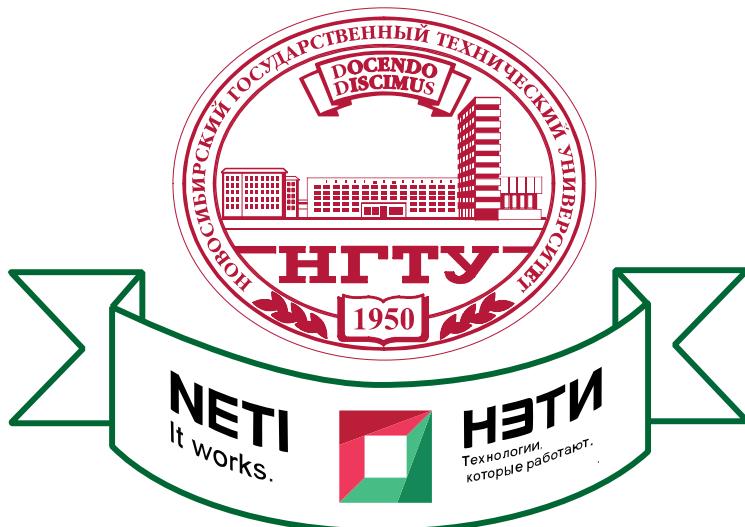


Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования

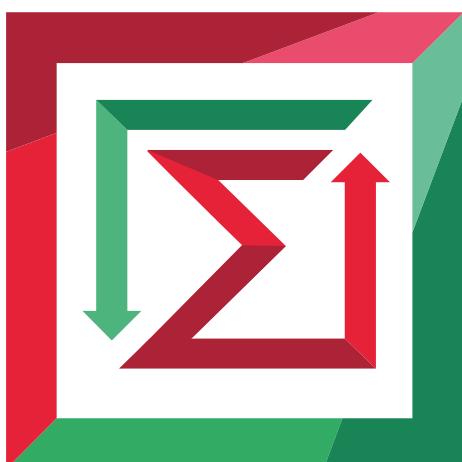
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



Теоретической и прикладной информатики

Лабораторная работа № 2
по дисциплине «Компьютерное моделирование»

Оптимизация работы сервера



Факультет: ПМИ
Группа: ПМИ-02
Студент: Сидоров Даниил,
Дюков Богдан
Преподаватель: Карманов Виталий Сергеевич

Новосибирск

2026

1. Формулировка задания

Построить модель работы сервера и определить математическое ожидание времени и вероятности обработки запросов.

Сервер, имеющий входной буфер на 4 запроса, обрабатывает запросы, поступающие с рабочих станций с временными интервалами, распределенными по показательному закону со средним значением 3 мин/с. Время обработки сервером одного запроса распределено по экспоненциальному закону со средним значением 2 мин.

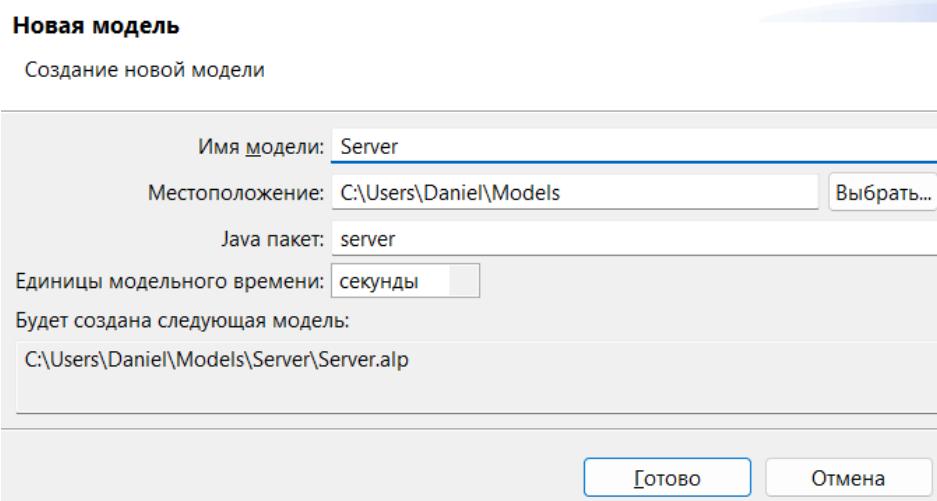
2. Цели работы

Построить имитационную модель работы сервера и оптимизировать ее параметры.

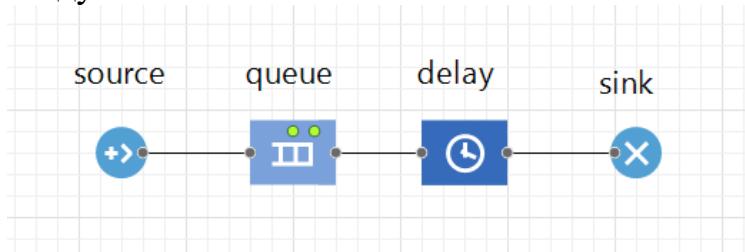
3. Описание выполненных действий

Создание модели

Зададим имя модели Server и установим единицы модельного времени – секунды. После создания модели откроется пользовательский интерфейс.



Добавим блоки библиотеки моделирования процессов на диаграмму и соединим их между собой.



Перейдем к установке свойств блоков модели и настройке модели. Изменим свойства объекта Source, учитывая, что время между прибытиями заявок определено в секундах согласно экспоненциальному закону:

source - Source

Имя:	source	<input checked="" type="checkbox"/> Отображать имя
<input type="checkbox"/> Исключить		
Прибывают согласно:	= Времени между прибытиями	
Время между прибытиями:	= exponential(1/120) секунды	
Первое прибытие происходит:	= после заданного времени	
Считать параметры агентов из БД:	= <input type="checkbox"/>	
За 1 раз создается несколько агентов:	= <input type="checkbox"/>	
Ограниченнное кол-во прибытий:	= <input type="checkbox"/>	

Определим свойства объекта Queue с учетом, что длина очереди (вместимость) равна пяти, и включим сбор статистики.

queue - Queue

Имя:	queue	<input checked="" type="checkbox"/> Отображать имя
<input type="checkbox"/> Исключить		
Вместимость:	= 5	
Максимальная вместимость:	= <input type="checkbox"/>	
Место агентов:	= <input type="checkbox"/>	
Специфические		
Очередь:	= FIFO	
Разрешить уход по таймауту:	= <input type="checkbox"/>	
Разрешить вытеснение:	= <input type="checkbox"/>	
Вернуть агента в исходную точку:	= <input checked="" type="checkbox"/>	
Включить сбор статистики:	= <input checked="" type="checkbox"/>	

Изменим свойства объекта Delay, считая, что время задержки определено согласно экспоненциальному закону и включим сбор статистики:

delay - Delay

Имя:	delay	<input checked="" type="checkbox"/> Отображать имя	<input type="checkbox"/> Исключить
Тип задержки:	= <input checked="" type="radio"/> Определенное время <input type="radio"/> До вызова функции stopDelay()		
Время задержки:	= exponential(1/120.0)		секунды
Вместимость:	= 1		
Максимальная вместимость:	= <input type="checkbox"/>		
Место агентов:	= <input type="checkbox"/> node		<input type="checkbox"/> <input type="checkbox"/>
Специфические			
Выталкивать агентов:	= <input type="checkbox"/>		
Вернуть агента в исходную точку:	= <input checked="" type="checkbox"/>		
Включить сбор статистики:	= <input checked="" type="checkbox"/>		

Изменим свойства Simulation:Main, установив следующие параметры:

- режим выполнения: виртуальное время (максимальная скорость);
- остановка: в заданное время;

- конечное время: 3600 с;
- генератор случайных чисел: фиксированное начальное число (воспроизводимые прогоны), равное 4.

Simulation - Простой эксперимент

Имя: Исключить

Агент верхнего уровня:

Максимальный размер памяти: Мб

Пропустить экран эксперимента и запустить модель

Параметры

Модельное время

Режим выполнения: Виртуальное время (максимальная скорость)
 Реальное время со скоростью

Остановить:

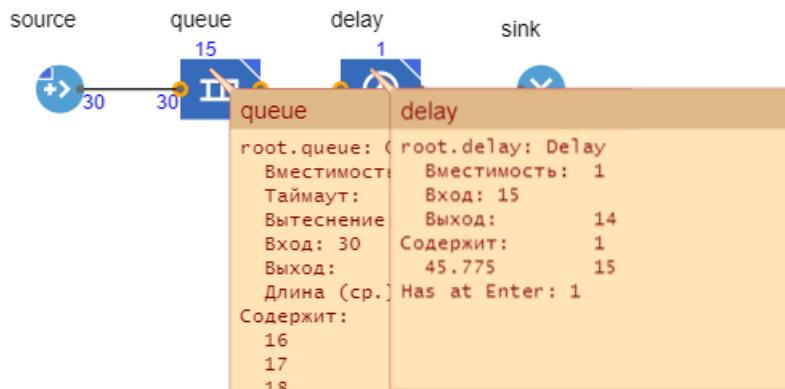
Начальное время: Конечное время:

Начальная дата: Конечная дата:

Случайность

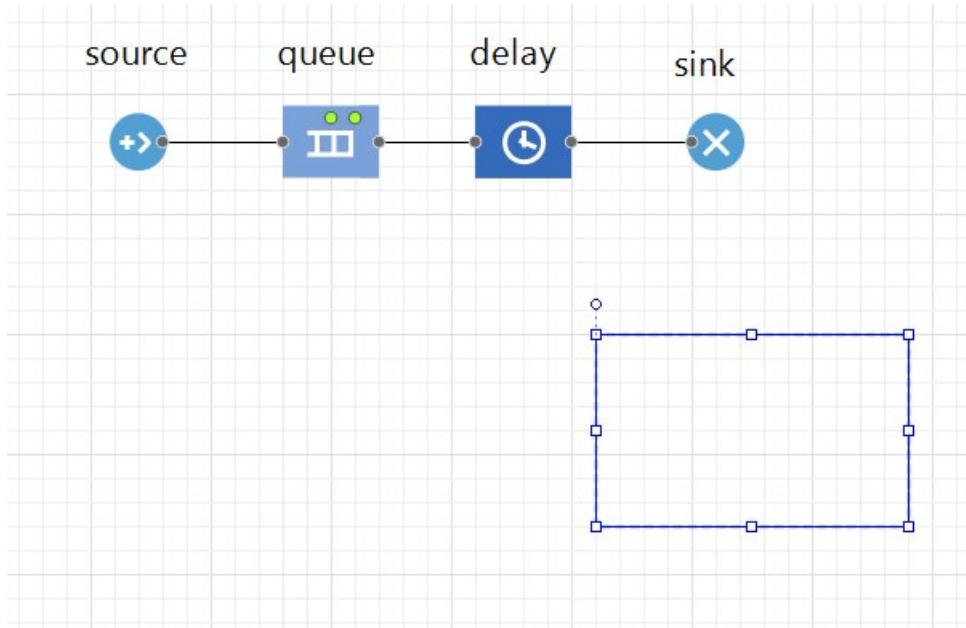
Генератор случайных чисел:
 Случайное начальное число (уникальные "прогоны")
 Фиксированное начальное число (воспроизводимые "прогоны") Начальное число:
 Нестандартный генератор (подкласс класса Random):

Окно свойств объектов имеет такой вид:



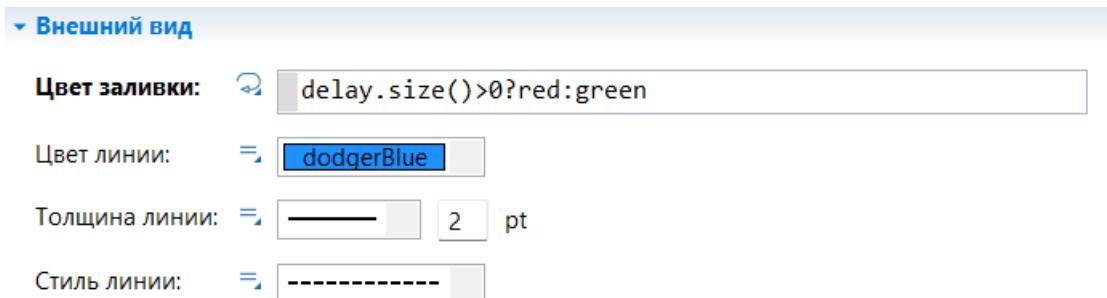
Создание анимации модели

Выберем прямоугольный узел и обозначим сервер.

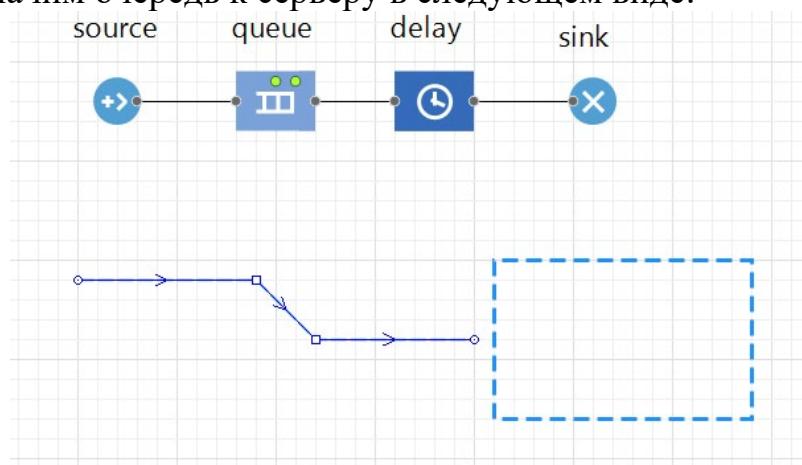


В свойствах прямоугольного узла установим динамический цвет заливки, который изменяется в зависимости от того, обрабатывает ли сервер в данный момент запрос или нет:

`delay.size()>0?red:green`



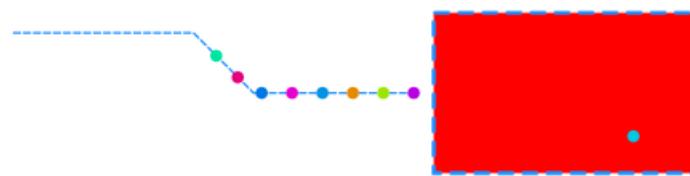
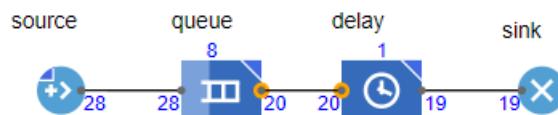
Обозначим очередь к серверу в следующем виде:



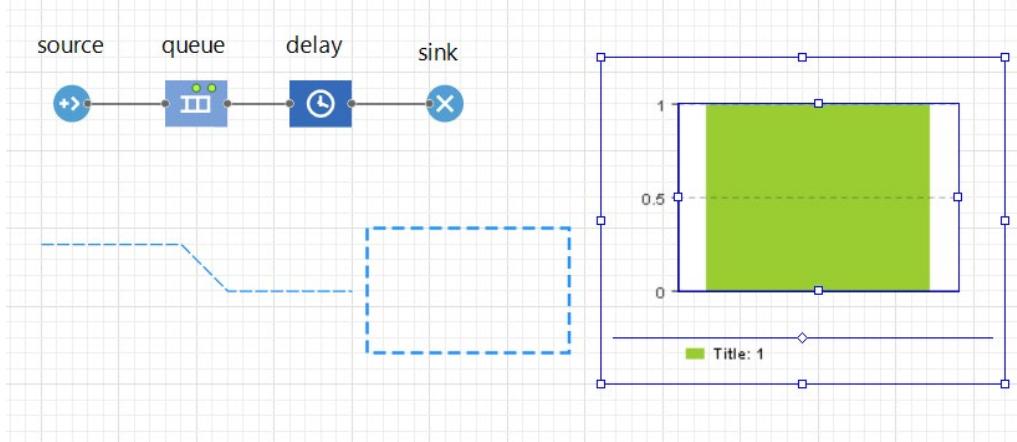
queue - Queue

Имя:	queue	<input checked="" type="checkbox"/> Отображать имя	<input type="checkbox"/> Исключить
Вместимость:	= 15		
Максимальная вместимость:	= <input type="checkbox"/>		
Место агентов:	= <input type="checkbox"/> path		
delay - Delay			
Имя:	delay	<input checked="" type="checkbox"/> Отображать имя	<input type="checkbox"/> Исключить
Тип задержки:	= <input checked="" type="radio"/> Определенное время <input type="radio"/> До вызова функции stopDelay()		
Время задержки:	= <input type="checkbox"/> exponential(1/120.0)	секунды	
Вместимость:	= 1		
Максимальная вместимость:	= <input type="checkbox"/>		
Место агентов:	= <input type="checkbox"/> node		
Специфические			
Выталкивать агентов:	= <input type="checkbox"/>		
Вернуть агента в исходную точку:	= <input checked="" type="checkbox"/>		
Включить сбор статистики:	= <input checked="" type="checkbox"/>		

Запустим модель:



Для отображения среднего коэффициента использования сервера добавим следующую диаграмму.



Установим свойства диаграммы.

▼ Данные

Заголовок:	Server Utilization
Цвет:	yellowGreen
Значение:	delay.statsUtilization.mean()

Укажем параметры диаграммы:

▼ Местоположение и размер

Уровень:	level		
X:	590	Ширина:	200
Y:	40	Высота:	360

▼ Легенда

<input checked="" type="checkbox"/> Отображать легенду					
Высота:	30				
Цвет текста:	black				
Расположение:	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

▼ Область диаграммы

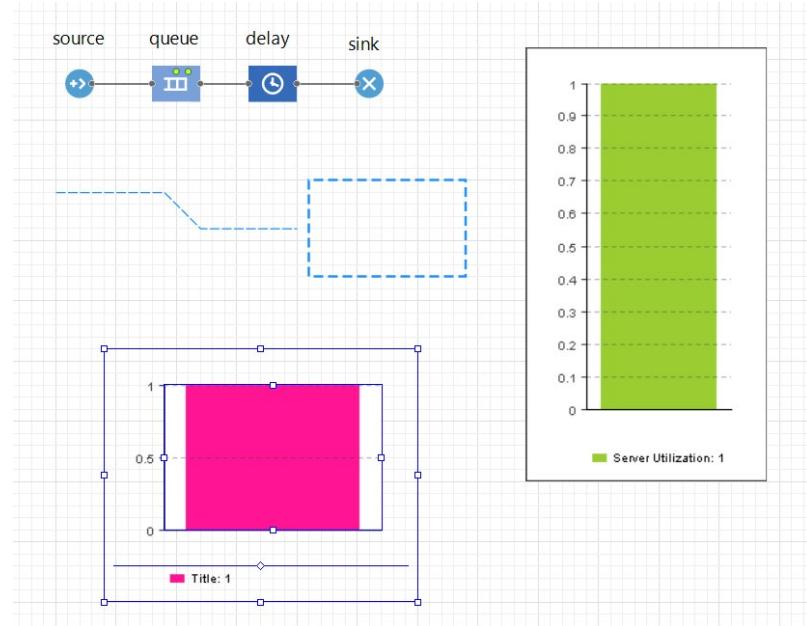
Смещение по оси X:	50	Ширина:	120
Смещение по оси Y:	30	Высота:	270
Цвет фона:	white		
Цвет границы:	black		

и выберем внешний вид диаграммы:

▼ Внешний вид

Направление столбцов:	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Относительная ширина столбцов:	<input type="range" value="80"/> 80%			
Положение подписей у осей:	Слева			
Цвет фона:	white			
Цвет границы:	darkGray			
Цвет меток:	darkGray			
Цвет сетки:	darkGray			

Добавим еще одну диаграмму для отображения средней длины очереди.



Также установим свойства диаграммы:

▼ Данные

Заголовок:	Queue length
Цвет:	deepPink
Значение:	queue.statsSize.mean()

+ X ↑ ↓

и выберем внешний вид диаграммы:

▼ Внешний вид

Направление столбцов:



Относительная ширина столбцов:



Положение подписей у осей:

Снизу

Цвет фона:

Нет заливки

Цвет границы:

darkGray

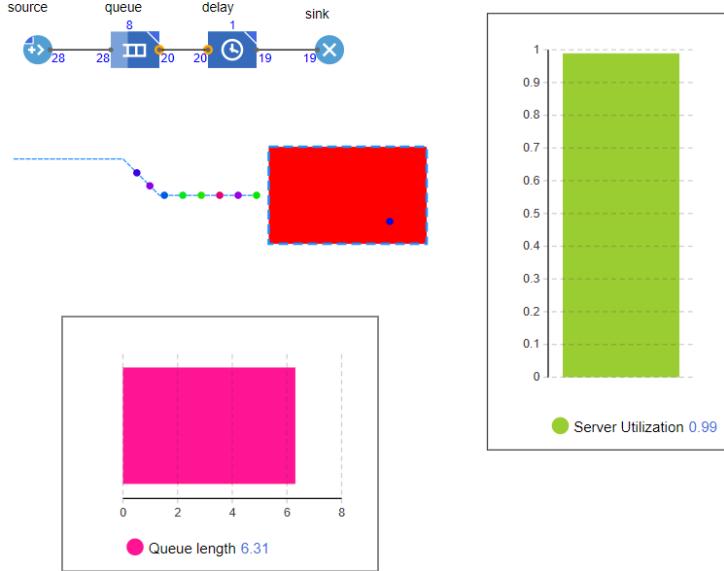
Цвет меток:

darkGray

Цвет сетки:

darkGray

Запустив модель, получим:

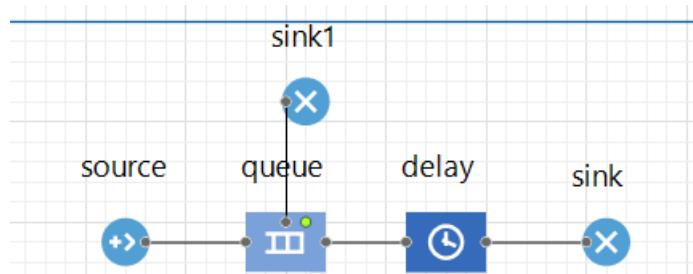


Все запросы, вырабатываемые объектом Source, имеют один и тот же приоритет, поэтому при полном заполнении накопителя (пять запросов) теряется будет последний запрос. Уточним модель, изменив свойства объекта Source, и разрешим вытеснение.

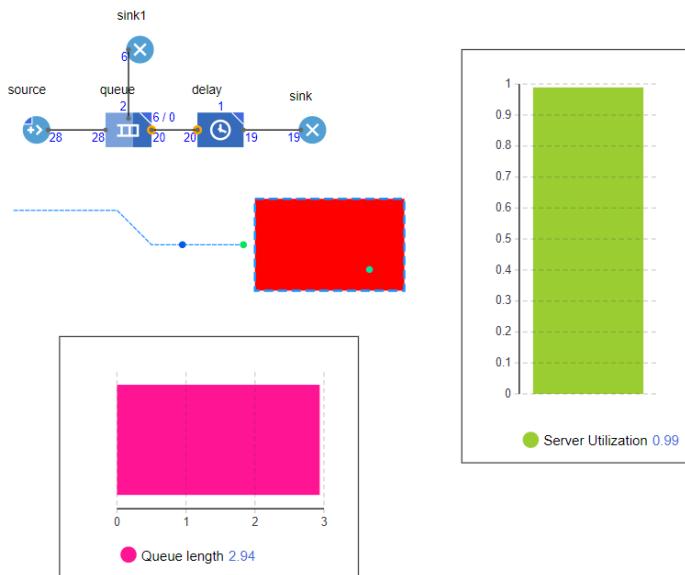
queue - Queue

Имя:	queue	<input checked="" type="checkbox"/> Отображать имя	<input type="checkbox"/> Исключить
Вместимость:	= 5		
Максимальная вместимость:	=	<input type="checkbox"/>	
Место агентов:	= path		
Специфические			
Очередь:	= FIFO		
Разрешить уход по таймауту:	=	<input type="checkbox"/>	
Разрешить вытеснение:	=	<input checked="" type="checkbox"/>	
Вернуть агента в исходную точку:	=	<input checked="" type="checkbox"/>	
Включить сбор статистики:	=	<input checked="" type="checkbox"/>	

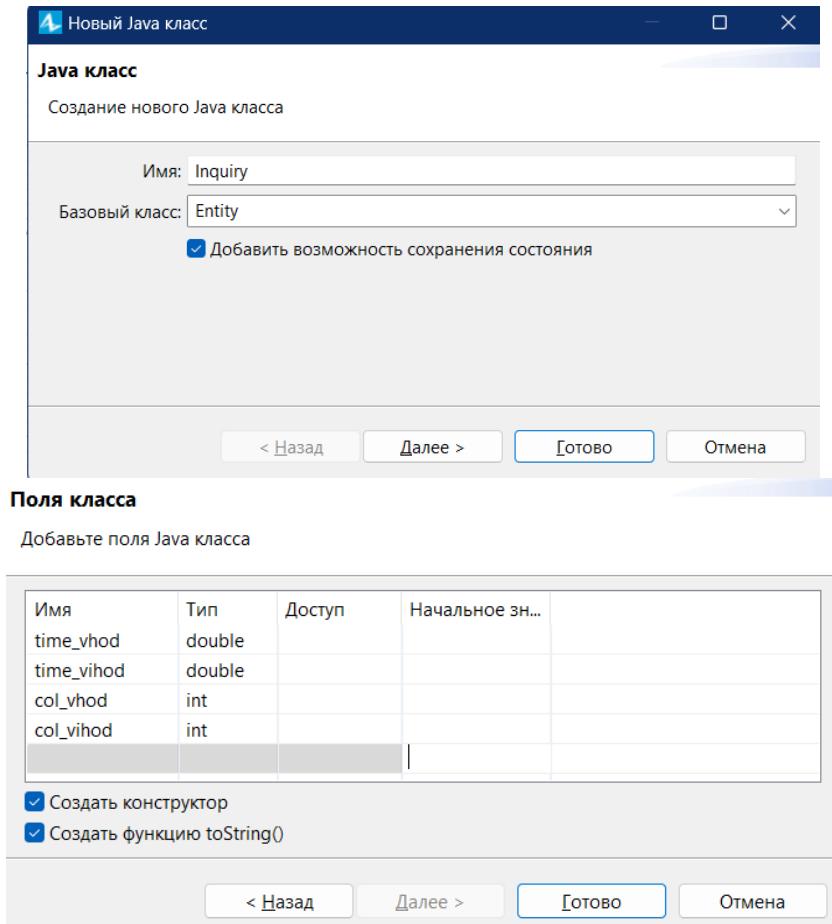
Для уничтожения потерянных запросов вследствие полного заполнения накопителя нужно добавить второй объект Sink.



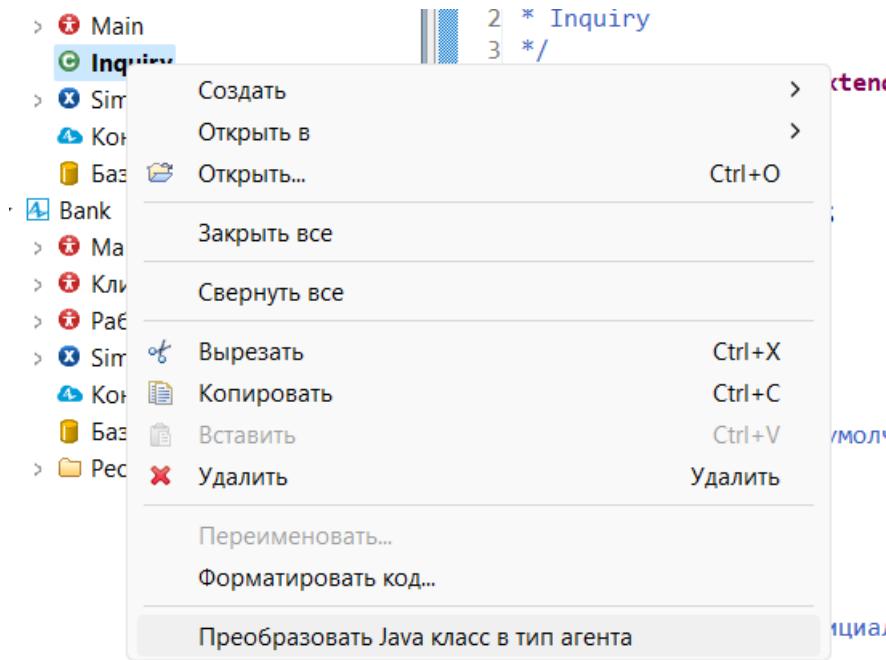
Соединим порт outPreempted объекта Queue с входным портом InPort блока Sink1.



Далее создадим тип заявок Inquiry. Для этого в панели «Проекты» правой кнопкой мыши выберем «Создать – Java-класс» с именем Inquiry и базовым классом Entity.



Преобразуем класс в тип агента следующим образом:



Окно с автоматически созданными параметрами нестандартного типа заявок Inquiry выглядит так:

```

1 /**
2 * Inquiry
3 */
4 public class Inquiry extends Entity implements Serializable {
5
6     double time_vhod;
7
8     double time_vihod;
9
10    int col_vhod;
11
12    int col_vihod;
13
14    /**
15     * Конструктор по умолчанию
16     */
17    public Inquiry() {
18    }
19
20    /**
21     * Конструктор, инициализирующий поля
22     */
23    public Inquiry(double time_vhod, double time_vihod, int col_vhod, int col_vihod) {
24        this.time_vhod = time_vhod;
25        this.time_vihod = time_vihod;
26        this.col_vhod = col_vhod;
27        this.col_vihod = col_vihod;
28    }
29
30    @Override
31    public String toString() {
32        return
33            "time_vhod = " + time_vhod + " "
34            "time_vihod = " + time_vihod + " "
35            "col_vhod = " + col_vhod + " "
36            "col_vihod = " + col_vihod + " ";
37    }
38
39    /**
40     * Это число используется при сохранении состояния модели<br>
41     * Его рекомендуется изменить в случае изменения класса
42     */
43    private static final long serialVersionUID = 1L;
44}
45

```

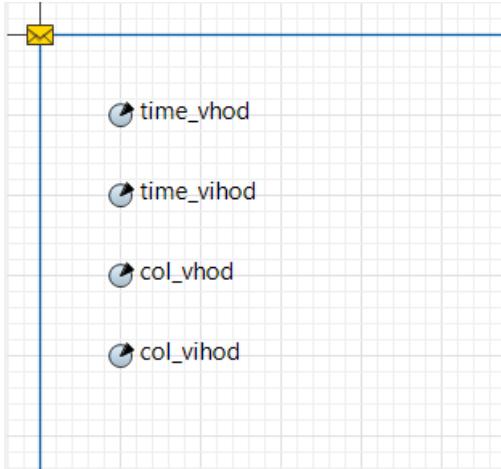
Приведем по шагам действия, необходимые для формирования нестандартного типа заявки:

- 1) создадим тип заявок Inquiry;
- 2) откроем палитру «Библиотека моделирования процессов»;
- 3) поместим элемент Agent в графический редактор;
- 4) в поле «Имя нового агента» введем Inquiry;
- 5) выберем анимацию агента, установив 2D, и выберем из выпадающего списка «Сообщение»;
- 6) выполним настройку параметров агента.

Параметр	Тип
time_vhod	double

time_vihod	double
col_vhod	int
Col_vihod	int

Окно с автоматически созданными параметрами нестандартного типа заявок Inquiry выглядит так:



Для сбора статистических данных о времени обработки запросов сервером необходимо добавить элемент статистики. Этот элемент будет запоминать значения времени для каждого запроса, предоставляя пользователю стандартную статистическую информацию (среднее, минимальное, максимальное из измеренных значений, среднеквадратичное отклонение и др.). Добавим элемент «Данные гистограммы», выбрав имя time_обработки, количество интервалов 50, начальный размер интервала 0,01.

time_обработки - Данные гистограммы

Имя:	<input type="text" value="time_обработки"/>	<input checked="" type="checkbox"/> Отображать имя	<input type="checkbox"/> Исключить
Видимость:	<input checked="" type="checkbox"/> да		
Значение:	<input type="text"/>		
Кол-во интервалов:	<input type="text" value="50"/>		
<input checked="" type="checkbox"/> Считать CDF <input type="checkbox"/> Вычислять процентили: Нижний: <input type="text" value="10"/> Верхний: <input type="text" value="10"/>			
<input checked="" type="checkbox"/> Вести журнал в базе данных Вести журнал выполнения модели			
▼ Диапазон значений			
<input checked="" type="radio"/> Выбирается автоматически <input type="radio"/> Фиксированный Нач. размер интервала: <input type="text" value="0.01"/>			

Добавим еще один элемент «Данные гистограммы» с именем ver_обработки, количеством интервалов 50 и начальным размером интервала 0,01.

ver_obrabortki - Данные гистограммы

Имя: Отображать имя Исключить

Видимость: да

Значение:

Кол-во интервалов:

Считать CDF

Вычислять процентили: Нижний: Верхний:

Вести журнал в базе данных

[Вести журнал выполнения модели](#)

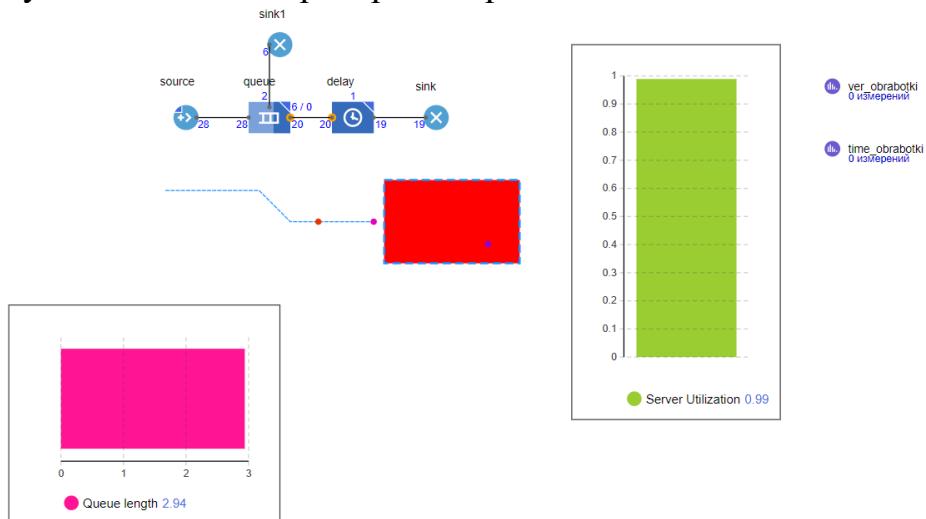
▼ Диапазон значений

Выбирается автоматически

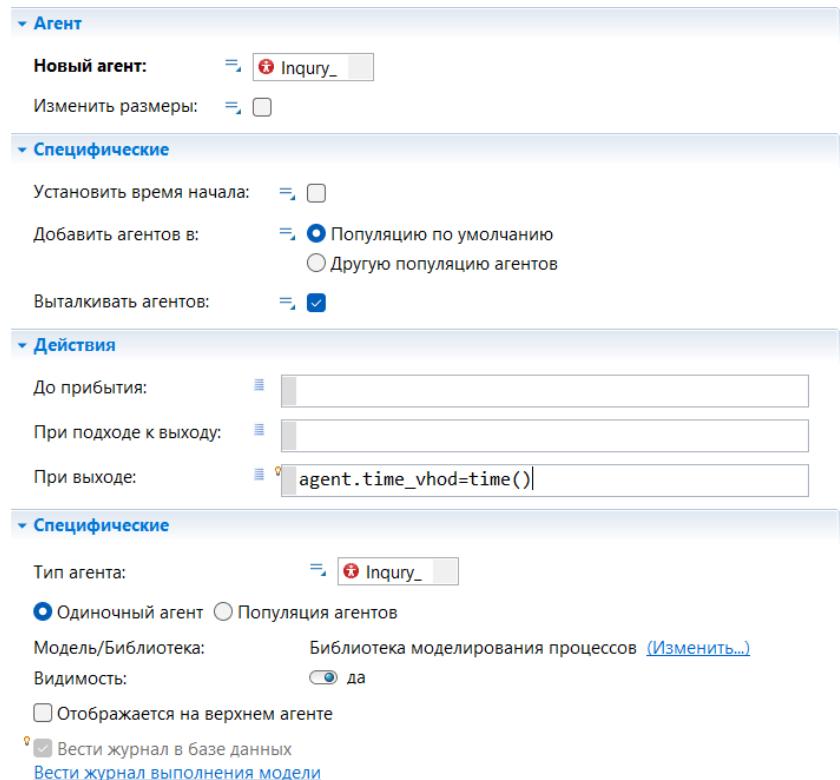
Фиксированный

Нач. размер интервала:

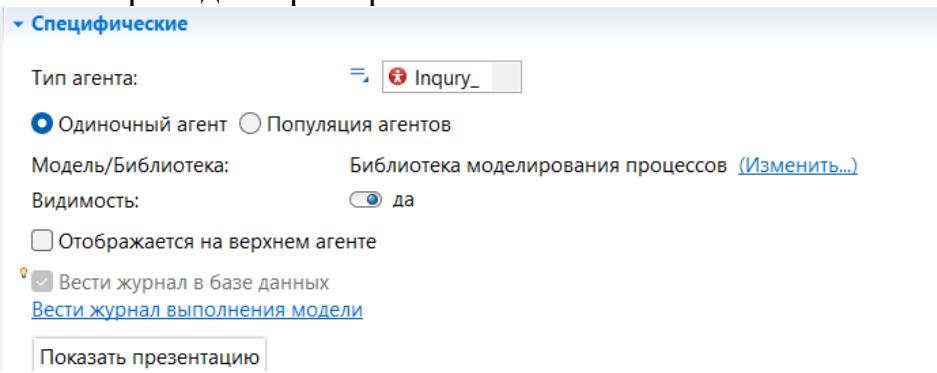
Запустим модель и проверим ее работоспособность:



Изменим в свойствах объекта Source тип заявки на Inquiry_.



Установим `agent.time_vhod=time()` в поле действие при выходе. Для остальных объектов поле «Тип агента» автоматически изменится на `Inquiry_`, поэтому ниже приведем пример объекта Sink.



Установим следующие действия при входе в объект Sink:

- `time_obrabotki.add(time() - agent.time_vhod)` – добавляется время обработки одного запроса в объект сбора данных гистограммы `time_obrabotki`. Данное время определяется как разность между текущим модельным временем `time()` и временем входа запроса в модель;
- `agent.col_vihod=sink.count(); agent.col_vhod=source.count()` – определяется количество запросов, вошедших в блок Sink и вышедших из блока Source соответственно;
- `ver_obrabotki.add(agent.col_vihod/agent.col_vhod)` – добавляется относительная доля обработанных запросов в объект сбора данных гистограммы `ver_obrabotki` при поступлении каждого обработанного запроса в блок Sink. На основе множества относительных долей обработанных запросов определяется математическое ожидание вероятности обработки запросов сервером.

Действия

При входе: `time_обработки.add(time()-agent.time_vhod);agent.col_vihod`

Изменим тип параметров на double.

col_vhod - Параметр

Имя: `col_vhod` Отображать имя Исключить

Видимость: да

Тип: `double`

Значение по умолчанию:

Массив системной динамики

col_vihod - Параметр

Имя: `col_vihod` Отображать имя Исключить

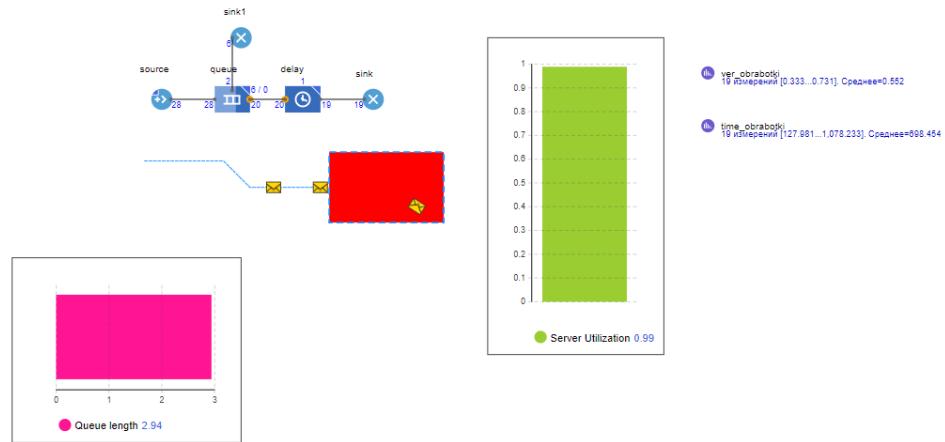
Видимость: да

Тип: `double`

Значение по умолчанию:

Массив системной динамики

Запустим модель:



Создадим параметр объекта Delay со следующими свойствами:

- имя: `time_mean` (среднее время);
- значение по умолчанию: 120;
- тип: double.

⌚ time_mean - Параметр

Имя:	time_mean	<input checked="" type="checkbox"/> Отображать имя	<input type="checkbox"/> Исключить
Видимость:	да		
Тип:	double		
Значение по умолчанию:	= 120		
<input type="checkbox"/> Массив системной динамики			

Изменение свойств объекта Delay:

⌚ delay - Delay

Имя:	delay	<input checked="" type="checkbox"/> Отображать имя	<input type="checkbox"/> Исключить
Тип задержки:	= <input checked="" type="radio"/> Определенное время <input type="radio"/> До вызова функции stopDelay()		
Время задержки:	= exponential(1/time_mean)	секунды	
Вместимость:	= 1		
Максимальная вместимость:	= <input type="checkbox"/>		
Место агентов:	= node		

Добавим бегунок, с помощью которого будем изменять среднее время обработки запросов объекта Delay.

⌚ slider - Бегунок

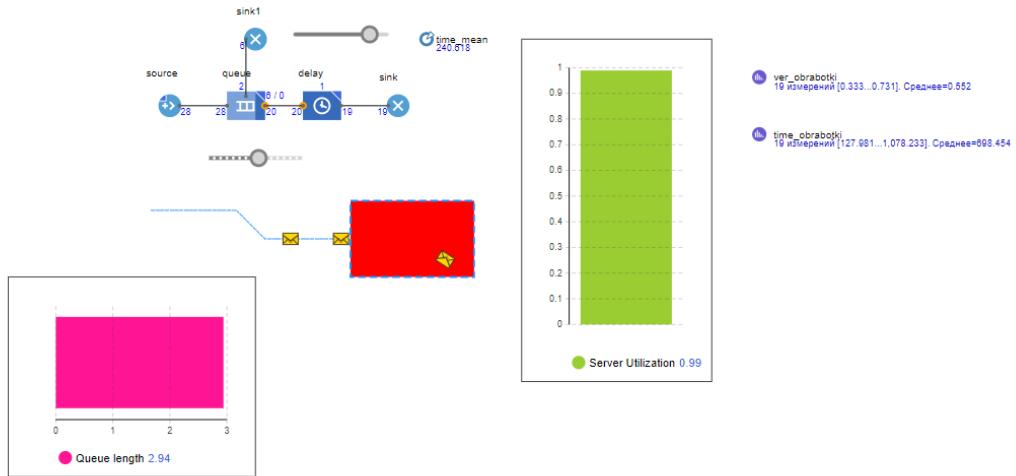
Имя:	slider	<input type="checkbox"/> Исключить
<input checked="" type="checkbox"/> Отображается на верхнем агенте	<input type="checkbox"/> Блокировать	
Ориентация:	<input type="radio"/> Вертикальная <input checked="" type="radio"/> Горизонтальная	
<input checked="" type="checkbox"/> Связь с:	= <input checked="" type="radio"/> time_mean	
Минимальное значение:	= 1	
Максимальное значение:	= 300	
Шаг:	= <input type="checkbox"/>	
Доступность:	= <input checked="" type="radio"/> да	
Добавить метки...		

Добавим бегунок под элементом Queue, что позволит изменять емкость буфера в ходе моделирования.

⌚ slider1 - Бегунок

Имя:	slider1	<input type="checkbox"/> Исключить
<input checked="" type="checkbox"/> Отображается на верхнем агенте	<input type="checkbox"/> Блокировать	
Ориентация:	<input type="radio"/> Вертикальная <input checked="" type="radio"/> Горизонтальная	
<input checked="" type="checkbox"/> Связь с:	= <input checked="" type="radio"/> queue	
Параметр:	= capacity	
Минимальное значение:	= 0	
Максимальное значение:	= 15	
Шаг:	= <input type="checkbox"/>	
Доступность:	= <input checked="" type="radio"/> да	
Добавить метки...		

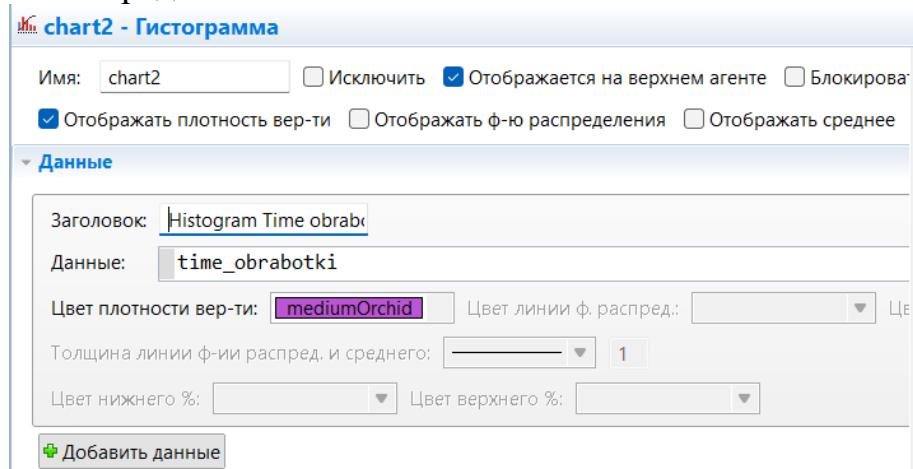
Теперь в процессе моделирования можно изменять емкость входного буфера и среднее время обработки запросов с помощью бегунков. Запустим модель:



Добавим на диаграмму рассматриваемого потока гистограмму, отображающую собранную временную статистику.

Изменим свойства гистограммы:

- заголовок: Histogram Time obrabotki;
- данные: time_обработки;
- отображать среднее.



Добавим еще одну гистограмму, отображающую собранную вероятностную статистику и изменим ее свойства:

- заголовок: Histogram Ver obrabotki;
- данные: ver_обработки;
- отображать среднее.

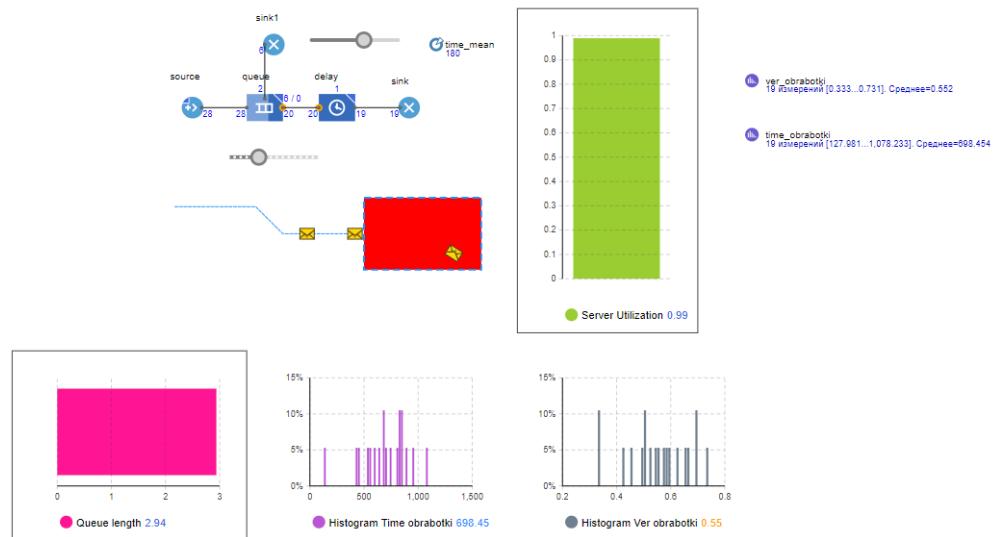
chart3 - Гистограмма

Имя: chart3 Исключить Отображается на верхнем агенте Блокировано
 Отображать плотность вер-ти Отображать ф-ю распределения Отображать среднее

Данные

Заголовок: Histogram Ver obrabotki
 Данные: ver_обработки
 Цвет плотности вер-ти: slateGray Цвет линии ф. распред.: ЦЕ
 Толщина линии ф-ии распред. и среднего: 1
 Цвет нижнего %: Цвет верхнего %:
+ Добавить данные

Запустим окончательную модель:



4. Полученные результаты и их анализ

Количество запросов	Макс длина очереди	Среднее значение интервала времени между заявками и п., мин	Среднее время обслуживания заявки, мин	Вероятность обслуживания	Мат ожидание времени обработки запросов, сек	Длина очереди, запросов	Загрузка сервера
4	5	3	1	0.92	60.35	0.09	0.2
4	5	3	2	0.92	140.73	0.28	0.37
4	5	3	3	0.88	221.82	0.46	0.56
10	5	3	2	0.9	179.34	0.45	0.74
4	5	2	2	0.92	118.08	0.32	0.51
4	5	1	2	0.35	662.42	4.05	0.99
4	1	3	2	0.08	125.61	0.17	0.37

Вероятность обслуживания 0.92, это означает, что почти все запросы обрабатываются. Если мы хотим увеличить этот показатель, то необходимо уменьшить время обработки запроса или обрабатывать запросы в многопоточном режиме. Если увеличить среднее время обслуживания заявки до 3 мин, это негативно скажется на работе сервера. Если уменьшить среднее значение интервала времени между заявками до 1 мин, то Сервер будет работать на пределе, будет постоянная очередь, высокое мат ожидание времени обработки запросов. Если увеличить число начальных запросов, это особо не скажется на работе сервера. При уменьшении макс. длины очереди количество необработанных заявок увеличивается.

5. Вывод

В работе построена имитационная модель работы сервера, позволяющая моделировать систему обслуживания с переменной длиной очереди и возможностью отказов. Определены вероятность и математическое ожидание времени обработки запросов