Лабораторная работа 11

Модель системы массового обслуживания M |M |1

Горяйнова АА

Содержание

# 1 Цель работы

Реализовать модель в CPN tools.

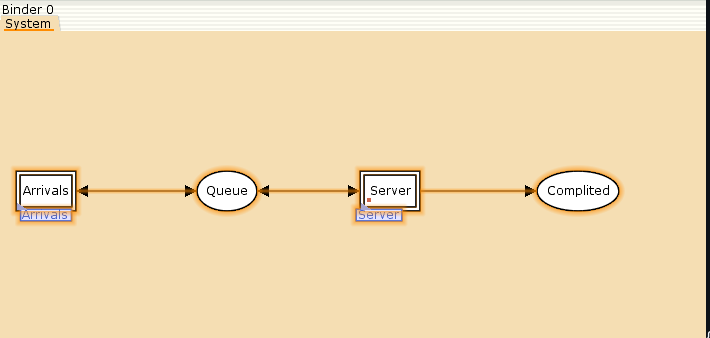
# 2 Задание

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

# 3 Выполнение лабораторной работы

**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.

Будем использовать три отдельных листа: на первом листе опишем граф системы (рис. **¿fig:001?**), на втором — генератор заявок (рис. 1), на третьем — сервер обработки заявок (рис. 2). 

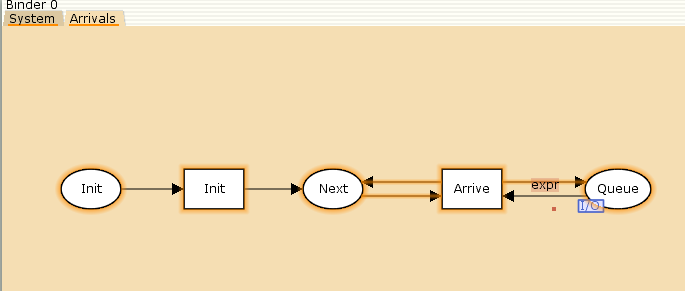


Рис. 1: Граф генератора заявок системы

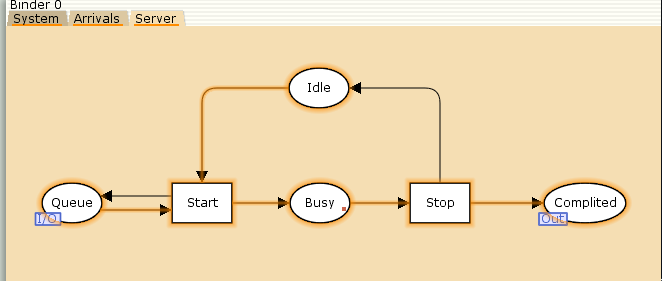


Рис. 2: Граф процесса обработки заявок на сервере системы

Зададим декларации системы (рис. 3).

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 — A и B;
* 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.

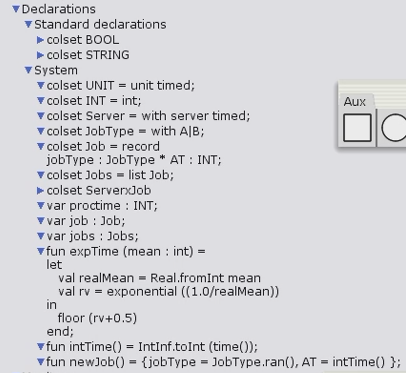


Рис. 3: Задание деклараций системы

Зададим параметры модели на графах сети. На листе System (рис. 4): – у позиции Queue множество цветов фишек — Jobs; начальная маркировка 1`[] определяет, что изначально очередь пуста. – у позиции Completed множество цветов фишек — Job.

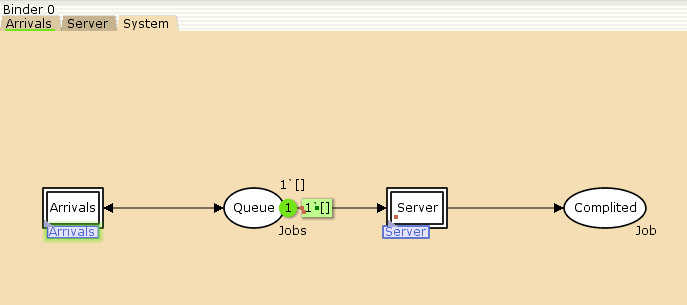


Рис. 4: Параметры элементов основного графа системы обработки заявок в очереди

На листе Arrivals (рис. 5): – 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;

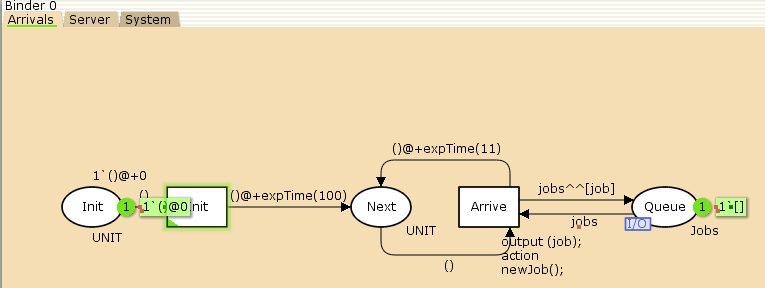


Рис. 5: Параметры элементов генератора заявок системы

На листе Server (рис. 6): – у позиции Busy: множество цветов фишек — Server, начальное значение мар- кировки — 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 об- служивания 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 заяв- ки на сервере; – 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.

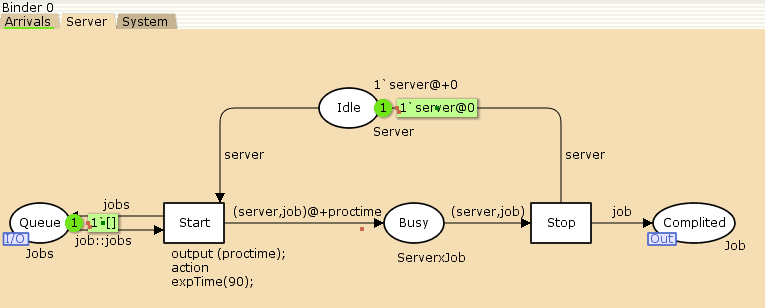


Рис. 6: Параметры элементов обработчика заявок системы

Выбираем Break Point (точка останова) и уста- навливаем её на переход Start. После этого в разделе меню Monitor появится новый подраздел, который назовём Ostanovka. В этом подразделе необходимо внести изме- нения в функцию Predicate, которая будет выполняться при запуске монитора(рис. 7)

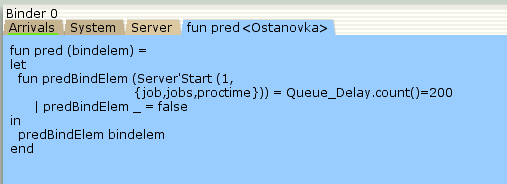


Рис. 7: Функция Predicate монитора Ostanovka

Функция Observer выполняется тогда, когда функция предикатора выдаёт значе- ние 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. 8)

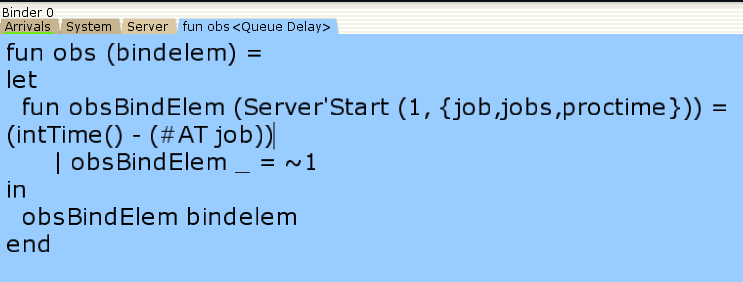
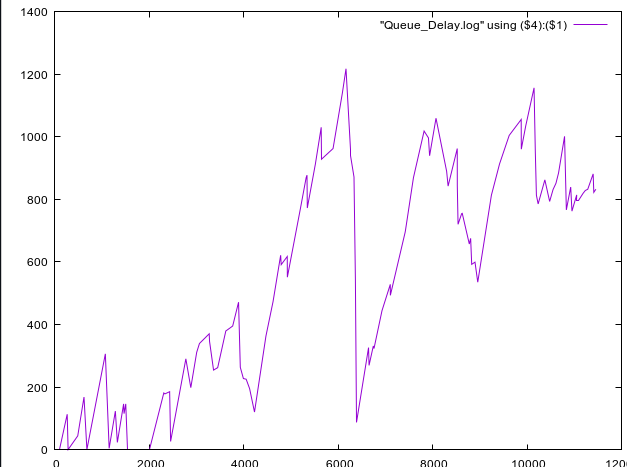
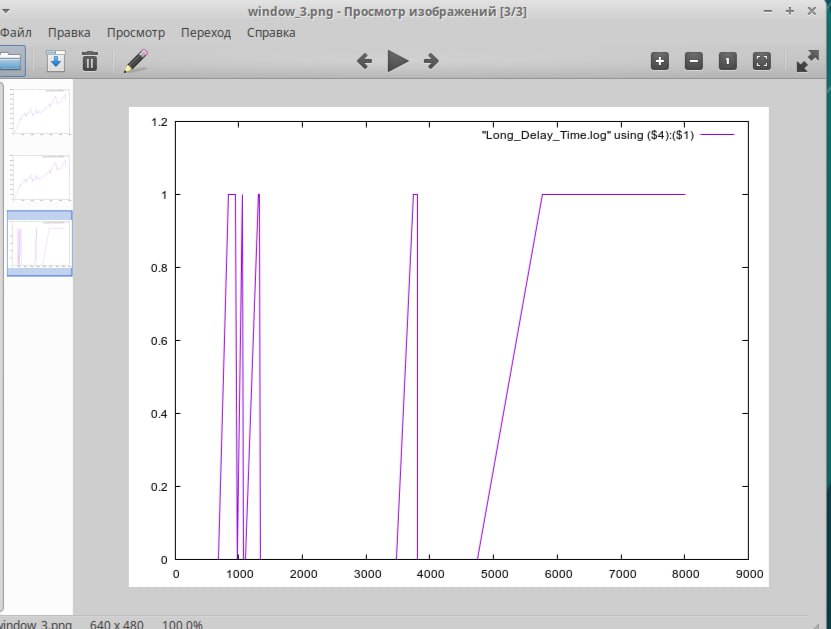


Рис. 8: Функция Observer монитора Queue Delay

После запуска программы на выполнение в каталоге с кодом программы появит- ся файл Queue\_Delay.log, содержащий в первой колонке — значение задержки очереди, во второй — счётчик, в третьей — шаг, в четвёртой — время. С помощью 114 Лабораторная работа 11. Модель системы массового обслуживания M |M |1 gnuplot можно построить график значений задержки в очереди (рис. **¿fig:010?**), выбрав по оси x время, а по оси y — значения задержки. 

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. 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. Посчитаем, сколько раз задержка превысила заданное значение. С помощью палит- ры Monitoring выбираем Data Call и устанавливаем на переходе Start. Монитор называем Long Delay Time Если значение монитора Queue Delay превысит некоторое заданное значение, то функция выдаст 1, в противном случае — 0. Восклицательный знак означает разыменование ссылки. При этом необходимо в декларациях задать глобальную переменную (в форме ссылки на число 200): longdelaytime: globref longdelaytime = 200; 116 Лабораторная работа 11. Модель системы массового обслуживания M |M |1 Если значение монитора Queue Delay превысит некоторое заданное значение, то функция выдаст 1, в противном случае — 0. Восклицательный знак означает разыменование ссылки. При этом необходимо в декларациях задать глобальную переменную (в форме ссылки на число 200): longdelaytime: globref longdelaytime = 200; С помощью gnuplot можно построить график (рис. **¿fig:011?**), демонстрирующий, в какие периоды времени значения задержки в очереди превышали заданное значение 200: gnuplot plot [0:][0:1.2] “Long\_Delay\_Time.log” using ($4):($1) with lines quit 

# 4 Выводы

Я реализовала модель в CPN tools.

# Список литературы