунок 11. График, показывающий факт преодоления порога в 200 единиц (1 – порог преодолен, 0 – нет)

Вывод

В результате выполнения лабораторной работы построили модель системы массового обслуживания M|M|1 с помощью CPN Tools, а также научились мониторить параметры модели во время симуляции.