

Лабораторная работа № 11

Модель системы массового обслуживания М|М|1

Дворкина Ева Владимировна

Содержание

| | |
|--|-----------|
| 1 Введение | 4 |
| 1.1 Цели и задачи | 4 |
| 2 Теоретическое введение | 5 |
| 3 Выполнение лабораторной работы | 6 |
| 3.1 Постановка задачи | 6 |
| 3.2 Реализация модели системы массового обслуживания $M M 1$ в CPN Tools | 6 |
| 3.3 Мониторинг параметров моделируемой системы | 14 |
| 4 Выводы | 19 |
| Список литературы | 20 |

Список иллюстраций

| | | |
|------|---|----|
| 3.1 | Граф сети системы обработки заявок в очередь | 7 |
| 3.2 | Граф генератора заявок системы | 7 |
| 3.3 | Граф процесса обработки заявок на сервере системы | 8 |
| 3.4 | Определения множества цветов системы | 9 |
| 3.5 | Определение переменных модели | 10 |
| 3.6 | Определение функций системы | 10 |
| 3.7 | Параметры элементов основного графа системы обработки заявок в очереди | 11 |
| 3.8 | Параметры элементов генератора заявок системы | 12 |
| 3.9 | Параметры элементов обработчика заявок системы | 13 |
| 3.10 | Запуск системы обработки заявок в очереди | 13 |
| 3.11 | Функция Predicate монитора Ostanovka | 14 |
| 3.12 | Функция Observer монитора Queue Delay | 15 |
| 3.13 | Функция Observer монитора Queue Delay Real | 15 |
| 3.14 | Функция Observer монитора Long Delay Time | 16 |
| 3.15 | График изменения задержки в очереди | 17 |
| 3.16 | Периоды времени, когда значения задержки в очереди превышали заданное значение | 18 |

1 Введение

1.1 Цели и задачи

Цель работы

Реализовать в CPN Tools модель системы массового обслуживания $M|M|1$.

Задание

- Реализовать в CPN Tools модель системы массового обслуживания $M|M|1$.
- Настроить мониторинг параметров моделируемой системы и нарисовать графики очереди.

2 Теоретическое введение

CPN Tools — специальное программное средство, предназначенное для моделирования иерархических временных раскрашенных сетей Петри. Такие сети эквивалентны машине Тьюринга и составляют универсальную алгоритмическую систему, позволяющую описать произвольный объект [1].

CPN Tools позволяет визуализировать модель с помощью графа сети Петри и применить язык программирования CPN ML (Colored Petri Net Markup Language) для формализованного описания модели.

Назначение CPN Tools:

- разработка сложных объектов и моделирование процессов в различных прикладных областях, в том числе:
- моделирование производственных и бизнес-процессов;
- моделирование систем управления производственными системами и роботами;
- спецификация и верификация протоколов, оценка пропускной способности сетей и качества обслуживания, проектирование телекоммуникационных устройств и сетей.

3 Выполнение лабораторной работы

3.1 Постановка задачи

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 [2].

3.2 Реализация модели системы массового обслуживания M|M|1 в CPN Tools

Модель состоит из трех отдельных листов: на первом листе опишем граф системы (рис. 3.1); на втором — генератор заявок (рис. 3.2); на третьем — сервер обработки заявок (рис. 3.3).

Сеть имеет 2 позиции (очередь — Queue, обслуженные заявки — Complited) и два перехода (генерировать заявку — Arrivals, передать заявку на обработку серверу — Server). Переходы имеют сложную иерархическую структуру, задаваемую на отдельных листах модели (с помощью соответствующего инструмента меню — Hierarchy).

Между переходом Arrivals и позицией Queue, а также между позицией Queue и переходом Server установлена дуплексная связь. Между переходом Server и позицией Complited — односторонняя связь.

Граф генератора заявок имеет 3 позиции (текущая заявка — Init, следующая заявка — Next, очередь — Queue из листа System) и 2 перехода (Init — определяет распределение поступления заявок по экспоненциальному закону с интенсивностью 100 заявок в единицу времени, Arrive — определяет поступление заявок в очередь).

Граф процесса обработки заявок на сервере имеет 4 позиции (Busy — сервер занят, Idle — сервер в режиме ожидания, Queue и Completed из листа System) и 2 перехода (Start — начать обработку заявки, Stop — закончить обработку заявки).

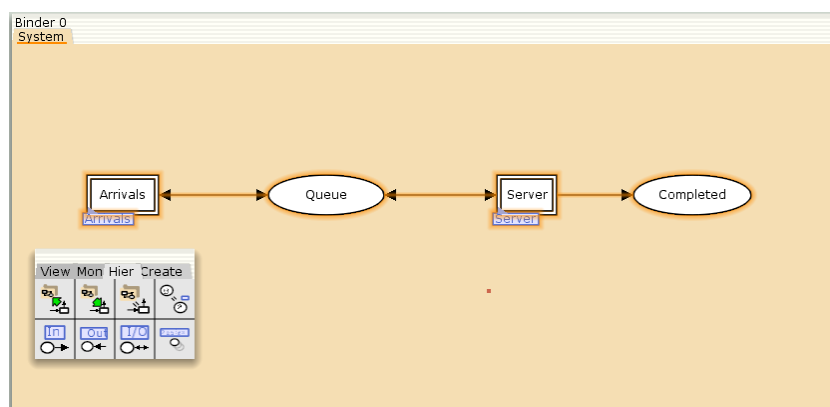


Рис. 3.1: Граф сети системы обработки заявок в очередь

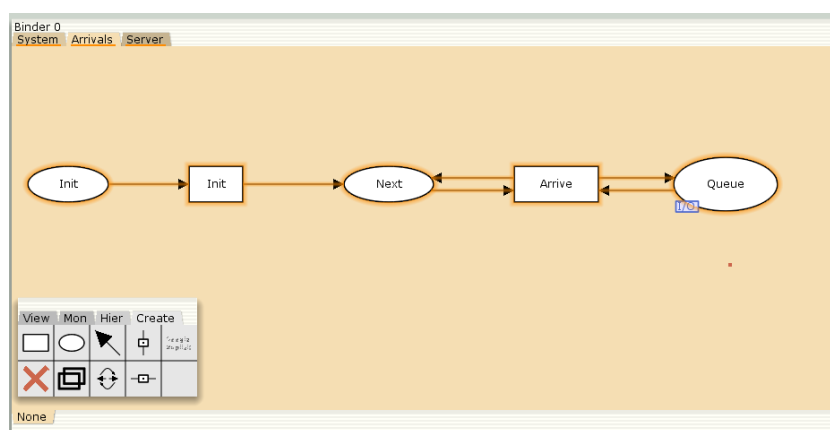


Рис. 3.2: Граф генератора заявок системы

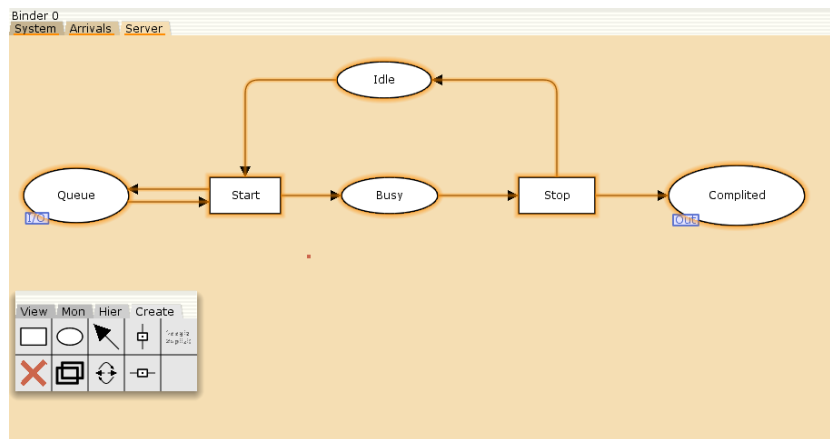


Рис. 3.3: Граф процесса обработки заявок на сервере системы

Зададим декларации системы (рис. 3.4-3.6).

Определим множества цветов системы (colorset):

- фишки типа UNIT определяют моменты времени;
- фишки типа INT определяют моменты поступления заявок в систему.
- фишки типа JobType определяют 2 типа заявок — А и В;
- кортеж Job имеет 2 поля: jobType определяет тип работы, соответственно имеет тип JobType, поле AT имеет тип INT и используется для хранения времени нахождения заявки в системе;
- фишки Jobs — список заявок;
- фишки типа ServerxJob — определяют состояние сервера, занятого обработкой заявок.

Переменные модели:

- proctime — определяет время обработки заявки;
- job — определяет тип заявки;
- jobs — определяет поступление заявок в очередь

Функции модели:

- функция `expTime` описывает генерацию целочисленных значений через интервалы времени, распределённые по экспоненциальному закону;
- функция `intTime` преобразует текущее модельное время в целое число;
- функция `newJob` возвращает значение из набора `Job` — случайный выбор типа заявки (A или B)

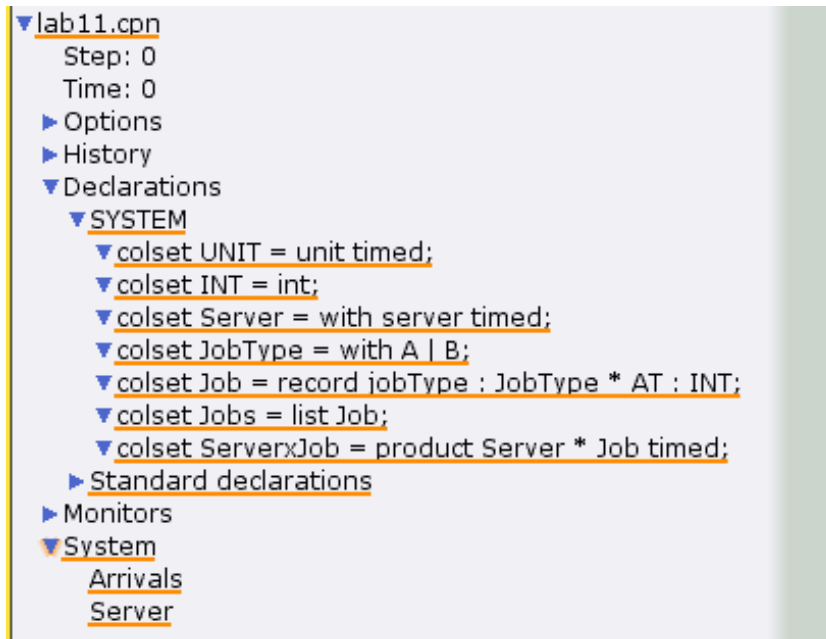


Рис. 3.4: Определения множества цветов системы

- ▶ Options
- ▼ lab11.cpn
 - Step: 0
 - Time: 0
 - ▶ Options
 - ▶ History
 - ▼ Declarations
 - ▼ SYSTEM
 - ▼ colset UNIT = unit timed;
 - ▼ colset INT = int;
 - ▼ colset Server = with server timed;
 - ▼ colset JobType = with A | B;
 - ▼ colset Job = record jobType : JobType * AT : INT;
 - ▼ colset Jobs = list Job;
 - ▼ colset ServerxJob = product Server * Job timed;
 - ▼ var proctime : INT;
 - ▼ var job : Job;
 - ▼ var jobs : Jobs;
 - ▶ Standard declarations
 - ▶ Monitors
 - ▼ System
 - Arrivals
 - Server

Рис. 3.5: Определение переменных модели

- ▼ Declarations
 - ▼ SYSTEM
 - ▼ colset UNIT = unit timed;
 - ▼ colset INT = int;
 - ▼ colset Server = with server timed;
 - ▼ colset JobType = with A | B;
 - ▼ colset Job = record jobType : JobType * AT : INT;
 - ▼ colset Jobs = list Job;
 - ▼ colset ServerxJob = product Server * Job timed;
 - ▼ var proctime : INT;
 - ▼ var job : 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 () = IntInf.toInt (time());
 - ▼ fun newJob () = {jobType = JobType.ran(), AT = intTime()};

Рис. 3.6: Определение функций системы

Зададим параметры модели на графах сети.

На листе System (рис. 3.7):

- у позиции Queue множество цветов фишек — Jobs; начальная маркировка $1'[]$ определяет, что изначально очередь пуста.
- у позиции Completed множество цветов фишек — Job.

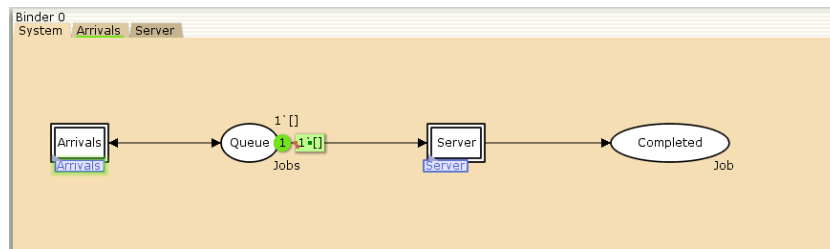


Рис. 3.7: Параметры элементов основного графа системы обработки заявок в очереди

На листе Arrivals (рис. 3.8):

- у позиции Init: множество цветов фишек — UNIT; начальная маркировка $1'() [0?]$ определяет, что поступление заявок в систему начинается с нулевого момента времени;
- у позиции Next: множество цветов фишек — UNIT;
- на дуге от позиции Init к переходу Init выражение $()$ задаёт генерацию заявок;
- на дуге от переходов Init и Arrive к позиции Next выражение $()@+expTime(100)$ задаёт экспоненциальное распределение времени между поступлениями заявок;
- на дуге от позиции Next к переходу Arrive выражение $()$ задаёт перемещение фишки;

- на дуге от перехода Arrive к позиции Queue выражение $jobs^{1^1}$ задаёт поступление заявки в очередь;
- на дуге от позиции Queue к переходу Arrive выражение jobs задаёт обратную связь.

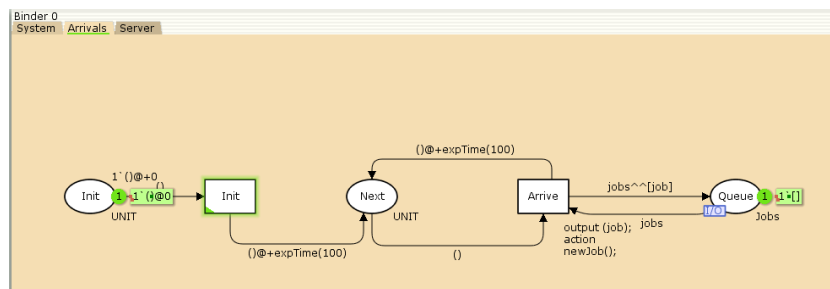


Рис. 3.8: Параметры элементов генератора заявок системы

На листе Server (рис. 3.9):

- у позиции Busy: множество цветов фишек — Server, начальное значение маркировки — $1'server@0$ определяет, что изначально на сервере нет заявок на обслуживание;
- у позиции Idle: множество цветов фишек — ServerxJob;
- переход Start имеет сегмент кода `output(proctime); action expTime(90);` определяющий, что время обслуживания заявки распределено по экспоненциальному закону со средним временем обработки в 90 единиц времени;
- на дуге от позиции Queue к переходу Start выражение `job::jobs` определяет, что сервер может начать обработку заявки, если в очереди есть хотя бы одна заявка;
- на дуге от перехода Start к позиции Busy выражение `(server,job)@+proctime` запускает функцию расчёта времени обработки заявки на сервере;

¹job

- на дуге от позиции Busy к переходу Stop выражение (server,job) говорит о завершении обработки заявки на сервере;
- на дуге от перехода Stop к позиции Completed выражение job показывает, что заявка считается обслуженной;
- выражение server на дугах от и к позиции Idle определяет изменение состояние сервера (обрабатывает заявки или ожидает);
- на дуге от перехода Start к позиции Queue выражение jobs задаёт обратную связь.

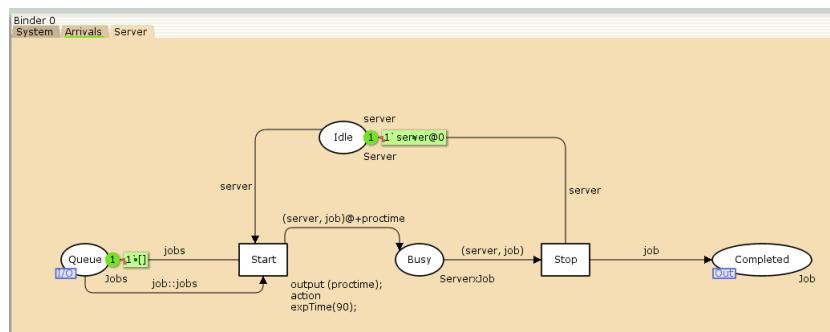


Рис. 3.9: Параметры элементов обработчика заявок системы

После добавления всех параметров система начинает работать (рис. 3.10)

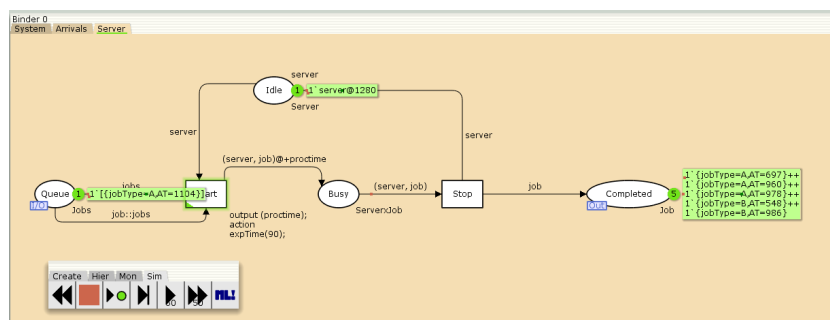


Рис. 3.10: Запуск системы обработки заявок в очереди

3.3 Мониторинг параметров моделируемой системы

Потребуется палитра Monitoring. Выбираем Break Point (точка останова) и устанавливаем её на переход Start. После этого в разделе меню Monitor появится новый подраздел, который назовём Ostanovka. В этом подразделе необходимо внести изменения в функцию Predicate, которая будет выполняться при запуске монитора. Зададим число шагов, через которое будем останавливать мониторинг. Для этого true заменим на `Queue_Delay.count()=200` (рис. 3.11).

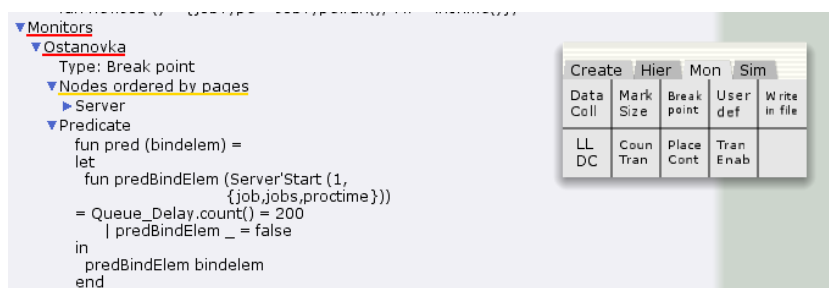


Рис. 3.11: Функция Predicate монитора Ostanovka

Необходимо определить конструкцию `Queue_Delay.count()`. С помощью палитры Monitoring выбираем Data Call и устанавливаем на переходе Start. Появившийся в меню монитор называем Queue Delay (без подчеркивания). Функция Observer выполняется тогда, когда функция предикатора выдаёт значение true. По умолчанию функция выдаёт 0 или унарный минус (~1), подчёркивание обозначает произвольный аргумент. Изменим её так, чтобы получить значение задержки в очереди. Для этого необходимо из текущего времени `intTime()` вычесть временную метку AT, означающую приход заявки в очередь (рис. 3.12)..

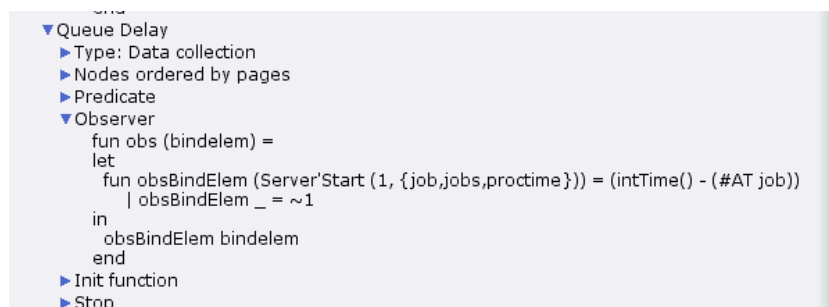


Рис. 3.12: Функция Observer монитора Queue Delay

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. Функцию Observer изменим следующим образом (рис. 3.13):

По сравнению с предыдущим описанием функции добавлено преобразование значения функции из целого в действительное, при этом obsBindElem _ принимает значение ~1.0. После запуска программы на выполнение в каталоге с кодом программы появится файл Queue_Delay_Real.log с содержимым, аналогичным содержимому файла Queue_Delay.log, но значения задержки имеют действительный тип.

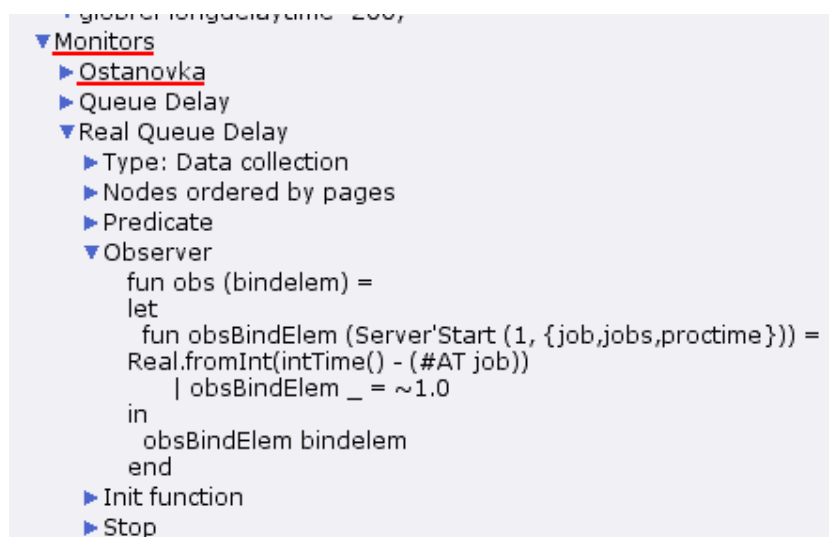


Рис. 3.13: Функция Observer монитора Queue Delay Real

Посчитаем, сколько раз задержка превысила заданное значение. С помощью па-

литры Monitoring выбираем Data Call и устанавливаем на переходе Start. Монитор называем Long Delay Time. Функцию Observer изменим следующим образом(рис. 3.14):

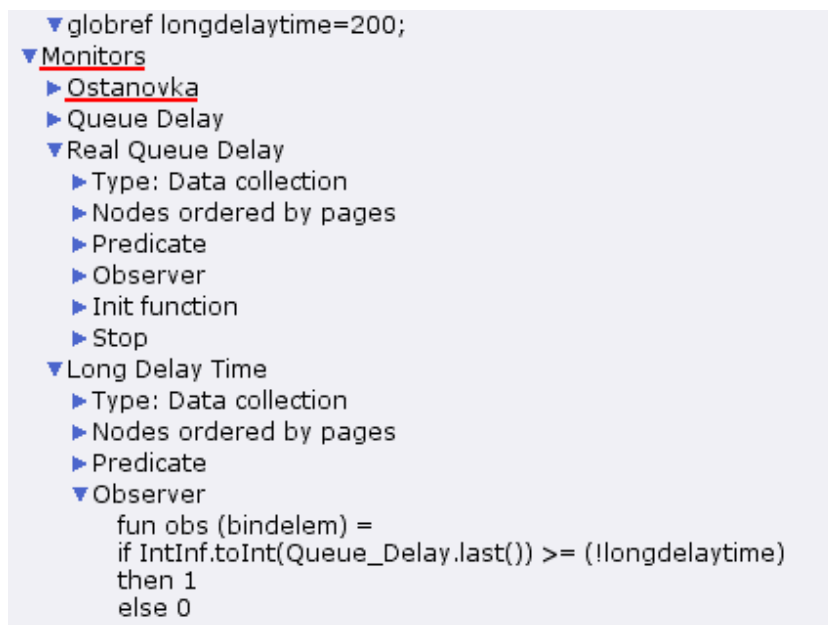


Рис. 3.14: Функция Observer монитора Long Delay Time

После запуска программы на выполнение в каталоге с кодом программы появится файл Queue_Delay.log, содержащий в первой колонке — значение задержки очереди, во второй — счётчик, в третьей — шаг, в четвёртой — время. С помощью gnuplot можно построить график значений задержки в очереди (рис. 3.15), выбрав по оси x время, а по оси y — значения задержки:

```

#!/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 стиль линии
  
```



```
set style line 2
plot "Queue_Delay.log" using ($4):($1) with lines
```

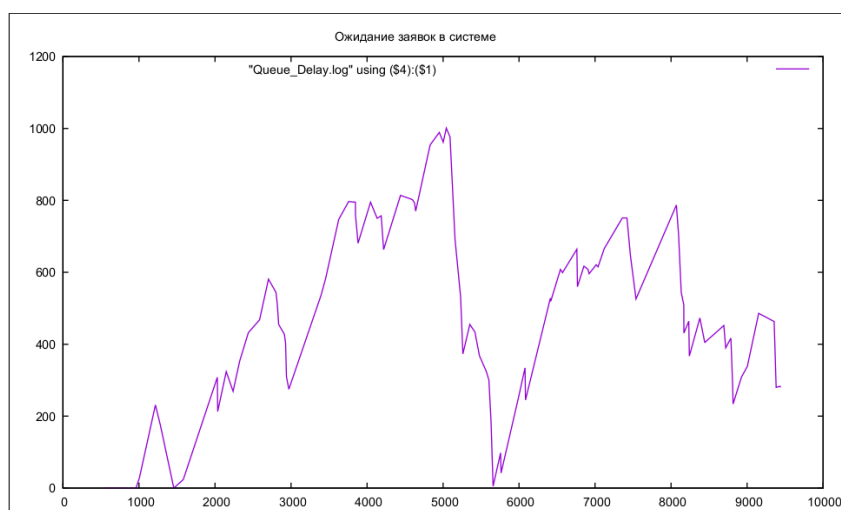


Рис. 3.15: График изменения задержки в очереди

С помощью gnuplot можно построить график (рис. 3.16), демонстрирующий, в какие периоды времени значения задержки в очереди превышали заданное значение 200.

```
#!/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 стиль линии
set style line 2
plot [0:] [0:1.2] "Long_Delay_Time.log" using ($4):($1) with lines
```



Рис. 3.16: Периоды времени, когда значения задержки в очереди превышали заданное значение

4 Выводы

В результате выполнения работы была реализована в CPN Tools модель системы массового обслуживания $M|M|1$.

Список литературы

1. Королькова А.В., Кулябов Д.С. Сети Петри. Моделирование в CPN Tools [Электронный ресурс].
2. Королькова А.В., Кулябов Д.С. Лабораторная работа 11. Модель системы массового обслуживания $M|M|1$ [Электронный ресурс].