

Лабораторная работа 11

Мажитов М. А.

25 мая 2024

Российский университет дружбы народов, Москва, Россия

В систему поступает поток заявок двух типов, распределённый по пуассоновскому закону. Заявки поступают в очередь сервера на обработку. Дисциплина очереди - FIFO. Если сервер находится в режиме ожидания (нет заявок на сервере), то заявка поступает на обработку сервером.

1. Рисуем граф сети.

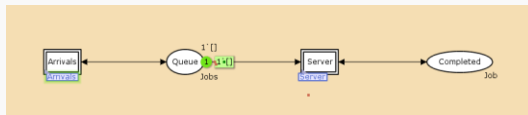


Рис. 1: Граф сети модели

Выполнение лабораторной работы

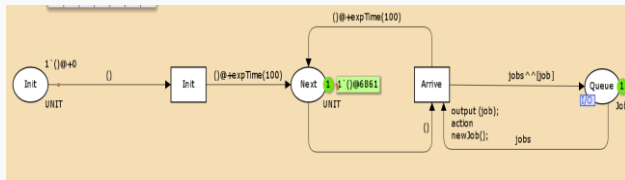


Рис. 2: Граф Arrivals

Выполнение лабораторной работы

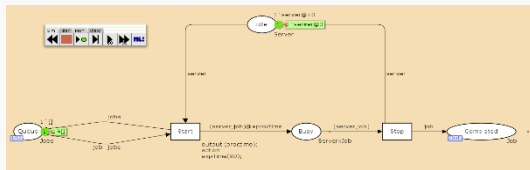



Рис. 3: Граф Sevrer

2. Зададим декларации модель.



```
▼ Declarations
  ► Standard declarations
  ▼ globref longdelaytime = 200;
  ▼ colset UNIT = unit timed;
  ▼ colset INT = int;
  ► colset Server
  ► colset JobType
  ▼ colset Job = record jobType : JobType *
    AT:INT;
  ▼ colset Jobs = list Job;
  ▼ colset ServerxJob = product Server * Job timed;
  ► var proctime
  ► var job
  ▼ var jobs: Jobs;
  ▼ fun expTime (mean: int) =
    let
      val realMean = Real.fromInt mean
      val rv = exponential ((1.0/realMean))
    in
      floor (rv+0.5)
    end;
  ► fun intTime
```

Рис. 4: Декларации модели

3. Если прокрутить моделирование, то сможешь увидеть как пакеты поступают в систему и обрабатываются.

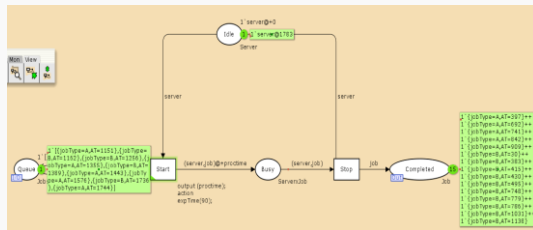


Рис. 5: Моделирование

4. Добавим мониторы. Изменим предикат, задав число шагов, через которое будем останавливать мониторинг.

```
fun pred (bindelem) =  
  let  
    fun predBindElem (Server'Start (1, {job,jobs,proctime})) = Queue_Delay.count()=200  
      | predBindElem _ = false  
  in  
    predBindElem bindelem  
  end
```

Рис. 6: Функция Predicate монитора Ostanovka

5. Добавим Data call.

```
fun obs (bindelem) =  
  let  
    fun obsBindElem (Server'Start (1, {job,jobs,proctime}))  
      = (intTime() - (#AT job))  
      | obsBindElem _ = ~1  
  in  
    obsBindElem bindelem  
  end
```

Рис. 7: Функция Observer монитора Queue Delay

Выполнение лабораторной работы

6. Запустив, мы получим log файл, при помощи которого мы можем построить график изменения задержки в очереди.

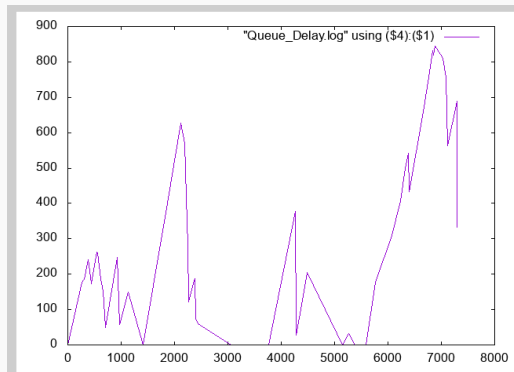


Рис. 8: График изменения задержки в очереди

7. Посчитаем задержку в действительных значениях. С помощью палитры Monitoring выбираем Data Call и устанавливаем на переходе Start. Появившийся в меню монитор называем Queue Delay Real.

```
fun obs (bindelem) =  
  let  
    fun obsBindElem (Server'Start (1, {job,jobs,proctime}))  
      = Real.fromInt(intTime() - (#AT job))  
      | obsBindElem _ = ~1  
  in  
    obsBindElem bindelem  
  end
```

Рис. 9: Функция Observer монитора Queue Delay Real

8. Запустив, мы получим log файл, при помощи которого мы можем построить график изменения задержки в очереди.

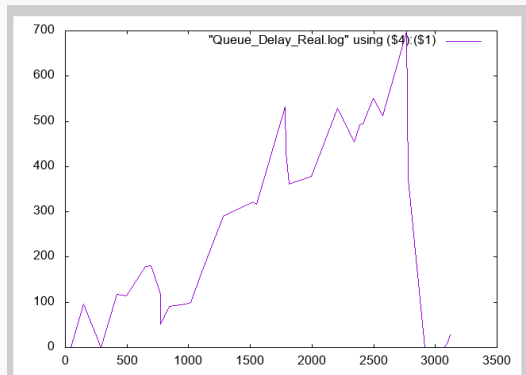


Рис. 10: График изменения задержки в очереди

9. Посчитаем, сколько раз задержка превысила заданное значение. С помощью палитры Monitoring выбираем Data Call и устанавливаем на переходе Start. Монитор называем Long Delay Time.

Выполнение лабораторной работы

10. Запустив, мы получим log файл, при помощи которого мы можем построить график изменения задержки в очереди.

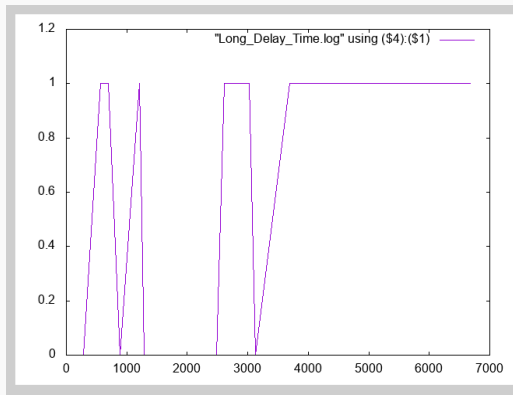


Рис. 11: График изменения задержки в очереди

Во время выполнения лабораторной работы, я провел моделирование $M|M|1$.