Лабораторная работа № 11

Модель системы массового обслуживания M|M|1

Мугари Абдеррахим

Содержание

Список иллюстраций

0.1 Цель и задачи

0.1.1 Цель работы

Создание модели системы массового обслуживания M|M|1 с использованием среды моделирования CPN Tools.

0.1.2 Задание

- Построить модель M|M|1 в CPN Tools.
- Настроить мониторинг ключевых параметров работы системы.
- Визуализировать графики изменения длины очереди во времени.

0.2 Теоретическое введение

CPN Tools представляет собой специализированную программную среду для построения и анализа иерархических временных раскрашенных сетей Петри. Такие сети обладают вычислительной мощностью, аналогичной машине Тьюринга, что позволяет использовать их для моделирования любых алгоритмически представимых процессов [1].

Одной из сильных сторон CPN Tools является возможность графической визуализации моделей на основе сетей Петри, а также поддержка формального описания с использованием языка CPN ML (Colored Petri Net Markup Language).

0.2.1 Области применения CPN Tools:

- моделирование сложных объектов и процессов;
- анализ производственных и бизнес-процессов;
- проектирование и исследование систем управления (включая промышленные роботы);
- формализация и проверка сетевых протоколов;
- оценка параметров телекоммуникационных систем, включая пропускную способность и качество обслуживания.

Эти возможности делают CPN Tools мощным инструментом для решения задач в инженерии, бизнесе и телекоммуникациях.

1 Выполнение лабораторной работы

1.1 Постановка задачи

Необходимо смоделировать систему, в которую поступают заявки двух типов. Поток заявок подчиняется пуассоновскому распределению. Все заявки направляются в очередь на обработку, где применяется дисциплина обслуживания FIFO (первым пришёл — первым обслужен). Если сервер свободен, он немедленно начинает обработку поступившей заявки [2].

1.2 Реализация модели M|M|1 в CPN Tools

Модель реализована в среде CPN Tools и разделена на три отдельных листа:

1. **System** — основная схема модели (см. рисунок 1.1);
2. **Generator** — схема генерации заявок (см. рисунок 1.2);
3. **Server** — схема обработки заявок (см. рисунок 1.3).

1.2.1 Основной граф модели (лист System)

На листе описывается структура системы:

- **Позиции:**

- `Queue` — очередь заявок;
- `Complited` — завершённые заявки.

- **Переходы:**
  - `Arrivals` — генерация новых заявок;
  - `Server` — передача заявки на обработку.

Оба перехода имеют иерархическую структуру, которую можно настроить с помощью инструмента *Hierarchy*.
Между `Arrivals` и `Queue`, а также между `Queue` и `Server` — двусторонняя связь. Между `Server` и `Complited` — односторонняя.

1.2.2 Граф генерации заявок (лист Generator)

Здесь реализована логика поступления заявок:

- **Позиции:**
  - `Init` — текущая заявка;
  - `Next` — следующая заявка;
  - `Queue` — ссылка на позицию очереди из листа *System*.

- **Переходы:**
  - `Init` — моделирует поступление заявок с экспоненциальным распределением (интенсивность: 100 заявок в единицу времени);
  - `Arrive` — подача заявки в очередь.

1.2.3 Граф обработки заявок (лист Server)

Этот лист описывает поведение сервера:

- **Позиции:**
  - `Busy` — сервер в работе;
  - `Idle` — сервер свободен;
  - `Queue` и `Complited` — позиции из листа *System*.
- **Переходы:**
  - `Start` — начало обработки заявки;
  - `Stop` — завершение обработки.

1.2.4 Иллюстрации

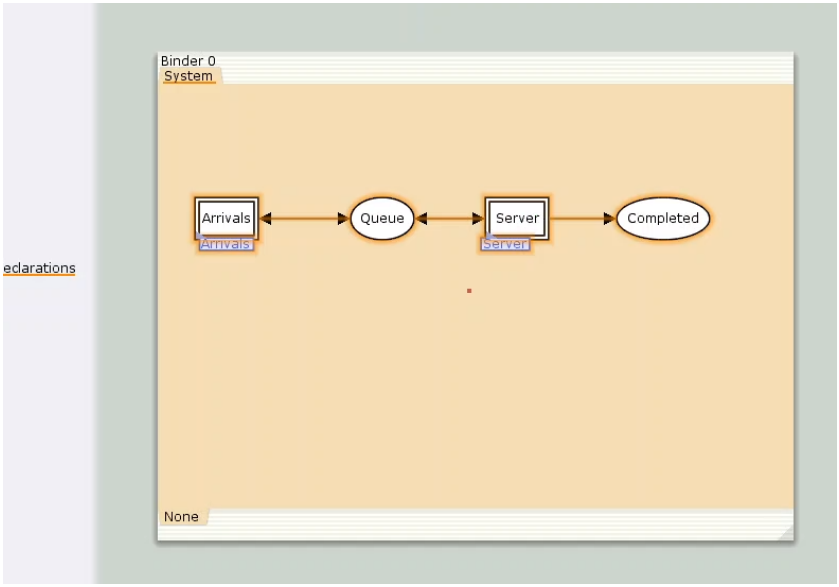

Рис. 1.1: Граф сети системы обработки заявок в очередь

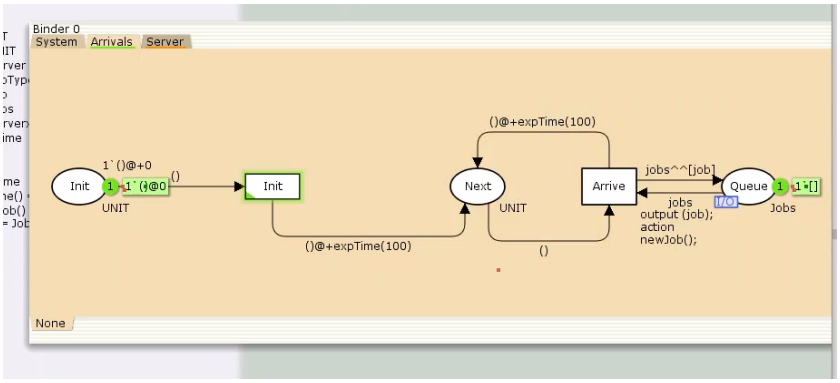

Рис. 1.2: Граф генератора заявок системы

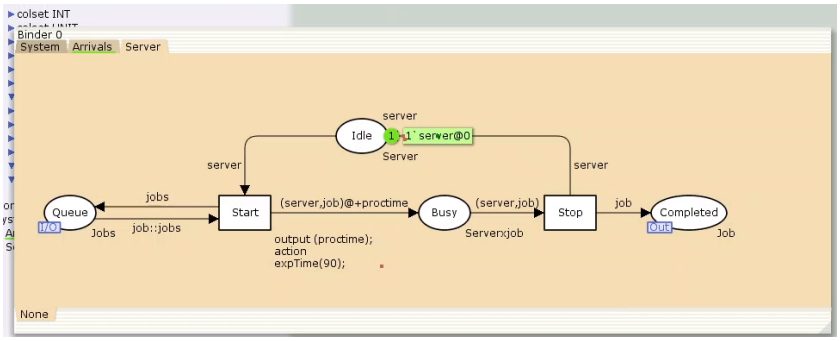

Рис. 1.3: Граф процесса обработки заявок на сервере системы

В дальнейшем необходимо определить декларации для каждой части модели (см. рисунки 1.4-1.6).

1.3 Декларации модели

1.3.1 Множества цветов (colorset)

В модели определены следующие типы данных (множества цветов), используемые для представления различных сущностей в системе:

- **UNIT** — используется для фиксации моментов времени.
- **INT** — обозначает моменты поступления заявок в систему.
- **JobType** — определяет два возможных типа заявок: `A` и `B`.

- **Job** — кортеж, состоящий из двух полей:
  - `jobType` (тип `JobType`) — обозначает тип заявки;
  - `AT` (тип `INT`) — хранит время, в течение которого заявка находится в системе.
- **Jobs** — список заявок (тип — список объектов `Job`).
- **ServerxJob** — состояние сервера, когда он занят конкретной заявкой.

1.3.2 Переменные модели

Модель использует следующие переменные:

- `proctime` — время, затрачиваемое на обработку одной заявки;
- `job` — отдельная заявка;
- `jobs` — список заявок, поступивших в очередь.

1.3.3 Встроенные функции модели

Для корректной генерации и обработки заявок определены следующие функции:

- `expTime()` — возвращает случайные значения, моделирующие интервалы времени между поступлениями заявок на основе экспоненциального распределения;
- `intTime()` — преобразует текущее модельное время в целое число;
- `newJob()` — создает новую заявку (`Job`), случайным образом выбирая тип (`A` или `B`).

1.3.4 Иллюстрации

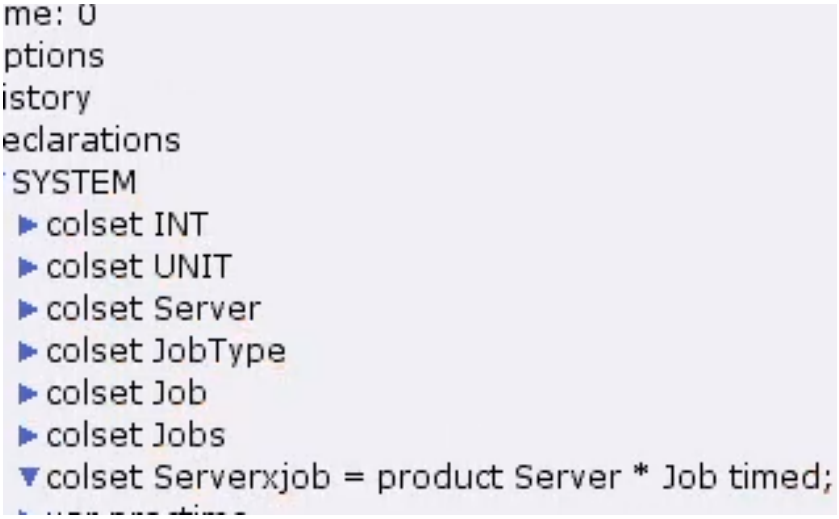

Рис. 1.4: Определения множества цветов системы

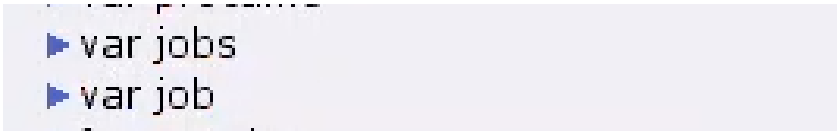

Рис. 1.5: Определение переменных модели

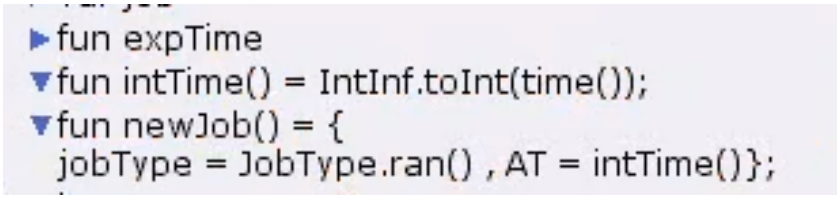

Рис. 1.6: Определение функций системы

1.4 Настройка параметров модели в CPN Tools

Для корректной работы модели в CPN Tools были заданы параметры элементов на всех трех листах: **System**, **Arrivals** и **Server**.

1.4.1 Лист System

На данном листе представлены основные элементы сети массового обслуживания:

- **Позиция Queue**:
  - Цветовое множество: Jobs.
  - Начальная маркировка: 1'[], что указывает на пустую очередь в начале моделирования.
- **Позиция Completed**:
  - Цветовое множество: Job.

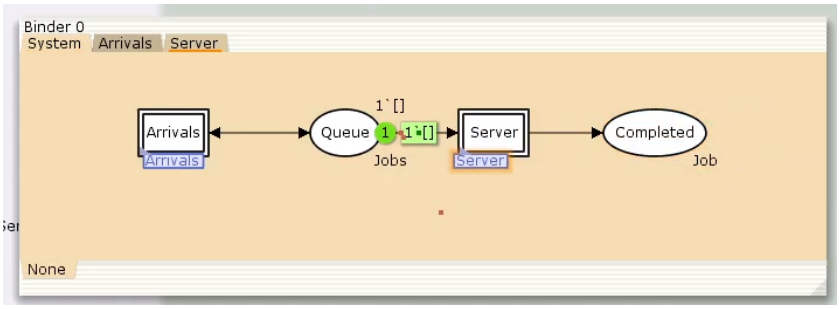

Рис. 1.7: Параметры элементов основного графа системы обработки заявок в очереди

1.4.2 Лист Arrivals — генератор заявок

На этом листе формируются заявки, поступающие в систему:

- **Позиция Init**:
  - Цветовое множество: UNIT.
  - Начальная маркировка: 1'()@0, что означает начало генерации заявок с нулевого времени.

- **Позиция `Next`**:
  - Цветовое множество: `UNIT`.
- **Переход `Init`**:
  - На дуге из `Init`: выражение `()` — инициирует генерацию заявок.
  - На дугах от `Init` и `Arrive` к `Next`: `()@+expTime(100)` — задаёт интервалы между заявками по экспоненциальному распределению со средней интенсивностью 100.
- **Переход `Arrive`**:
  - На дуге из `Next`: `()` — передаёт сигнал генерации.
  - На дуге к `Queue`: `jobs^^[job]` — добавляет заявку в очередь.
  - Обратная связь с `Queue`: `jobs`.

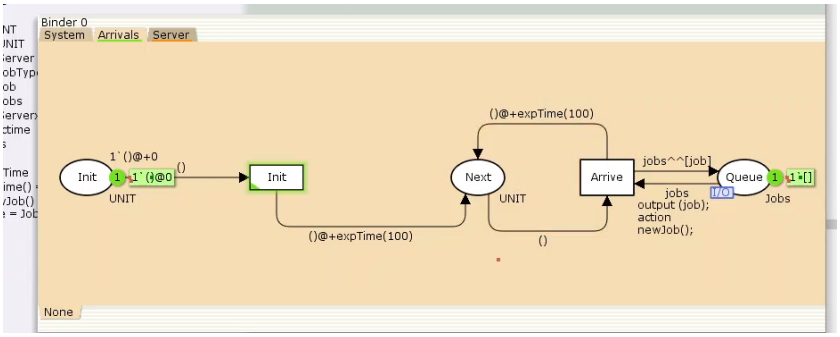

Рис. 1.8: Параметры элементов генератора заявок системы

1.4.3 Лист `Server` — обработка заявок

На этом листе смоделирован сервер, обрабатывающий заявки:

- **Позиция `Busy`**:
  - Цветовое множество: `Server`.

- Начальная маркировка: `1'server@+0` — сервер свободен в начале моделирования.

- **Позиция `Idle`**:
  - Цветовое множество: `ServerxJob`.

- **Переход `Start`**:
  - Output: `proctime`.
  - Action: `expTime(90)` — определяет, что время обработки заявки подчиняется экспоненциальному распределению со средним значением 90.
  - На дуге от `Queue`: `job::jobs` — позволяет начать обработку, если есть заявки.
  - К `Busy`: `(server, job)@+proctime` — передача заявки на сервер с учетом времени обработки.
  - Обратная связь в `Queue`: `jobs`.

- **Переход `Stop`**:
  - От `Busy`: `(server, job)` — завершение обработки заявки.
  - К `Completed`: `job` — заявка считается обслуженной.
  - Состояние сервера обновляется через `Idle`: `server`.

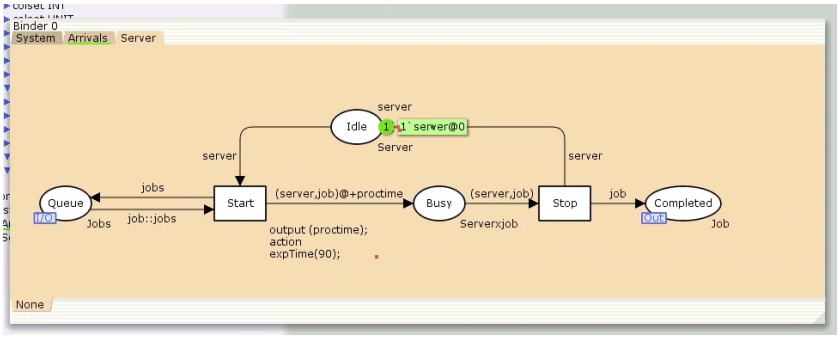

Рис. 1.9: Параметры элементов графа обработки заявок

После задания всех необходимых параметров компоненты обработчика заявок активируются, и система начинает функционировать, как показано на рисунке ниже.

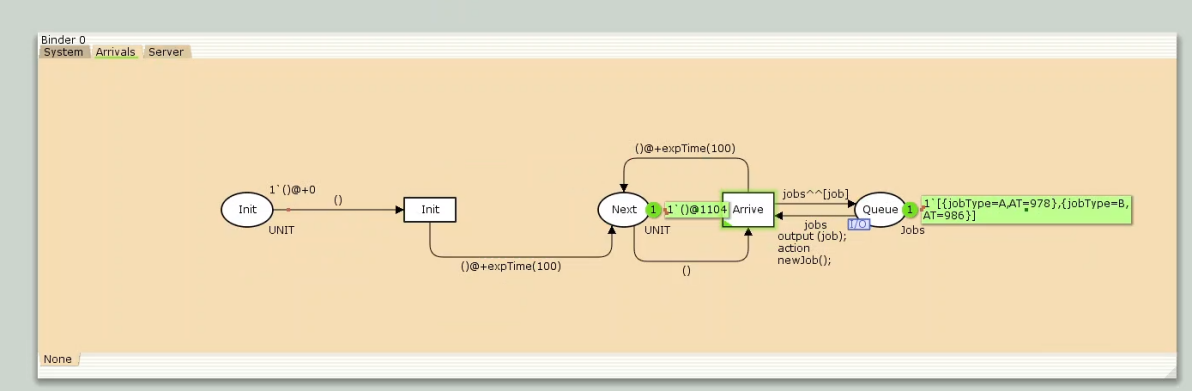

Рис. 1.10: Запуск системы обработки заявок в очереди

1.5 Мониторинг параметров моделируемой системы

Для отслеживания параметров используется палитра **Monitoring**. В первую очередь добавляется **Break Point**, который размещается на переходе `Start`. После этого в меню **Monitor** появляется новый раздел, назовём его `Ostanovka`.

В этом разделе необходимо изменить функцию `Predicate`, отвечающую за условие активации монитора. Стандартное значение `true` заменяется на выражение `Queue_Delay.count()=200`, чтобы монитор срабатывал каждые 200 заявок.

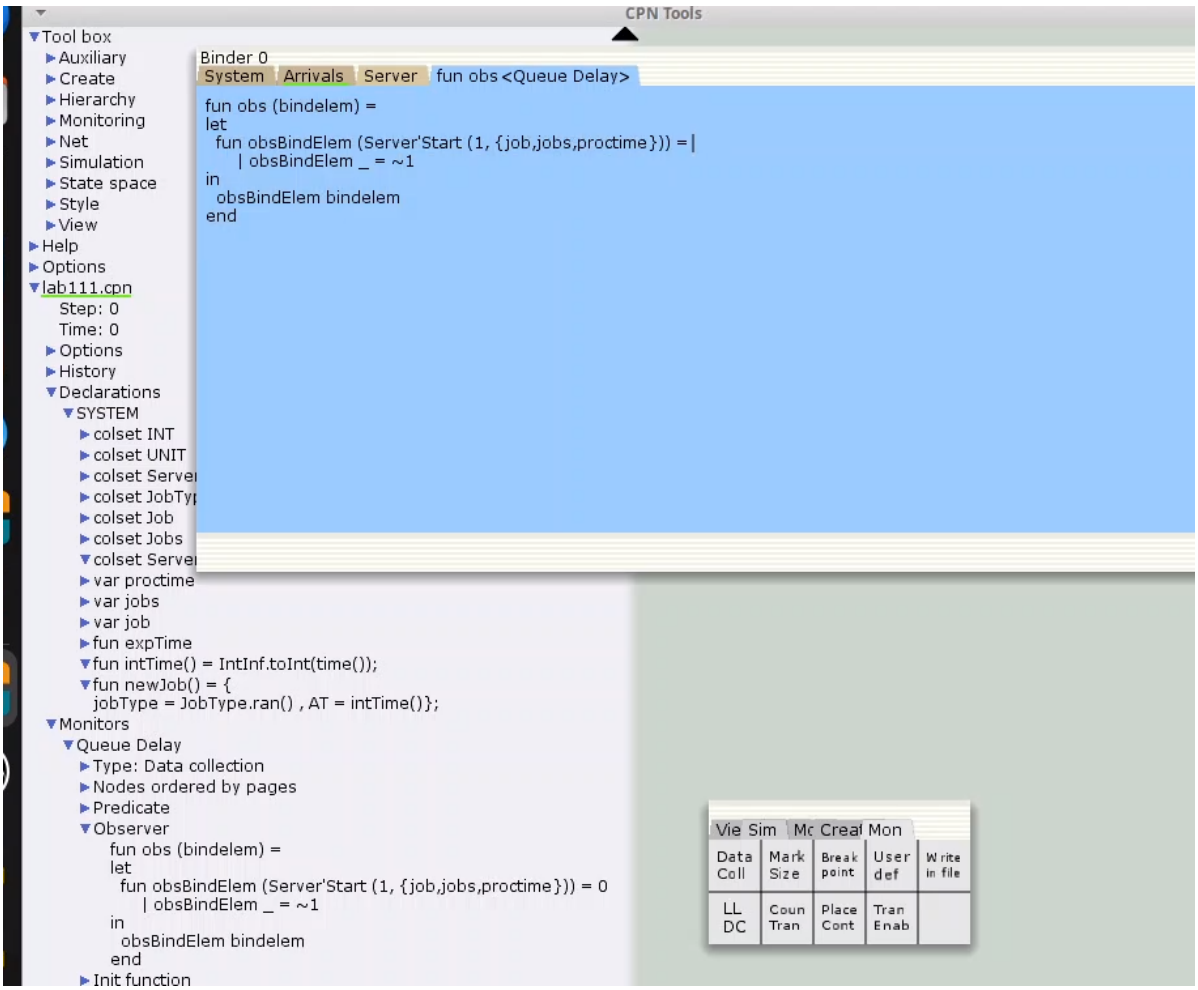

Рис. 1.11: Функция Predicate монитора Ostanovka

1.5.1 Определение функции `Queue_Delay.count()`

С помощью палитры **Monitoring** выбирается элемент **Data Call**, который размещается также на переходе `Start`. Новый монитор следует назвать `Queue Delay` (без подчеркивания).

Функция `Observer` будет выполняться, когда предикат возвращает `true`. По умолчанию она возвращает 0 или ~1, где подчёркивание обозначает произвольный аргумент. Чтобы получить значение задержки, нужно из текущего времени `intTime()` вычесть временную метку `AT`, которая указывает момент поступления заявки в очередь.

Рис. 1.12: Функция Observer монитора Queue Delay

1.5.2 Вычисление задержки в действительных значениях

Для получения задержки в виде действительных чисел, снова используется **Data Call** на переходе `Start`. Новый монитор называется `Queue Delay Real`. Функцию `Observer` следует изменить так, чтобы результат преобразовывался в тип `real` (например, используя `~1.0`).

После запуска системы в каталоге проекта появится файл `Queue_Delay_Real.log` с данными, аналогичными файлу `Queue_Delay.log`, но с действительными значениями.

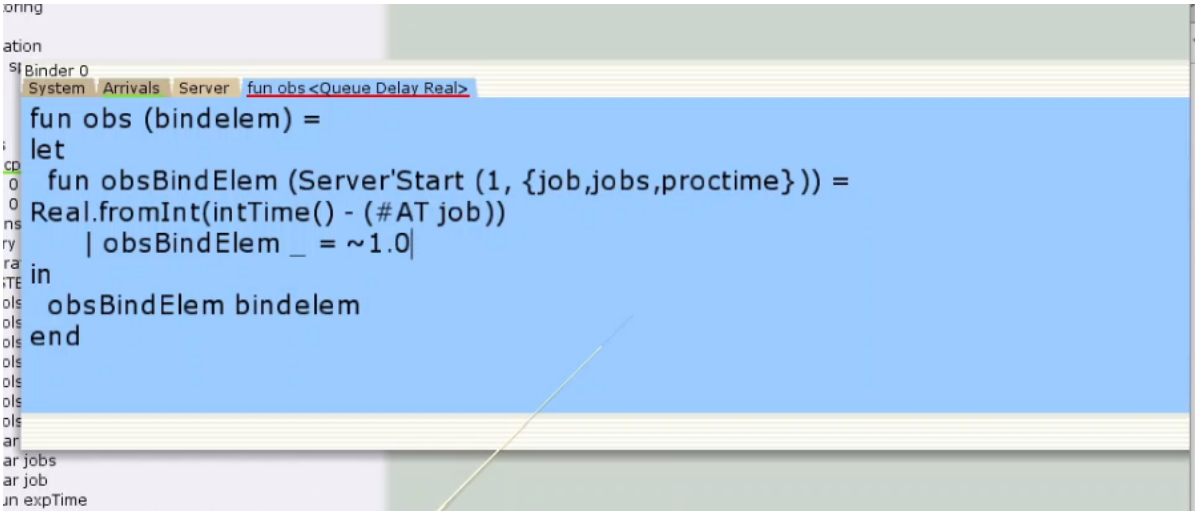

Рис. 1.13: Функция Observer монитора Queue Delay Real

1.5.3 Подсчёт случаев превышения задержки

Чтобы посчитать количество случаев, когда задержка превысила заданное значение, снова используется **Data Call** на переходе `Start`. Новый монитор следует назвать `Long Delay Time`, и изменить в нём функцию `Observer`, как показано ниже.

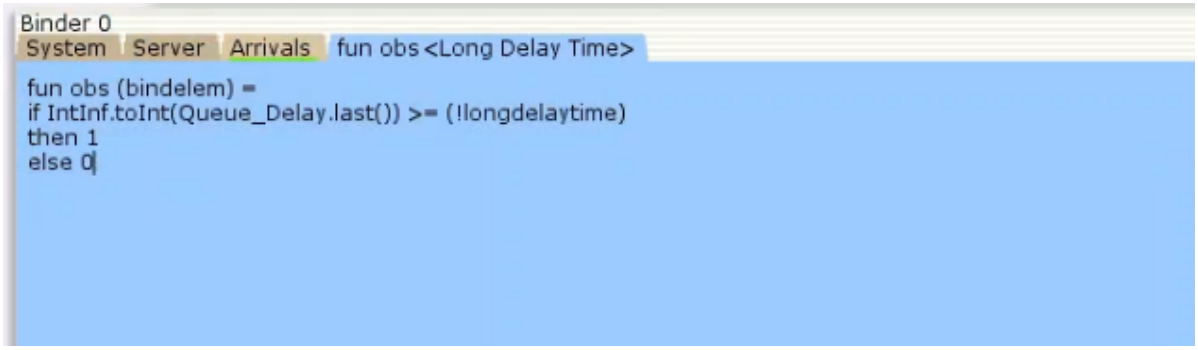

Рис. 1.14: Функция Observer монитора Long Delay Time

После запуска системы создается файл `Queue_Delay.log`, где: - первая колонка — значение задержки, - вторая — счётчик, - третья — шаг, - четвёртая — время.

С помощью **gnuplot** можно построить график изменения задержки, используя: - по оси X — время, - по оси Y — значения задержки.

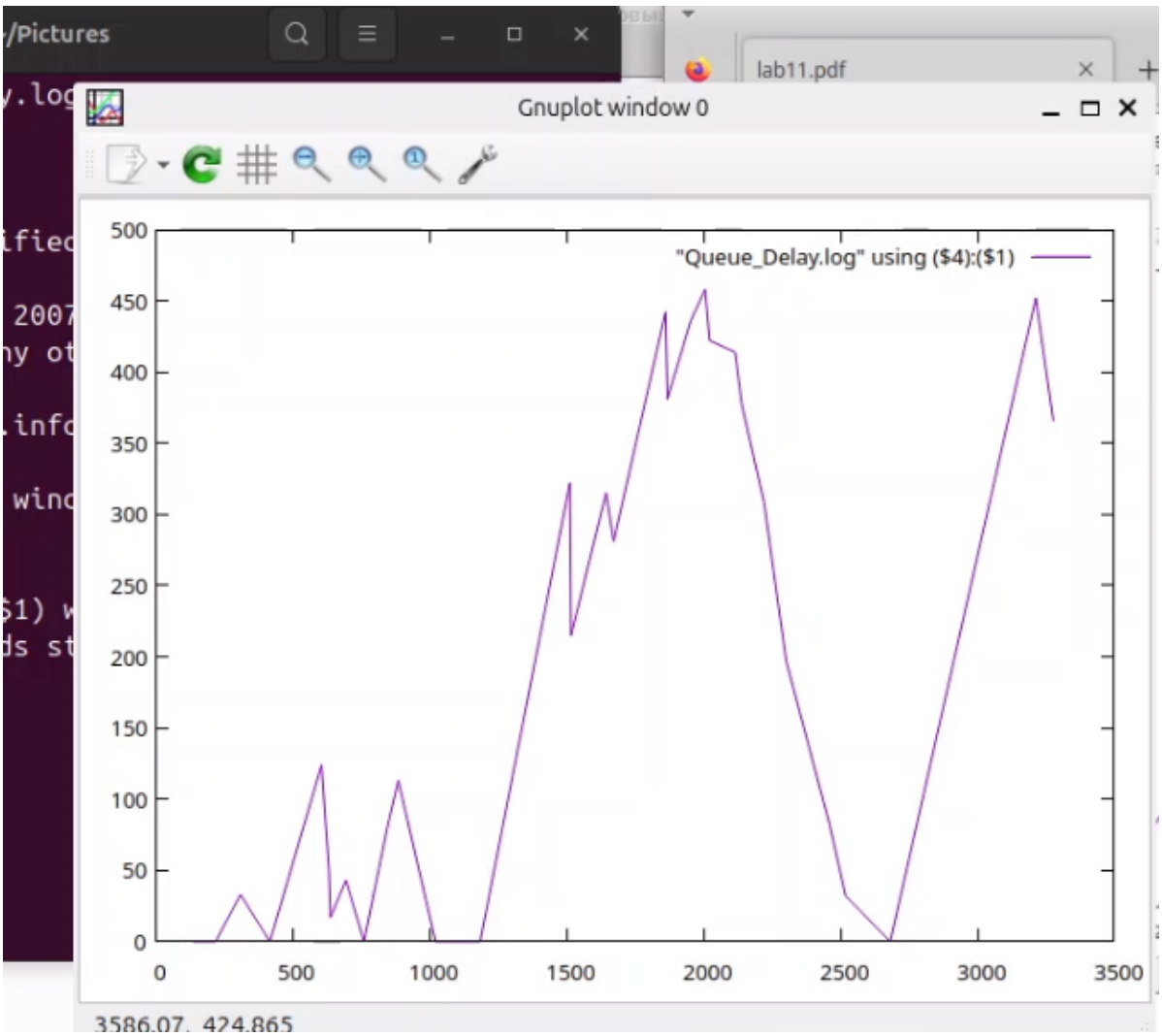

Рис. 1.15: График изменения задержки в очереди

С помощью **gnuplot** можно построить график, иллюстрирующий периоды времени, в которые значения задержки в очереди превышали установленный порог — 200. Этот график помогает визуализировать моменты перегрузки в системе (см. рисунок ниже).

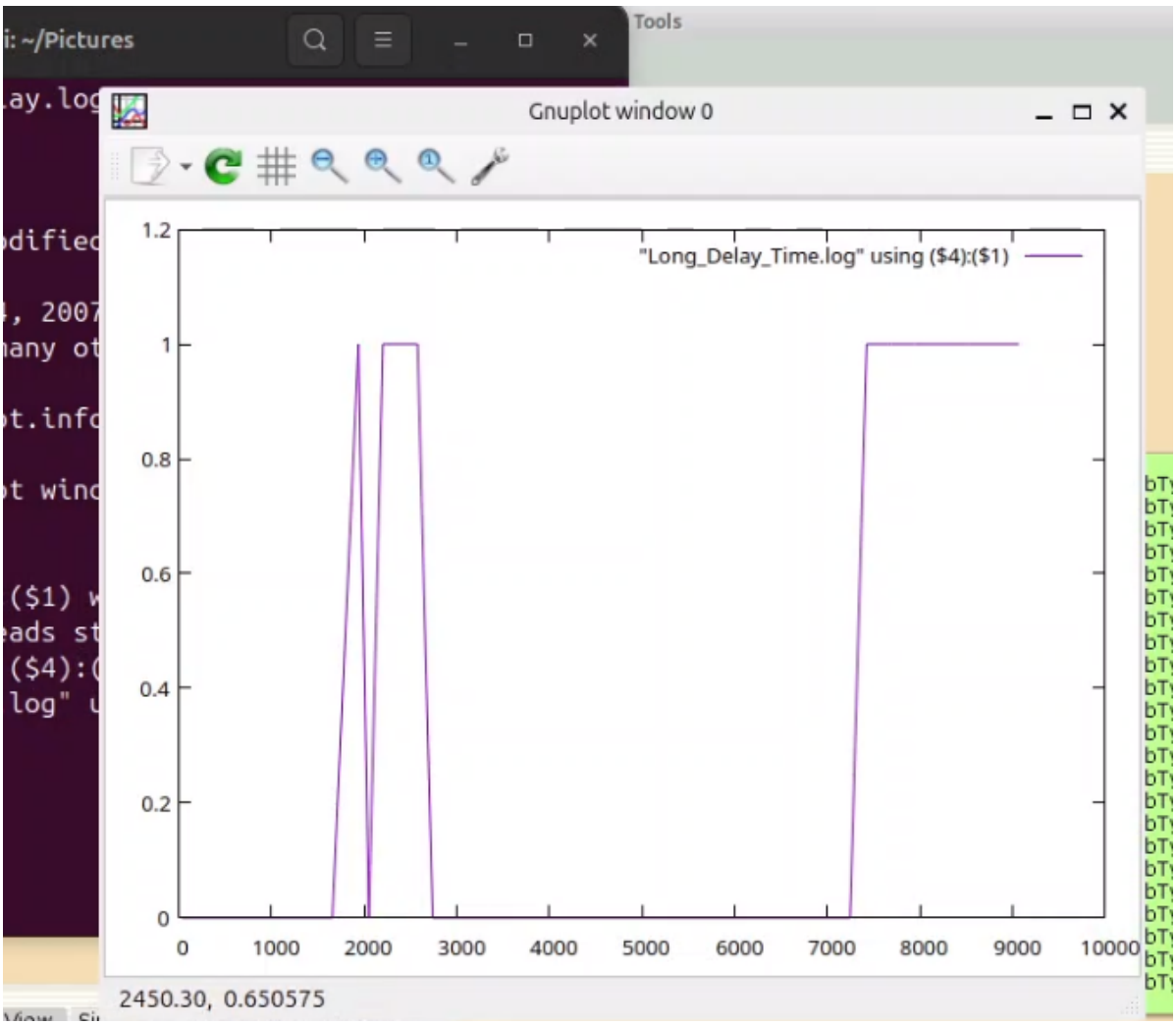

Рис. 1.16: Периоды времени, когда значения задержки в очереди превышали заданное значение

2 Выводы

В рамках выполненной работы была разработана и реализована модель системы массового обслуживания **M|M|1** с использованием среды **CPN Tools**.

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].