Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А.Бонч-Бруевича

Парамонов А.И., Маколкина М.А., Киричёк Р.В., Выборнова А.И.

Математические модели в сетях связи

Раздел: Математическое моделирование

Учебное пособие

Направление подготовки 09.03.04 Программная инженерия

основная профессиональная образовательная программа

Инфокоммуникационные технологии и системы связи

кафедра

СЕТЕЙ СВЯЗИ И ПЕРЕДАЧИ ДАННЫХ:

Санкт-Петербург

СПб ГУТ)))

2017

УДК 621.391

А.И. Парамонов, Маколкина М.А., Киричёк Р.В., Выборнова А.И. Математические модели в сетях связи. Раздел Лабораторный практикум. СПб: Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А.Бонч-Бруевича, 2017. – 58 с.

В пособии рассматриваются вопросы применения теоретических основ в практических методах моделирования телекоммуникационных систем и сетей в части параметров сетей связи и анализа трафика, овладение которыми будет полезно бакалаврам, занимающимся задачами проектирования и эксплуатации телекоммуникационных систем и сетей связи.

Пособие адресовано студентам, изучающим дисциплину “Математические модели в сетях связи ”. Направление подготовки 09.03.04 Программная инженерия.

Рекомендовано к печати редакционно-издательским советом университета, протокол № __/__ от __ января 2017г.

Рецензент: профессор, д.т.н. Кучерявый А.Е., зав кафедрой сетей связи и передачи данных СПбГУТ им. проф. М.А. Бонч-Бруевича.

СПбГУТ - Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича является старейшим и известнейшим вузом Российской Федерации, общепризнанным лидером российской высшей школы в области подготовки специалистов для отрасли связи и телекоммуникаций. В университете актуализированы, оптимизированы и аккредитованы основные направления и профили подготовки бакалавров, магистров и специалистов, а также выпускников колледжей для отраслей телекоммуникаций и IT – отрасли. Реализуется система непрерывного образования «лицей-колледж-университет» на основе интегрированных профильных образовательных программ.

Введение

Построение и эксплуатация сетей связи это комплексная задача, связанная с выбором логической и физической структуры сети, точек размещения оборудования, выбор способа и схемы построения линейных сооружений, изучения потребностей в услугах связи, прогнозировании спроса на них и выбора требуемых параметров оборудования и линий связи. Решение каждой из этих подзадач отражается на характеристиках сети связи. Назначение сети как технической системы состоит в выполнении работы по доставке трафика при предоставлении услуг связи. Качество предоставления услуг является основной характеристикой функционирования этой системы. Каждая из услуг может иметь специфические требования к качеству обслуживания, производимого ее пользователями трафика.

В этом пособии рассматриваются математические методы, применимые для оценки и анализа параметров сетей связи с коммутацией каналов и коммутацией пакетов. Приведенные в пособии материалы дают лишь необходимые сведения для пояснения приводимых понятий и методов. Для углубленного изучения теоретических основ и расширения области знаний о задачах связанных с расчетом параметров сетей связи следует обратиться к и рекомендуемой и дополнительной литературе.

В первой главе пособия приведены основные понятия, показатели качества и математические модели, применяемые при анализе технических решений и расчете канальной емкости в сетях с коммутацией каналов и коммутацией пакетов.

В последующих главах пособия приведены материалы, посвященные применению средств имитационного моделирования.

Имитационное моделирование представляет собой разновидность математического моделирования, которая позволяет получить численные оценки параметров исследуемых процессов и явлений. В отличие от аналитического моделирования, которое позволяет получить точные общие аналитические зависимости, имитационное моделирование позволяет получить лишь частные решения для конкретных условий. Однако, благодаря относительно низкой трудоемкости построения имитационных моделей и высокой скорости вычислений оно дает возможность получения достаточно точного описания тех же зависимостей. К тому же точные аналитические модели удается получить далеко не во всех случаях, и тогда имитационное моделирование становится, практически, единственным методом исследования. Следует отметить, что чаще всего аналитические методы моделирования и имитационное моделирование взаимно дополняют друг друга. Часто имитационные модели используются для проверки теоретических гипотез и моделей.

Технологии имитационного моделирования глубоко проникли в современную жизнь. Компьютерные игры, виртуальная реальность, дополненная реальность это лишь примеры имитационных моделей, которые встречаются на бытовом уровне. В области научных исследований,

разработок, производства и образования также используется множество, как правило, программных средств, которые можно отнести к системам имитационного моделирования. Системы имитационного моделирования позволяют, в ряде случаев, заменить натурное (физическое) моделирование и тем самым значительно облегчить решение задач проектирования.

Области применения имитационного моделирования весьма разнообразны, однако нас интересует вполне конкретная сфера деятельности, а именно построение телекоммуникационных систем и сетей. Круг решаемых в этой области задач тоже весьма широк: от уровня свойств физической среды, электрических сигналов и т.д. до уровня поведения пользователя (человека). В настоящее время нет единой системы имитационного моделирования, которая бы давала возможность глубоко и досконально моделировать процессы и явления, происходящие на всех этих уровнях.

В данном учебном пособии даются основные сведения, необходимые для построения имитационных моделей и анализе результатов моделирования с использованием двух систем имитационного моделирования AnyLogic и OMNeT++. По мнению автора, на момент написания пособия, эти системы являются наиболее проработанными и достаточно документированными. Обе системы доступны для свободного использования (или ограниченного использования, достаточного для решения рассматриваемых в настоящем пособии задач). Выбор двух систем объясняется их особенностями и особенностями задач в области теории и практики развития телекоммуникационных систем. Первая рассматриваемая система AnyLogic является системой имитационного моделирования общего назначения и не содержит готовых средств ориентированных именно на телекоммуникационные приложения. Однако, особенность ее реализации, выражающаяся в широких функциональных возможностях, простоте освоения, наглядности результатов и некоторых других, позволяют найти ей применение в задачах исследования процессов, описываемых моделями систем массового обслуживания. Такое описание часто используется при описании элементов телекоммуникационных систем и сетей связи.

Вторая система имитационного моделирования OMNeT++ ориентирована на построение моделей сетей связи и содержит широкий набор библиотечных элементов, позволяющий упростить задачи моделирования сложных телекоммуникационных систем. Ее использование позволяет достаточно простыми средствами создавать модели близкие по свойствам к реальным сетевым устройствам.

Выбор той или иной системы для решения конкретной задачи определяется исследователем (разработчиком модели), поэтому навыки работы с этими системами позволят достаточно свободно ориентироваться в выборе средств решения конкретных задач.

В пособии приводятся достаточно подробные описания процесса построения нескольких, наиболее наглядных, по мнению автора, имитационных моделей. Разумеется, материалы данного пособия не охватывают всего множества возможных вариантов построения моделей, а

дают лишь начальные сведения и позволяют получить начальные навыки работы, что может быть весьма полезно для дальнейшей углубленной работы с данными системами. Для углубленного изучения способов построения моделей и систем имитационного моделирования следует обратиться к документации этих систем и дополнительной литературе, в частности, приведенной в перечне рекомендуемой литературы.

Материалы данного учебного пособия могут быть полезны для самостоятельного изучения систем имитационного моделирования, а также при выполнении лабораторного практикума по специальности 11.04.02 в рамках учебного плана.

1 Сети связи: общие сведения и модели

1.1 Сети с коммутацией каналов

1.1.1 Структура сетей с коммутацией каналов и сетевое оборудование

Технология коммутации каналов (КК), традиционно, применялась как базовая технология построения сетей телефонной связи.

Основная особенность технологии коммутации каналов заключается в том, что единицей измерения ресурса сети связи является канал связи. При предоставлении услуги связи канал (или несколько каналов) предоставляются на все время предоставления услуги. Это время определяется длительностью занятия, которая в свою очередь, в сети телефонной связи, в наибольшей степени зависит от длительности разговора.

Ввиду того, что технология коммутации каналов постепенно вытесняется технологий с коммутацией пакетов. Вопросы построения сети с КК целесообразно рассмотреть с точки зрения разработанных ранее и существующих по сей день решений. Традиционно, оборудование и линейные сооружения, используемые для построения телефонных сетей связи, принято подразделять на первичные и вторичные сети.

Первичная сеть – это каналообразующая сеть, определяющая структуру сети связи.

Вторичная сеть – это сеть коммутации, включающая в себя узлы коммутации, подключенные к первичной сети.

Примером структуры сетей с коммутацией каналов является структура телефонной сети общего пользования (ТфОП) [1]. Основным коммутационным элементом ТфОП являются АТС (автоматические телефонные станции).

Исторически, городские сети телефонной связи строились с использованием архитектуры без узлообразования (при малом числе абонентов), в которой все АТС сети соединялись по принципу «каждая с каждой». Второй архитектурой является архитектура ТфОП с узлообразованием, в которую вводятся еще два вида узлов коммутации: узлы входящих сообщений и узлы исходящих сообщений (УВС и УИС). Пара таких узлов образует узловой район. Сети содержали узлы спецслужб (УСС), и междугородные автоматические телефонные станции (АМТС).

Все узлы связаны между собой соединительными линиями первичной сети, каждая из которых позволяет организовать некоторое количество независимых каналов (иногда эту группу каналов называют пучком соединительных линий).

С точки зрения локализации систем коммутации относительно населенных пунктов принято различать следующие уровни иерархии сетей: местный (городские и сельские сети), зоновый (группа населенных пунктов на территории одной зоны нумерации), междугородный и международный.

В процессе модернизации сетей связи, связанной с распространение цифровых систем коммутации появились новые и изменились некоторые

общепринятые названия узлов сети [1]. Районные АТС (РАТС) в цифровой сети обычно именуются ОПС (Опорная станция), узлы УИС и УВС – ТС (транзитная станция) и ОПТС (Опорно-транзитная станция).

Основная задача проектирования сети связи – обеспечение требований к качеству предоставления услуг и надежности (устойчивости), которая решается выбором необходимого объема ресурсов: числа каналов, узлов коммутации и структуры их связей.

1.1.2 Понятие абонентской нагрузки в сетях с коммутацией каналов

При обслуживании потока вызовов коммутационной системой каждый вызов занимает выход системы на некоторый промежуток времени. Если, например, выход одновременно обслуживает только один вызов, то загрузка выхода может характеризоваться суммарным временем обслуживания всех вызовов, а коэффициент полезного действия или использование выхода можно оценивать отношением суммарного времени обслуживания всех вызовов ко времени действия выхода. В теории телетрафика суммарное время обслуживания вызовов принято называть нагрузкой.

Следует различать нагрузки: поступающую, обслуженную и потерянную [2].

Обслуженная коммутационной системой за промежуток времени $[t_1, t_2)$ нагрузка $Y_0(t_1, t_2)$ представляет собой сумму времен занятия всех выходов коммутационной системы, обслуживающей поступающий на ее входы поток вызовов за рассматриваемый промежуток времени [2].

Обслуженная за интервал времени нагрузка измеряется в часо-занятиях или минуто-занятиях.

Интенсивность нагрузки – нагрузка за единицу времени, (в телефонии, обычно за 1 ч). За единицу измерения интенсивности нагрузки принят эрланг (Эрл) по имени А. К. Эрланга. Один эрланг представляет собой нагрузку в одно часо-занятие за 1 ч.

В теории и практике расчета пропускной способности коммутационных систем обычно используется средняя интенсивность нагрузки, которую для краткости называют интенсивностью нагрузки.

Под поступающей на коммутационную систему за промежуток времени $[t_1, t_2)$ нагрузкой $Y(t_1, t_2)$ понимается такая нагрузка, которая была бы обслужена коммутационной системой за рассматриваемый промежуток времени, если бы каждому поступающему вызову тотчас было предоставлено соединение со свободным выходом.

Потерянная коммутационной системой в течение промежутка времени $[t_1, t_2)$ нагрузка $Y_п(t_1, t_2)$ представляет собой разность между поступающей и обслуженной нагрузками за рассматриваемый промежуток времени.

Час наибольшей нагрузки. Интенсивность нагрузки в сетях связи не постоянна во времени, поэтому для практических расчетов часто используют

ее наибольшее ожидаемое значение. В качестве такого значения выбирают значение в час наибольшей нагрузки ЧНН. Под часом наибольшей нагрузки понимают интервал времени, продолжительностью 60 минут, в течение которого нагрузка максимальна в среднем, за достаточно продолжительный интервал времени.

Удельная абонентская нагрузка. В практических задачах часто используют понятие удельной абонентской нагрузки. Это интенсивность нагрузки, создаваемой одним абонентом, в среднем в достаточно большой группе абонентов. Она также измеряется в эрлангах (Эрл).

Связь интенсивности нагрузки, интенсивности вызовов и среднего времени занятия.

$y = c\bar{t}$ Эрл, (1.1)

где c – интенсивность вызовов (вызовов в час),

$\bar{t}$ – среднее время занятия (час).

Значение удельной абонентской нагрузки может быть специфично для различных абонентов. Иногда вводят условную классификацию абонентов, например, квартирные абоненты (квартирный сектор), бизнес абоненты (учрежденческий сектор), таксофоны и т.д. Также имеет место специфика и для абонентов сетей фиксированной и подвижной связи.

В качестве типового примера интенсивности удельной абонентской нагрузки для сетей фиксированной связи является значение 0,1 Эрл, а для сетей подвижной связи 0,03 Эрл. Более детальные рекомендации по выбору значений удельной нагрузки даны в [10].

1.1.3 Показатели качества функционирования сети телефонной связи

Показателями качества услуги телефонной связи являются: качество передачи речи и вероятность отказа установления соединения (коэффициент потерь вызовов и время установления соединения).

Качество передачи речи. Для оценки *качества передачи речи* используют методы экспертных оценок используя рейтинговые оценки (MOS – Mean Opinion Score, R – фактор). Усредненная экспертная оценка характеризует субъективное качество восприятия услуги (QoE – QualitY of Experience). При использовании стандартных технических средств, отклонение от нормы качества передачи речи происходит только при их неисправности. При проектировании сети связи, как правило, этот показатель не оценивается.

Коэффициент потерь вызовов. Коэффициент потерь вызовов характеризует вероятность отказа установления соединения из-за нехватки ресурсов сети, т.е. нехватки каналов.

При измерениях *коэффициент потерь* оценивается средней величиной вероятности потерь

$$\bar{p} = \frac{n_f}{n_0} \quad (1.2)$$

где n_0 – общее число попыток вызовов,

n_f- количество отказов.

Коэффициент потерь, обычно приводится в % или ‰ (промилле, 1‰=0,1%=0,001).

Для услуги телефонной связи установлены отраслевые нормативы (Приказ №113 Минкомсвязи РФ от 27.09.2007 г.) [3].

Время установления соединения. Время установления соединения условно подразделяется на несколько этапов:

-задержка получения ответа станции (время отклика узла связи, время с момента снятия телефонной трубки до получения сигнала «ответ станции»);

-время установления соединения (время с момента окончания набора номера вызываемого абонента до момента получения сигнала «посылка вызова»);

-время выполнения соединения (время с момента снятия вызываемым абонентом трубки до момента установления разговорного соединения);

-время разъединения (время с момента посылки сигнала разъединения до момента полного освобождения ресурсов).

Эти параметры также нормированы (Приказ №113 Минкомсвязи РФ от 27.09.2007 г.) [3].

1.1.4 Обеспечение качества функционирования, математические модели

Основной задачей при проектировании сети связи является обеспечение баланса между абонентским трафиком (спросом на услуги), объемом ресурсов сети (количества каналов) и качеством предоставления услуги (коэффициентом потерь вызовов).

При решении данной задачи рассматривают два уровня (модели ВОС): сетевой и канальный.

Сетевой уровень. *На сетевом уровне* рассматриваются маршруты пропуска трафика в сети. Для этого сеть связи удобно описать моделью графа [5], в которой узлы сети (АТС и узлы связи) соответствуют вершинам графа, а линии связи дугам графа.

Каждая из дуг графа характеризуется интенсивностью нагрузки $y_{i,j}$. Значения интенсивностей нагрузки определяются распределением трафика в сети связи между оконечными узлами (АТС).

Интенсивность нагрузки, производимая абонентами АТС *i* зависит от удельной интенсивности нагрузки и количества абонентов. Например, если для абонентов введена классификация по *k* секторам

$$y_i = n_i^{(кв)} y_0^{(кв)} + n_i^{(уч)} y_0^{(уч)} + \cdots + n_i^{(k)} y_0^{(k)} \quad (1.3)$$

$$n_i^{(кв)} + n_i^{(уч)} + \cdots + n_i^{(k)} = n_i$$

где $n_i^{(k)}$ - количество абонентов k-го сектора, включенных в данный узел;

n_i - общее количество абонентов, включенных в данный узел.

Доля трафика, производимого абонентами узла *i*, направляемая на узел *j* определяется коэффициентами распределения k_{ij}, *j*=1..*d*, где *d* – количество направлений связи. В данном случае рассматриваются только оконечные узлы или узлы, связывающие данную сеть с другой сетью (например, АМТС, УСС).

Таким образом, данная модель сети позволяет создать описание сети в виде таблицы нагрузок между оконечными узлами связи, в которой

$$Y_{i,j} = y_i^{(\text{исх})} k_{ij} \qquad (1.4)$$

где $y_i^{(исх)}$ интенсивность исходящей нагрузки, обычно, в расчетах принимают

$$y_i^{(исх)} = \frac{1}{2} y_i . \qquad (1.5)$$

Канальный уровень. На данном уровне требуется оценить необходимое количество каналов, образуемых линиями связи между узлами сети. Для этого необходимо знать интенсивность трафика, обслуживаемого линиями связи, которая получена на предыдущем этапе, и норматив на коэффициент потерь вызовов, который задается существующими отраслевыми документами или техническим заданием.

Задача решается методами теории телетрафика (теории массового обслуживания). Поток вызовов от абонентов рассматривается как случайный поток заявок на обслуживание, каналы рассматриваются как обслуживающие устройства, которые занимаются входящими вызовами на некоторое случайное время равное времени занятия (приблизительно времени разговора). Модель такой системы должна описывать взаимодействие двух случайных процессов: процесса поступления заявок (и занятия каналов) и процесса освобождения каналов и называется системой массового обслуживания (СМО). При поступлении заявки в момент, когда все каналы заняты, заявка получает отказ (теряется). Такая дисциплина обслуживания называется дисциплина обслуживания с потерями (отказами). Цель построения математической модели в том, чтобы связать интенсивность абонентского трафика, количество каналов и вероятность потерь (отказов).

Подробно модели СМО описаны, например, в [2, 8, 11]. Здесь рассмотрим лишь основные свойства некоторых из них.

Модели СМО связывают показатели качества с параметрами потока заявок и характеристиками процесса их обслуживания. Они разработаны для потоков и процессов, имеющих определенные свойства. Поэтому, выбор той или иной модели зависит от свойств тех процессов, которые она должна описывать.

1. Модель потока заявок. Модель потока вызовов от абонентов (потока заявок) в телефонии, обычно, описывают моделью простейшего потока.

Простейший поток заявок представляет собой временную последовательность независимых случайных событий. Под событием понимается поступление вызова. Этот поток имеет три свойства:

-стационарный,

-ординарный,

-без последействия.

Стационарность потока – вероятность поступления k заявок за интервал времени (τ) зависит только от величины этого интервала и не зависит от того, где на оси времени он выбран.

Ординарность потока – вероятность поступления двух и более заявок за интервал времени, стремящийся к нулю, тоже стремится к нулю.

Отсутствие последействия – независимость настоящего от прошлого, т.е. процесс поступления заявок не зависит ни от процесса поступления до настоящего момента, ни от состояния системы обслуживания.

Для простейшего потока вероятность поступления k заявок за интервал времени *t* является случайной величиной, имеющей распределение Пуассона

$$p_k = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \quad (1.6)$$

где λ - интенсивность потока (заявок/ед. времени).

Интервалы времени между заявками в таком потоке также случайны и имеют экспоненциальное распределение вероятности

$$f(x) = 1 - e^{-\lambda x} \quad (1.7)$$

2. Модель процесса обслуживания. В этой модели предполагается, что обслуживающее устройство (канал) занимается заявкой на случайное время, которое имеет экспоненциальное распределение вероятности. Такое предположение достаточно точно описывает реальные телефонные вызовы.

$$f(x) = 1 - e^{-\mu x} \quad (1.8)$$

где μ - интенсивность обслуживания (заявок/ед. времени).

$\bar{t} = \frac{1}{\mu}$ - среднее время обслуживания (занятия канала).

3. Вероятность отказов (потерь вызовов). В условиях описанных моделей вероятность отказов определяется как первая формула Эрланга (В – формула Эрланга)

$$p = \frac{\frac{y^v}{v!}}{\sum_{j=0}^{v} \frac{y^j}{j!}} \quad (1.9)$$

Где y – это интенсивность нагрузки (Эрл),

v – количество каналов.

Используя выражение (9) для полученной выше таблицы 5 можно вычислить для каждой из линий связи необходимое количество каналов, при заданной вероятности потерь.

Таким образом, алгоритм расчета числа каналов для сети с коммутацией каналов можно определить следующим образом:

1. Вычислить интенсивность абонентской нагрузки, производимой в оконечных узлах связи.
2. Вычислить таблицу распределения нагрузки между оконечными узлами связи.
3. Вычислить таблицу распределения нагрузки по имеющимся линиям связи.
4. Для полученных значений нагрузки и заданной величины потерь вычислить требуемое число каналов для каждой из линий связи.

1.1.5 Контрольные вопросы по изучаемой теме

1. Структура сетей с коммутацией каналов, узлообразование.
2. Иерархия местной, междугородной, зоновой и международной сетей.
3. Понятие абонентской нагрузки, обслуженная, поступающая, потерянная нагрузка.
4. Интенсивность абонентской нагрузки, удельная абонентская нагрузка, час наибольшей нагрузки, типовые значения удельной абонентской нагрузки, принятые в телефонии единицы измерения.
5. Показатели качества функционирования сети телефонной связи, связь с качеством восприятия услуги.
6. Математические модели сетевого уровня.
7. Математические модели канального уровня.
8. Модель потока вызовов (заявок), свойства потока.
9. Модель процесса обслуживания вызовов (заявок).
10. Модель оценки вероятности отказов. Первая формула Эрланга.

1.2 Сети с коммутацией пакетов

1.2.1 Структура сетей с коммутацией пакетов

Современные сети связи, как правило, строятся с применением технологии с коммутацией пакетов.

Основная особенность технологии коммутации пакетов заключается в том, что единицей измерения ресурса сети связи является время передачи пакета. Вся передаваемая в сети информация представлена пакетами данных. Предоставление услуги связи осуществляется передачей пакетов данных через сеть связи.

Основными элементами сети связи являются узлы связи, осуществляющие маршрутизацию (коммутацию) пакетов, передавая их в соответствии с адресной информацией и правилами маршрутизации.

Структура сети с пакетной коммутацией можно условно выделить три уровня: уровень доступа, уровень распределения (агрегирования) трафика и уровень ядра сети.

Сети принято классифицировать согласно масштабу на: WAN – глобальный, MAN – уровня города, LAN – уровня организации, PAN – персональные, BAN – нательные.

1.2.2 Трафик и услуги в сетях передачи данных, IP телефония

Трафик в сети передачи данных представляет собой поток пакетов, его обычно описывают моделью случайного потока заявок. Под заявкой понимают поступление пакета данных.

Трафик характеризуется такими параметрами как:

-интенсивность поступления пакетов λ (пакетов/с)

-и средняя длина пакета $\bar{L}$ (бит, байт)

-интенсивность трафика $a = \lambda\bar{L}$ (бит/с).

По аналогии с сетями с коммутацией каналов, трафик в сети с коммутацией пакетов также создает нагрузку на сеть связи. Эта нагрузка определяется той работой, которую сеть выполняет при передаче пакетов данных. Обслуживание пакетов в узлах связи заключается в их коммутации, т.е. направлении на определенный выход (порт) согласно правилам маршрутизации, а также в передаче пакета по линии связи. Время, необходимое для передачи пакета (а также время коммутации) определяют пропускную способность сети связи и зависят от параметров используемого оборудования. Обычно, пакет данных передается модемом или иным устройством узла связи последовательным способом. Время передачи пакета определяется его размером (количеством передаваемых бит) и скоростью передачи, которая определяется параметрами модема и линии связи. Пакеты, поступающие в то время, когда передается очередной пакет, помещаются в буферную память узла связи (очередь) где ожидают своей очереди на передачу. Если количество ожидающих в очереди пакетов достигает некоторой заданной величины, то вновь поступающие пакеты теряются

(стираются). В этом случае имеет место комбинированная дисциплина обслуживания – с ожиданием и потерями (отказами).

Для предоставления услуги связи в сети с коммутацией пакетов требуется доставка пакетов, по крайней мере, между двумя узлами связи. При этом доставка может производиться по маршруту, содержащему несколько участков (узлов и линий связи). Как было отмечено выше, доставка пакета на каждом из этих участков требует затрат времени которые, в общем случае, определяются временем распространения сигнала, временем передачи пакета по линии связи и временем ожидания пакета в очереди в узле связи

$$T_D = T_{pr} + T_{tr} + T_W \quad (1.10)$$

где T_{pr} – время распространения сигнала;

T_{tr} – время передачи пакета;

T_W – время ожидания в узле связи.

Время распространения сигнала в большинстве систем связи определяется временем распространения электрического (электромагнитного поля) или оптического сигнала. С достаточной для практических приложений точностью скорость распространения сигнала описывают величиной близкой к скорости распространения света. В практических расчетах проводных сетей принимают время распространение равное 5 мкс/км.

$$T_{pr} = 5d \text{ мкс}, \quad (1.11)$$

где d – расстояние км.

В сетях относительно малой протяженности (PAN, LAN, MAN) этой составляющей часто пренебрегают, т.к. задержка распространения на малых расстояниях пренебрежимо мала по сравнению с задержкой на передачу и ожидание в узле связи. При построении сетей большой протяженности время распространения может составлять существенную долю задержки. Например, расстояние от Калининграда до Владивостока, по прямой составляет примерно 7500 км, задержка распространения составит 37,5 мс, что может превышать другие составляющие. Другим примером может служить линия связи с использование искусственного спутника земли (ИСЗ), протяженность которой может превышать 35000 км (высота орбиты, между наземными станциями более 70000 км), а время распространения более 200 мс.

Время передачи пакета определяется скоростью передачи данных по линии связи и длиной пакета

$$T_{tr} = \frac{L}{b} \quad (1.12)$$

где b – скорость передачи данных (бит/с),

L – размер пакета данных (бит).

Из сказанного следует, что время распространения – величина постоянная, по крайней мере, с достаточной для практических расчетов точностью. Если рассматривать стабильное состояние канала связи, при котором скорость передачи данных постоянна (обычно это допустимо для проводных технологий, но не всегда допустимо для беспроводных) то время передачи будет зависеть только от длины пакета. Длина пакета может быть

различна для различных услуг. Поэтому, в общем случае, время передачи случайно.

Время ожидания в узле связи это время, которое пакет проводит в очереди (буферной памяти) ожидая передачи. Оно зависит от интенсивности трафика, времени передачи и других параметров, методы его оценки будут рассмотрены ниже. Это время также случайно.

Услуги современных сетей связи можно условно классифицировать на три группы: интерактивные, потоковые и фоновые.

Интерактивные услуги предполагают, что пользователь совершает активные действия и ожидает реакцию на них. Качество восприятия услуги зависит и задержки реакции на действие пользователя. В качестве примера такой услуги можно привести WEB-серфинг. Пользователь «кликает» ссылку на просматриваемой странице и ожидает загрузки очередной страницы, чем быстрее он получает данные, тем комфортнее воспринимается услуга.

Потоковые услуги предполагают, что пользователь передает или получает регулярный пакетов. Качество услуги определяется не только задержкой доставки пакета, но и изменением задержки для различных пакетов. В качестве примера можно привести услугу потокового видео (IPTV). Кодек на стороне отправителя формирует последовательность пакетов, каждый из пакетов должен быть доставлена получателю не позднее определенного времени, иначе произойдет остановка воспроизведения видео. Интервалы времени между пакетами на стороне получателя не должны значительно отличаться от интервалов на стороне отправителя. Это отличие определяется случайным характером задержки, т.к. каждый из пакетов может быть доставлен с разной величиной задержки.

Фоновые услуги не связаны с жесткими временными ограничениями. Примерами таких услуг могут быть электронная почта, загрузка файлов и др. В этом случае качество восприятия услуги определяется, в основном, временем необходимым для выполнения работы по доставке данных текстового сообщения или файла, т.е. скоростью передачи данных.

Следует заметить, что трафик, создается пользователем в сети не постоянно, а в течение времени предоставления услуги (время разговора, просмотра видео и т.д.). Назовем это время сессией. Таким образом, для различных услуг будут различные характеристики интенсивности и продолжительности сессий. Например, для услуги IP телефонии (передачи речи VoIP) следует ожидать, что сессии будут аналогичны телефонным вызовам, производимым в сети с коммутацией каналов. Описать поток сессий можно интенсивностью нагрузки, аналогично описанию для сети с коммутацией каналов (1.1). Тогда интенсивность нагрузки для сессий будет равна

$$s = c\bar{t} \qquad (1.13)$$

где с – интенсивность сессий (сессий/час),

$\bar{t}$- средняя продолжительность сессии (час)

Таким образом, общее время доставки пакета также является случайной величиной, зависящей от параметров сети и трафика. Иными словами, пакет доставляется получателю с некоторой случайной задержкой, кроме этого с некоторой вероятностью он может быть потерян на маршруте и вообще не будет доставлен получателю. Вполне логично ожидать, что оба эти явления влияют на качество восприятия услуги связи. С целью обеспечения и поддержания качества услуг, введены показатели качества функционирования сети, которые определяют ряд параметров в наибольшей степени влияющих на качество восприятия услуг связи.

1.2.3 Показатели качества функционирования сетей передачи данных

Исходя из свойств предоставляемых услуг Международным союзом электросвязи (ITU-T) в рекомендациях Y.1540 [6] и Y.1541 [7] был определен перечень параметров (показателей) качества функционирования сетей связи. Эти такие параметры как:

-средняя задержка доставки пакета данных (IPTD),

-вариация задержки доставки данных (джиттер) (IPDV),

-коэффициент потерь пакетов (IPLR),

-коэффициент ошибок (IPER).

Этот перечень показателей также закреплен отраслевым нормативным документом (Приказ №113 Минкомсвязи РФ от 2007 г. [3]).

1.2.4 Трафик передачи данных. Математические модели

Как и в случае сети с коммутацией каналов, основной задачей при проектировании сети передачи данных является обеспечение баланса между трафиком (спросом на услуги), объемом ресурсов сети (пропускной способностью) и качеством предоставления услуги (параметрами функционирования).

При решении данной задачи рассматривают два уровня (модели ВОС): сетевой и канальный.

Сетевой уровень. *На сетевом уровне* рассматриваются маршруты пропуска трафика в сети. Для этого сеть связи удобно описать моделью графа [5] (в данном случае неориентированного), в которой узлы сети (маршрутизаторы) соответствуют вершинам графа, а линии связи дугам графа.

Каждое из ребер графа характеризуется интенсивностью нагрузки $s_{i,j}$. Значения интенсивностей нагрузки определяются распределением трафика в сети связи между оконечными узлами (узлами доступа).

Интенсивность нагрузки, производимая пользователями, включенными в узел доступа *i,* зависит от спроса на услуги и набора предоставляемых услуг.

$$s_i = n_i^{(1)} s_0^{(1)} + n_i^{(2)} s_0^{(2)} + \cdots + n_i^{(m)} s_0^{(m)} \qquad (1.14)$$

где $n_i^{(j)}$ - количество пользователей *j*-й услуги, включенных в данный узел;

n_i - общее количество пользователей, включенных в данный узел.

m – количество предоставляемых услуг.

$s_0^{(j)}$ - удельная интенсивность сессий j-й услуги.

Доля трафика, производимого абонентами узла i, направляемая на узел j определяется коэффициентами распределения k_{ij}, j=1..d, где d – количество направлений связи. В данном случае рассматриваются только оконечные узлы или узлы, связывающие данную сеть с другой сетью.

Таким образом, данная модель сети позволяет создать описание сети в виде таблицы нагрузок между оконечными узлами связи, в которой

$$S_{i,j} = s_i k_{ij} \tag{1.15}$$

Канальный уровень. На данном уровне требуется оценить необходимую пропускную способность, линий связи между узлами сети. Для этого необходимо знать интенсивность трафика, обслуживаемого линиями связи, которая получена на предыдущем этапе, и нормативы на качество обслуживания.

Задача решается методами теории телетрафика (теории массового обслуживания). На предыдущем шаге мы получили интенсивности нагрузки сессий $s_{i,j}$. Далее от полученных значений нагрузки необходимо перейти к интенсивности трафика (бит/с).

Если нам известны данные о предоставляемых услугах, интенсивности трафика, производимого этими услугами $a_0^{(j)}$ – удельная интенсивность трафика, производимого j-й услугой (во время сессии), то общая интенсивность трафика может быть получена как

$$a_{ij} = v_{ij}\left(\eta_1 a_0^{(1)} + \eta_2 a_0^{(2)} + \cdots + \eta_r a_0^{(r)}\right) \tag{1.16}$$

где v_{ij} – число сессий, которые требуется обслужить,

$a_0^{(j)}$ – удельная интенсивность трафика j-й услуги (бит/с),

η_j – доля нагрузки сессий, производимой j-й услугой.

Значение v_{ij}, фактически, означает число сессий, обслуживание которого должна обеспечивать линия связи. Оно может быть определено аналогично тому, как определяется число необходимых каналов в сети с коммутацией каналов, т.е. с помощью 1 й формулы Эрланга (см. п.1). В результате чего может быть получена таблица распределения трафика по линиям связи.

Данные из этой таблицы являются основой для следующего этапа расчета.

Поток пакетов рассматривается как случайный поток заявок на обслуживание, линия связи рассматриваются как обслуживающее устройство, которое занимаются передаваемыми пакетами на некоторое случайное время равное времени передачи пакета. Модель такой системы

должна описывать взаимодействие двух случайных процессов: процесса поступления заявок и процесса освобождения т.е. является моделью системы массового обслуживания (СМО). При поступлении заявки в момент, когда устройство занято, заявка ставится на ожидание. Когда число ожидающих заявок достигло некоторого заданного значения (размера буфера) заявка теряется. Такая дисциплина обслуживания называется комбинированной дисциплиной обслуживания (с ожиданием и отказами). Цель построения математической модели в том, чтобы связать интенсивность трафика, пропускную способность канала со временем ожидания и вероятностью потерь (отказов).

Подробно модели СМО описаны, например, в [2,8,9]. Здесь рассмотрим лишь основные свойства некоторых из них.

Модели СМО связывают показатели качества с параметрами потока заявок и характеристиками процесса их обслуживания. Они разработаны для потоков и процессов, имеющих определенные свойства. Поэтому, выбор той или иной модели зависит от свойств тех процессов, которые она должна описывать.

1. Модель потока заявок. Модель потока пакетов (потока заявок) в сетях передачи данных, обычно, описывают моделью случайного потока.

Из теории телетрафика (массового обслуживания) известны решения для некоторых видов случайных потоков и моделей СМО.

Наибольшее число известных решений связано с моделью простейшего потока.

Простейший поток заявок представляет собой временную последовательность независимых случайных событий. Под событием понимается поступление вызова. Этот поток имеет три свойства:

-стационарный,

-ординарный,

-без последействия.

Стационарность потока – вероятность поступления k заявок за интервал времени (τ) зависит только от величины этого интервала и не зависит от того, где на оси времени он выбран.

Ординарность потока – вероятность поступления двух и более заявок за интервал времени, стремящийся к нулю, тоже стремится к нулю.

Отсутствие последействия – независимость настоящего от прошлого, т.е. процесс поступления заявок не зависит ни от процесса поступления до настоящего момента ни от состояния системы обслуживания.

Для простейшего потока вероятность поступления k заявок за интервал времени *t* является случайной величиной, имеющей распределение Пуассона

$$p_k = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \qquad (1.17)$$

где λ - интенсивность потока (заявок/ед. времени).

Интервалы времени между заявками в таком потоке также случайны и имеют экспоненциальное распределение вероятности

$$f(x) = 1 - e^{-\lambda x} \qquad (1.18)$$

2. Модель процесса обслуживания. В этой модели предполагается, что обслуживающее устройство (канал) занимается заявкой на случайное время.

3. Время задержки пакета (в очереди на обслуживание). В условиях описанных моделей среднее время задержки пакета на участке сети определяется формулой Поячека-Хинчина [2, 8, 11]

$$T = \frac{\rho\bar{t}}{2(1-\rho)}\left(1 + \frac{\sigma^2}{\bar{t}^2}\right) + \bar{t} \qquad (1.19)$$

где $\rho = a\bar{t}$

а – интенсивность пакетов,

$\bar{t} = \frac{\bar{L}}{b}$- среднее время обслуживания пакета;

σ^2- дисперсия времени обслуживания;

$\bar{L}$- средняя длина пакета (бит);

b – скорость передачи (бит/с).

В частных случаях, например, когда время обслуживания имеет экспоненциальное распределение, то $\frac{\sigma^2}{\bar{t}^2} = 1$, тогда

$$T_{exp} = \frac{\bar{t}}{1-\rho} \qquad (1.20)$$

Когда время обслуживания постоянно, то $\sigma^2 = 0$

$$T_D = \frac{\rho\bar{t}}{2(1-\rho)} + \bar{t} \qquad (1.21)$$

Если свойства потока отличаются от простейшего, то может быть применена приближенная формула [8]

$$T_G = \frac{\rho\bar{t}}{2(1-\rho)}\left(\frac{\sigma_a^2 + \sigma_s^2}{\bar{t}^2}\right)\left(\frac{\bar{t}^2 + \sigma_s^2}{\bar{a}^2 + \sigma_s^2}\right) + \bar{t} \qquad (1.22)$$

где σ_a^2, σ_s^2 - дисперсии интервалов времени между пакетами и времени обслуживания, соответственно, $\bar{a}$ – среднее значение интервала между пакетами, $\bar{t}$ – среднее время обслуживания.

3. Вероятность отказов (потерь пакетов).

В общем случае для оценки вероятности потерь может быть использована приближенная формула [9]

$$p = \frac{1-\rho}{1-\rho^{\frac{2}{C_a^2+C_s^2}n_b+1}}\rho^{\frac{2}{C_a^2+C_s^2}n_b}, \qquad (1.23)$$

где C_a^2 и C_s^2 – квадратичные коэффициенты вариации соответственно распределений входящего потока и времени обслуживания, n_b – размер буфера, ρ – загрузка системы.

Таким образом, алгоритм расчета пропускной способности для сети с коммутацией пакетов можно определить следующим образом:

1. Вычислить интенсивность нагрузки, производимой в оконечных узлах связи (сессий).
2. Вычислить таблицу распределения нагрузки между оконечными узлами связи.
3. Вычислить таблицу распределения нагрузки по имеющимся линиям связи.
4. Вычислить таблицу распределения интенсивностей трафика по имеющимся линиям связи.
5. Для полученных значений интенсивностей трафика и заданной величины задержки вычислить требуемые пропускные способности линий связи.
6. Проверить выполнение норм по коэффициенту потерь пакетов.

2 Построение имитационных моделей в системе AnyLogic

2.1 Краткое описание системы имитационного моделирования AnyLogic

Система имитационного моделирования AnyLogic поддерживает различные подходы к созданию имитационных моделей: процессно-ориентированный (дискретно-событийный), системно динамический и агентный, а также любую их комбинацию. Гибкость языка моделирования, предоставляемого AnyLogic, позволяет учитывать различные аспекты моделируемой системы с различным уровнем детализации. Система имеет графический интерфейс, инструменты и библиотеки, которые упрощают процесс создания моделей для широко круга задач.

Графическая среда разработки значительно ускоряет процесс создания моделей. Возможность создания библиотек позволяет разработчику многократно использовать уже написанные модули. AnyLogic поддерживает как дискретный, так и непрерывный подходы.

Основой системы моделирования является Java платформа, которая обеспечивает расширяемость моделей за счет программирования на Java, создания пользовательских библиотек и работы с базами данных.

Система включает в себя возможность создания интерактивной анимации для улучшения наглядности моделей.

Система разработана на языке программирования Java, поэтому она является мультиплатформенным программным продуктом. Среда разработки и модели, созданные в ней, могут работать на Windows, Mac OS и Linux.

Широкие возможности данной системы делают ее привелекательной для построения моделей различных процессов и явлений.

С точки зрения решения задач моделирования процессов, происходящих в сетях связи, данная система также может быть эффективно использована. Модели, которые будут рассмотрены в данном пособии, являются дискретно-событийными. Теоретически, в системе AnyLogic можно описать «с нуля» достаточно сложные процессы. Однако, на практике целесообразная область использования той или иной системы определяется трудоемкостью построения моделей. Поэтому разработчик модели должен произвести выбор в пользу той или иной системы имитационного моделирования исходя из конкретной задачи и возможностей доступных ему систем. По мнению автора настоящего пособия, в области моделирования телекоммуникационных систем и сетей использование данной системы целесообразно при построении моделей устройств и систем относительно малой сложности. Под сложностью стоит понимать логику работы моделей. Например, детальное моделирование телекоммуникационных протоколов может потребовать слишком много времени и ресурсов. В то время как в специализированных системах эта задача может быть уже решена в виде готовых библиотечных модулей. Однако при моделировании отдельных решений, систем массового обслуживания, различных дисциплин обслуживания и проверки адекватности аналитических моделей, изучении их

свойств данная система дает возможность быстрого и наглядного получения результатов. Поэтому ее стоит рассматривать как весьма полезный инструмент в решении научных и инженерных задач.

Как было отмечено выше данная система может быть инсталлирована в различных операционных системах Windows, MAC и Linux.

Система имеет графический редактор, в котором может быть построена модель исследуемой системы с использованием типовых элементов (из палитры элементов), при минимальном использовании программирования. Основными элементами системы, с точки зрения разработчика модели, являются графический редактор и палитры (библиотеки) элементов. Можно выделить следующие основные этапы построения модели:

-выбор элементов и построение структуры системы;

-определение свойств элементов: численных (начальных) значений параметров и логики функционирования;

-определение измеряемых параметров, методов получения статистических результатов и выбор соответствующих элементов;

-описание имитационного эксперимента;

-компиляция модели;

-выполнение имитационного эксперимента.

Модель может иметь иерархическую структуру, в которой построенная модель может быть оформлена в виде модуля, который может быть использован при построении более сложной модели, наравне с типовыми модулями.

В данном учебном пособии рассмотрена только малая доля возможностей данной системы. Далее будет рассмотрено построение двух моделей систем массового обслуживания в среде AnyLogic. Для углубленного изучения возможностей и освоения приемов работы с данной системы имитационного моделирования следует обратиться к справочной информации и документации на данную систему.

Инсталляция данной системы, как правило, не вызывает сложностей, поэтому она отдельно не рассматривается.

2.2 Моделирование СМО с отказами

2.2.1 Построение модели

Для моделирования СМО с отказами требуется определить поток заявок с заданными свойствами и несколько обслуживающих устройств, которые принимают заявки на обслуживание. Обслуживание длится в течение определенного (в общем случае случайного) времени. Поступившая заявка принимается на обслуживание, если в момент ее поступления имеется хотя бы одно свободное обслуживающее устройство. Если свободных обслуживающих устройств нет, то заявка получает отказ (теряется) и покидает систему. В общем случае, с некоторой вероятностью заявка может вернуться в систему, но в рассмотренной ниже модели мы этот вариант рассматривать не будем.

Таким образом, для построения модели нам нужно иметь:

-источник заявок (потока);

-набор элементов, имитирующих обслуживающие устройства;

-средства сбора статистики о работе модели.

1. Запуск среды разработки AnyLogic (в данном примере используется AnyLogic 7.0.2)

2. Создание новой модели (Файл->Создать->Модель) в диалоговом окне зададим имя модели MMV.

3. Выбор и соединение основных элементов модели. Перейдем в элемент проекта «Main». Откроем палитру и выберем библиотеку «Библиотека моделирования процессов» В окне редактора модели добавим нужные элементы: «source», «Select output», «delay» и «sink».

Затем выберем библиотеку «основная» и добавим из нее переменную и дадим ей имя «p».

Соединим выход «source» со входом «Select output», выход «Select output» со входом «delay», выход «delay» со входом «sink» соединительными линиями.

Модель будет выглядеть как приведено на рисунке 2.1.

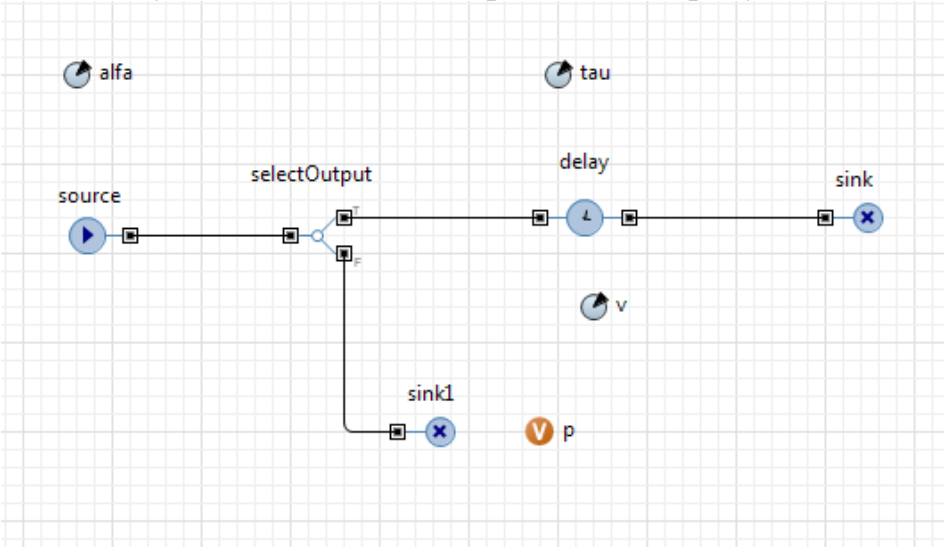


Рисунок 2.1 - Вид модели в редакторе

4. Изменение свойств элементов модели. Чтобы открыть свойства элемента нужно выделить его, при этом справа или внизу (в зависимости от настоек среды) откроется окно свойств выделенного элемента.

4.1 Свойства источника потока и обслуживающего устройства

Введем три параметра: для источника заявок и процесса обслуживания, одну переменную и переименуем их, как показано на рисунке 4 («alfa», «tau», «v» и «p» соответственно).

В свойствах параметров введем значения по умолчанию (на данном этапе установим значения alfa=1; tau=1; v=1.1).

Откроем свойства элемента «source» и внесем следующие изменения (рисунок 2.2).

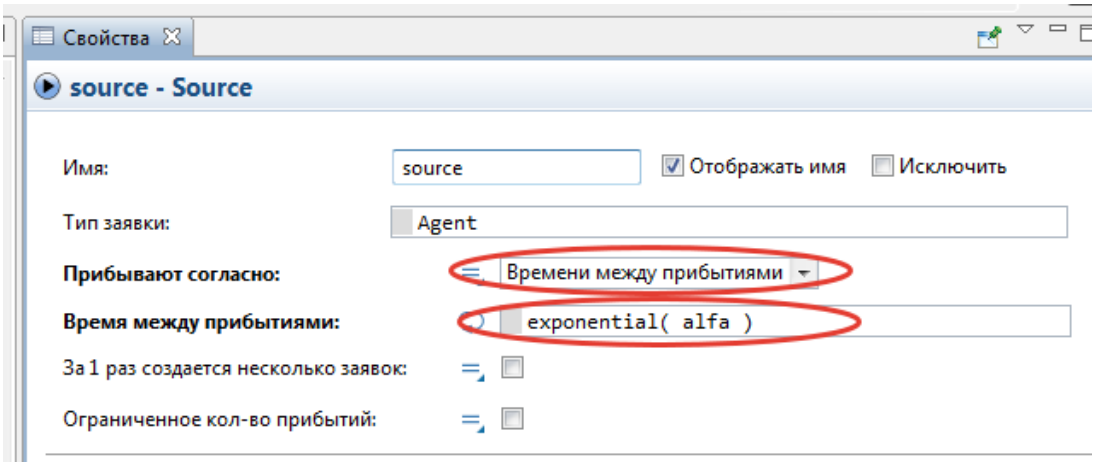


Рисунок 2.2 - Свойства элемента «source»

Откроем свойства элемента «delay» и внесем следующие изменения (рисунок 2.3)

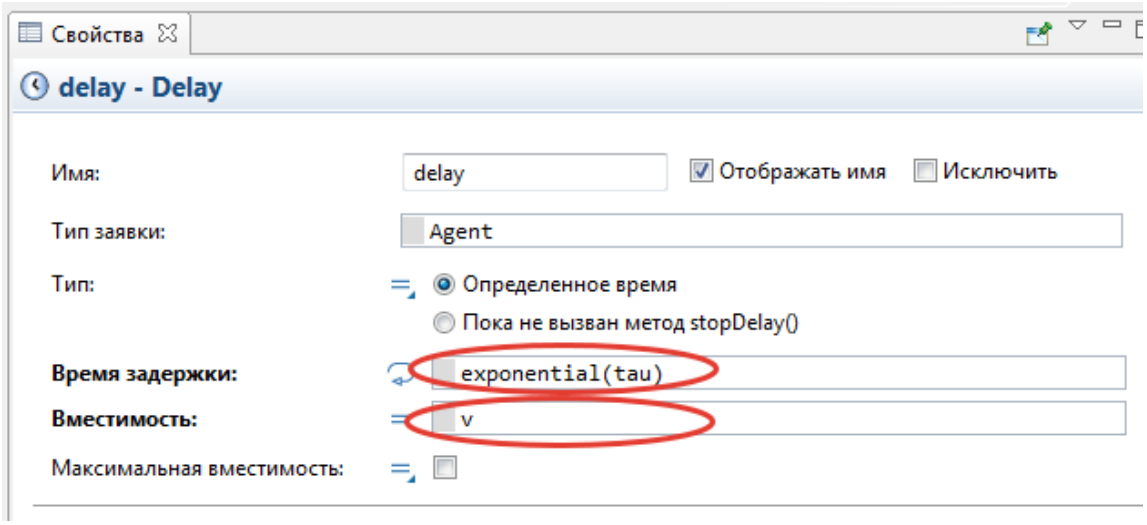


Рисунок 2.3 - Свойства элемента «delay»

4.2 Свойства элемента «sink1»

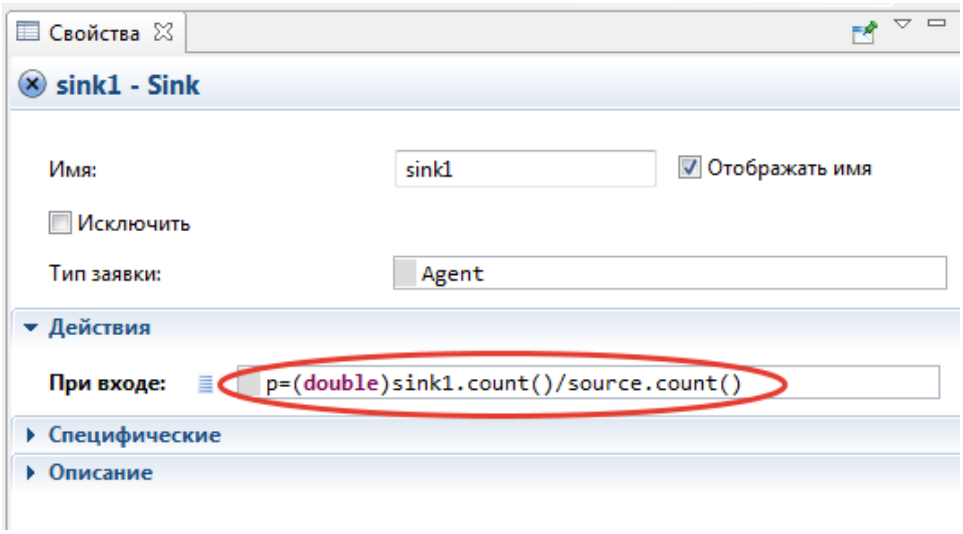


Рисунок 2.4 - Свойства элемента «sink1»

4.2 Свойства элемента «selectOutput»

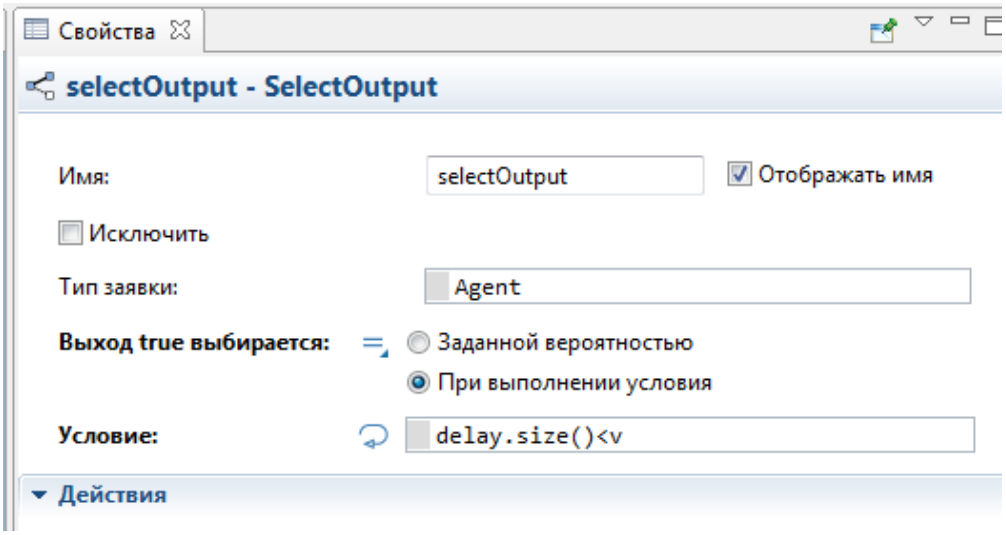


Рисунок 2.5 - Свойства элемента «selectOutput»

После проделанных действий можно проверить работоспособность модели. Для этого выберем в панели инструментов имя модели для запуска процесса имитации (рисунок 2.6).

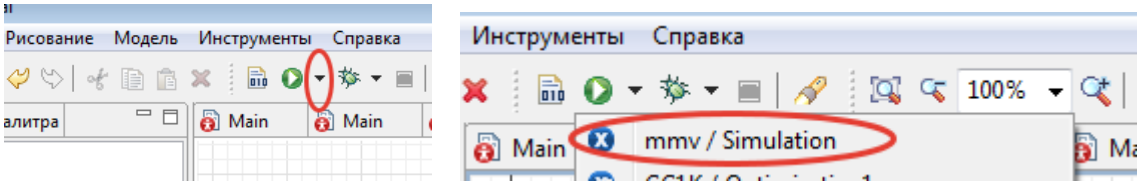


Рисунок 2.6 - Компиляция модели

Если модель введена без ошибок, то будет открыто окно имитационной модели (рисунок 2.7).

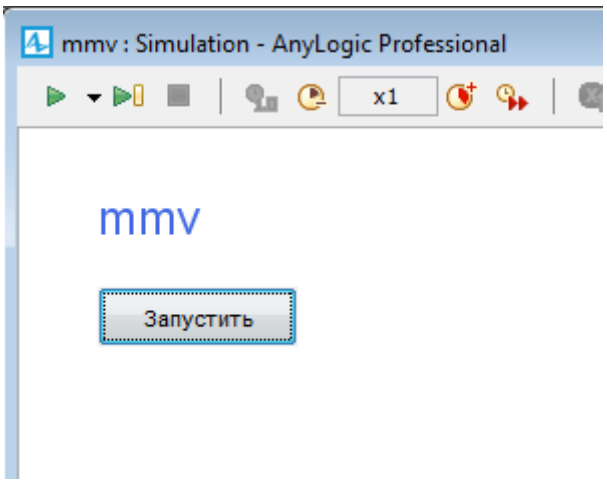


Рисунок 2.7 - Окно имитационной модели

Далее нажмем кнопку «Запустить».

В результате будет отображена анимация работы модели (рисунок 2.8).

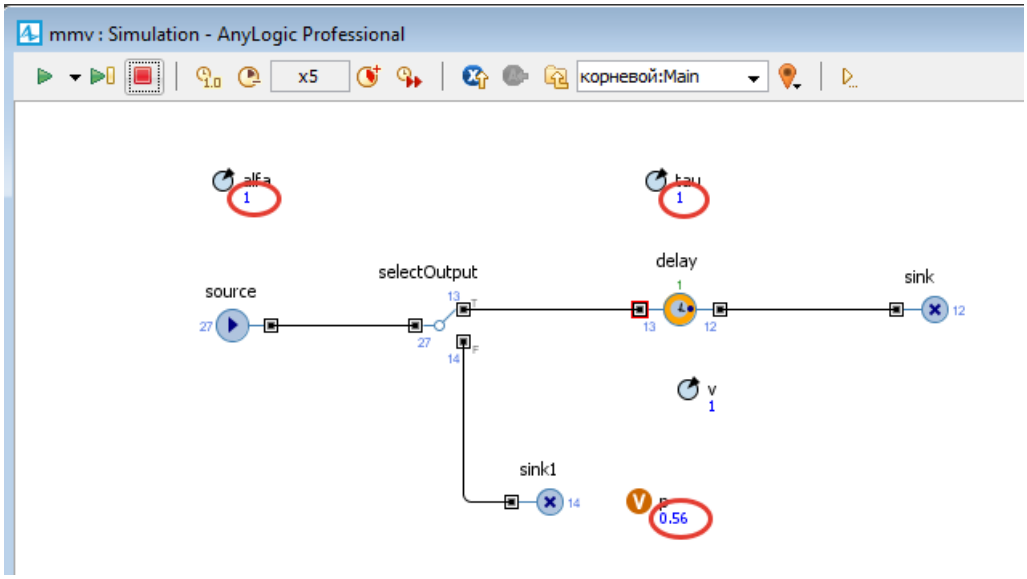


Рисунок 2.8 - Окно имитационной модели

Работу модели можно ускорить, нажав кнопку с двойной стрелкой, выделенной на рисунке 10. Значение переменной «p» должно быть близко к 0,5 (Значение, которое дает первая формула Эрланга [2] при интенсивности нагрузки равной 1 Эрл и одном обслуживающем устройстве.

Внимание. *Если на этапах запуска (рисунки 9 и 10) были выведены сообщения об ошибках, то следует внимательно проверить состав модели и свойства ее элементов.*

Изменяя значение параметра «alfa» мы можем изменять интенсивность поступающего трафика, а изменяя значение параметра «v» можно изменять количество обслуживающих устройств.

2.2.2 Получение результатов моделирования

В данном случае параметром функционирования модели является оценка вероятности потерь, которая вычисляется в переменной «p». Например, изменяя параметры модели можно получить зависимость вероятности потерь от нагрузки. Выберем значение v=10. Далее проведем ряд имитационных экспериментов с различными значениями интенсивности нагрузки «alfa». Выберем набор значений {4, 6, 8, 10, 12, 14, 16}. Результаты сведем в таблицу 2.1.

Таблица 2.1 – Результаты оценки коэффициента потерь

№	Интенсивность нагрузки	Коэффициент потерь
1	4	0,005
2	6	0,043
3	8	0,121
4	10	0,214
5	12	0,301
6	14	0,377
7	16	0,441

По полученным данным построим график зависимости коэффициента потерь от интенсивности нагрузки, рисунок 2.9.

Рисунок 2.9 - Зависимость коэффициента потерь от интенсивности нагрузки (при числе обслуживающих устройств 10)

2.3 Моделирование СМО с ожиданием

2.3.1 Построение имитационной модели

1. Запуск среды разработки AnyLogic (в данном примере используется AnyLogic 7.0.2)

2. Создание новой модели (Файл->Создать->Модель) в диалоговом окне зададим имя модели GG1.

3. Создаем новый тип агента (Файл->Создать->Тип агента) в диалоговом окне введем имя Packet, нажимаем «Далее». Анимация агента, выберем «Нет», нажимаем «Далее». Параметры агента, нажимаем «добавить», назовем добавленный параметр «origTime», выберем тип «double», значение по умолчанию «0». Снова нажмем «добавить», следующий параметр назовем «length», тип «double», значение по умолчанию «1». Нажмем «Готово». В результате будет создан новый тип агента «Packet». Он будет отображаться в дереве проекта.

4. Выбор и соединение основных элементов модели. Перейдем в элемент проекта «Main». Откроем палитру и выберем библиотеку «Библиотека моделирования процессов» В окне редактора модели добавим нужные элементы: «source», «queue», «delay» и «sink». Затем выберем библиотеку «статистика» и добавим из нее три элемента «данные гистограммы», и три элемента «гистограмма».

Соединим выход «source» со входом «queue», выход «queue» со входом «delay», выход «delay» со входом «sink» соединительными линиями.

Модель будет выглядеть как приведено на рисунке 2.10.

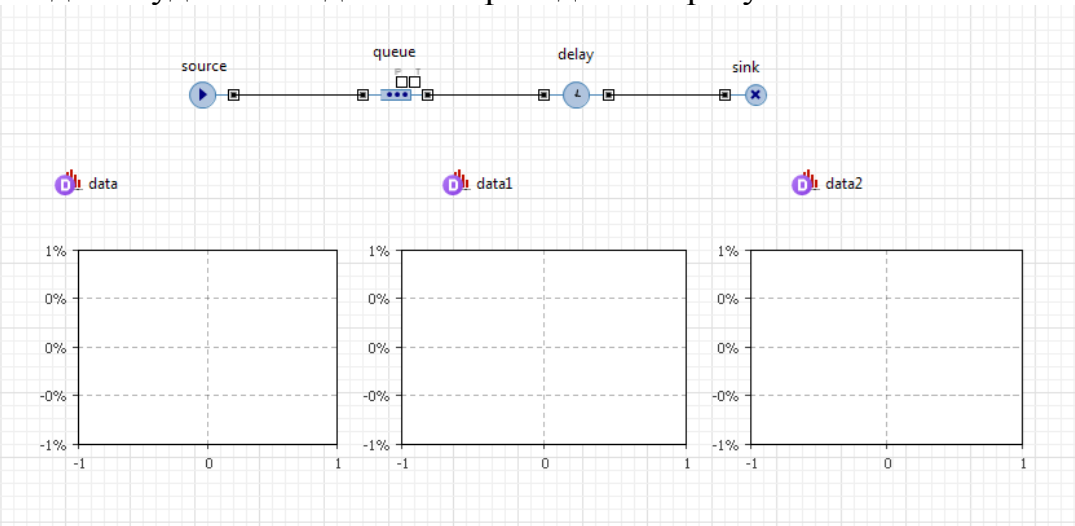


Рисунок 2.10 - Вид модели в редакторе

5. Изменение свойств элементов модели. Чтобы открыть свойства элемента нужно выделить его, при этом справа или внизу (в зависимости от настоек среды) откроется окно свойств выделенного элемента.

5.1 Гистограммы. Переименуем элементы «data1», «data2» и «data2» на «iat», «srvt» и «delivt» (время между прибытиями, время обслуживания и время доставки, соответственно).

Подключим первую гистограмму к элементу «iat», вторую к «srvt» и третью к «delivt». Для этого в свойствах каждой гистограммы нажмем кнопку «Добавить данные» и введем соответствующее название.

В результате модель будет выглядеть, приблизительно, как изображено на рисунке 2.11.

Рисунок 2.11 - Вид модели после редактирования ее свойств

5.2 Свойства источник потока и обслуживающего устройства

Введем четыре параметра: для источника заявок и процесса обслуживания и переименуем их, как показано на рисунке 2.12.

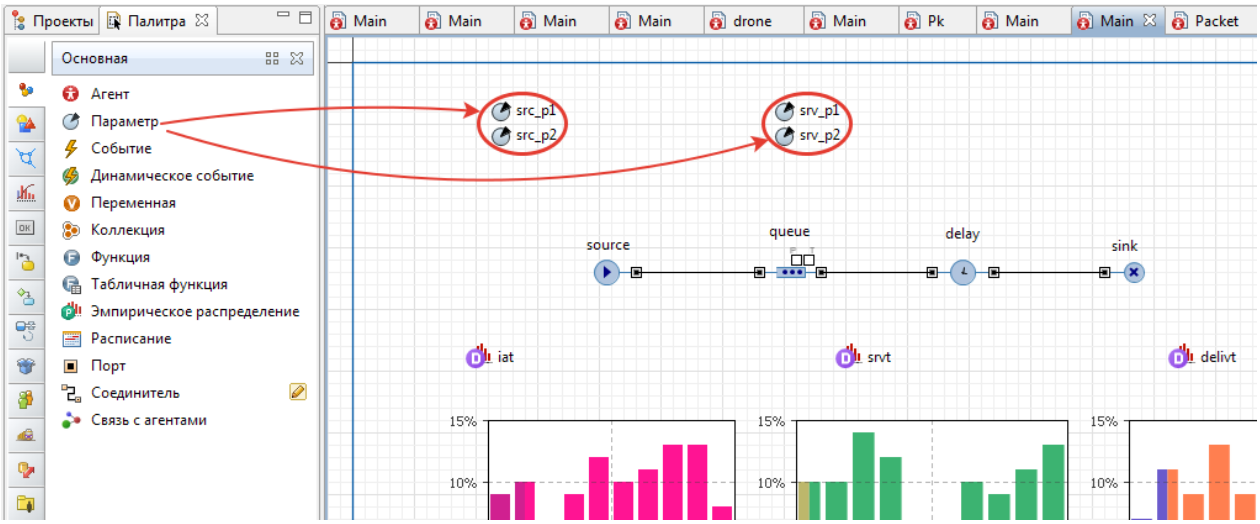


Рисунок 2.12 - Добавление параметров источника и обслуживания

В свойствах параметров введем значения по умолчанию (на данном этапе установим значения src_p1=1.1; src_p2=0.3; srv_p1=1.1; srv_p2=0.14).

Откроем свойства элемента «source» и внесем следующие изменения (рисунок 2.13). (**Внимание**! В более поздних версиях AnyLogic вместо «entity» следует писать «agent»).

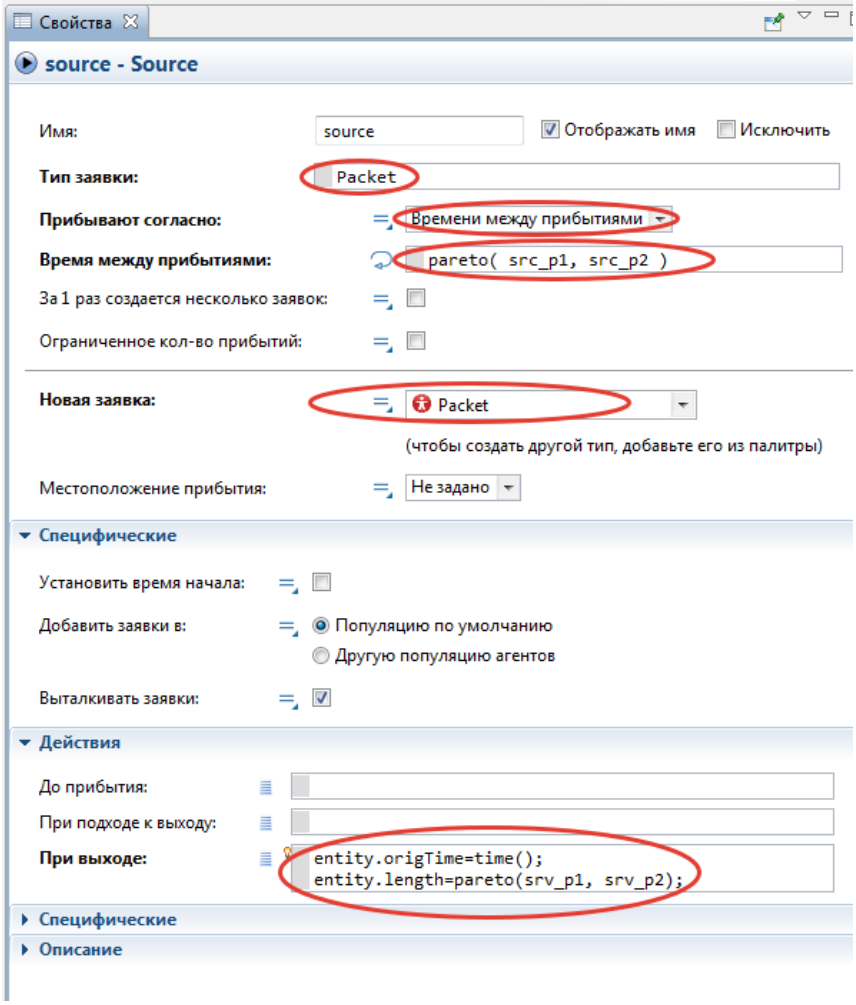


Рисунок 2.13 - Изменение свойств источника

Откроем свойства элемента «delay» и внесем следующие изменения (рисунок 2.14)

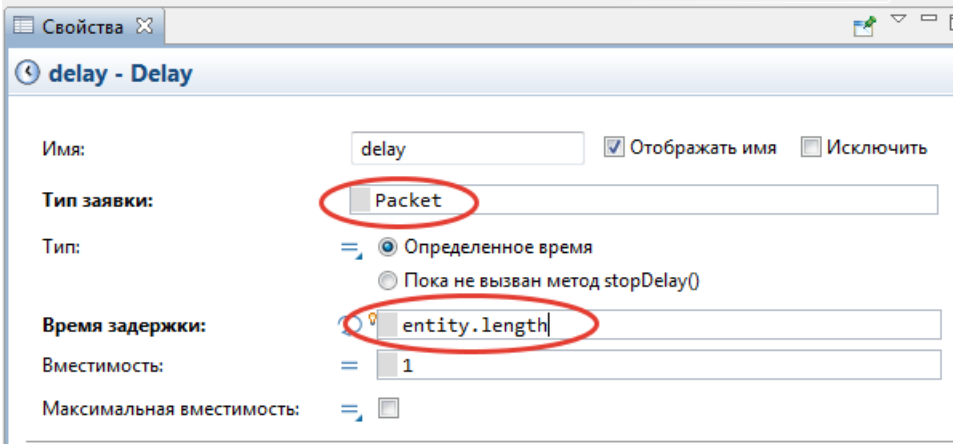


Рисунок 2.14 - Изменение свойств элемента «delay»

5.3 Свойства очереди

Откроем свойства элемента «queue» и внесем следующие изменения (рисунок 2.15)

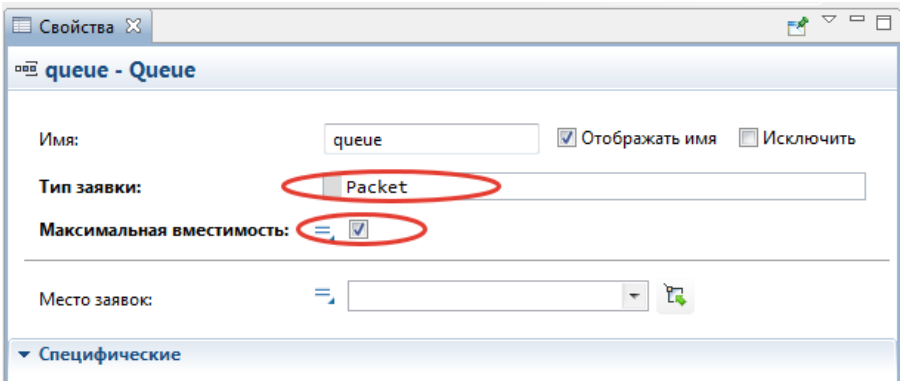


Рисунок 2.15 - Изменение свойств элемента «queue»

После проделанных действий можно проверить работоспособность модели. Для этого выберем в панели инструментов имя модели для запуска процесса имитации (рисунок 2.16).

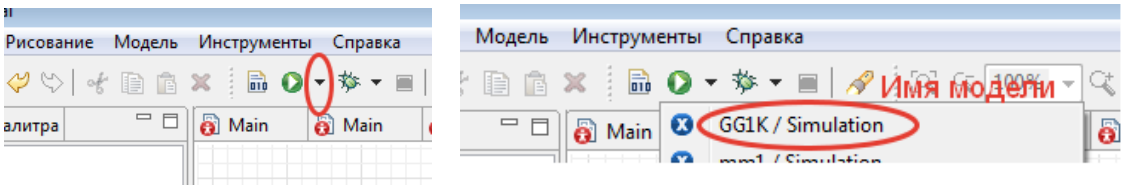


Рисунок 2.16 - Компиляция модели

Если модель введена без ошибок, то будет открыто окно имитационной модели (рисунок 2.17).

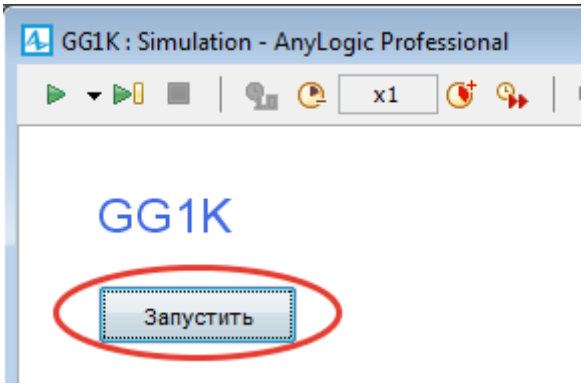


Рисунок 2.17 - Окно имитационной модели

Далее нажмем кнопку «Запустить».

В результате будет отображена анимация работы модели (рисунок 2.18).

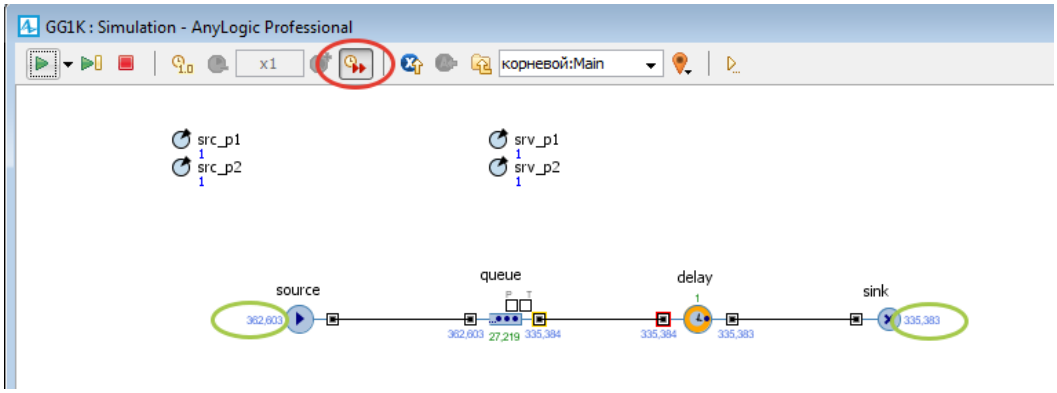


Рисунок 2.18 - Окно имитационной модели

Работу модели можно ускорить, нажав кнопку с двойной стрелкой, выделенной на рисунке 10.

Внимание. *Если на этапах запуска (рисунки 9 и 10) были выведены сообщения об ошибках, то следует внимательно проверить состав модели и свойства ее элементов.*

5.4 Изменение свойств модели для сбора статистики

Введем в модель две переменные «viat» и «vro», рисунок 2.19.

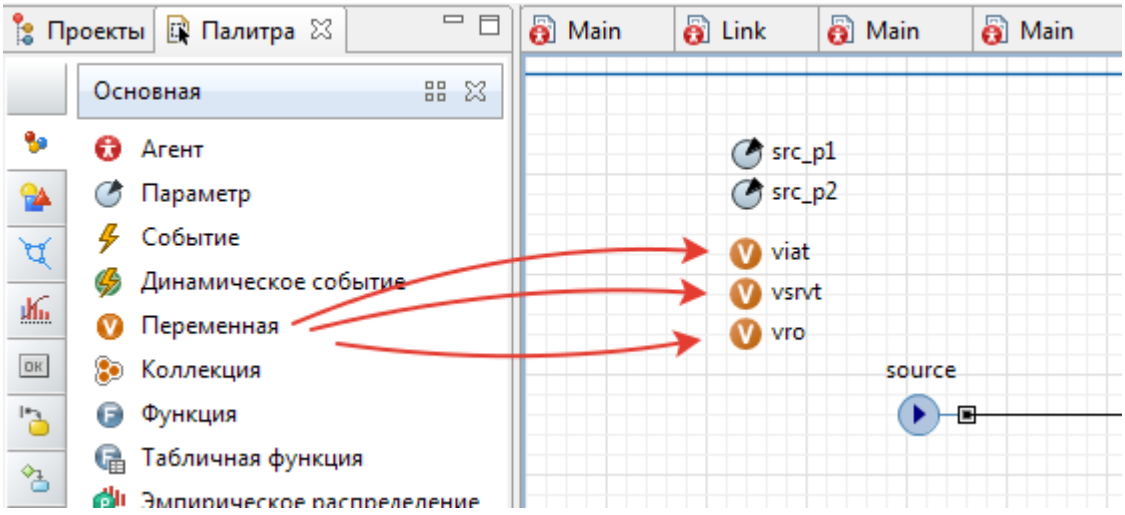


Рисунок 2.19 - Введение двух переменных

Откроем свойства элемента «queue» и внесем следующие дополнения, показанные на рисунке 2.20.

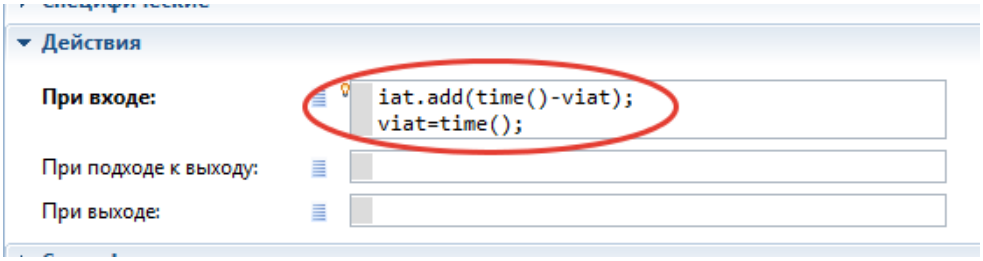


Рисунок 2.20 - Сбор статистики о входном потоке в элементе «queue»

Откроем свойства элемента «delay» и внесем следующие дополнения, показанные на рисунке 2.21.

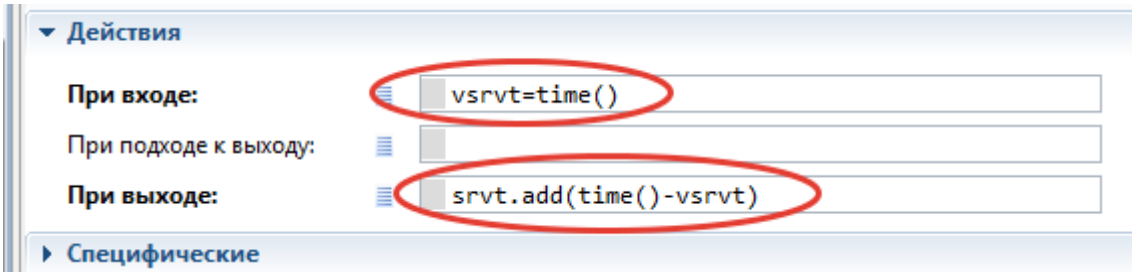


Рисунок 2.21 - Сбор статистики об обслуживании в элементе «delay»

Откроем свойства элемента «sink» и внесем следующие дополнения, показанные на рисунке 2.22.

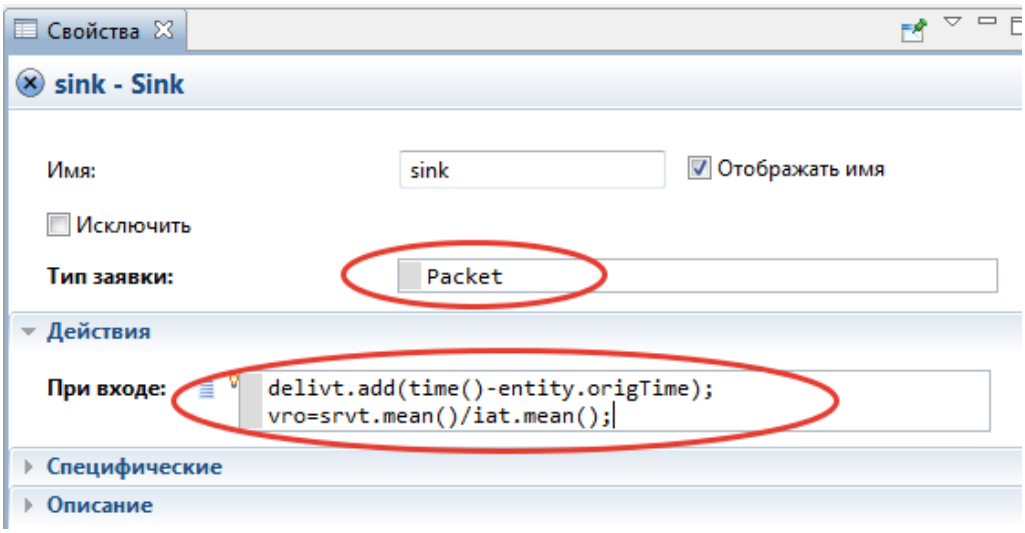


Рисунок 2.22 - Сбор статистики о времени доставки и нагрузке в элементе «sink»

2.3.2 Выполнение имитационного эксперимента

2.3.2.1 Установка заданного значения нагрузки

Для определения значения параметра распределения воспользуемся методом оптимизации, реализованным в AnyLogic.

Например, нужно установить значение нагрузки равное 0,7. В модели значение нагрузки вычисляется в ходе имитационного эксперимента в переменной «vro».

Значение интенсивности нагрузки определяется средним временем обслуживания и интенсивностью трафика (интервалом времени между прибытиями пакетов). В модели зададим среднее время обслуживания равное (приблизительно) единице. При распределении Парето такое значение достигается при значениях параметров srv_p1=2.5 и srv_p2=0.6. Укажем эти значения по умолчанию для данных параметров и не будем их изменять в ходе эксперимента.

Зададим значение параметра src_p1=2.1 как значение по умолчанию, его тоже не будем изменять в ходе эксперимента.

Величиной трафика будем управлять с помощью параметра src_p2.

Для выбора значения данного параметра для заданной величины трафика нужно решить следующую оптимизационную задачу

$$p2 = \arg\min_{p2} |vro - a| \qquad (2.1)$$

Для этого создадим оптимизационный эксперимент Файл->Создать->Эксперимент->Оптимизация.

В свойствах оптимизационного эксперимента введем следующие изменения (рисунок 2.23).

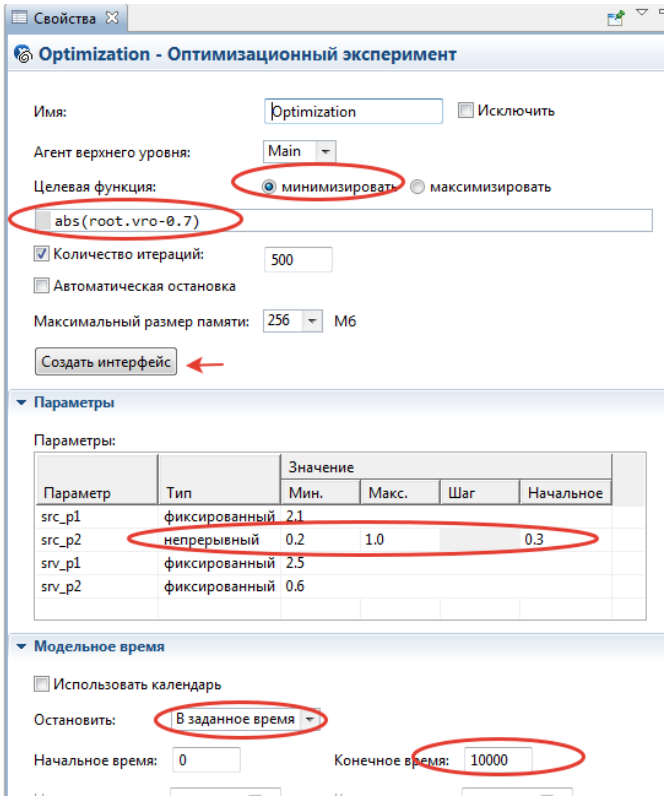


Рисунок 2.23 - Выбор значения параметра для нагрузки 0,7

Далее через панель инструментов запустим оптимизационный эксперимент, рисунок 2.24.

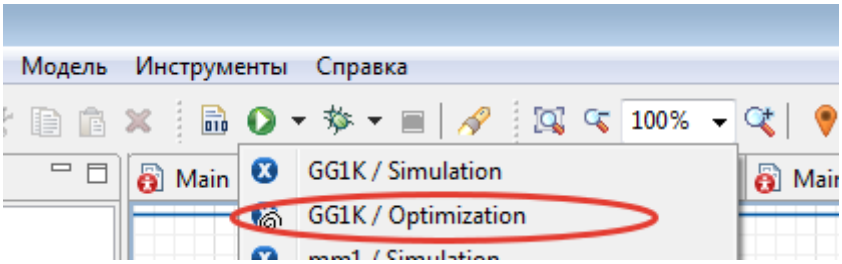


Рисунок 2.24 - Запуск оптимизационного эксперимента

В результате выполнения оптимизационного эксперимента получим значение параметра src_p2 при котором значение нагрузки равно 0,7 (рисунок 2.25).

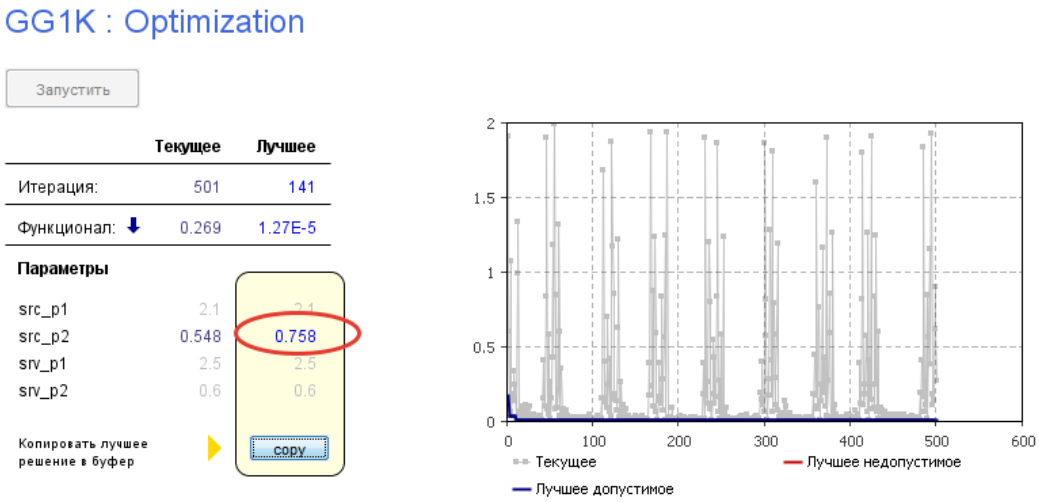


Рисунок 2.25 - Результат оптимизационного эксперимента

Установим полученное значение в качестве значения по умолчанию параметра src_p2 и запустим имитацию, рисунок 2.26.

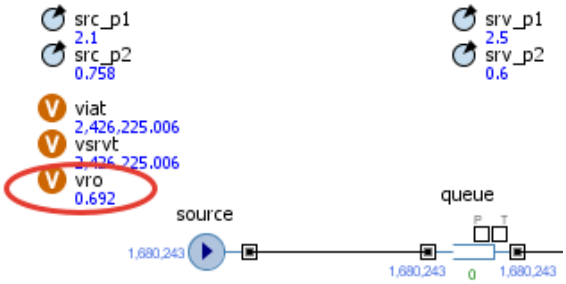


Рисунок 2.26 - Имитация с полученным значением параметра

Из рисунка 2.26 видно, что получаемое значение нагрузки 0,692 близко к заданному (0,7), следовательно, полученное значение параметра верно.

2.3.2.2 Получение зависимости задержки доставки пакета от нагрузки

Для получения данной зависимости необходимо выполнить серию имитационных экспериментов для различных значений нагрузки.

Для удобства снятия результатов, введем в модель еще несколько переменных, рисунок 2.27.

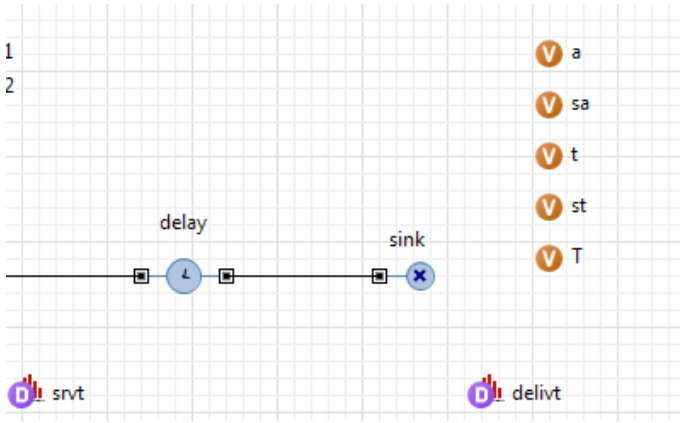


Рисунок 2.27 - Дополнительные переменные для отображения результатов

Значения переменным будем присваивать в элементе «sink», дополним его кодом, как показано на рисунке 2.28.

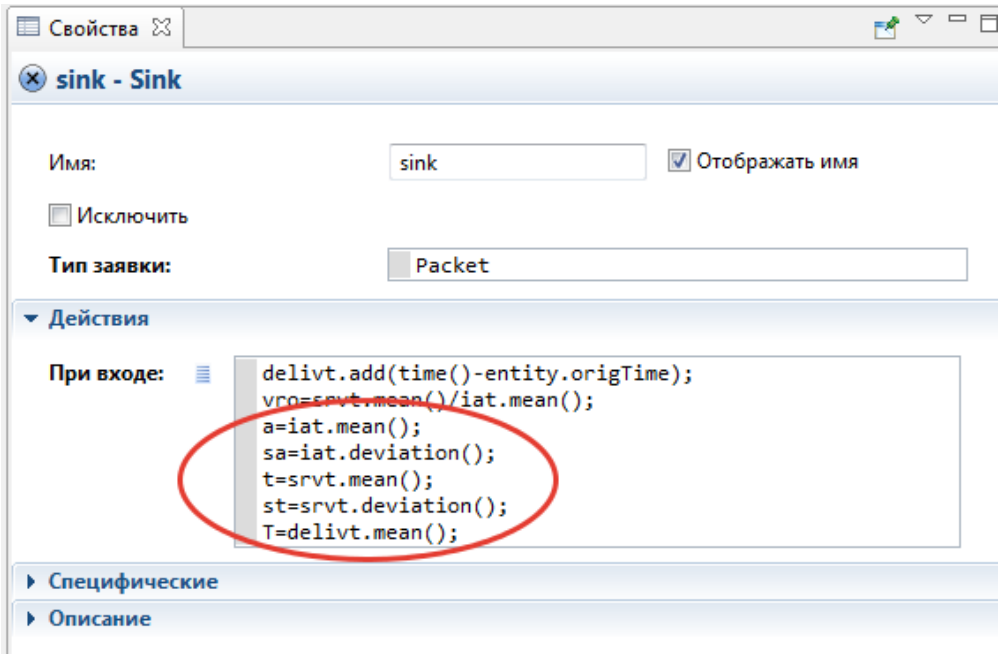


Рисунок 2.28 - Дополнение элемента «sink»

Результаты экспериментов следует занести в таблицу 2.2.

Таблица 2.2 – Результаты исследования

№	Интенс. нагрузки		Значение параметра	Интервал между пакетами	СКО интервала между пакетами	Время обслуж.	СКО времени обслуживания	Время доставки
		vro	src_p2	a	σ_a	t	σ_t	T
1	0,1	0,111	4,721	9,01	16,23	1,00	0,95	1,03
2	0,2	0,197	2,676	5,11	10,35	1,00	0,85	1,06
3	0,3	0,296	1,767	3,37	5,51	1,00	0,85	1,12
4	0,4	0,405	1,292	2,47	3,38	1,00	0,83	1,23
5	0,5	0,488	1,075	2,05	3,03	1,00	0,86	1,39
6	0,6	0,598	0,877	1,68	2,72	1,00	0,98	1,83
7	0,7	0,691	0,758	1,446	2,123	1,00	0,862	2,198
8	0,8	0,798	0,652	1,25	1,80	1,00	0,84	3,47
9	0,9	0,904	0,580	1,11	1,55	1,00	0,86	8,29
10	0,95	0,949	0,554	1,06	1,67	1,00	0,87	16,65

На рисунке 2.29 приведена полученная зависимость времени доставки от интенсивности нагрузки.

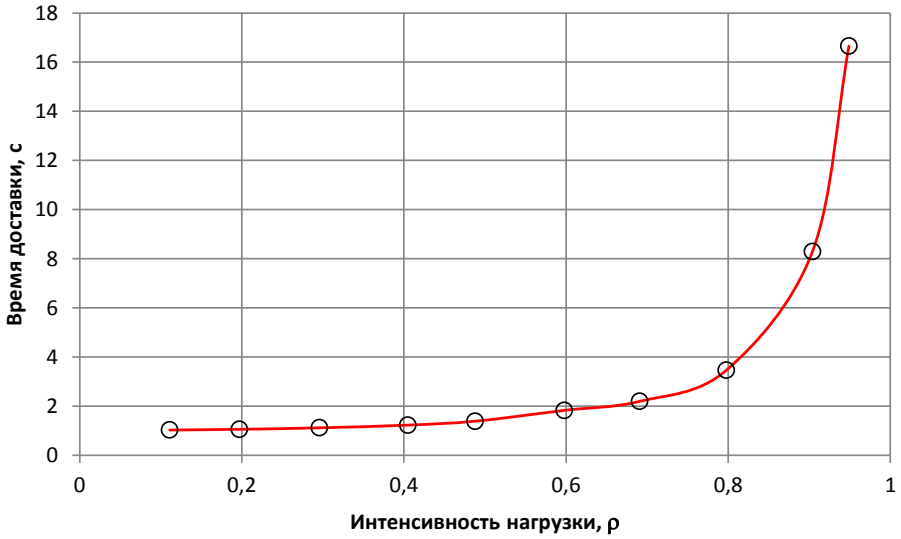


Рисунок 2.29 - Зависимость времени доставки от нагрузки

2.3.3 Оптимизация размера буфера

Дополним модель одним элементом «sink» и изменим параметры очереди, тем самым преобразуем модель в СМО G/G/1/K с конечным размером очереди и комбинированной дисциплиной обслуживания.

Полагаем, что задержка доставки большая, чем пороговое время доставки (в примере 10 с) эквивалентна потере.

Выберем вместимость очереди, такой при которой достигается минимум суммы потерь и превышений порогового времени доставки при значении нагрузки 0,95.

Для этого модифицируем модель как показано на рисунках 2.30 – 2.34.

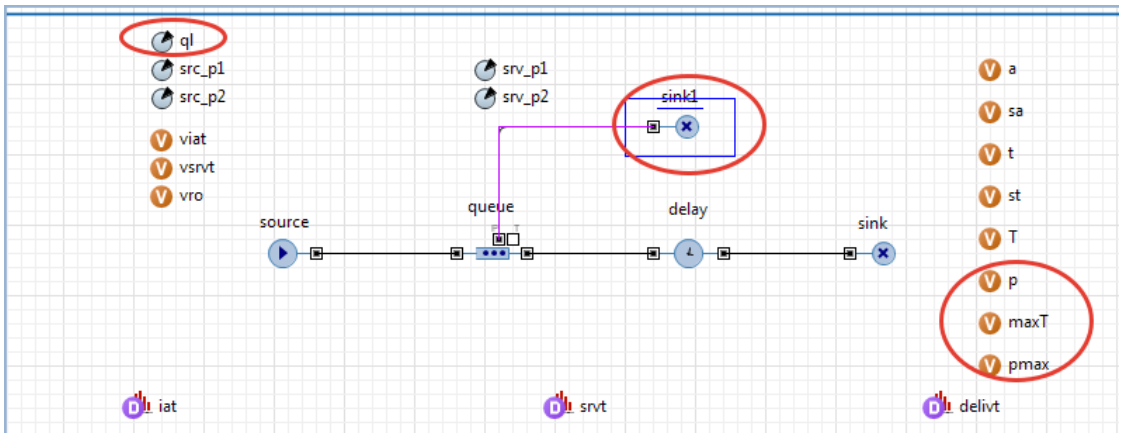


Рисунок 2.30 - Добавленные в модель элементы

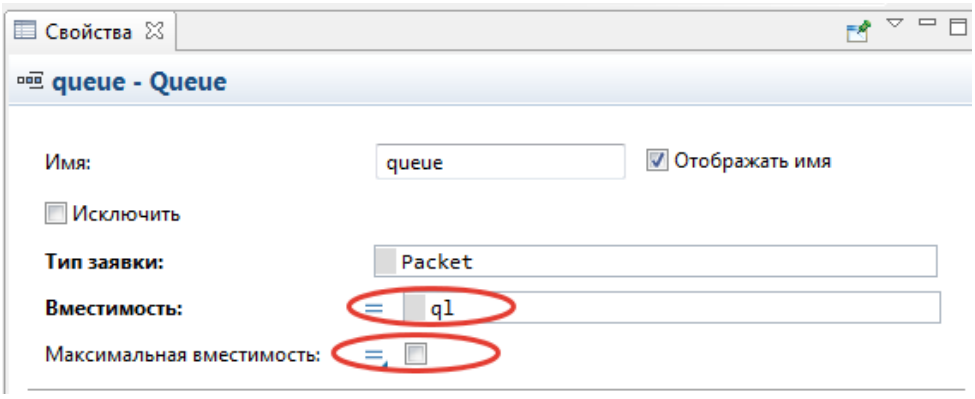


Рисунок 2.31 - Изменение свойств очереди

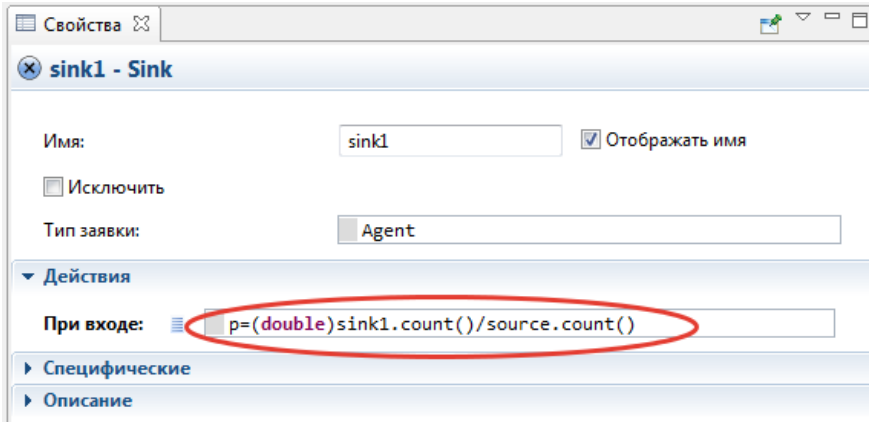


Рисунок 2.32 - Изменение свойств «sink1»

Создадим еще один оптимизационный эксперимент, рисунок 2.33.

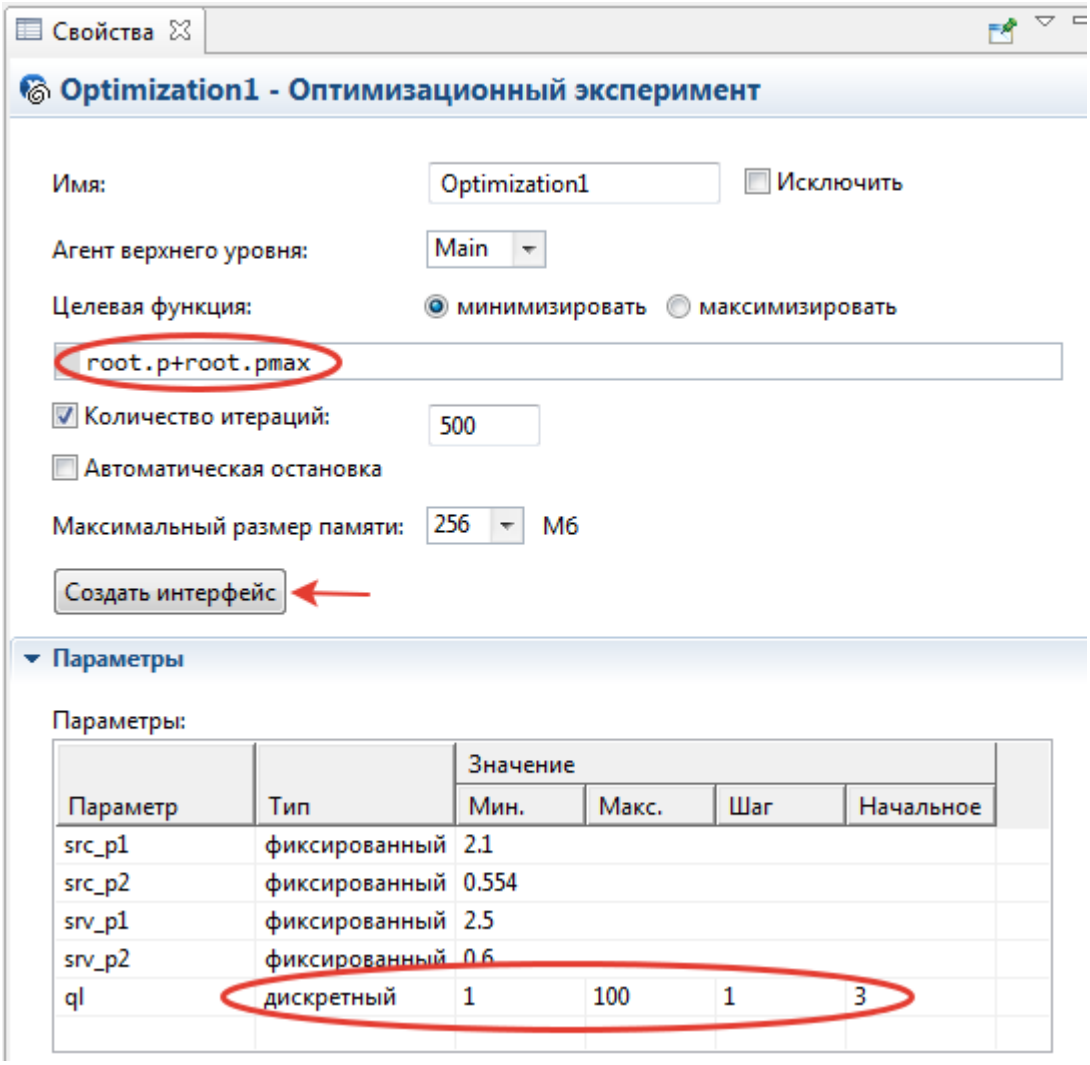


Рисунок 2.33 - Оптимизация вместимости очереди

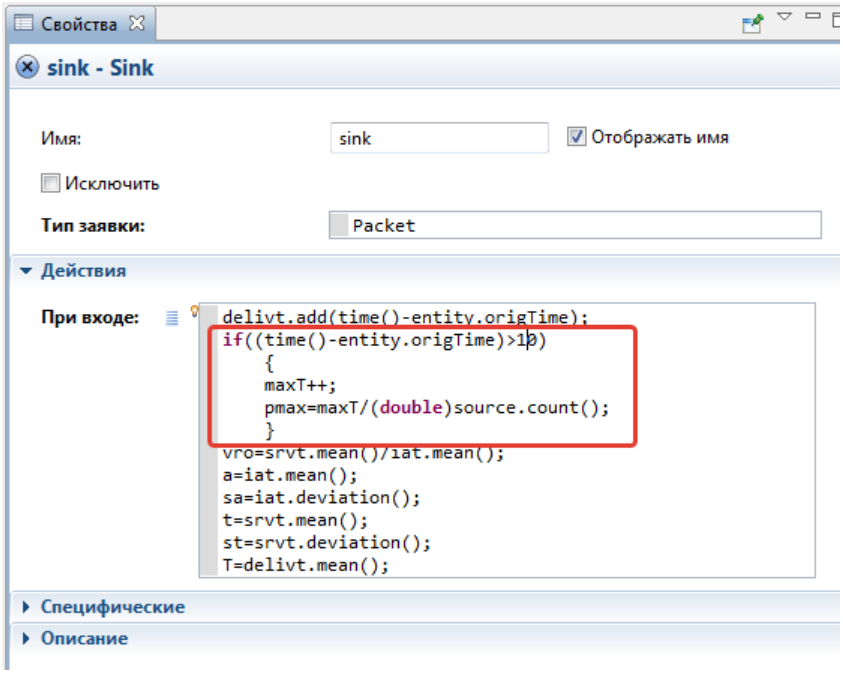


Рисунок 2.34 - Изменения в элементе «sink»

Запустим оптимизационный эксперимент.

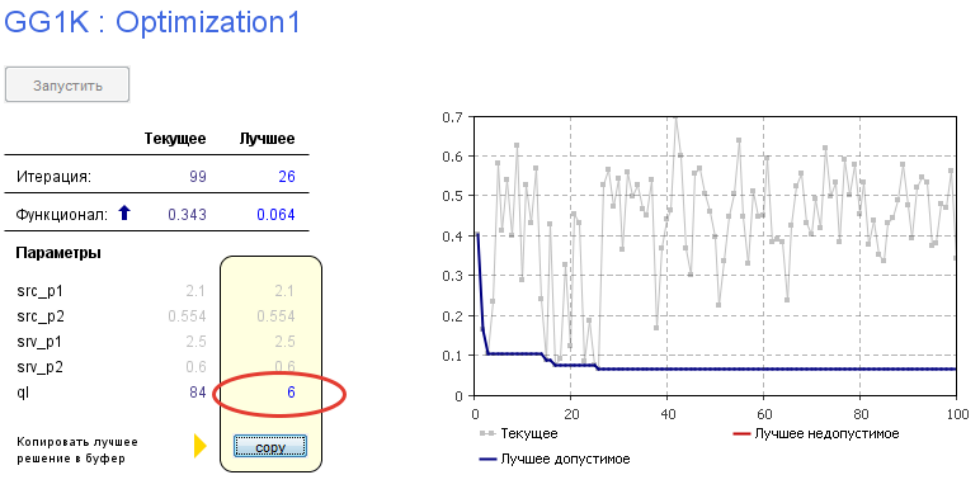


Рисунок 2.35 - Результат оптимизации вместимости очереди

Полученное значение 6.

Выводы

В результате выполнения работы была построена и исследована модель СМО G/G/1, в которой входной поток и время обслуживания были заданы распределением Парето.

1. Освоены основные методы работы с системой моделирования AnyLogic и разработана имитационная модель СМО G/G/1.
2. Проведены имитационные и оптимизационные эксперименты, в результате которых получены необходимые значения параметров СМО.
3. Исследована зависимость времени доставки от интенсивности нагрузки. Время доставки увеличивается с ростом интенсивности нагрузки. Полученная зависимость находится между зависимостями для СМО типов M/M/1 и M/D/1.
4. Построена модель СМО с комбинированной дисциплиной обслуживания G/G/1/K.
5. Найдена оптимальная вместимость очереди, которая при нагрузке 0,95 равна 6.

3 Имитационное моделирование в системе OMNeT++

3.1 Краткое описание системы моделирования OMTeT++

OMNeT ++ представляет собой систему дискретного событийного моделирования модульной структуры. Он ориентирован на моделирование сетей связи и использует объектно-ориентированный подход. Благодаря гибкой архитектуре системы может быть, использована в различных проблемных областях, например таких как:

-моделирование проводных и беспроводных сетей связи;

-моделирование протоколов;

-моделирование систем и сетей массового обслуживания;

-моделирование многопроцессорных систем и других распределенных аппаратных систем;

-моделирование и проверка аппаратных архитектур;

-оценка производительности сложных программных систем.

Обобщая, можно сказать, что данная система применима в тех областях, где изменение состояния моделируемой системы может быть описано потоком дискретных событий.

Сам по себе OMNeT ++ не является симулятором, а предоставляет собой инфраструктуру и набор инструментов для создания имитационных моделей. Одним из базовых принципов построения этой инфраструктуры является модульность. Модели собираются из модулей, которые могут быть многократно использованы. Модули представляют собой отлаженные программы, которые могут объединяться различными способами.

Модули могут быть соединены друг с другом через шлюзы (в других системах эти элементы могут называться портами), и образовывать новые модули. Глубина вложенности модуля не ограничена. Модули взаимодействуют друг с другом посредством передачи сообщений, которые могут содержать произвольные структуры данных. Модули могут передавать сообщения по определенным путям через шлюзы и соединения, или напрямую к месту назначения. Последнее полезно, например, при моделировании беспроводных технологий. Модули могут иметь параметры, которые используются для настройки их поведения и / или определения топологии модели. Модули, расположенные на самом низком уровне иерархии (по отношению к вложенности), называются простыми модулями, они включают в себя модель поведения. Простые модули реализованы в виде программы на языке C ++ и используют библиотеки моделирования.

Модели OMNeT ++ могут выполняться работать под различными пользовательскими интерфейсами. Графический интерфейс, очень полезен для демонстрации и отладки моделей, а интерфейс командной строки подходит для выполнения пакета моделей.

Симулятор, инструменты и пользовательские интерфейсы обладают высокой переносимостью переносимости. Они испытываются на наиболее распространенных операционных систем (Linux, Mac OS / X, Windows), и

могут быть установлены из коробки или после тривиальных модификаций на большинстве Unix-подобных операционных систем.

OMNeT ++ поддерживает параллельное распределенное моделирование. В OMNeT ++ может использоваться несколько механизмов связи между разделами распределенного параллельного моделирования, например, MPI или именованные каналы. Алгоритм параллельного моделирования может быть легко расширен, также могут быть добавлены новые алгоритмы параллельного моделирования. OMNeT++ можно использовать для презентации в классе параллельных алгоритмов моделирования, так как моделирование может работать параллельно даже в графическом интерфейсе, который предоставляет детальную информацию о том, что происходит.

OMNEST является коммерчески поддерживаемой версии OMNeT ++. OMNeT ++ является бесплатным только для академических и некоммерческих использования; в коммерческих целях, необходимо получить лицензии OMNEST от Simulcraft Inc.

3.2 Инсталляция OMNET++

Ubuntu 14.04 LTS

1. Загрузка. Загрузить с сайта http://omnetpp.org файл omnetpp-5.0-src.tgz. Скопировать файл в каталог home/username, где username имя пользователя, под которым Вы вошли в систему.

2. Распаковка. Откройте окно терминала, нажав Ctrl+Alt+T. Распакуйте архив, введя в окне терминала $ tar xvfz omnetpp-5.0-src.tgz. После этого в папке home/username будет создан каталог omnetpp-5.0.

3. Переменные окружения. Устанавливаем переменные окружения. Набираем в окне терминала

$ gedit ~/.bashrc , в конец файла добавим строку

export PATH=$PATH:$HOME/omnetpp-5.0/bin

Закрываем редактор с сохранением файла. После этого закроем и вновь откроем терминал.

4. Обновление системы. Введем в окне терминала

$ sudo apt-get update

5. Установка пакетов. Введем в окне терминала

5.1 $ sudo apt-get install build-essential gcc g++ bison flex perl \
tcl-dev tk-dev libxml2-dev zlib1g-dev default-jre \
doxygen graphviz libwebkitgtk-1.0-0

5.2 $ sudo apt-get install qt4-qmake libqt4-dev libqt4-opengl-dev \
openscenegraph libopenscenegraph-dev openscenegraph-plugin-osgearth \
osgearth osgearth-data libosgearth-dev

6. Конфигурирование. Введем в окне терминала

$ ./configure

7. Сборка. Введем в окне терминала

$ make

8. Проверка. Введем в окне терминала

$ cd samples/dyna

$ ./dyna

3.3 Процесс построение модели «с нуля» (модель tictoc)

Рассмотренная ниже модель (tictoc) подробно описана в документации OMNeT++, а ее реализация входит в набор примеров стандартного инсталляционного пакета. Однако, рекомендуется выполнить весь процесс ее построения самостоятельно с целью закрепления навыков работы с системой моделирования и понимания назначения отдельных элементов модели и процесса ее построения.

Начнем рассмотрение процесса построения модели с сети, которая состоит всего из двух узлов. Пусть эти узлы выполняют простейшие операции. Эти операции заключаются в том, что один из узлов создает сообщение (пакет) и передает его второму узлу. Второй узел принимает сообщение и отправляет его обратно первому узлу. Первый узел ведет себя аналогичным образом. В результате функционирование узлов сводится к приему сообщения и отправке его обратно. Этот процесс можно сравнить с игрой в пинг-понг, где игроки это узлы, а шарик передаваемое сообщение. Назовем эти узлы "tic" и "toc" (с целью избежать путаницы, здесь и далее будем придерживаться тех наименований и определений, которые приведены в документации OMNeT++).

Рассмотрим те этапы, которые нужно выполнить, чтобы реализовать этот проект с нуля:

1. Создадим рабочий каталог с именем проекта, и перейдем в него. Создадим его в каталоге ~/omnetpp-4.6/samples. Целесообразно, выбирать имя каталога таки же, как имя модели, но в нашем случае каталог «tictoc» уже существует в исходной инсталляции OMNeT++, поэтому мы отступим от этого правила и назовем каталог нашего проекта «mytictoc» (это не повлияет на процесс создания и запуска модели). Для создания каталога в Linux откроем окно терминала (Ctrl+Alt+T) и перейдем в каталог ./home/opnetpp-4.6/samples (при стандартном размещении каталогов OMNeT++, если у Вас оно отличается, то это нужно учесть) и выполним команду mkdir mytictoc. Далее перейдем в созданный каталог, выполнив команду cd mytictoc.

2. Опишем топологию моделируемой сети путем создания файла топологии. Файл описания топологии это текстовый файл, который содержит текст описания на языке NED (Network Definition). В нем определены узлы сети и связи между ними. Его можно создать любимым текстовым редактором. Именно это и рекомендуется сделать. Можно использовать, например, gedit в Linux или Notepad в Windows, а также подобные им редакторы. Не следует использовать такие редакторы (текстовые процессоры) как Word и ему подобные, т.к. они используют некоторые служебные (невидимые на экране) символы форматирования, которые могут повлиять на дальнейший процесс построения проекта или просто вызвать ошибки.

Замечание. Подробное описание языка NED можно найти в разделе 3 OMNeT ++ вручную (каталоге «DOC» в установке OMNeT ++).

Назовем этот файл tictoc1.ned. Его содержимое приведено ниже.

(Для создания наберем в терминале команду gedit tictoc1.ned и введем в окно редактора текст, приведенный ниже).

```
simple Txc1
{
   gates:
      input in;
      output out;
}
//
//tic и toc экземпляры класса Txc1, они соединены друг с другом
//в обоих направлениях. tic и toc будут передавать сообщения
//друг другу.
//
network Tictoc1
{
   submodules:
      tic: Txc1;
      toc: Txc1;
   connections:
      tic.out --> {  delay = 100ms; } --> toc.in;
      tic.in <-- {  delay = 100ms; } <-- toc.out;
}
```

После ввода следует закрыть окно редактора с сохранением введенного текста.

Содержимое файла понятнее, если его читать снизу вверх. Рассмотрим определение конфигурации сети (network ... {...}), во второй половине файла. Сам элемент Tictoc1 определен как экземпляр класса network (сеть). В его определении можно увидеть две секции: submodules (субмодули) и connections (соединения). В секции submodules определены два субмодуля tic и toc. Субмодули (tic и toc) являются экземплярами одного и того же типа модулей Txc1. В секции connections мы подключаем выходной шлюз подмодуля tic (out) ко входному шлюзу toc (in), и наоборот. Мы решили, что соединение вносит задержку доставки 100 мс в обоих направлениях (это время, которое требуется для доставки сообщения по линии связи между модулями, конкретное численное значение выбрано исключительно как пример и не несет в себе иного смысла).

Рассмотрим конструкцию (simple ... {...}) в начале файла. Здесь тип Txc1 определен как простой (simple) модуль (это означает, что он является атомарным на уровне языка NED, и должен быть реализован на C ++). В описании типа Txc1 есть только одна секция gates (шлюзы), в которой определены два шлюза: один входной (типа input) с именем «in», и один выходной (типа output) с именем «out».

Замечание. Более подробную информацию о простых модулях можно найти в Разделе 4 руководства OMNeT ++.

3. Теперь нужно реализовать функциональность простого модуля Txc1. Для этого аналогичным образом создадим файл C++ «txc1.cc». Его содержимое приведено ниже.

```
#include <string.h>
#include <omnetpp.h>
class Txc1 : public cSimpleModule
{
  protected:
    // Переопределение виртуальных функций, определяющих алгоритм.
    virtual void initialize() override;
    virtual void handleMessage(cMessage *msg) override;
};
// Класс модуля необходимо зарегистрировать в OMNeT++
Define_Module(Txc1);
void Txc1::initialize()
{
    //Инициализация выполняется в начале процесса моделирования.
    //Для запуска tic-toc-tic-toc, одному из модулей нужно
    //отправить первое сообщение. Пусть это будет `tic'.
    //Какой это модуль Tic или Toc?
    if (strcmp("tic", getName()) == 0) {  //Если Tic, создаем и отправляем
        //первое сообщение в шлюз "out". Строка "tictocMsg" это
        // произвольная строка, которая является именем объекта message.
        cMessage *msg = new cMessage("tictocMsg");
        send(msg, "out");
    }
}
void Txc1::handleMessage(cMessage *msg)
{
    //Метод handleMessage() вызывается при приходе каждого
    //сообщения. Мы отправляем его другому модулю через шлюз
    //`out'. Модули`tic' and `toc' делают одно и то же, сообщение
    // будут пересылаться между ними.
    send(msg, "out"); // Отправить сообщение
}
```

Поясним этот текст. Тип простого модуля Txc1 определяется как класс C++ Txc1, который является подклассом cSimpleModule, он должен быть зарегистрирован в OMNeT ++ с помощью макроса Define_Module ().

В этом классе переопределяются (перегружаются) два метода класса cSimpleModule: initialize() (инициализация) и handleMessage() (обработка сообщения). При работе модели они вызываются из ядра программы моделирования. Первый вызывается только один раз при инициализации, а второй каждый раз, когда модуль получает сообщение.

Далее следует определение функциональности этих методов. В методе Initialize() сначала выполняется проверка имени модуля, это нужно для того чтобы понять, что делать дальше. Мы решили, что первое сообщение будет передавать модуль с именем «tic». Выполняемая функция «не знает» какому модулю она принадлежит, поэтому и нужна проверка. Если проверка

показала, что это модуль «tic», то создается сообщение, т.е. объект (cMessage), который отправляется из модуля через шлюз out. Поскольку этот шлюз подключен к входному шлюзу другого модуля (toc), ядро моделирования доставит это сообщение другому модулю как аргумент метода handleMessage() (после того, как задержит на 100 мс в линии связи). Если проверка показала, что имя модуля не «tic», то функция просто завершается, не выполнив никаких действий.

Второй модуль просто отправляет полученное сообщение обратно (также с задержкой 100 мс). Это приводит к непрерывному циклическому процессу похожему на «пинг-понг».

Сообщения (пакеты, кадры, заявки и т.д.) и события (таймеры, тайм-ауты) в OMNeT++ представлены объектами cMessage (или его подклассами). После того, как их отправляют или распределяют, они помещаются ядром программы моделирования в список "запланированных событий" (scheduled events) или "список будущих событий" (future events) пока не наступит их время, тогда они будут переданы соответствующим модулям через метод handleMessage().

Обратите внимание, что не существует ни одно условия остановки процесса. Процесс моделирования будет продолжаться «бесконечно». Его можно остановить вручную с помощью графического интерфейса. (Также возможно указать ограничение по времени моделирования или ограничение по времени центрального процессора в файле конфигурации, но в данном примере мы этого не делается.)

4. Теперь создадим файл сборки Makefile, который нужен для компиляции и компоновки программы, и получения исполняемого файла tictoc:

Для этого выполним следующую команду

$ opp_makemake

Эта команда должна создать файл сборки (Make) в рабочем каталоге mytictoc.

5. Далее откомпилируем и скомпонуем программу моделирования с помощью команды:

$ make

Если были выданы сообщения об ошибках компиляции, их необходимо исправить и повторить операции компиляции о компоновки. Повторяем процесс, пока сборка не будет выполнена без ошибок компиляции и компоновки.

Замечание. Для компиляции и компоновки проекта может быть удобнее использовать IDE OMNeT++. Процессы создания нового проекта, добавления файлов и сборка проекта в IDE полностью автоматизированы. При этом нет необходимости вручную создавать файл Make. Однако, если Вы хотите закрепить навыки работы с системой моделирования,

рекомендуется провести описанные выше действия вручную в командной строке терминала.

6. Если сейчас запустить исполняемый файл, он выдаст сообщение, что не может найти файл omnetpp.ini. Это файл инициализации, его тоже необходимо создать. Файл omnetpp.ini сообщает программе моделирования какую сеть следует имитировать (в одной и той же программе может быть несколько сетей), передает значения параметров модели, явно указывает начальные значения для генераторов случайных чисел и т.д.

Можем создать, аналогичным образом, следующий простой файл omnetpp.ini:

```
[General]
network = Tictoc1
```

7. После того, как выполнены описанные выше действия, мы можем запустить моделирование, выполнив следующую команду:

```
$ ./mytictoc
```

Теперь мы должны увидеть окно моделирования OMNeT ++.

Замечание. Для того, чтобы запустить моделирование из IDE, нужно просто щелкнуть правой кнопкой мыши на omnetpp.ini и выбрать запуск, моделирования в OMNeT ++.

8. Чтобы запустить модель нужно нажать кнопку Run на панели инструментов. Мы увидим анимацию процесса обмена сообщениями между tic и toc (рисунок 3.1).

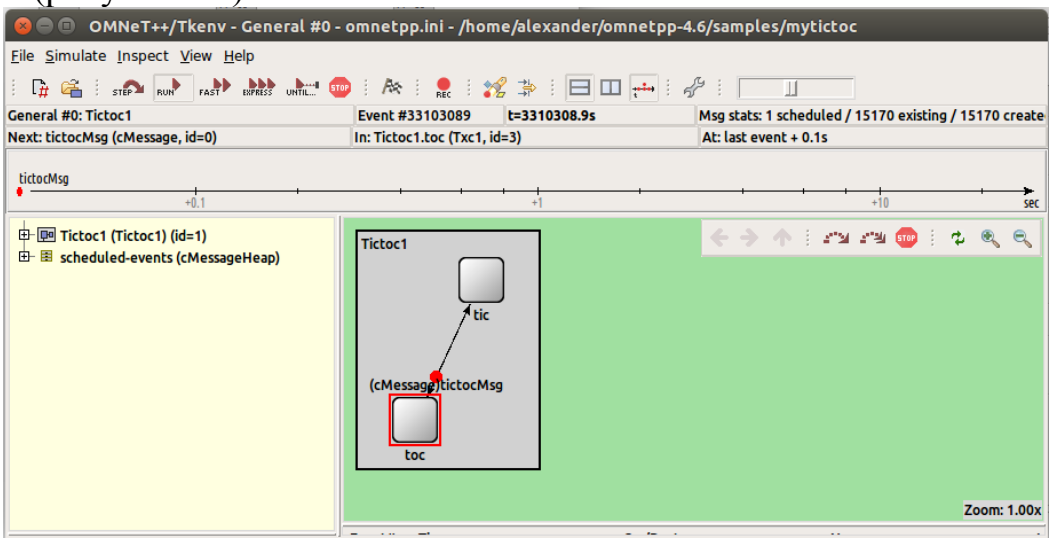


Рисунок 3.1 - Окно моделирования

На основной панели инструментов отображается модельное время. Это виртуальное время, оно не имеет ничего общего с реальным временем, это время, которое используется для выполнения процесса имитации. На самом

деле время, которое потребуется для выполнения моделирования зависит от производительности компьютера (аппаратного обеспечения) и сложности самой имитационной модели.

Обратите внимание, что для обработки сообщения узлом не затрачивается модельное время (нулевое время). Единственное на что тратится время в данном примере модели это задержка доставки сообщения по линии связи.

9. В программе моделирования можно замедлять или ускорять анимацию с помощью кнопок в верхней части графического окна. Можно остановить моделирование с помощью клавиши F8 (эквивалент кнопки «Stop» на панели инструментов), выполнять моделирование пошагово (F4), запустить анимацию (F5) или выключить ее (F6). Режим F7 (экспресс-режим) полностью отключает функции трассировки для обеспечения максимальной скорости. В строке состояния отображаются показатели скорости выполнения процесса моделирования событие/с и simsec/c (отображаются, когда модель работает в режимах «fast» или «express»).

10. Из программы моделирования можно выйти, щелкнув значок Закрыть или выбрав в меню File | Exit.

Резюме. На данном этапе, можно сказать что, мы прошли весь путь создания имитационной модели, практически «с нуля». Слово практически все же стоит оставить, т.к. за уровень «нуля» можно принимать различные вещи. Мы построили модель сети и определили тип простого модуля, т.е. минимально возможный строительный элемент модели. Однако, при его определении мы использовали родительский класс «cSimpleModule» (из библиотеки OMNeT++, его определение находится в файле csimplemodule.h), от которого наш новый тип «Txc1» унаследовал свойства и методы (), и нам не пришлось их определять самим (часть работы уже была сделана за нас авторами системы моделирования). Это замечательное качество (полиморфизм) языка объектно-ориентированного языка (C++) позволяет значительно облегчить процесс построения модели. (Но для этого нужно знать состав, свойства и методы классов из библиотек OMNeT++.)

Итак, обобщим то, что мы сделали:
-создали сеть (NED файл);
-создали тип простого модуля (файл C++);
-создали конфигурацию модели (файл ini);
-откомпилировали и скомпоновали программу (op_makemake, make);
-запустили имитационную модель (./mytictoc).

3.4 Построение модели СМО (использованием IDE)

Запуск среды. Запускаем интегрированную среду разработки (IDE) кликнув на значок OMNeT++ на рабочем столе или из командной строки, набрав строку /home/username/omnetpp-4.6/bin/omnetpp, где username имя каталога пользователя. При запуске будет выведено диалоговое окно с предложением указать (выбрать) рабочий каталог. По умолчанию будет предложен каталог /home/username/omnetpp-4.6/samples. Вы можете выбрать любой удобный для Вас каталог. Мы в данном примере согласимся с предложенным каталогом и нажмем кнопку OK. По истечении некоторого времени, которое зависит от производительности вашего компьютера, откроется главное окно Simulation – OMNeT++ - IDE. Главное окно содержит меню, панель инструментов (Toolbar) и несколько, вложенных окон.

IDE системы моделирования построено на основе Eclipce. В зависимости от умолчаний и предшествующих сеансов использования программы, IDE может быть запущена в различных конфигурациях. Для построения модели нам нужна конфигурация Simulation. Название конфигурации отображается в заголовке (имени) главного окна. Если конфигурация отличается от требуемой, то следует выбрать Меню –>Window->Open perspective -> Simulation. В заголовке окна будет отображаться Simulation – OMNeT++ IDE.

1. Создание проекта. Меню->File->New->OMNeT++ Project. В диалоговом окне мастера создания проекта укажем имя проекта в поле «Project name». Введем имя «lesson_01» (без кавычек). Нажимаем кнопку далее «Next». В следующем диалоговом окне предлагается выбрать шаблон модели. Выбор шаблона позволяет сократить количество последующих действий создания модели. В нашем примере мы не будем использовать шаблонов, а создадим пустой проект, выбрав пункт «Empty project» и нажав «Next». В следующем диалоге предлагается выбрать тип проекта «Project type» и набор инструментов «Toolchain», выбираем OMNeT++ Simulation и GCC for OMNeT++, соответственно, нажимаем «Next». В следующем диалоговом окне мы можем выбрать конфигурации, просто нажмем «Next».

После описанных действий в рабочей папке создается каталог проекта с именем lesson_01, который содержит файл package.ned. Данный файл содержит описание конфигурации сети.

2. Подключение библиотек. Определим предметную область и цель построения модели. Начнем с построения модели наиболее простой системы массового обслуживания (СМО), которая в принципе, может быть составной частью модели элемента сети связи. В качестве такой модели выберем модель M/M/1 [2,8]. Для ее построения целесообразно использовать простые модули библиотеки «queueinglib». Выберем в дереве проекта «Project explorer» каталог «lesson_01», нажимаем правую кнопку мыши и выбираем пункт свойства «properties» (или Меню->File->Properties). В диалоговом окне свойств выбираем «Project references», отмечаем галочкой «queueinglib», нажимаем «OK». В результате к проекту подключается библиотека очередей

«queueinglib». В IDE отображается окно редактора модели package.net, в котором на данный момент нет никаких элементов. В правой части экрана IDE (по умолчанию) размещено окно палитры «Palette». В этом окне можно выбрать режим работы: «Selector» выделение модулей или «Connector» соединение модулей. Также возможно выбрать типы создаваемых модулей «Types» и субмодули «Submodules».

3. Создание модели. На верхнем уровне иерархии модели находится модуль типа сеть «Network», модули всех остальных типов являются вложенными по отношению к нему. Чтобы построить модель и выполнить моделирование наш проект должен включать в себя модуль данного уровня. Выберем в окне «Types» тип «Network» щелчком левой клавиши мыши. Переведем указатель в область редактирования и вновь нажмем левую клавишу. В области редактора появится прямоугольник, который можно растянуть и разметить удобным для нас образом в области редактирования, далее в нем будем размещать другие модули нашей модели.

Для построения нашей модели нам потребуются несколько элементов, которые имитируют поток заявок (источник заявок), их обслуживание (обслуживающее устройство), ожидание в очереди (буфер) и завершение заявок (обработку обслуженных заявок). Эти элементы имеются в виде субмодулей в библиотеке «queueinglib». Для построения модели в окне «Palette» откроем папку «Submodules», в ней отображаются субмодули библиотеки «queueinglib». Если их нет в окне «Palette», то нужно перейти в окно «Project Explorer» и выбрать библиотеку «queueinglib», нажатием левой клавиши мыши, далее снова переходим в окно «Palette». Выберем субмодуль «Source» (источник). Разместим его в области, ограниченной прямоугольником, только что созданного модуля «Network». Аналогичным образом добавим субмодули «Queue» и «Sink». Субмодуль - очередь «Queue» имитирует два элемента СМО: саму очередь (буфер) и обслуживающее устройство. Обслуживание заявки имитируется ее задержкой, а очередь постановкой заявки на ожидание, если обслуживающее устройство занято (обслуживанием предыдущей заявки). Элемент «Sink» (слив) не несет функционального назначения в модели СМО, его роль состоит в завершении тех заявок, которые покинули обслуживающее устройство, т.е. были обслужены и далее не нужны для анализа ее функционирования.

Соединим размещенные в редакторе субмодули между собой. Для этого выберем в окне «Palette» режим «Connection». Наведем указатель мыши на субмодуль «Source» и кликнем левую клавишу мыши. Переведем указатель на субмодуль «Queue», при перемещении за указателем буде следовать соединительная линия, «закрепленная» в элементе «Source», снова нажмем левую клавишу. При этом появится всплывающее окно, в котором будут отображаться варианты подключения, т.е. перечень шлюзов элемента «Queue», к которым может быть подключена линия. В данном случае имеется только один вариант «source.out -> queue.in++», выберем его. После этого линия связи будет зафиксирована, а выход «Source» будет подключен к

входу «Queue». Выполним аналогичные действия для подключения выхода «Source» к входу «Sink» (рисунок 3.2).

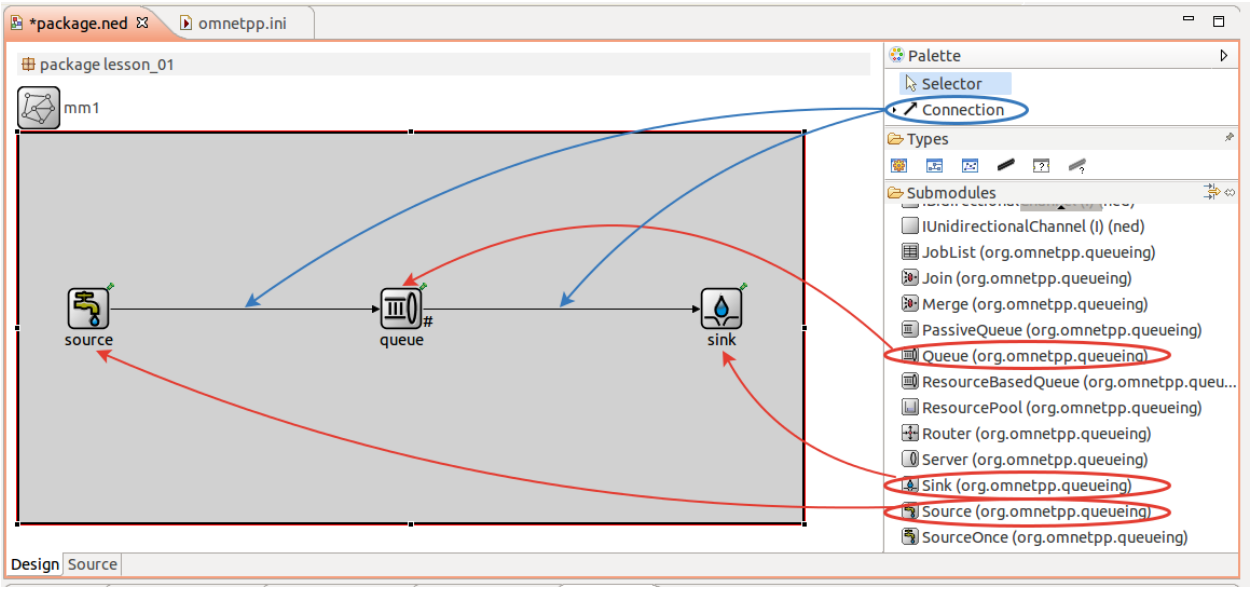


Рисунок 3.2 - Построение модели

После этого нашу модель можно считать построенной. Вернее сказать, мы определили ее состав и связи между ее элементами. Но перед запуском модели требуется сделать еще пару шагов: во-первых необходимо определить значения параметров тех субмодулей, из которых мы собрали модель, а также определить какие данные и каким образом мы хотим получить о работе нашей модели.

4. Определение параметров модели. Для определения параметров модели нужно создать ini файл, содержащий описание параметров модели. Для этого выберем Меню->File->New->Initialization File (ini). В окне редактора будет открыта форма, в которой можно определить параметры модели. В левой части формы выберем пункт «General» в правой части можно задать такие параметры как: имя сети, для которой выполняется модель «Network to simulate», ограничение максимального значения модельного времени «Simulation time limit», использование процессора «CPU time limit», точность представления модельного времени «Simulation tame precision». Определим в этой форме «Network to simulate» введя в соответствующее поле «Network», т.е. название нашего модуля верхнего уровня. В поле «Simulation time limit» введем число 20000s, что будет верхним пределом модельного времени, по истечении которого моделирование будет автоматически остановлено.

Выберем в левой части формы пункт «Parameters» в правой части формы отображается таблица, в которую мы можем добавить необходимые параметры субмодулей нашей модели. Нажмем кнопку «Add…», откроется диалоговое окно, в котором приведен перечень всех параметров используемых в модели субмодулей. По умолчанию, все параметры выбраны (отмечены галочкой). В верхней части формы можно выбрать опции для

различных способов отображения (фильтрации) списка параметров. Отменим выбор всех параметров, нажатием клавиши «Deselect all» внизу формы. Среди всех параметров выберем «**.source.InterArrivalTime» и «**.queue.serviceTime», нажмем кнопку «OK». После этого в таблицу будут добавлены два выбранных параметра. Первый параметр означает величину интервала между моментами поступления заявок, т.е. задает поток заявок. Второй параметр определяет продолжительность обслуживания заявки обслуживающим устройством. В поле «Value» этой таблицы нам следует задать численное значение или выражение для определения параметров. Для параметра «**.source.InterArrivalTime» в поле «Value» введем выражение «exponential(1.25s)» (без кавычек), а для параметра «**.queue.serviceTime» введем «exponential(1s)». Введенные выражения означают то, что значения параметров являются случайными числами, имеющими экспоненциальное распределение вероятности со средним значением 1,25 с и 1 с, соответственно. Таким образом, мы определили модель СМО M/M/1, с нагрузкой 1/1,25=0,8.

4. Имитационный эксперимент. Теперь можно приступить к исполнению процесса имитационного моделирования. Выберем Меню->Run->Run, откроется форма, в которой можно определить или выбрать различные конфигурации. В нашем случае согласимся со всеми умолчаниями и нажмем кнопку «Run». После этого запускается процесс компиляции модели и ее запуск, это может занять ощутимое время, в зависимости от производительности компьютера. После запуска откроется окно имитационного эксперимента, в котором будет представлено изображение нашей модели, а также окна с дополнительной информацией о ее работе. В панели инструментов нажимаем кнопку «Run», после этого начинается процесс имитации. Мы можем управлять скоростью выполнения имитационного процесса, выбирая возможные опции в панели инструментов «Run», «Fast run», «Express» или «Until …». Также возможно выполнять моделирование пошагово с помощью клавиши «Step». Нажмем «Run», при этом запустится процесс моделирования и анимации. В окне модели мы сможем наблюдать процесс передачи заявок между ее элементами (рисунок 3.3).

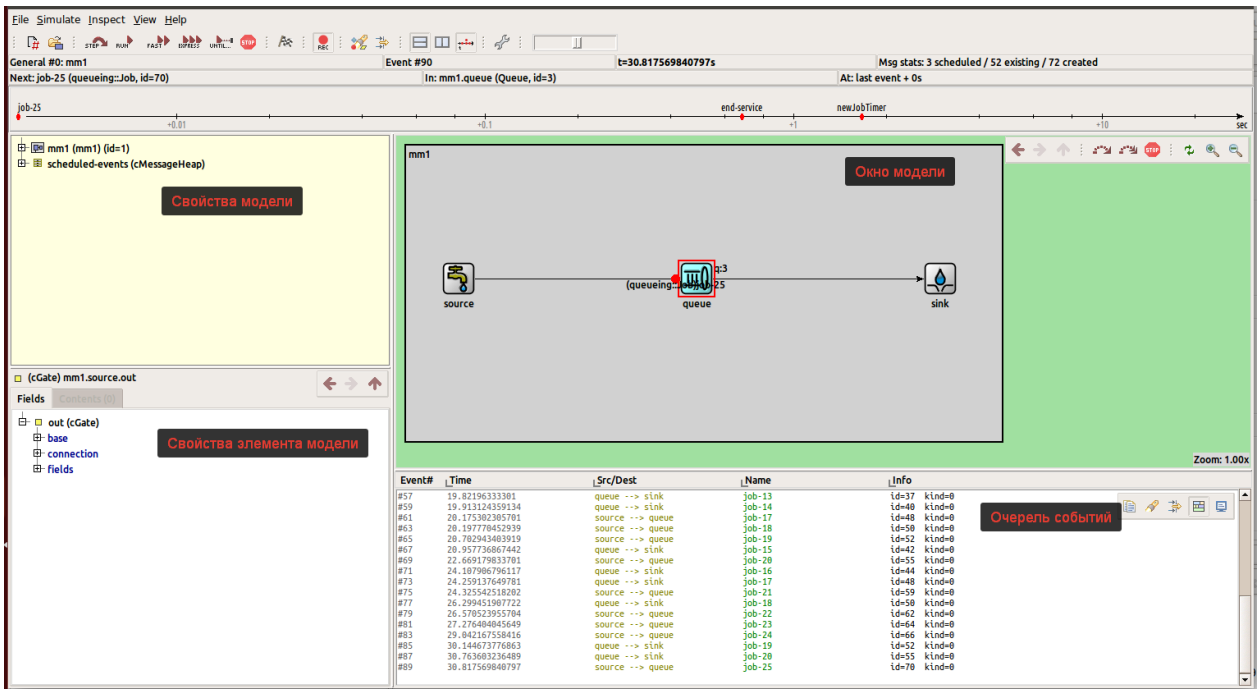


Рисунок 3.3 - Окно имитационного эксперимента

Под окном модели расположено окно, в котором отображается очередь событий модели. В ней мы можем найти подробную информацию о каждом событии, произошедшем в процессе моделирования. В верхней части окна графически представлена ось времени, на которой точками отображаются события, происходящие в ближайшем времени, по отношению к текущему модельному времени. Слева расположены окна свойств модели, модуля или события. Выделяя элемент в окне модели, мы можем посмотреть его свойства в процессе моделирования. Двойной щелчок по модулю в окне модели открывает его содержимое. В нашем случае используются простые модули (низкого уровня), поэтому при их открытии мы увидим лишь пустой прямоугольник. Вернуться обратно можно через контекстное меню, вызываемое нажатием правой кнопки мыши, в контекстном меню выбираем «Go Up».

Рассмотренный нами интерфейс имитационного эксперимента представляет весьма полные данные о работе модели и очень полезен при отладке модели.

Остановим выполнение модели кнопкой «Stop». После этого мы вновь можем запустить выполнение эксперимента, с того момента, когда он был остановлен, нажав одну из описанных выше кнопок запуска.

Кратко рассмотрим смысловое назначение очереди событий. Остановим эксперимент и выделим, например первое событие в очереди событий. В окне очереди событий мы видим поля: номер события, модельное время, модули источника и назначения, имя события, его идентификатор и вид события (рисунок 3.4).

Event#	Time	Src/Dest	Name	Info
#1	0.875461955022	source --> queue	job-1	id=2 kind=0
#3	1.863878382354	source --> queue	job-2	id=4 kind=0
#5	2.131392713069	queue --> sink	job-1	id=2 kind=0
#7	3.054615844539	queue --> sink	job-2	id=4 kind=0
#9	3.909443015636	source --> queue	job-3	id=8 kind=0
#11	4.696644155617	queue --> sink	job-3	id=8 kind=0
#13	6.056142885108	source --> queue	job-4	id=11 kind=0
#15	6.607191371778	queue --> sink	job-4	id=11 kind=0
#17	8.123003344731	source --> queue	job-5	id=14 kind=0

Рисунок 3.4 - Окно очереди событий

Первое событие произошло в момент, когда модельное время было равно 0,8754…, это событие соответствует выходу заявки из модуля «Source» и передачи ее на вход модуля «queue». Имя события «job-1», его идентификатор id=2, вид kind=0.

В окне свойств мы можем видеть перечень свойств данного события и их значения (рисунок 3.5).

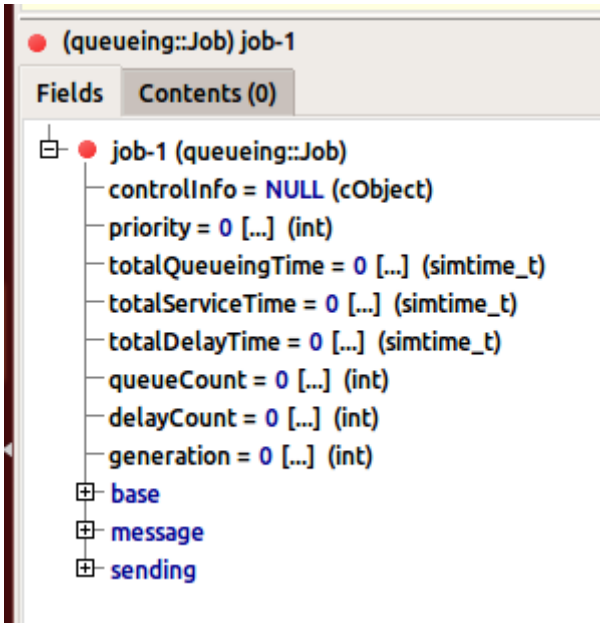


Рисунок 3.5 - Окно свойств

Например, мы видим приоритет «priority=0» события (мы его не задавали при построении модели), общее время задержки в очереди «totalQueueingTime=0», общее время обслуживания «totalServiceTime=0» они равны нулю, т.к. заявка только что покинула источник и нигде больше не была.

Посмотрим на второе событие. Оно соответствует выходу второй заявки из источника и ее передачи на вход очереди.

Рассмотрим третье событие в очереди. По идентификатору id=2, мы можем определить, что это первая заявка, только в данный момент событие соответствует ее выходу из модуля «Queue» и передача ее на вход модуля «Sink». Видим, что в окне свойств изменилось значение свойства «totalServiceTime=1.2559…», оно соответствует времени, на которое заявка была задержана в обслуживающем устройстве, в модуле «queue». Это время

является временем обслуживания, заявка не стояла в очереди «totalQueueingTime=0», это очевидно, т.к. при поступлении первой заявки очередь была пуста.

Изучая последовательность событий, мы можем выполнить анализ логики функционирования построенной модели и выявить возможные ошибки. Таким образом, интерфейс имитационного эксперимента весьма информативен и полезен при отладке новых и анализе имеющихся моделей.

Однако целью имитационного моделирования является получение статистических оценок параметров функционирования модели. Само по себе построение и отладка модели являются лишь средством достижения этого результата. Хотя очередь событий дает нам исчерпывающую информацию о работе модели, использовать ее для статистического анализа свойств модели может быть затруднительно, особенно в сложных моделях. Поэтому, далее рассмотрим средства анализа результатов эксперимента.

4. Анализ результатов эксперимента. Анализ результатов эксперимента является важным элементом процесса моделирования, т.к. именно для их получения и создается модель.

Для получения результатов эксперимента необходимо создать в проекте файл анализа «Analysis File». Для этого перейдем в наш проект, выбрав в дереве редактора проекта «lesson_01». Далее создаем этот файл, выбирая Меню->File->New-> Analysis File (anf), указав в окне диалога на папку «results»с результатами в папке проекта «lesson_01», т.е. путь lesson_01/results (рисунок 3.6).

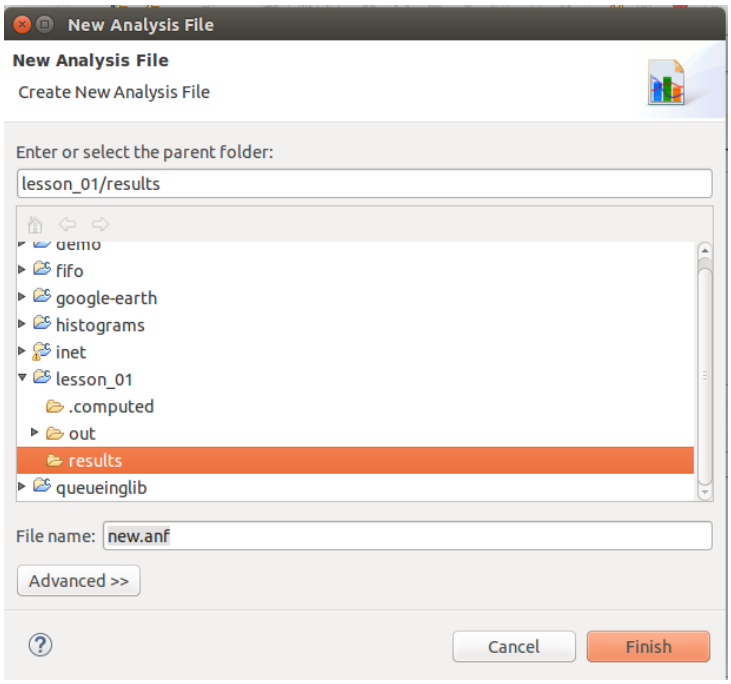


Рисунок 3.6 - Создание файла анализа

После создания файла в папке проекта появится файл с именем new.anf (имя по умолчанию). В редакторе форма для заполнения этого файла выглядит, как показано на рисунок 3.7.

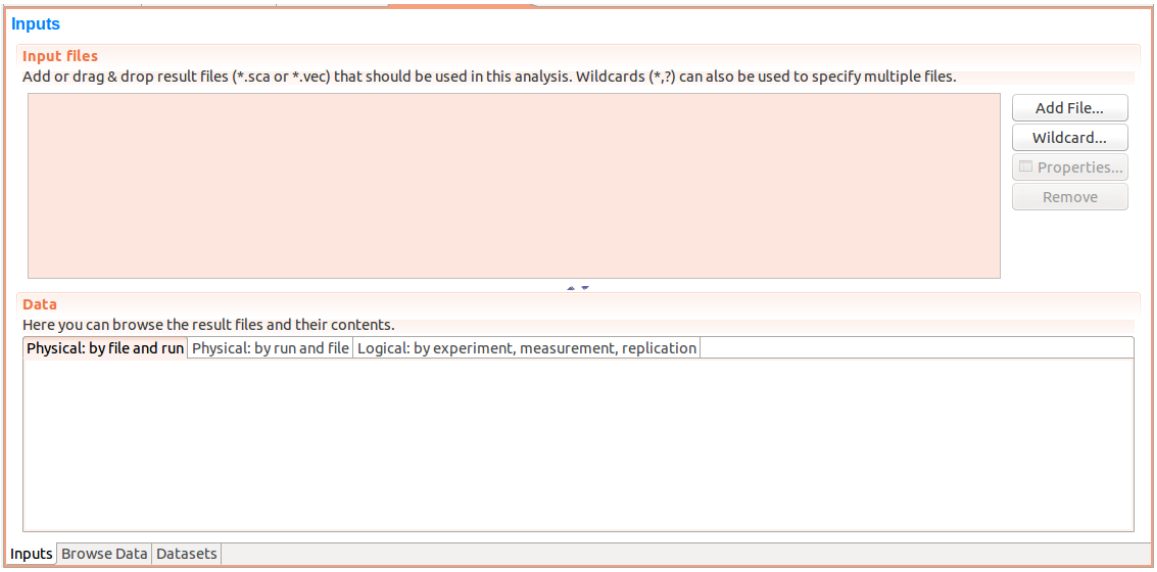


Рисунок 3.7 - Окно формы редактирования файла анализа

В форме имеются два поля: верхнее – «входные файлы» («Input Files») и нижнее «Данные» («Data»). В верхнее поле нам нужно добавить ссылки на файлы, которые являются входными для анализа, т.е. файлы с результатами эксперимента. Перечень этих файлов задается в файле инициализации (ini). В данный момент у нас есть файлы, созданные в результате эксперимента по умолчанию, это файлы скалярных значений («*.sca») и векторных значений («*.vec»). Для того чтобы выбрать эти файлы нажмем кнопку «Wildcard …» (фильтр по группе символов). В открывшемся диалоговом окне нажмем кнопку «OK». В поле «Input Files» будут добавлены два файла (рисунок 3.8).

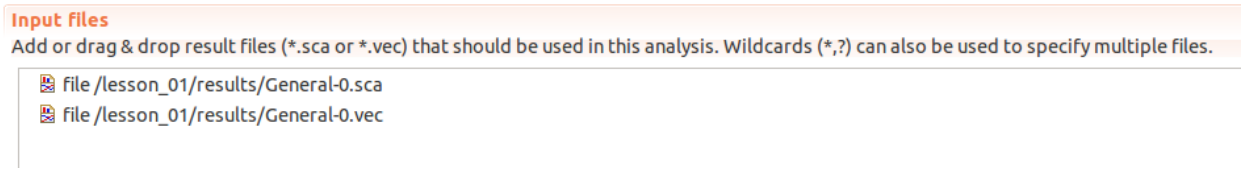


Рисунок 3.8 - Входные файлы

В поле «Data» будут добавлены две записи (рисунок 3.9).

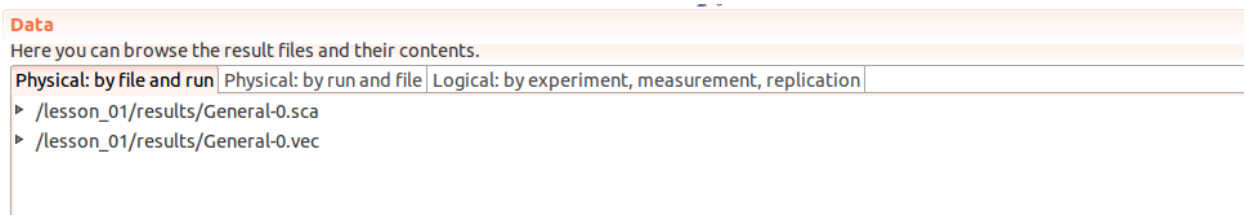


Рисунок 3.9 - Данные

Рассмотрим скалярные данные. Откроем двойным щелчком левой клавиши мыши первую строку в поле «Data». После чего перейдем в окно «Browse Data». Откроем записи «General», а затем «mm1.queue» (рисунок 3.10).

Мы можем видеть перечень скалярных параметров и их численных значений, полученных в результате измерений. В частности, обратим внимание на параметр «queueingTime:mean (Scalar)» и соответствующее значение 3,985… (среднее время ожидания в очереди). Если вспомнить аналитическую модель системы M/M/1 [2, 8], теоретическое значение этого параметра должно быть равно 4,0. Результаты моделирования представляют собой результаты измерения, проведенного на модели системы, и содержат ряд ошибок, как случайных, так и систематических. Эти ошибки обусловлены как не идеальностью самой модели, так и конечным размером выборки. Однако, сравнивая полученный результат с теоретическим значением, можно заметить, что в данном случае, отклонение от теоретического значения составило не более 0,02. В ряде случаев такое отклонение можно считать вполне допустимым. Стоит отметить, что при имитационном моделировании отклонение будет всегда. Поэтому, в дальнейшем, параметры имитационного эксперимента следует выбирать с учетом допустимой точности результатов.

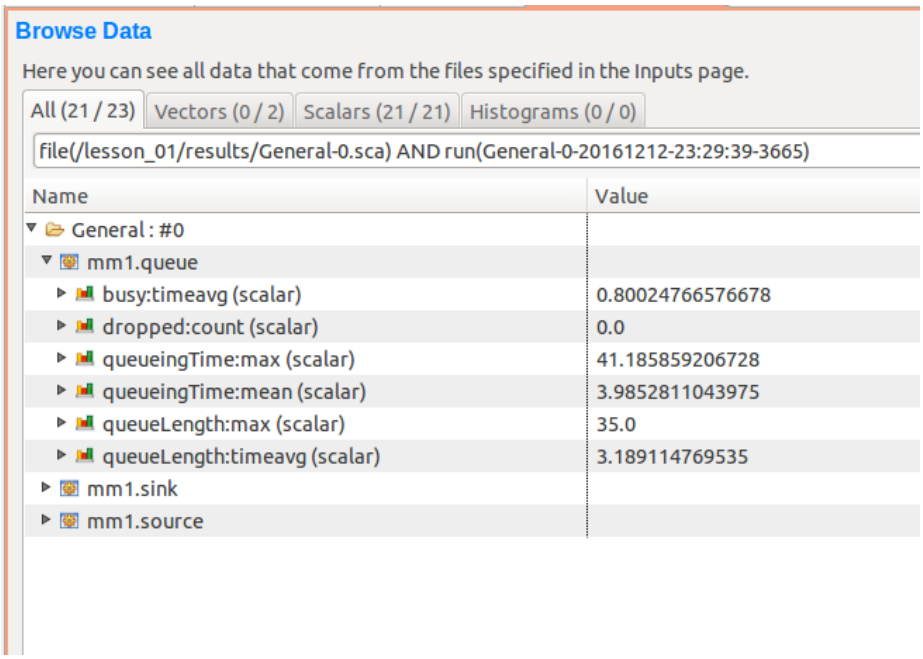


Рисунок 3.10 Скалярные данные

Кроме рассмотренного среднего времени ожидания в очереди в данном перечне приведены также и другие значения, например, нагрузка (использование) «busy:timeavg (Scalar)», максимальное время ожидания «queueingTime:max (Scalar)».

Аналогичным образом можем рассмотреть скалярные данные, полученные для модулей «sink» и «source».

Скалярные данные позволяют получить оценки параметров модели, измеренные за интервал модельного времени, равный продолжительности имитационного эксперимента. Если мы будем повторять один и тот же эксперимент (вновь и вновь запуская моделирование), то сможем заметить, что получаемые результаты будут идентичны результатам предшествующих экспериментов. При проведении измерений на реальной системе, мы бы не смогли бы получить идентичные результаты. В имитационной модели процессы не случайные, а псевдослучайные, они могут быть повторены абсолютно точно при определенной настройке генератора случайных чисел. Это может быть удобно при построении и отладке моделей. Псевдослучайная последовательность может быть абсолютно точно повторена при запуске генератора псевдослучайных чисел с одними и теми же начальными условиями. Так и происходит при повторном запуске процесса имитационного моделирования. Начальные условия старта генератора псевдослучайных чисел задаются начальным значением «Seed» в файле «ini» на вкладке «Random Numbers». Изменяя это значение, мы можем получить различные реализации псевдослучайной последовательности.

В полученных в результате эксперимента данных мы можем найти средние значения основных параметров модели. По ним мы можем судить о ее поведении. Наблюдаемые значения характеризуют исследуемые случайные величины. Среднее значение это лишь одна точечная оценка случайной величины. Во многих случаях, для получения достаточно полного представления о случайной величине ее недостаточно. Второй точечной оценкой, случайной величины является среднеквадратическое отклонение (стандартное отклонение) [20]. Она дает представление о разбросе значений и необходимо для получения интервальной оценки - доверительного интервала.

Для получения значений среднеквадратического отклонения изменим режим записи результатов «Results recording modes» в разделе «Result Recording» файла «ini». Введем в данное поле «default, stats», сохраним файл и повторим эксперимент. Снова откроем файл «anf» и перейдем на «Scalars». Просматривая таблицу результатов, мы можем найти среднее и среднеквадратическое значения параметров, рисунок 3.11.

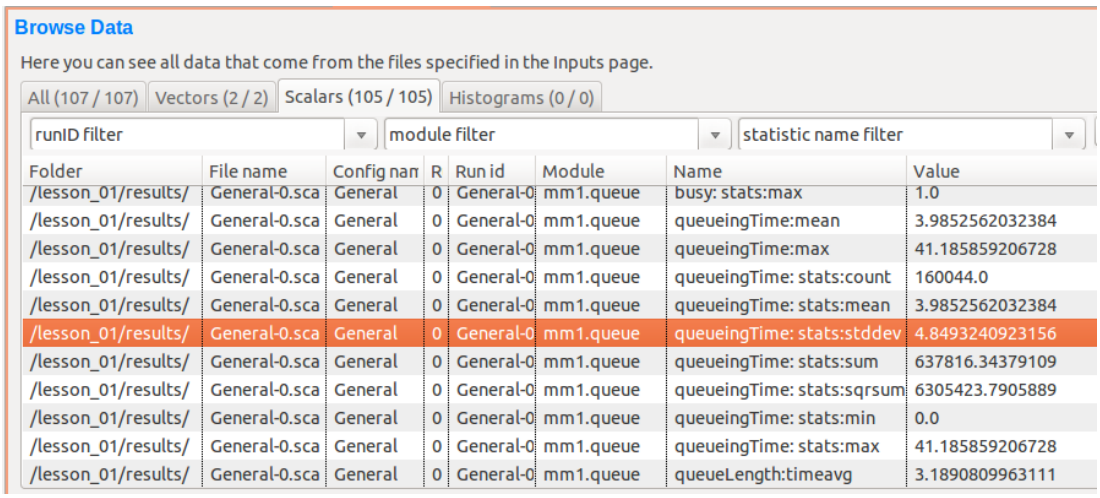


Рисунок 3.11 - Данные статистики

Например, «queingTime:stats:mean» и «queingTime:stats:stdev». Первое из них повторяет полученное ранее среднее время ожидания в очереди, а второе его среднеквадратическое отклонение. Значение «queingTime:stats:count» содержит общее число наблюдений, для которых были получены эти значения. Используя эти три значения, мы можем построить доверительный интервал для среднего времени ожидания в очереди [20,21]. Так для доверительной вероятности 0,95 получим 3,99±0,02 с (более подробно смотрите в разделе 4 «Обработка результатов имитационного моделирования»). Как видим полученный доверительный интервал [3,97; 4,01] с «накрывает» теоретическое значение (с вероятностью 0,95).

Наряду со скалярными данными в результате эксперимента собираются векторные данные (файл «*.vec»). Под векторными данными понимается набор связанных друг с другом значений. Например, длина очереди и модельное время. В этом случае значения длины очереди связаны со значениями модельного времени, которые представлены в хронологическом порядке. Перейдем в поле «Inputs» и откроем аналогичным образом записи векторных данных. Выберем векторные данные, как показано на рисунок 3.12.

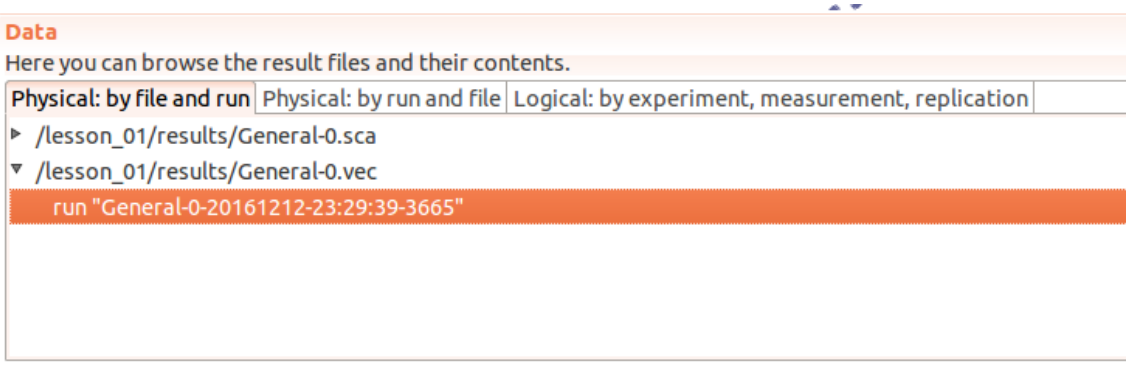


Рисунок 3.12 - Выбор векторных данных

Двойным щелчком левой клавиши мыши откроем обозреватель данных, рисунок 3.13.

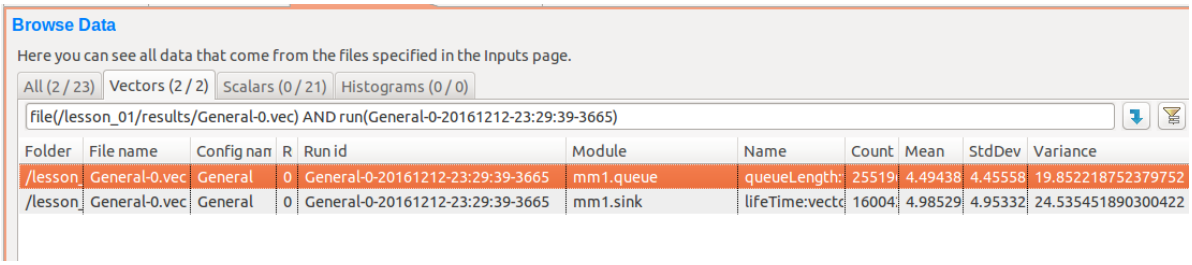


Рисунок 3.13 - Выбор векторных данных

Двойным щелчком левой клавиши мыши по верхней строке откроем график, построенный по данным для «mm1.queue». Эта диаграмма иллюстрирует изменение длины очереди во время проведения эксперимента, рисунок 3.14.

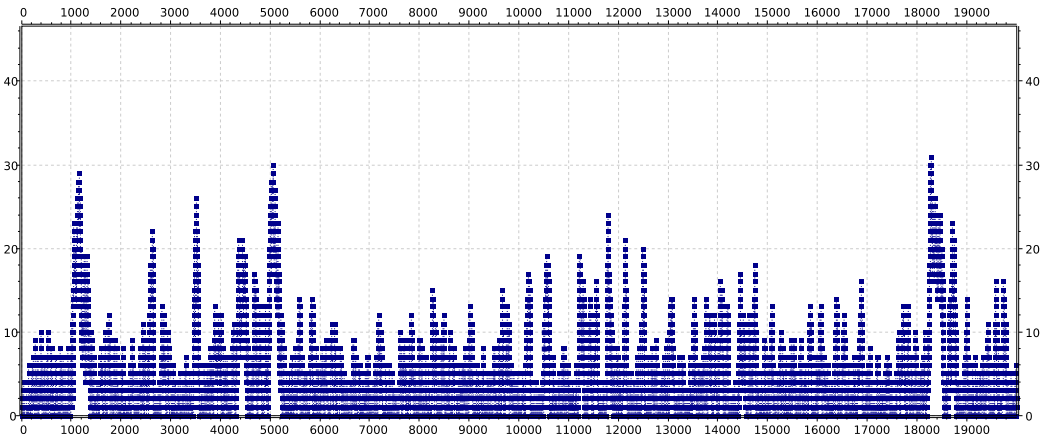


Рисунок 3.14 - Диаграмма изменения длины очереди во времени

Если вид приведенной диаграммы кажется не слишком презентабельным, то ее представление можно изменить, используя доступные опции. Наведем указатель на гистограмму и нажмем правую клавишу мыши, из контекстного меню выберем свойства «Properties…». Изменим тип линии «Line type» на «Pins», «Symbol type» изменим на «None». Исследуя свойства диаграммы можно изменить шрифт, подписи осей и масштаб. Теперь вид диаграммы изменится, рисунок 3.15.

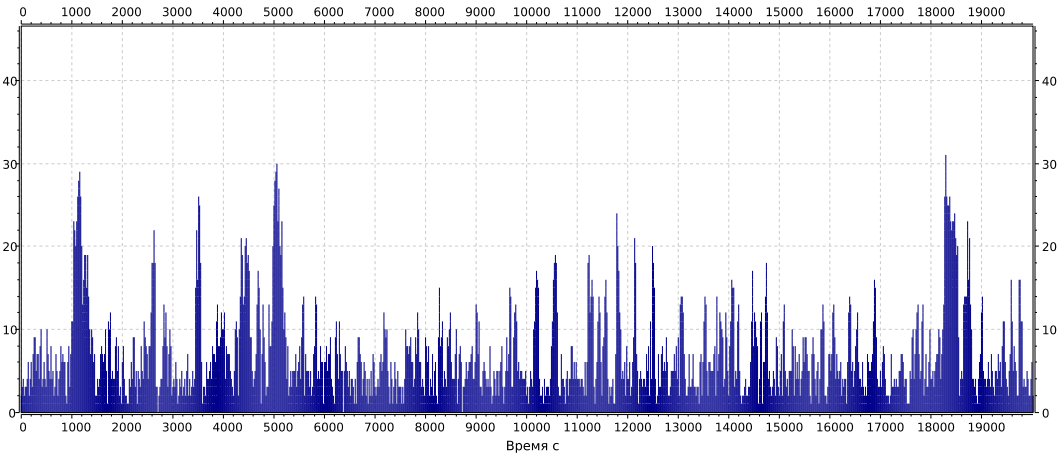


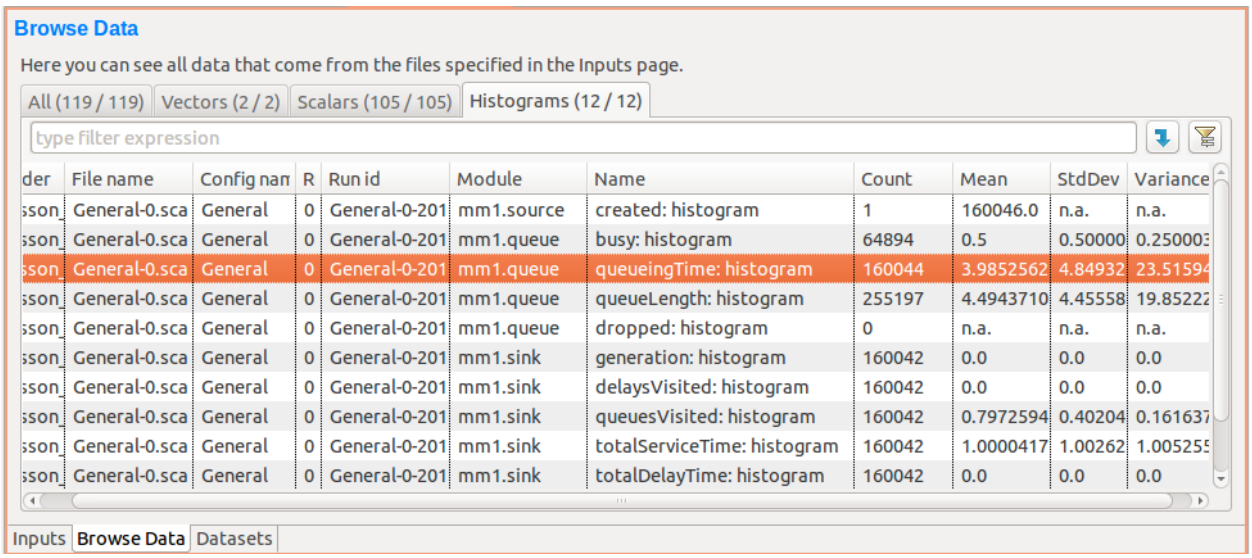
Рисунок 3.15 - Измененная диаграмма изменения длины очереди во времени

Если диаграмму планируется использовать в каких-либо документах, ее можно сохранить через контекстное меню в формате векторной графики (SVG), выбрав опцию «Export to SVG». Конечно, можно сделать «скриншот», но следует отметить, что этот (векторный) формат значительно удобнее, чем форматы растровых изображений, который получают с помощью «скриншотов». Главные его достоинства это масштабирование без потери качества изображения, а также относительно малый размер файла. Его достаточно легко преобразовать в другие необходимые форматы с помощью различных графических редакторов или свободных онлайн сервисов.

Рассмотренные выше возможности анализа позволяют получить численные точечные и интервальные оценки параметров модели, а также наблюдать динамику их изменения в течение модельного времени.

Мы уже отмечали, что значения исследуемых параметров являются случайными величинами. Наиболее полную характеристику случайной величины дает ее функция распределения или плотность вероятности [20]. В исследованиях в области связи наиболее часто используют плотность вероятности, т.к. она дает наглядное представление о случайной величине. Описание плотности вероятности по экспериментальным данным, т.е. получение эмпирической функции плотности вероятности производят путем построения гистограммы для значений случайной величины [20]. Такая возможность есть и в средствах анализа OMNeT++.

Для построения гистограммы перейдем в файл «ini», в раздел «Result Recording», поле «Results recording modes» впишем «default, histogram», сохраним файл и снова выполним эксперимент. Откроем файл «anf», вкладку «Histograms», рисунок 3.16.



der	File name	Config nam	R	Run id	Module	Name	Count	Mean	StdDev	Variance
sson	General-0.sca	General	0	General-0-201	mm1.source	created: histogram	1	160046.0	n.a.	n.a.
sson	General-0.sca	General	0	General-0-201	mm1.queue	busy: histogram	64894	0.5	0.50000	0.250003
sson	General-0.sca	General	0	General-0-201	mm1.queue	queueingTime: histogram	160044	3.9852562	4.84932	23.51594
sson	General-0.sca	General	0	General-0-201	mm1.queue	queueLength: histogram	255197	4.4943710	4.45558	19.85222
sson	General-0.sca	General	0	General-0-201	mm1.queue	dropped: histogram	0	n.a.	n.a.	n.a.
sson	General-0.sca	General	0	General-0-201	mm1.sink	generation: histogram	160042	0.0	0.0	0.0
sson	General-0.sca	General	0	General-0-201	mm1.sink	delaysVisited: histogram	160042	0.0	0.0	0.0
sson	General-0.sca	General	0	General-0-201	mm1.sink	queuesVisited: histogram	160042	0.7972594	0.40204	0.161637
sson	General-0.sca	General	0	General-0-201	mm1.sink	totalServiceTime: histogram	160042	1.0000417	1.00262	1.005255
sson	General-0.sca	General	0	General-0-201	mm1.sink	totalDelayTime: histogram	160042	0.0	0.0	0.0

Рисунок 3.16 - Выбор данных гистограммы

Выберем данные «queueingTime:histogram» и сделаем двойной клик левой клавиши мыши. В результате откроется окно гистограммы, рисунок 3.17. Внешний вид гистограммы также можно несколько изменить, как было описано выше. Если средство построения гистограммы нас не устраивает, по каким либо причинам, мы можем экспортировать данные гистограммы в файл с разделением полей данных запятыми «*.csv». Это можно сделать через контекстное меню (открываемое правой клавишей мыши) из вкладки «Histograms».

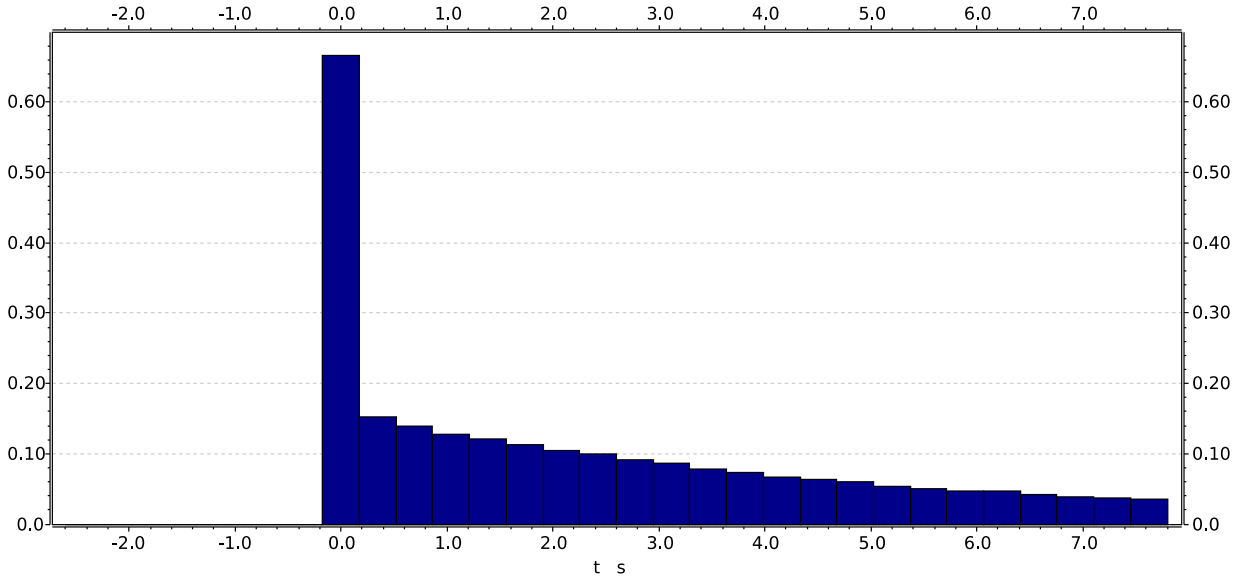


Рисунок 3.17 - Гистограмма времени ожидания в очереди

Изменяя свойства гистограммы, мы можем выбрать различные виды ее представления: количество наблюдений, плотность вероятности «pdf» или

функция распределения «cdf», более подробно эти варианты рассмотрены в разделе 4 «Обработка результатов имитационного моделирования».

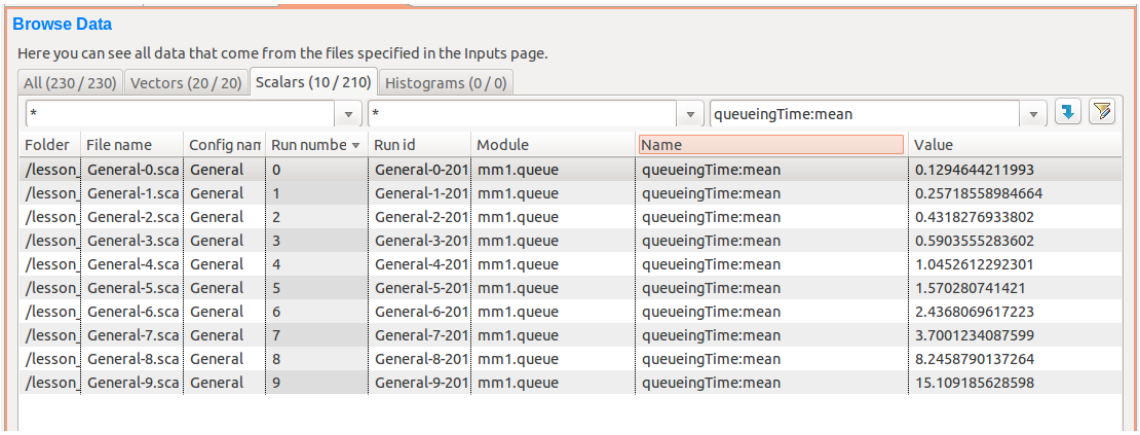
Рассмотренные выше способы представления параметров и характеристик позволяют достаточно полно описать значения параметров, получаемые в результате моделирования.

В ряде случаев требуется исследование зависимости одной или нескольких оценок от параметров модели. Тогда, целесообразно провести несколько испытаний с разными значениями входных параметров модели. Система моделирования позволяет это сделать в рамках одного эксперимента, описав требуемую последовательность в файле инициализации.

5. Серия испытаний. Последовательность испытаний может быть определена в файле «ini». Например, нам нужно исследовать зависимость времени ожидания в очереди от интенсивности нагрузки. Согласно аналитической модели M/M/1 [2,8], интенсивность трафика может изменяться от 0 до величины близкой к единице. Исследуемая величина непрерывна, это значит, что для ее полного описания нам нужно выполнить бесконечно большое число измерений, что практически невозможно. Поэтому результатом может быть только аппроксимация исследуемой зависимости, построенная по конечному числу значений. Для исследования выберем 10 значений интенсивности нагрузки в диапазоне от 0,1 до 0,95. Значение 0 мы выбрать не можем, т.к. в этом случае заявки вообще не будут создаваться, и в модели не будет происходить никаких событий, следовательно, и модельное время не будет изменяться. Такая модель не будет работать. Значение 1 мы могли бы выбрать, но в этом случае число заявок в очереди будет непрерывно расти на протяжении всего испытания, что также может отрицательно сказаться на функционировании модели. Поэтому, ограничимся набором значений интенсивности нагрузки {0,1; 0,2; 0,3; 0,4; 0,5; 0,6; 0,7; 0,8; 0,9; 0,95}. В «ini» файле нам нужно указать средние значения интервала времени между заявками, которые можно вычислить как отношение среднего времени обслуживания (которое у нас равно 1c) к интенсивности трафика. Тогда получим следующий набор средних значений интервалов времени между заявками {1/0,1; 1/0,2; 1/0,3; 1/0,4; 1/0,5; 1/0,6; 1/0,7; 1/0,8; 1/0,9; 1/0,95}. Запишем эту последовательность в поле «Value» параметра, «**.source.InterArrivalTime» в форме файла «ini» в следующем виде «exponential(${10s, 5s, 3.33s, 2.5s, 2s, 1.67s, 1.43s, 1.25s, 1.11s, 1.05s}). Сохраним файл «ini». Откроем конфигурации эксперимента Меню->«Run configurations …». В форме выбора конфигурации, в поле «Run number» введем символ «*» (звездочка), в области «Options» выберем «Command line». При такой конфигурации эксперимент будет запускаться на максимальной скорости, в отличие от предыдущего эксперимента, не будет отображаться интерфейс пользователя, а количество испытаний будет равно количеству введенных нами значений интенсивности трафика, т.е. 10. О ходе выполнения задачи мы можем судить по индикатору в панели информации в нижней части окна. После окончания эксперимента будут получены 10

файлов результатов «General-X.sca» и 10 фалов «General-X.vec», где X – номер испытания. Кроме этих файлов также будут сгенерированы файлы «*.vci» и «*.elog».

Откроем файл анализа «*.anf», выберем вкладку «Scalars» и выберем в поле фильтра «queueingTime:mean» (рисунок 3.18).



Folder	File name	Config nam	Run numbe	Run id	Module	Name	Value
/lesson_	General-0.sca	General	0	General-0-201	mm1.queue	queueingTime:mean	0.1294644211993
/lesson_	General-1.sca	General	1	General-1-201	mm1.queue	queueingTime:mean	0.25718558984664
/lesson_	General-2.sca	General	2	General-2-201	mm1.queue	queueingTime:mean	0.4318276933802
/lesson_	General-3.sca	General	3	General-3-201	mm1.queue	queueingTime:mean	0.5903555283602
/lesson_	General-4.sca	General	4	General-4-201	mm1.queue	queueingTime:mean	1.0452612292301
/lesson_	General-5.sca	General	5	General-5-201	mm1.queue	queueingTime:mean	1.570280741421
/lesson_	General-6.sca	General	6	General-6-201	mm1.queue	queueingTime:mean	2.4368069617223
/lesson_	General-7.sca	General	7	General-7-201	mm1.queue	queueingTime:mean	3.7001234087599
/lesson_	General-8.sca	General	8	General-8-201	mm1.queue	queueingTime:mean	8.2458790137264
/lesson_	General-9.sca	General	9	General-9-201	mm1.queue	queueingTime:mean	15.109185628598

Рисунок 3.18 - Данные серии испытаний

В окне будут представлены значения среднего времени ожидания в очереди, полученные в результате 10 испытаний при различных значениях интенсивности нагрузки (численные значения могут отличаться от значений, приведенных на рисунке). Порядковый номер испытания показан в поле «Run number».

Эти данные также можно представить графически или экспортировать во внешние приложения. Более подробно об этом в разделе 4 «Обработка результатов имитационного моделирования»

6. Резюме. Обобщая рассмотренный выше процесс создания модели, можем резюмировать, что при описании модели необходимо создать и наполнить содержимым три файла:

1. Файл модели (*.ned);
2. Файл инициализации (*.ini);
3. Сценарий эксперимента
4. Файл анализа (*.anf).

Текстовое описание модели.

Рассмотренный выше способ построения модели с использованием IDE нагляден, достаточно интуитивно понятен и позволяет избежать многих ошибок. Но иногда все же может оказаться быстрее написать или изменить модель, редактируя файлы проекта в текстовом формате. Это можно сделать и в IDE, выбрав внизу окна редактора вкладку «Source». Посмотрим на содержимое файла «packege.ned».

Содержимое файла «lesson_01.ned»

```
package lesson_01;                          //#Пакет проекта

import org.omnetpp.queueing.Queue;          //#Ссылка на субмодуль Queue
import org.omnetpp.queueing.Sink;           //#Ссылка на субмодуль Sink
import org.omnetpp.queueing.Source;         //#Ссылка на субмодуль Source

@license(LGPL);                             //#Текст лицензии LGPL
//
// TODO documentation                       Документация к проекту
//
network mm1                                 //#Начало описания модуля Network
{
   @display("bgb=726,360");                 //#Ширина и длина на экране
   submodules:                              //#Подключаемые субмодули
      source: Source {                      //#Субмодуль источника
         @display("p=64,162");              //#Координата в модуле Network
      }
      queue: Queue {                        //#Субмодуль очереди
         @display("p=354,162");             //#Координата в модуле Network
      }
      sink: Sink {                          //#Субмодуль Sink
         @display("p=651,162");             //#Координата в модуле Network
      }
   connections:                             //#Слединения
      queue.out --> sink.in++;              //#Вых. очереди соединить с вх. sink
      source.out --> queue.in++;            //#Вых. источника соединить со вх. sink
}                                           //#Конец описания модуля Network
```

Синтаксис файла напоминает синтаксис программы на языке С. В начале файла объявляется пакет «lesson_01», следующие три строки выполняют импорт трипов Queue, Sink и Sours, т.е. используемых в модели модулей. Далее текст лицензии. Комментарии в файле определяются двумя слешами «//». Следует отметить, что такие комментарии будут включены в документацию, которая создается в составе модели. Если комментарий не следует включать в документацию, его следует начинать символами «//#».

Далее следует определение модуля «lesson_01» (тип network). Область модуля ограничена фигурными скобками. Первая строка в области модуля определяет параметры его отображения «@display("bgb=726,360")» ширину и высоту прямоугольника, которым отображается модуль в графическом редакторе и в графическом интерфейсе при выполнении имитации. Далее следует секция «submodules:» в которой приводятся описания включенных в модель субмодулей «Source», «Queue» и «Sink». В каждом определении аналогично приводятся параметры отображения, в которых указаны относительные координаты расположения графических символов в области модуля. За определением субмодулей следует определение связей «connections:». Односторонняя связь между шлюзами элементов определяется символами «-->». Если шлюз допускаем подключение нескольких связей, например, как входя очереди или элемента «Sink», то после его описания ставится знак инкремента «++» (два знака плюс).

Рассмотрим файл инициализации «ini».

Файл omnetpp.ini

```
[General]
network = mm1
cmdenv-runs-to-execute = *
cmdenv-status-frequency = 500s
record-eventlog = true
sim-time-limit = 200000s
warmup-period = 1s
**.param-record-as-scalar = false
**.result-recording-modes = default, histogram
**.source.interArrivalTime = exponential(1.25s )
**.queue.serviceTime = exponential(1s)
```

В файле приведены: имя раздела «[General]», имя нашей модели «mm1», какие испытания (прогоны) должны быть выполнены «cmdenv-runs-to-execute», период выведения состояния на консоль «cmdenv-status-frequency», включить запись log-файла событий, ограничение модельного времени, время прогрева «warmup-period», отключение записи параметров, установка режима записи результатов «default, histogram», определение параметра «interArrivalTime» источника заявок и времени обслуживания «serviceTime».

Более подробно о содержимом файла «ini» в описании OMNeT++.

3.5 Модель канального уровня

3.5.1 Канальный уровень

В общем случае, задача протокола канального уровня состоит в доставке данных непосредственно между сетевыми устройствами. Доставляемые порции данных на канальном уровне принято называть кадрами. К функциям канального уровня можно отнести:

-доступа к среде передачи,

-определение границ кадра,

-адресация источников и получателей,

-обеспечение достоверности принимаемых данных,

-адресация протокола верхнего уровня.

В зависимости от уровня сети (локальный или глобальный) набор функций канального уровня различен.

Функции доступа к среде (управления доступом) характерны для протоколов локальных сетей. Эти функции обеспечивают совместное использование общей среды передачи (распространения сигнала). В этом случае, в канальном уровне выделяют два подуровня:

-управление логическим каналом (Logical Link Control, LLC).

-управление доступом к среде (Media Access Control, MAC),

Подуровень LLC обеспечивает достоверность передачи данных, а также обеспечивает взаимодействие с протоколом сетевого уровня.

Подуровень MAC выполняет функции управления доступом к общей среде.

Протоколы канального уровня ориентированы на строго определенную топологию сети. К топологиям локальных сетей относятся общая шина, кольцо и звезда.

Примеры протоколов канального уровня для локальных сетей Token Ring, Ethernet, Fast Ethernet.

В сетях глобального уровня канальный уровень обеспечивает обмен сообщениями между двумя сетевыми устройствами, соединенными индивидуальной линией связи, т.е. с топологией "точка-точка". Примерами таких протоколов являются PPP, SLIP, LAP-B, LAP-D.

Сетевые устройства, взаимодействуя на канальном уровне выполняют следующие действия:

-Формирование кадра. Кадр содержит данные, принятые от протокола верхнего уровня, а также заголовок. Кадр содержит контрольную сумму, MAC-адреса отправителя и получателя, данные о типе протокола верхнего уровня, пакет которого упакован в поле данных кадра, и возможно некоторую другую информацию. Кадры могут быть информационными или служебными. Служебные кадры формируются различными сетевыми устройствами.

-Анализ заголовка и содержимого кадра. В результате анализа может быть обнаружена ошибка передачи и др.

-Прием и отправка кадра связаны с функциями управлением доступом к среде передач. Принимаемые кадры помещаются в буфер приемника, а при отправке - выбираются из буфера передатчика.

3.5.2 Построение модели LAN

На канальном уровне мы можем оперировать кадрами, т.е. моментами их отправки, размером, адресами назначения. Адресом назначения является MAC адрес сетевого устройства (в данном случае компьютера). Характер трафика в этом случае полностью определяется процессом отправки кадров и их характеристиками, т.к. в данном случае работает только протокол канального уровня и не участвуют никакие иные протоколы, которые могли бы внести изменения в процесс обмена данными.

Построим модель, состоящую из двух компьютеров, которые соединены между собой с помощью линии связи стандарта Ethernet, а именно 10 Мбит Ethernet. Вид модели в редакторе приведен на рисунке 3.19.

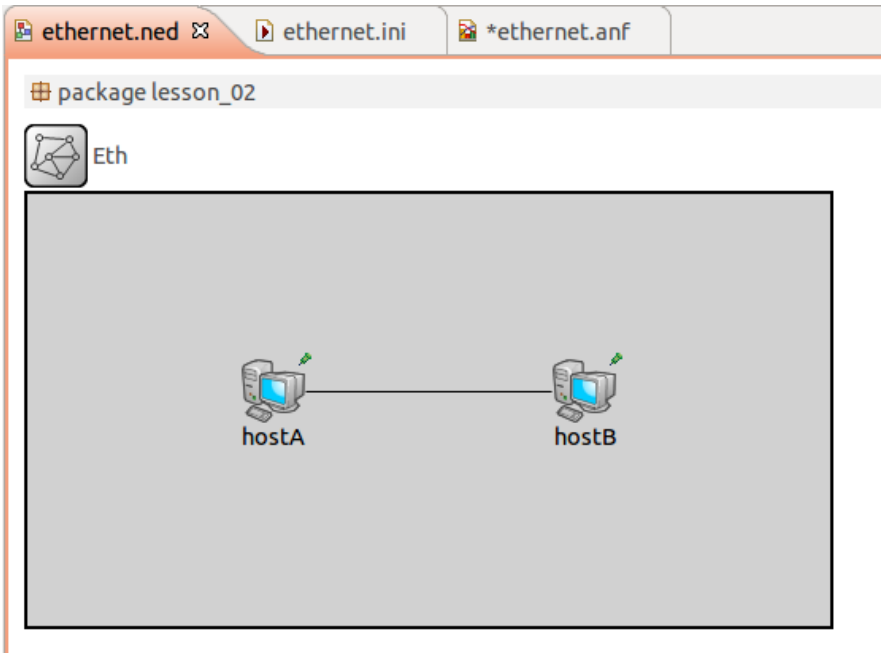


Рисунок 3.19 - Модель LAN

Содержимое файла NED приведено ниже.

```
package lesson_02;
import inet.node.ethernet.Eth10M;
import inet.node.ethernet.EtherHost;

//
// TODO documentation
//
network Eth
{
    @display("bgb=506,272");
```

```
    submodules:
        hostA: EtherHost {
            @display("p=136,58");
        }
        hostB: EtherHost {
            @display("p=365,108");
        }
        etherSwitch: EtherSwitch {
            @display("p=247,107");
        }
        hostC: EtherHost {
            @display("p=136,160");
        }
    connections:
        hostA.ethg <--> Eth10M <--> etherSwitch.ethg++;
        etherSwitch.ethg++ <--> Eth10M <--> hostB.ethg;
        hostC.ethg <--> Eth10M <--> etherSwitch.ethg++;
}
```

Содержимое файла «ini» приведено ниже.

```
[General]
network = lesson_02.Eth

sim-time-limit = 180s
tkenv-plugin-path = ../../../etc/plugins
**.vector-recording = true

**.hostA.cli.destAddress = "hostB"
**.hostB.cli.destAddress = "hostA"

**.cli.sendInterval = exponential(1s)

**.mac.address = "auto"

**.cli.reqLength = intuniform(50,1400)*1B
**.cli.respLength = truncnormal(3000B,3000B)
```

В файле определено время моделирования (180 с), запись векторного файла, адреса назначения отправляемых кадров, распределение времени между моментами отправки (экспоненциальное со средним значением 1 с). Распределение длины кадра содержащего запрос определено как равномерное в диапазоне от 30 до 1400 байт. Распределение длины кадра ответа определено как усеченное нормальное со средним значением 3000 байт и среднеквадратическим отклонением 3000 байт.

Выполнив имитационный эксперимент, получим результаты моделирования. В частности, ниже приведены графики отражающие характер потока (рисунок 3.20), изменение времени доставки (рисунок 3.21) и распределение времени доставки (рисунок 3.22).

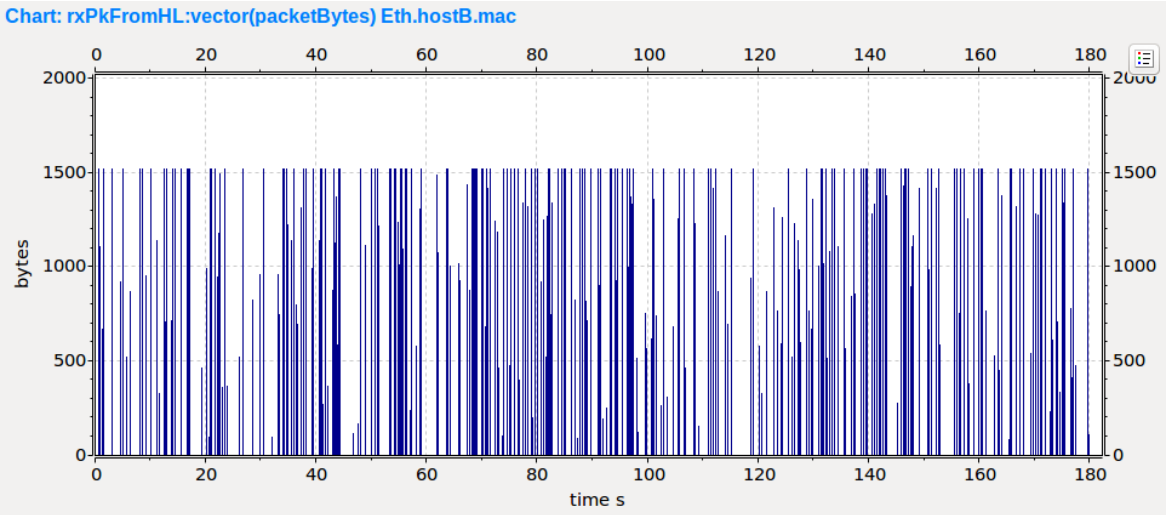


Рисунок 3.20 - Реализация потока кадров

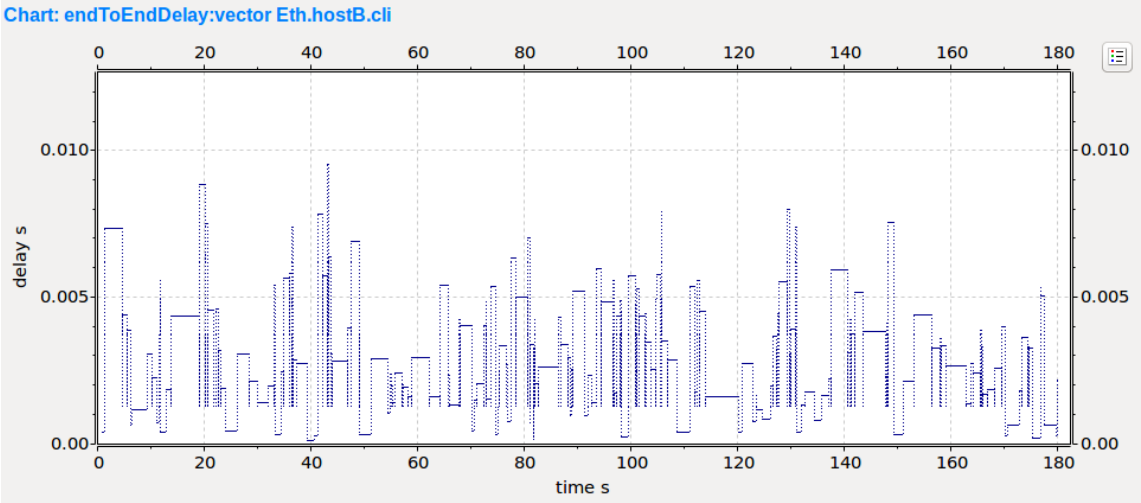


Рисунок 3.21 - Задержка из конца в конец

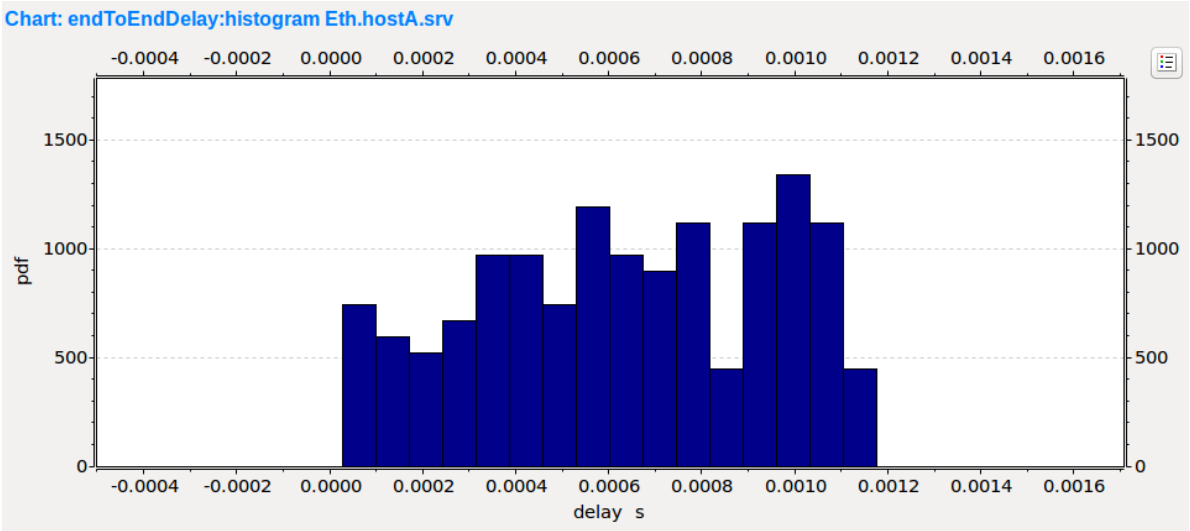


Рисунок 3.22 - Распределение времени доставки запроса

Интервал времени между моментами отправки кадров выбран настолько большим (1 с, или иначе говоря малая интенсивность трафика), что среднее время ожидания в буфере передатчика близко к нулю. Следовательно, время доставки кадра определяется временем его передачи по линии связи, т.е. длиной кадра и скоростью передачи данных. Так как скорость передачи постоянна, то распределение времени доставки определяется распределением длины кадра. Как видно из последнего рисунка (рис.), распределение времени доставки визуально напоминает равномерное распределение. В данном случае, заметное отличие от плотности вероятности равномерного распределения обусловлено не достаточно большим количеством переданных кадров.

Изменим конфигурацию сети, введя в нее коммутатор (Ethernet Switch) и еще один компьютер (hostC). Конфигурация сети приведена на рисунке 3.23.

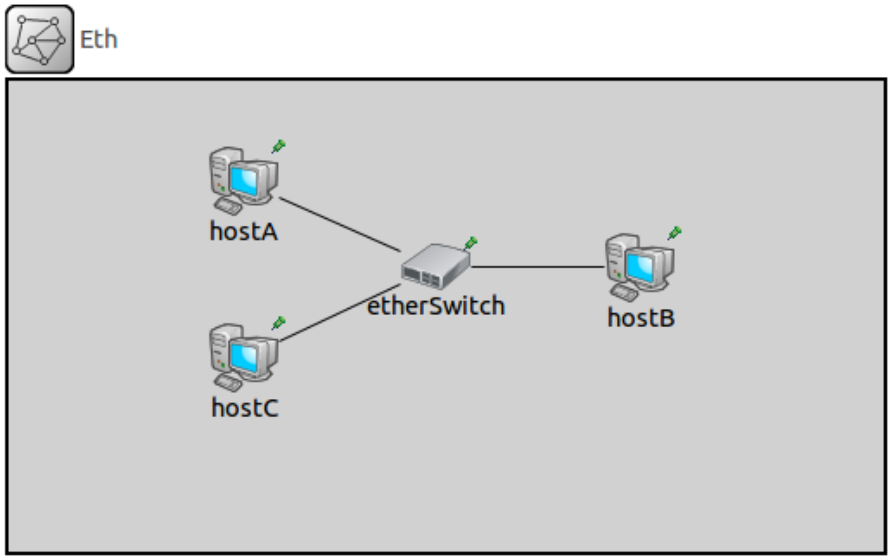


Рисунок 3.23 - Модель LAN (вариант 2)

Также изменим интервал времени между отправками сообщений, сделаем его равным 0,01 с. Содержимое файлов «NED» и «ini» приведено ниже.

Файл «ned»

```
package lesson_02;
import inet.node.ethernet.Eth10M;
import inet.node.ethernet.EtherHost;
import inet.node.ethernet.EtherSwitch;

//
// TODO documentation
//
network Eth
```

```
{
    @display("bgb=506,272");
    submodules:
        hostA: EtherHost {
            @display("p=119,66");
        }
        hostB: EtherHost {
            @display("p=377,131");
        }
        etherSwitch: EtherSwitch {
            @display("p=249,130");
        }
        hostC: EtherHost {
            @display("p=119,184");
        }
    connections:
        hostA.ethg <--> Eth10M <--> etherSwitch.ethg++;
        etherSwitch.ethg++ <--> Eth10M <--> hostB.ethg;
        hostC.ethg <--> Eth10M <--> etherSwitch.ethg++;
}
```

Файл «ini»

```
[General]
network = lesson_02.Eth

sim-time-limit = 180s
tkenv-plugin-path = ../../../etc/plugins
**.vector-recording = true

**.hostA.cli.destAddress = "hostB"
**.hostB.cli.destAddress = "hostA"
**.hostC.cli.destAddress = "hostB"

**.cli.sendInterval = exponential(0.01s)

#**.mac.txQueueLimit = 50

**.mac.address = "auto"

**.cli.reqLength = intuniform(50,1400)*1B
**.cli.respLength = truncnormal(3000B,3000B)
```

На рисунках 3.24 – 3.25 приведены плотности вероятности времени доставки запроса и ответа, соответственно. Время доставки запроса также имеет распределение близкое (визуально) к равномерному распределению, а время доставки ответа имеет распределение близкое к усеченному нормальному распределению, которые были заданы в качестве

распределений длин сообщений. Стоит отметить, что в этом случае форма полученных гистограмм визуально ближе к теоретическим законам.

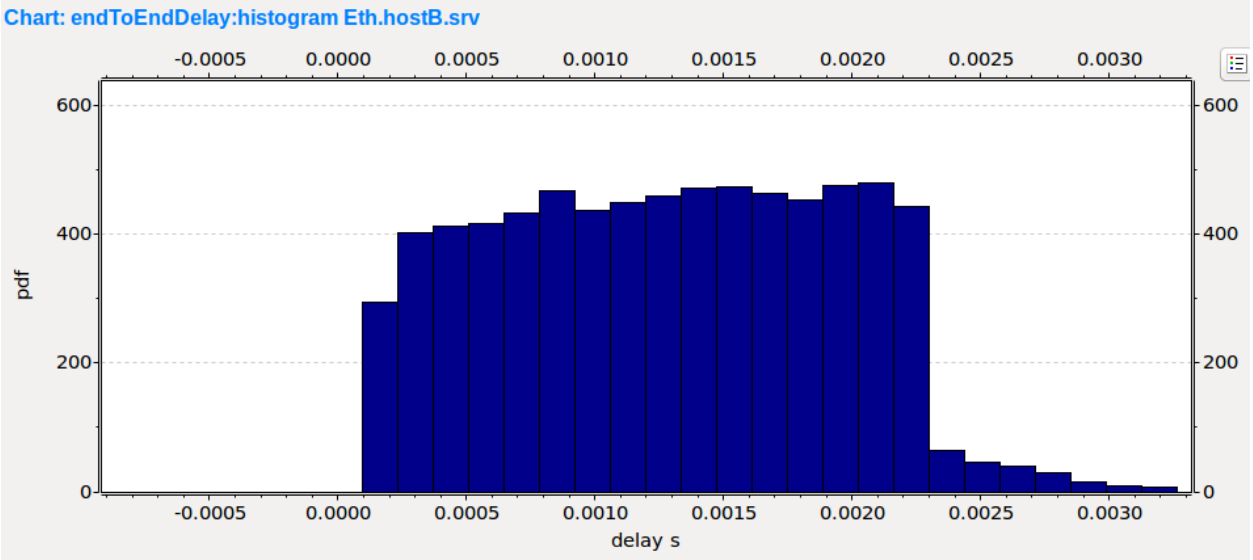


Рисунок 3.24 - Распределение времени доставки запроса

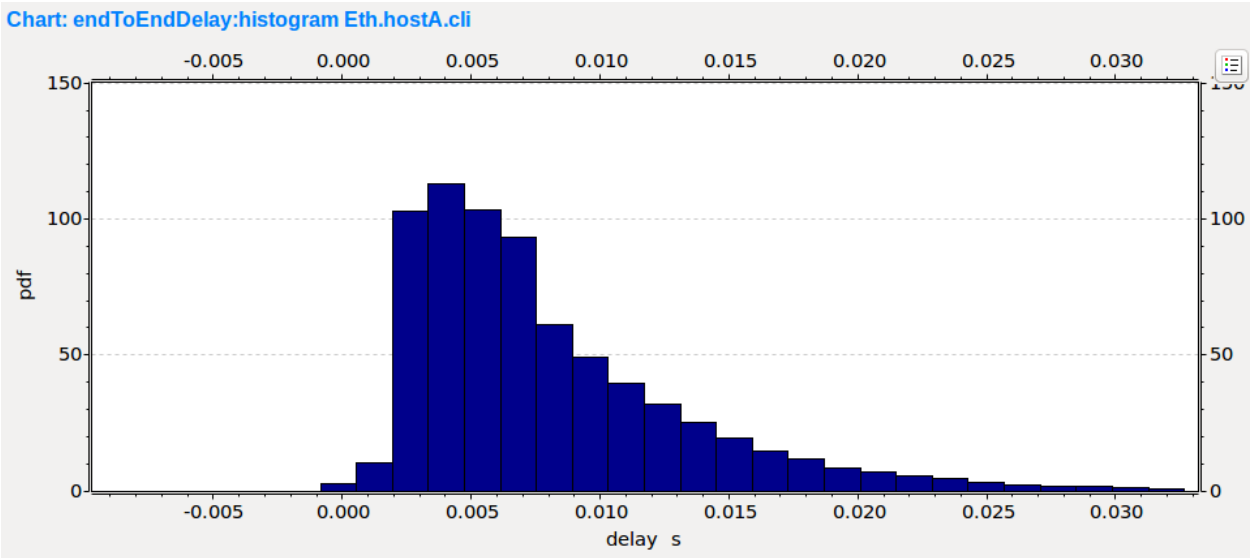


Рисунок 3.25 - Распределение времени доставки ответа

Это объясняется тем, что в данном случае размер выборки, по которой они построены гораздо больше, т.к. сократив интервал времени между отправками сообщений, мы увеличили интенсивность трафика, а ограничение модельного времени (180 с) осталось прежним.

3.6 Модель протокола TCP клиент-сервер

3.6.1 Кратко о протоколе TCP

Протокол TCP ориентирован на соединение, он использует для доставки данных IP-дейтограммы сетевого уровня, которые пересылаются посредством кадров канального уровня. Два устройства, между которыми происходит обмен данными, могут быть соединены напрямую или соединение может представлять собой маршрут в сети связи, включающий в себя несколько сетевых устройств. Элементы маршрута являются общими для многих клиентов и могут обслуживать потоки трафика различной интенсивности. К тому же маршрут может состоять из участков с различной битовой скоростью передачи данных. Таким образом, полоса пропускания, маршрута между двумя участниками обмена данными является случайной величиной. В следующем примере приведены расчеты, сделанные со многими допущениями и упрощениями, которые могут быть недопустимы в практических задачах, здесь их целью является лишь наглядность приводимых рассуждений. Представим маршрут между устройствами *a* и *b*, проходящий через маршрутизатор *c*. На участке *ac* скорость передачи данных равна 10 Гбит/с, а на участке *cb* 100 Мбит/с. Устройство *b* запрашивает у устройства *a* файл, размер которого равен 1 Гбайт. Если бы устройство *a* передавало файл со скоростью 10 Гбит/с, то ему потребовалось бы для передачи файла примерно 0,8 с, без учета «накладных расходов» на заголовки пакетов, задержки в буфере и повторы. Для передачи по каналу 100 Мбит/с потребуется уже 80 с. Маршрутизатор не выполняет функции промежуточного хранения данных. Его основная задача максимально быстро направлять поступающие пакеты данных на нужные порты. Максимум того что он может, это хранить несколько ожидающих передачи пакетов в буфере. Таким образом, возникает задача «согласования скоростей» передачи и приема данных. Ее можно достаточно просто решить, используя передачу с квитированием (подтверждением). Если устройство *a* будет передавать каждый следующий пакет только после получения подтверждения приема предыдущего пакета от устройства *b*. При таком подходе в буфере маршрутизатора будет находиться не более одного пакета. Но в этом случае на передачу одного пакета потребуется не только время его доставки от *a* к *b* t_{ab}, но и время доставки подтверждения t_{ba}, т.е. $t = t_{ab} + t_{ba}$. Если от *a* к *b* передается *n* пакетов, то средняя скорость передачи данных составит $\bar{a} \approx \frac{L}{t_{ab}+t_{ba}}$ (для всех пакетов кроме первого), где *L* длина пакета (бит). Очевидно, что максимально достижимая скорость будет иметь место, когда $t_{ba} = 0$, т.е. приблизительно равна $a_{max} \approx \frac{L}{t_{ab}}$. Тогда коэффициент снижения средней скорости за счет ожидания подтверждений будет равен $k = \frac{\bar{a}}{a_{max}} = \frac{t_{ab}}{t_{ab}+t_{ba}} = \frac{1}{1+\frac{t_{ba}}{t_{ab}}}$. Например, если $t_{ba} = t_{ab}$ (что вполне возможно в реальной сети), то $k = 0{,}5$, т.е. средняя скорость снизится в 2 раза. При этом пропускная

способность канала, во время передачи файла, будет использоваться примерно на 50%. Если подтверждение передавать не на один, а на группу из r пакетов, то снижение средней скорости будет равно $k(r) = \frac{1}{1+\frac{t_{ba}}{r\,t_{ab}}}$. Например, при передаче одного подтверждения на группу из 4 пакетов, снижение средней скорости при тех же условиях составит 20%, т.е. пропускная способность будет использоваться на 80%. Это уже лучше, но при таком подходе, который называется метод «*приостановки источника*», мы никогда не достигнем 100% (близкого к 100%) использования пропускной способности канала из-за необходимости приостановки передачи на время ожидания подтверждения t_{ba}. Достичь близкого к 100% использования пропускной способности можно только исключив это время ожидания. Повысить использование канала можно, используя метод «*скользящего окна*» (Sliding window).

Поскольку маршрутизатор способен хранить несколько пакетов в буфере, то логично предложить передачу данных на участке от *a* к *c* по одному или нескольку пакетов (пачками), а затем делать перерыв на время, пока эти пакеты не будут переданы по участку *cb* с меньшей скоростью. Передаваемая «пачка» пакетов, собственно и является тем самым окном. Если очередной пакет или пачка пакетов будут поступать в маршрутизатор до окончания передачи пакетов из предыдущей пачки, то канал *cb* будет использоваться на 100%. В этом случае проблема состоит в выборе интервала времени между пачками и количестве пакетов в пачке. При слишком малом интервале будет расти количество пакетов в буфере, что может привести к его заполнению и потере поступающих пакетов, а при слишком большом интервале будет снижаться использование канала *cb*. При этом подтверждения приема пакетов могут быть не привязаны жестко ко времени приема и передачи. Более подробно этот метод работает следующим образом. Отправитель, зная размер окна *w* (пакетов), передает все *w* пакетов, начиная с пакета 1 до пакета *w*, включительно. Если в процессе передачи он получает подтверждение, например, на первый переданный пакет, то он увеличивает окно на 1, т.е. окно уже будет от 1го до *w*+1 го пакета. Если далее, продолжая передачу, он получает подтверждения на 2 и 3й пакеты, то окно сместится на пакеты с 1го по *w*+3 й пакеты. Отправитель приостановит передачу только тогда, когда передав последний пакет текущего окна, он не получит подтверждения ни на один из переданных пакетов текущего окна. При получении подтверждения он вновь начинает передачу пакетов. При этом размер окна будет равен тому, при котором произошла приостановка передачи. Возможны различные процедуры уменьшения и увеличения размера окна.

Аналогичная идея и лежит в основе протокола транспортного уровня TCP (Transmission control protocol). Это протокол гарантированной доставки данных, поэтому квитирование является обязательным его элементом. Реализовано оно как было описано выше с помощью метода «скользящего окна». Наряду с методом скользящего окна в протоколе используется метод

управления перегрузкой (congestion control). Это позволяет обеспечить согласование скорости, гарантировать доставку данных и сохранить при этом достаточно высокую эффективность в части использования пропускной способности канала. Следует сказать, что TCP поддерживает дуплексный обмен данными, т.е. обе взаимодействующие стороны могут одновременно передавать друг другу данные, при этом с целью сокращения количества передаваемых сегментов, подтверждения получения сегмента могут передаваться не в виде отдельных единиц, а «с оказией», т.е. в заголовках передаваемых сегментов данных. Разумеется, если есть необходимость передавать в обратную сторону сегменты данных, если таковой нет, то подтверждения будут передаваться в виде отдельных сегментов, состоящих только из заголовка (с нулевой длиной поля данных). Поскольку процессы обмена в обе стороны независимы, то допускается, вернее даже вводится значительное отставание (0,1 … 0,5 с) моментов передачи подтверждений от моментов приема сегментов данных, за исключением нескольких случаев. К таким случаям относятся потеря сегмента данных или нарушение порядка их следования. В этих случаях подтверждение передается незамедлительно после их обнаружения. Все сегменты, передаваемые в рамках соединения, последовательно пронумерованы. Точнее, номером является номер первого байта, передаваемого в сегменте. Таким образом, имея данные о количестве байт в сегменте и номере первого байта, всегда можно определить номер байта следующего сегмента потока. После приема сегмента приемник проверяет, является ли этот сегмент следующим, по отношению к последнему принятому сегменту. Если да, то приемник отправит подтверждение на этот сегмент обычным образом. Если нет, т.е. предшествующий сегмент не был успешно принят (может быть потерян или нарушена последовательность сегментов), то приемная сторона вновь передает подтверждение на последний успешно принятый сегмент (появляется повторное подтверждение). Если следующий принятый сегмент вновь не является ожидаемым, то будет вновь передано подтверждение на последний успешно принятый сегмент. Таким образом, если сегмент потерян, то передающая сторона получит три идентичных подтверждения, это и является критерием, по которому он обнаруживает потерю сегмента (первое подтверждение на последний успешно принятый сегмент и два на сегменты, которые не были ожидаемыми). Вторым критерием является таймаут (RTO Retransmit Timeout), по истечении которого, если не пришло подтверждение, то сегмент считается потерянным. Иллюстрация приведена на рисунке 3.26.

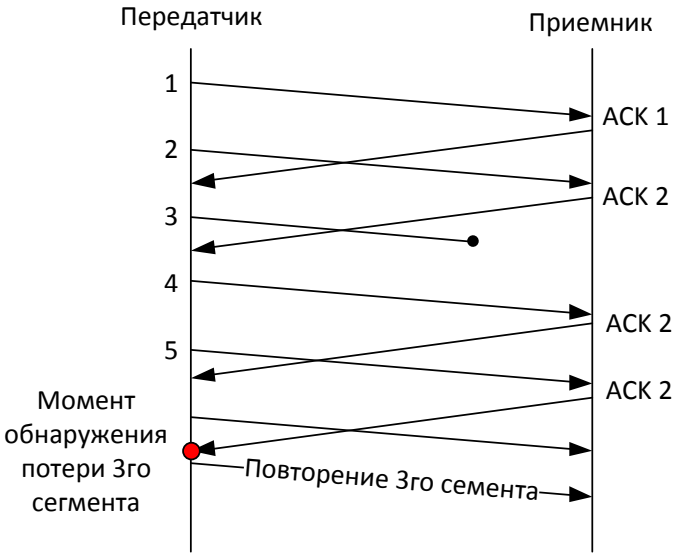


Рисунок 3.26 - Диаграмма обмена сегментами

Важными параметрами, которые используются при работе протокола, являются: размер окна приемника rwnd (Receiver Window), размер окна перегрузки cwnd (Congession Window), порог медленного запуска ssthresh (Slow Start Threshold), время отклика RTT (Round Trip Time), величина таймаута повторной передачи RTO (Retransmission Time-Out).

Управление размером окна перегрузки cwnd позволяет управлять размером скользящего окна. На его размер также влияет размер окна приема rwnd. Размер скользящего окна

$$W = \min(rwnd, cwnd) \quad (3.1)$$

В каждом сообщении о подтверждении приема (ACK) приемник присылает анонс окна (window advertisement) в котором указывается количество байтов, которые окно приемник может принять дополнительно. Фактически, это данные о состоянии буфера приемника, которые называют окном приемника (rwnd).

В новом соединении выбирается начальный размер скользящего окна *IW*.

$$IW = \min(4MSS, \max(2MSS, 4380)) \text{ байт} \quad (3.2)$$

где *MSS* – максимальный размер сегмента данных (байт).

Значение MSS выбирается как

$$MSS = \min(MSS_{RX}, MSS_{TX}) \text{ байт} \quad (3.3)$$

где MSS_{RX} - максимальный размер сегмента приемника,

MSS_{TX} - максимальный размер сегмента передатчика.

Значение ssthresh в начале соединения устанавливается равным 65535.

Во время фазы медленного запуска значение cwnd увеличивается на MSS с получением каждого ACK. Передатчик увеличивает окно до момента обнаружения потери сегмента, или до достижения значений ssthresh или rwnd. После этого значение cwnd остается постоянным и равным rwnd. При обнаружении потери сегмента по истечению RTO значение ssthresh уменьшается. Его значение приравнивается половине числа переданных, но

еще не подтвержденных байт (или 2MSS). Значение cwnd приравнивается MSS

$$ssthresh = \max(FlightSize/2, 2MSS)$$
$$cwnd = MSS \quad (3.4)$$

3.6.2 Построение модели

Для анализа функционирования TCP реализуем следующую модель. Создадим новую модель, как было описано в первом примере, дадим ей имя «WindowSize». В свойствах модели Меню->Properties подключим ссылку («Project References») на фреймворк INET (рисунок 3.27).

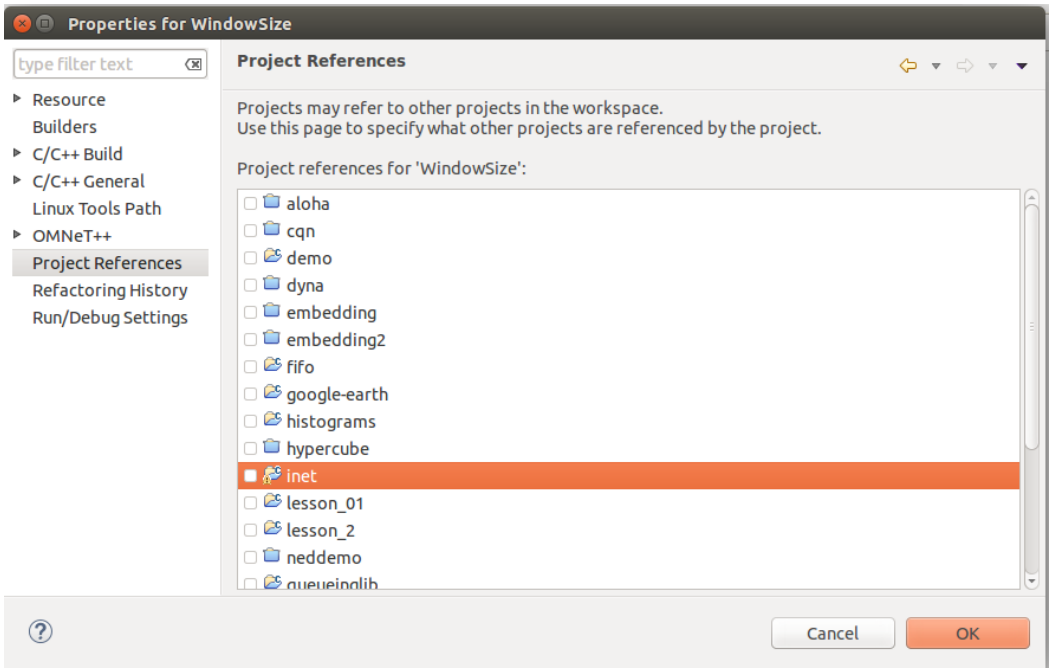


Рисунок 3.27 - Подключение INET

Далее создадим файл описания модели «NED» в папке WindowsSize/simulations с названием «WindowSize». При создании файла укажем «NED file with one item». В результате в окне редактора будем видеть следующую конфигурацию, рисунок 3.28. В палитре выберем модуль Network и добавим его в область редактора, откроем свойства добавленного модуля и переименуем его на «ClientServer».

Рисунок 3.28 - Созданный файл NED в окне редактора

Выберем удобный размер отображения модуля «ClientServer», выберем из палитры тип субмодуля «standardHost» и разместим два таких субмодуля в поле модуля сети. Изменим имя одного из них на «client1», второго на «server». Выберем субмодуль «server», откроем контекстное меню и выберем свойства «Properties..» в свойствах выберем другое изображение «Image» для отображения модуля «device/server».

Далее выберем в палитре субмодуль «IPv4NetworkConfigurator» и разместим его в окне модели. Выберем в палитре тип соединения «DatarateCnannel (ned)» кликнем по субмодулю «client», а затем по субмодулю «server», выберем в открывшемся контекстном меню «client.pppg++ <--> server.pppg++(PPPFrame-conn) (рисунок 3.29).

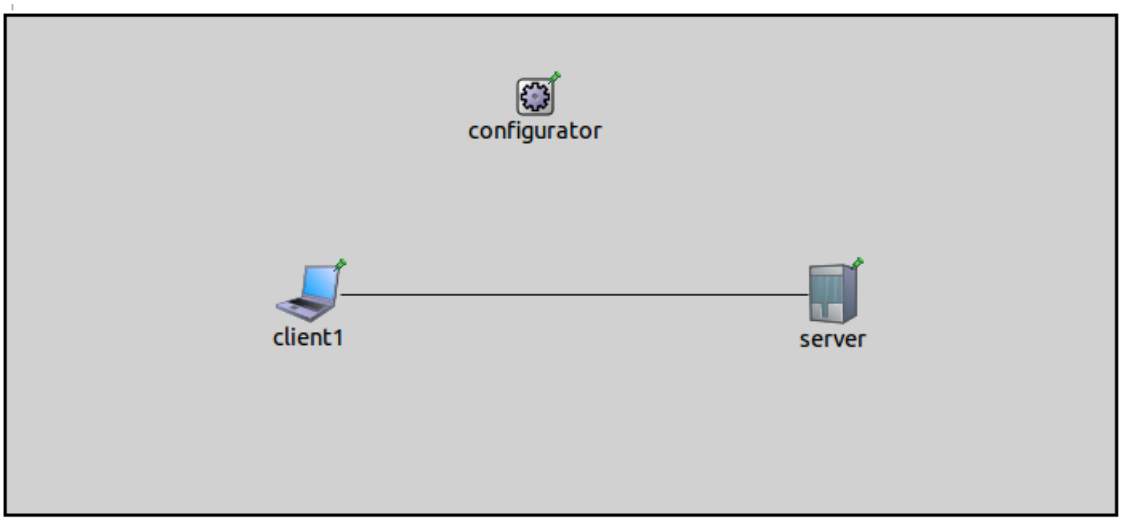


Рисунок 3.29 - Структура сети

Выберем соединительную линию, откроем контекстное меню (правая клавиша мыши), откроем форму параметров «Parameters…» изменим значения параметров «datarate» на 10Mbps и «delay» на 0.1us.

Далее выберем (или создадим) файл инициализации «ini». Откроем файл в редакторе, введем имя конфигурации «WindowSize», «Simulation time limit» введем значение 100s. Перейдем в текстовый режим редактирования «ini» файла, выбав вкладку «Source» и введем (или скопируем) следующие параметры.

```
[General]
description = "default_TCP <---> default_TCP"
network = ClientServer
total-stack = 7MiB
tkenv-plugin-path = ../../../etc/plugins
record-eventlog = true
**.server.numPcapRecorders = 1
**.server.pcapRecorder[0].pcapFile = "server.pcap"
**.client.numPcapRecorders = 1
**.client.pcapRecorder[0].pcapFile = "client.pcap"

## tcp apps
**.numTcpApps = 1
**.client*.tcpApp[*].typename = "TCPSessionApp"
**.client*.tcpApp[0].active = true
**.client*.tcpApp[0].localPort = -1
**.client*.tcpApp[0].connectAddress = "server"
**.client*.tcpApp[0].connectPort = 1000
**.client*.tcpApp[0].tOpen = 0.2s
**.client*.tcpApp[0].tSend = 0.4s
**.client*.tcpApp[0].sendBytes = 5000000B
**.client*.tcpApp[0].sendScript = ""
**.client*.tcpApp[0].tClose = 25s

**.server*.tcpApp[*].typename = "TCPEchoApp"
**.server*.tcpApp[0].localPort = 1000
**.server*.tcpApp[0].echoFactor = 0
**.server*.tcpApp[0].echoDelay = 0

# NIC configuration
**.ppp[*].queueType = "DropTailQueue" # in routers
**.ppp[*].queue.frameCapacity = 10  # in routers

*.configurator.config=xml("<config><interface hosts='*' address='192.168.1.x' netmask='255.255.255.0'/></config>")
```

Кратко о назначении указанных в «ini» файле параметров. Имя конфигурации "default_TCP <---> default_TCP". Имя сети «ClientServer», объем памяти, выделяемой процессу 7 Мбайт. Путь к плагинам tkenv, разрешение записи лога событий. Количество записей pcap файлов для сервера 1, количество записей pcap файлов для клиента 1. Имена файлов pcap для сервера и клиента.

Значение "TCPSessionApp" определяет TCP приложение, создающее одно соединение (сессию). Параметр «**client*.tcpApp[0].active = true» указывает на активное поведение приложения клиента, т.е. инициация сессии производится клиентом. Значение (минус) 1 для «**.client*.tcpApp[0].localPort» указывает на использование временного локального порта. Параметру «**.client*.tcpApp[0].connectAddress» присваивается значение «server», т.е. адрес установления связи будет равен адресу сервера. Локальный адрес порта, с которым производится соединение «**.server*.tcpApp[0]. connectPort» установлен

равным 1000. Время открытия соединения «**.client*.tcpApp[0].tOpen» равно 0,2 с. Задержка начала отправки данных «**.client*.tcpApp[0].tSend» равна 0,4 с. Количество байт, передаваемых клиентом «**.client*.tcpApp[0].sendBytes» зададим 5000000. Параметр «**.client*.tcpApp[0].sendScript = ""» равен пустой строке (это второй способ задания данных для отправки, в данном примере не используется). Время, через которое соединение будет закрыто «**.client*.tcpApp[0].tClose» равно 25 с. Приложение сервера «**.server*.tcpApp[*].typename ="TCPEchoApp"» эхо – приложение, которое возвращает клиенту полученные данные. Локальный порт сервера «**.server*.tcpApp[0].localPort» равен 1000. Параметр «**.server*.tcpApp[0].echoFactor» позволяет определить коэффициент на который умножается количество отправляемы обратно данных, в данном примере он равен 0 (обратно данные не отправляются). Задержка отправки данных обратно клиенту «**.server*.tcpApp[0].echoDelay» равна 0. Дисциплина обслуживания (в очереди) «**.ppp[*].queueType» (комбинированная) с ожиданием и отбрасыванием пакетов из хвоста очереди «**.ppp[*].queueType». Количество позиций ожидания в очереди «**.ppp[*].queue.frameCapacity» (вместимость) равна 10.

Параметры конфигуратора «*.configurator.config» определены как =xml("<config><interface hosts='*' address='192.168.1.x' netmask='255.255.255.0 '/></config>").

После ввода «NED» и «ini» файла можно запустить имитационный эксперимент, рисунок 3.30.

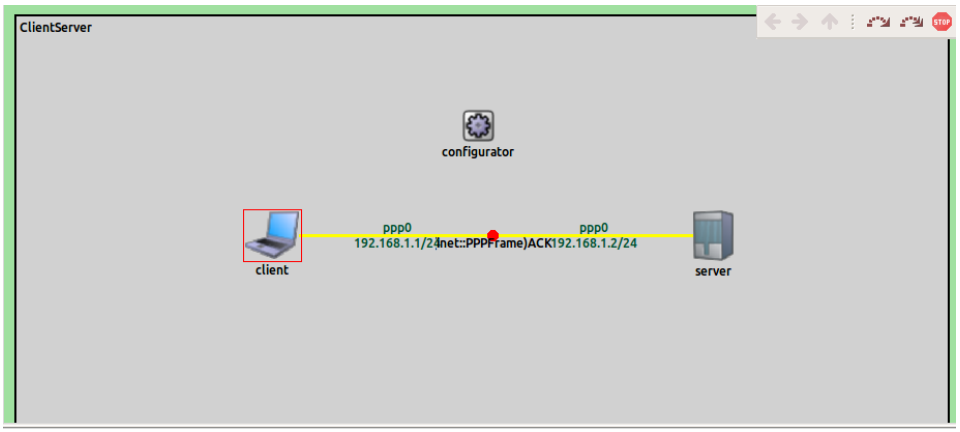


Рисунок 3.30 - Анимация в процессе имитационного эксперимента

В результате проведения эксперимента формируются файлы статистики, а также файлы «client.pcap» и «server.pcap», которые содержат подробные записи обо всех пакетах (сегментах) пересылаемых между клиентом и сервером. Эти файлы могут быть открыты программой «Wireshark» []. На рисунке 21 приведен пример отображения в Wireshark данных об обмене данными между клиентом и сервером во время проведения имитационного эксперимента. В таблице по каждому из сегментов отображаются номер пакета, время (модельное время), IP адреса источник и

получателя, протокол, длина и дополнительная информация. При выборе пакета из списка о нем может быть получена подробная информация о содержимом заголовка и тела пакета (рисунок 3.31).

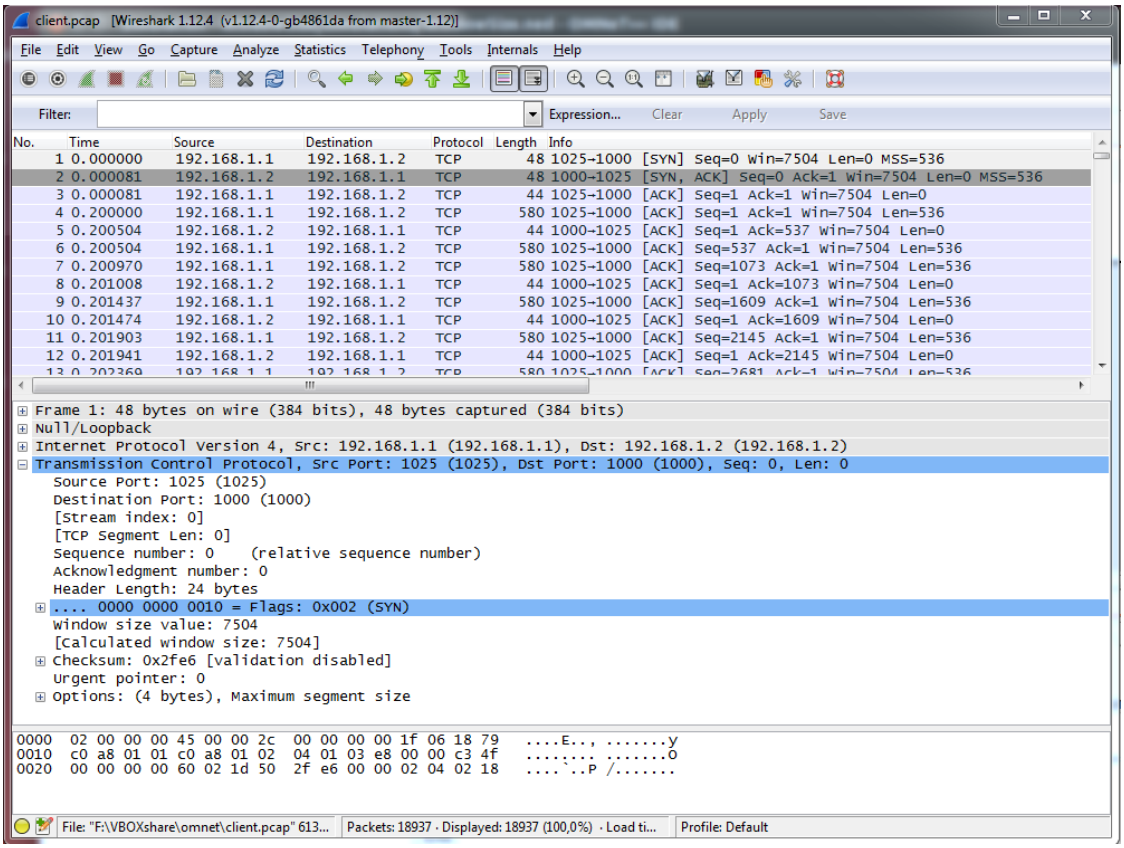


Рисунок 3.31 - Анализ потока в Wireshark

Эти данные позволяют провести детальный анализ обмена пакетами на протяжении всего TCP соединения. Например, рассматривая записи в начале списка можно увидеть сообщение от клиента к серверу SYN (первая запись). Сообщение сдержит запрос на установление соединения и размер MSS. Подтверждение, отправленное сервером клиенту ACK (вторая запись) и указанием установленного размера MSS. Следующая запись это подтверждение, переданное клиентом серверу о начале соединения. Аналогичным образом можно рассмотреть любые ппакеты переданные во время соединения.

Воспользуемся средством анализа Меню->Statistics->IO Graph. При выборе открывается окно, в котором можно в графическом виде получить иллюстрацию процесса обмена данными. Оперируя доступными органами управления, можно установить удобную форму отображения, а также нужный масштаб горизонтальной и вертикальной осей графика. На рисунке 3.32 приведена иллюстрация процесса передачи данных между клиентом и сервером.

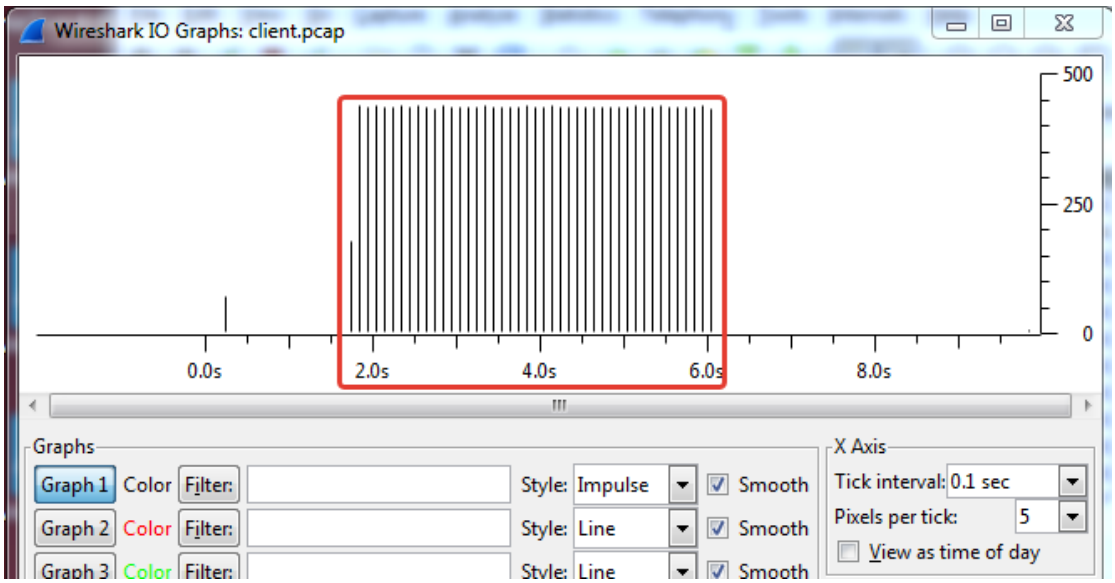


Рисунок 3.32 - Диаграмма потока пакетов в Wireshark

Из рисунка 3.32 видно, что фактическое время передачи файла размером 5 Мбайт, через канал с пропускной способностью 10 Мбит/с, заняло около 4,5 с. Это говорит о том, что пропускная способность канала использовалась приблизительно на 90%. Это можно считать приемлемой величиной использования ресурса.

На рисунке 3.33 приведена диаграмма, полученная в результате моделирования. На диаграмме приведена кривая, демонстрирующая изменение окна перегрузки сервера (cwnd), а также моменты потери сегментов данных (drop).

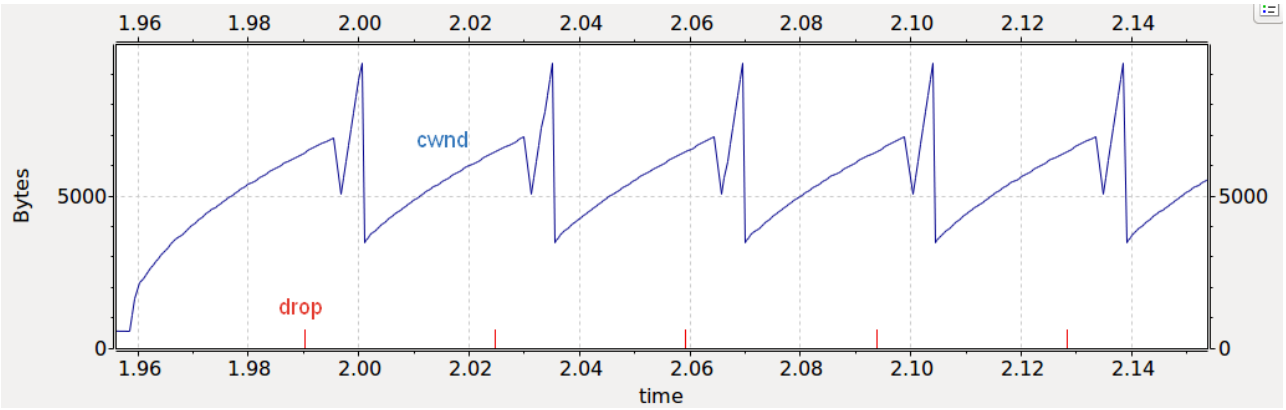


Рисунок 3.33 - Изменение cwnd и потерь от времени

Из диаграммы видно, что это изменение имеет периодический характер. Рассмотрим первый период. Значение cwnd возрастает до некоторого момента – это момент обнаружения потери сегмента. После этого момента уменьшается значение ssthresh до половины текущего значения cwnd. Окно перегрузки устанавливается на этот порог плюс три сегмента. Далее cwnd увеличивается на 1 сегмент с каждым полученным повторным

подтверждением, это приводит его быстрому увеличению. Когда приходит новое подтверждение окно устанавливается на значение ssthresh. После этого цикл повторяется.

Приведенные выше результаты получены для идеального случая, когда потери пакетов происходят только по причине переполнения буфера. В реальном случае потери могут возникнуть и по причине искажений передаваемых данных в линии связи, в наибольшей степени этот процесс выражен в беспроводных линиях связи. Модифицируем нашу модель, изменив свойства канала. Введем ненулевую вероятность искажения передаваемых бит. Для этого в коней файла «ini» добавим строку «**.ber=0.007», рисунок 3.34.

```
*.configurator.config=xml("<config><interface hosts='*' address='192.168.1.x' netmask='255.255.255.0'/></config>")
**.per = 0.007
```

Рисунок 3.34 - Изменение файла «ini»

Этим изменением мы задали вероятность потери пакета в линии связи равную $per = 7 \cdot 10^{-3}$.

Снова выполним имитационный эксперимент и воспользуемся данными из файла «server.pcap». Построим график аналогичный рисунку 22. Он приведен на рисунок 3.35.

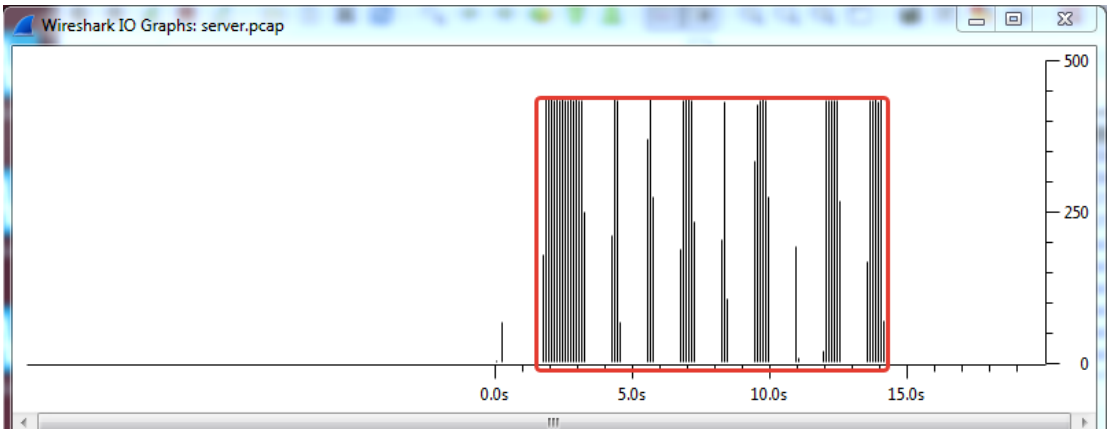


Рисунок 3.35 - Диаграмма потока пакетов при ненулевой вероятности ошибок

Аналогичную диаграмму можно получить, используя статисту из векторного файла, рисунок 3.36.

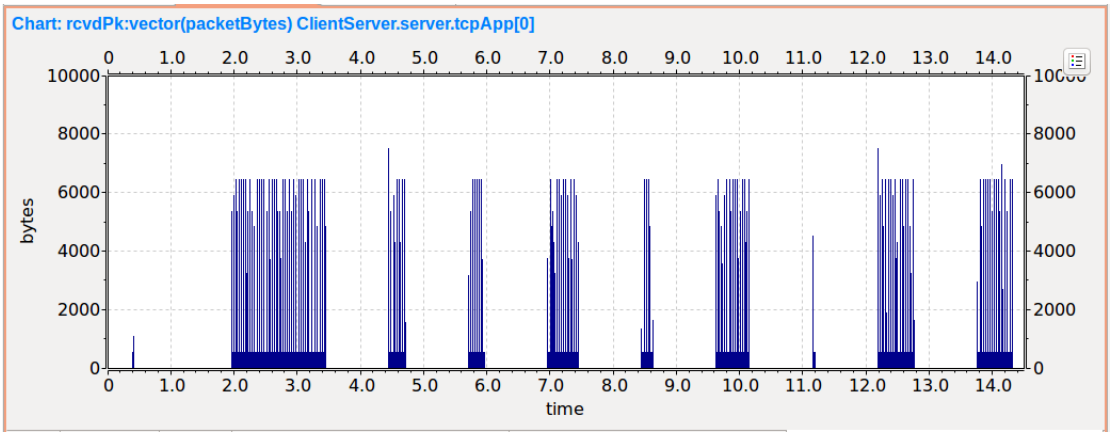


Рисунок 3.36 - Диаграмма потока пакетов по данным статистики

Из приведенных диаграмм видно, что полное время передачи файла составило около 12 с. Использование пропускной способности канала в этом случае составляет приблизительно 33%. Фактически, средняя скорость передачи данных 3,3 Мбит/с.

Проведенные эксперименты позволяют показать, что эффективность протокола в значительной степени зависит от свойств линии (канала) связи. Можно заметить, что при росте вероятности битовых ошибок в канале, полоса пропускания, при использовании протокола, TCP значительно уменьшается.

До вероятности потери пакета 5% для приблизительной оценки полосы пропускания можно использовать выражение

$$bw = \frac{1{,}22\ MTU}{RTT\ \sqrt{P_{loss}}}\ \text{бит/с} \qquad ()$$

где MTU – максимальный размер передаваемого кадра (бит);

RTT – «круговая» задержка (с);

P_{loss} - вероятность потери кадра.

Среднее значение RTT может быть найдено в результатах, оно равно 0,0038. Тогда согласно данной формуле $bw \approx 2{,}2$ Мбит/с. Это значение занижено, по сравнению с полученной ранее оценкой (3,3 Мбит/с). Выполнив ряд экспериментов можно получить зависимость пропускной способности от вероятности ошибки и сравнить ее с упомянутой формулой.

4 Обработка результатов имитационного моделирования

4.1 Результаты имитационного моделирования

Результатами имитационного моделирования являются результаты измерений значений параметров, выбранных в качестве показателей состояния исследуемой системы на этапе построения модели. Таким образом, получаемые результаты это результаты измерений, причем измеряемые параметры, как правило, являются случайными (псевдослучайными) величинами. Поэтому результатами исследования системы с помощью имитационной модели являются оценки и характеристики этих параметров. Для получения этих оценок следует использовать методы статистической обработки результатов измерений.

Действительно, если воспроизвести любую из приведенных выше моделей и получить в результате имитационного эксперимента значения оцениваемых параметров (вероятности потерь или задержки), то при каждой попытке это сделать мы будем получать различные значения. Здесь следует отметить, что это случиться тогда, когда генераторы псевдослучайных чисел, используемые в модели, будут стартовать из различных начальных состояний. Например, в OMNeT++ в свойствах эксперимента можно указать конкретное стартовое состояние генератора. Если это сделать, то результаты всегда будут точно повторяться, по той причине, что будет точно повторяться псевдослучайная последовательность, следовательно, и все процессы, происходящие в модели. Такая возможность удобна при построении, отладке и проверке модели. В реальной системе результаты измерений будут отличаться, поэтому, если целью является получение данных в серии независимых испытаний, то при имитационном эксперименте следует каждый раз выбирать различное значение начального состояния генератора случайных чисел.

Как уже было отмечено, процессы, происходящие в имитационной модели, строго говоря, не случайные, а псевдослучайные. Степень соответствия имитационной модели реальной системе определяется, в частности, и «схожестью» псевдослучайных процессов модели и случайных процессов реальной системы. Как правило, для систем имитационного моделирования выбираются достаточно хорошо проверенные и отработанные алгоритмы получения псевдослучайных чисел, которым в большинстве случаев, можно доверять. Однако следует помнить, что не идеальность псевдослучайной последовательности тоже может привести к ошибкам моделирования.

Чаще всего результатами измерений в имитационной модели являются:

-среднее значение параметра, полученное за время (модельное) выполнения эксперимента;

-стандартное отклонение или дисперсии какого-либо параметра (будут рассмотрены ниже);

-упорядоченное множество значений параметра, каждое из которых соответствует определенному моменту модельного времени (временной ряд);

-упорядоченное множество значений параметра, каждое из которых соответствует определенному состоянию модели, например, определенному значению какого-либо параметра;

-множество значений параметра, упорядоченное и агрегированное по какому либо признаку.

Параметрами могут быть как значения, каких-либо переменных модели, так и функции от этих значений.

Измерения при моделировании выполняются путем регистрации значения какой-либо переменной при выполнении определенного условия. Например, количество поступивших заявок, которое регистрируется в конце каждой секунды модельного времени или время ожидания в очереди регистрируется для заявки определенного приоритета и т.п.

Результаты, полученные в ходе имитационного эксперимента, называют выборкой, на основании которой затем получают оценки значений интересующих нас параметров. Количество значений в выборке называют размером выборки. Как будет показано ниже, от размера выборки зависит точность получаемых оценок. Размер выборки зависит от способа регистрации данных и от продолжительности эксперимента (в единицах модельного времени).

Поскольку результаты моделирования являются случайными (псевдослучайными) величинами то их можно описать соответствующими функциями распределения случайных величин и получить оценки параметров этих функций распределения.

4.2 Точечные оценки параметров

Оценка параметра называется точечной, если она выражается одним числом. Любая точечная оценка, вычисленная на основании опытных данных, является их функцией и поэтому сама должна представлять собой случайную величину с распределением, зависящим от распределения исходной случайной величины, в том числе от самого оцениваемого параметра и от числа опытов *n*.

Точечными оценками являются начальные и центральные моменты распределения случайной величины. Начальный момент распределения $m_k = M(X^k)$. Начальный момент при *k*=1 называется математическим ожиданием.

Центральный момент это $\mu_k = M[(X - m_k)^k]$. Центральный момент при *k=2* называется дисперсией случайной величины.

Наиболее распространенными точечными оценками являются математическое ожидание и дисперсия. Пусть в результате имитационного эксперимента получены данные в виде *n* значений какого-либо параметра $X = \{x_1, x_1, \dots, x_n\}$. По результатам измерений среднее значение оценивается как

$$M(x) = \bar{x} = \sum_{i=1}^{n} x_i, \quad i = 1 \dots n, \tag{4.1}$$

При неизвестном законе распределения часто делают допущение о том, что исследуемая случайная величина подчинена нормальному закону распределения, тогда дисперсия определяется как

$$D(x) = Dx = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2, \quad i = 1\ldots n. \tag{4.2}$$

Часто на практике вместо значения дисперсии приводят значение выборочного среднеквадратического отклонения (стандартное отклонение)

$$\sigma_x = \sqrt{Dx} \tag{4.3}$$

На практике также иногда применяют 3-й и 4-й моменты распределения, они характеризуют форму распределения. Величина $\frac{\mu_3}{\sigma^3}$ характеризует асимметрию распределения и называется коэффициентом асимметрии. Например, для нормального распределения коэффициент асимметрии равен нулю.

Величина $\frac{\mu_4}{\sigma^4} - 3$ характеризует остроту вершины (пика) распределения и называется коэффициентом эксцесса.

Моменты более высоких порядков на практике, обычно, не используются.

4.3 Интервальные оценки параметров

Интервальной называют оценку, которая определяется двумя числами концами интервала. Интервальные оценки позволяют установить точность и надежность оценок. При этом интервал, характеризующий оцениваемый параметр называется доверительным интервалом. При такой оценке задается доверительная вероятность, т.е. вероятность того интервал накрывает значение оцениваемого параметра. Ширина интервала определяет точность полученной оценки. Как правило, интервальные оценки используются для описания среднего значения и дисперсии.

Доверительный интервал для среднего значения может быть определен как

$$x = \bar{x} \pm \Delta \tag{4.4}$$

Где $\bar{x}$ среднее значение, согласно формуле (1);

$$\Delta = t_{n-1,1-\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}} \tag{4.5}$$

где

$t_{n-1,1-\frac{\alpha}{2}}$ - коэффициент Стьюдента для n-1 степени свободы и доверительной вероятности 1-α,

σ - среднеквадратическое отклонение;

n – количество наблюдений.

Если распределение исследуемой величины известно, то в (5) следует использовать выражение для σ соответствующего закона распределения. Например, оценивается средняя величина вероятности в серии испытаний, которая может быть описана схемой Бернулли, т.е. оценивается вероятность p успеха (неуспеха). Успехом считается какое-либо событие, происходящее в результате испытания, если событие не происходит, то говорим о неуспехе. В этом случае случайная величина подчинена биномиальному распределению, для которого среднеквадратическое отклонение определяется как

$$\sigma_p = \sqrt{\frac{\bar{p}(1-\bar{p})}{n}} \tag{4.6}$$

Тогда интервальная оценка для среднего значения вероятности будет определяться

$$p = \bar{p} \pm t_{n,1-\frac{\alpha}{2}} \frac{\sqrt{\bar{p}(1-\bar{p})}}{n} \tag{4.7}$$

При относительно большом количестве испытаний (более 100) коэффициент Стьюдента заменяют квантилем нормального распределения, тогда выражение (5) будет выглядеть как

$$\Delta = g_{1-\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}} \tag{4.8}$$

где $g_{1-\frac{\alpha}{2}}$ квантиль нормального распределения для доверительной вероятности 1-α..

Точность получаемой интервальной оценки определяется шириной доверительного интервала. Как правило, полученные оценки характеризуют относительной точностью оценки, которая определяется как отношение ширины половины доверительного интервала к среднему значению, выраженное в процентах

$$\delta = \frac{\Delta}{\bar{x}} 100\,\% \tag{4.9}$$

Из приведенных выше выражений видно, что ширина доверительного интервала определяется среднеквадратическим отклонением (дисперсией) оцениваемой величины и количеством наблюдений. Увеличение количества наблюдений приводит к сужению доверительного интервала. Поэтому, при планировании эксперимента, при наличии требований к точности, оценивают необходимое количество измерений (объем выборки).

Если оценку нужно получить с заданной точностью (величиной относительной ошибки δ), то необходимый размер выборки можно оценить как

$$n = \left(t_{n-1,1-\frac{\alpha}{2}} \frac{\sigma}{\bar{x}\,\delta}\right)^2 \tag{4.10}$$

Следует заметить, что в выражении (10) присутствуют сама средняя величина $\bar{x}$ и σ, которые требуется оценить по результатам измерений. Поэтому, в практических задачах сначала производят предварительную оценку средней величины $\bar{x}$ и стандартного отклонения σ, по выборке произвольного размера, получают их ожидаемые значения, а затем выполняют расчет уточненного размера выборки, используя эти значения.

Если размер выборки достаточно большой (практически, как правило, более 100 значений), то коэффициент Стьюдента заменяют квантилем нормального распределения $g_{1-\frac{\alpha}{2}}$, величина которого не зависит от размера выборки.

4.4 Правила записи численных значений результатов

Как правило, современные вычислительные средства позволяют производить вычисления с достаточно высокой точностью, т.е. количество знаков в числах определяется требованиями к точности и используемым программным обеспечением, которое обеспечивает требуемую точность для подавляющего большинства задач. В большинстве практических случаев приходится напротив ограничивать количество знаков в числах, представляющих результаты вычислений исходя из реальных требований к точности и точности получаемых оценок.

Правила записи и округления чисел определены в стандарте [32]. Далее приведена выдержка из этого стандарта, определяющая правила записи чисел.

Значащие цифры данного числа - это все цифры от первой слева, не равной нулю, до последней записанной цифры справа. При этом нули, следующие из множителя 10^n, не учитываются.

Примеры. 1. Число 12,0 имеет три значащие цифры;

2. Число 30 имеет две значащие цифры;

3. Число $120 \cdot 10^3$ имеет три значащие цифры;

4. Число $0{,}514 \cdot 10$ имеет три значащие цифры;

5. Число 0,0056 имеет две значащие цифры.

Когда необходимо указать, что число является точным, после числа должно быть указано слово "точно" или же последняя значащая цифра печатается жирным шрифтом.

Пример. В печатном тексте:

1 = 3600000 Дж (точно), или = 3600000 Дж

Следует различать записи приближенных чисел по количеству значащих цифр.

Примеры:

1. Следует различать числа 2,4 и 2,40. Запись 2,4 означает, что верны только цифры целых и десятых; истинное значение числа может быть

например 2,43 и 2,38. Запись 2,40 означает, что верны и сотые доли числа; истинное число может быть 2,403 и 2,398, но не 2,421 и не 2,382.

2. Запись 382 означает, что все цифры верны; если за последнюю цифру ручаться нельзя, то число должно быть записано.

3. Если в числе 4720 верны лишь две первые цифры, оно должно быть записано $47 \cdot 10^2$ или $4{,}7 \cdot 10^3$.

Число, для которого указывается допускаемое отклонение, должно иметь последнюю значащую цифру того же разряда как и последняя значащая цифра отклонения.

Примеры:

1. Правильно: 17,0±0,2

Неправильно: 17±0,2 или 17,00±0,2

2. Правильно: 12,13±0,17

Неправильно: 12,13±0,212, 1±0,17

3. Правильно: 46,40±0,15

Неправильно: 46,4±0,15 или 46,402±0,15

1.5. Числовые значения величины и ее погрешности (отклонения) целесообразно записывать с указанием одной и той же единицы физических величин.

Пример. 80,555±0,002 кг

Интервалы между числовыми значениями величин следует записывать:

От 60 до 100 или от 60 до 100

Свыше 100 до 120 или свыше 100 до 120

Как правило, результаты имитационного моделирования следует приводить с одинаковой точностью и одинаковым количеством знаков.

Количество приводимых знаков связано с точностью получения результатов.

Оцениваемые величины могут быть, как непрерывными, так и дискретными. Например, если в результате имитационного моделирования получили среднее значение задержки доставки пакета равное 23,021456 мс, полуширина доверительного интервала при доверительной вероятности 0,9 оказалась равной 0,347872 мс. Таким образом, относительная точность результата составила 1,5% (несколько ниже 0,01 от среднего значения). Применим тоже требование и к точности представления доверительного интервала, т.е. представим его двумя знаками после запятой. Тогда полученный результат можно записать как $\bar{t} = 23{,}02 \pm 0{,}35$ с.

Например, если в результате имитационного моделирования получили среднее значение длины пакета равное 1390,245603 байт, полуширина доверительного интервала при доверительной вероятности 0,9 оказалась равной 4,473873 байта. Таким образом, относительная точность результата составила 0,3% (ниже 0,001), т.е. мы можем ограничить число знаков в представлении доверительного интервала тремя после запятой, однако, природа случайной величины является дискретной, следовательно,

целесообразно ограничить приводимый результат только целой частью. Тогда полученный результат можно записать как $\bar{L} = 1390 \pm 4$ байта.

Если при выполнении моделирования имеются особые требования к числу знаков в приводимых результатах, то необходимо следовать этим требованиям.

4.5 Диаграммы и графики

Диаграмма это общее название геометрического изображения, позволяющего увидеть показатели в их сравнении или проследить динамику изменения какого-либо явления. Понятие графика является частным случаем диаграммы, т.е. это диаграмма выражающая взаимозависимость величин.

В результате имитационного моделирования или измерения в реальной системе могут быть получены значения некоторых параметров в различные моменты времени или модельного времени (временной ряд). Подобный ряд может быть получен не обязательно для моментов времени, а для значений какого-то иного параметра. Графическое представление такого ряда может наглядно иллюстрировать исследуемый процесс и быть полезным при описании результатов.

Графиком называют графическое представление функциональной зависимости от некоторой переменной. В нашем случае, как правило, имеет место набор данных измерений, которые можно рассматривать как функцию времени заданную таблицей. Описываемая величина может быть как непрерывной, так и дискретной это следует учитывать при построении графика. График может быть построен как для непрерывной, так и для дискретной величины.

Основные требования, которым нужно следовать при построении графика (по мнению автора пособия): наглядность, точность, информативность и минимальная достаточность. Прокомментируем их можно следующим образом.

Наглядность – график должен доходчиво демонстрировать то качество, которое автор намерен донести до читателя. График должен быть, по возможности, простым, чрезмерно насыщенный график (слишком много кривых и/или комментариев) менее нагляден, чем простой график. Все сказанное можно отнести к выбору цветов, типов начертания линий и т.д. здесь нужно следовать принципу минимальной достаточности.

Точность – единицы измерения, масштабы осей и способ изображения кривых должны быть выбраны так, чтобы было можно однозначно и достаточно точно получить значение функции по значению аргумента (и наоборот). Разумеется, что график должен содержать указания единиц измерения по каждой из осей координат.

Информативность – график должен быть, по возможности, максимально информативен. Например, если исследуемая величина не изменяется (всегда постоянна) и это есть то качество, которое автор хочет

донести до читателя, то график в виде горизонтальной прямой не будет информативным. Его вообще не следует приводить, достаточно описать это качество в тексте. Информации нет в постоянстве и периодичности, кроме той, что они имеют место, поэтому не следует приводить на графике слишком много повторяющихся участков процесса. Информативность графика можно повысить, приведя на нем несколько кривых, которые можно сравнивать между собой, но при этом следует учитывать требование к наглядности, которое описано выше. На практике, приведение более пяти кривых на одном графике снижает его наглядность, т.к. бри большом количестве объектов человеку (читателю) сложно произвести их сравнение.

Приведенные выше требования не являются жесткими и не охватывают все возможные случаи. Не всегда можно им строго следовать. Однако, они могут полезны во многих практических случаях.

Система координат. В большинстве тех случаев, которые связаны с задачами имитационного моделирования, рассмотренные в данном пособии, для графиков используется Декартова система координат. В большинстве случаев достаточно двух измерений (плоская кривая), но если это нужно можно использовать 3 измерения (3D поверхность), иногда 4 (раскрашенная 3D поверхность). Большее количество измерений использовать весьма затруднительно. Увеличение количества измерений существенно снижает точность представления. Построить наглядный 3-х мерный график довольно сложно. Поэтому, рекомендуется использовать более чем2-х мерные графики только в тех случаях, когда это действительно необходимо.

Для построения графиков можно использовать различные приложения (MS Office Excel, Libre Office, Open Office, Mathcad, Matlab и др.), в том числе сами системы имитационного моделирования, как правило, имеют такие возможности.

Существует нормативный документ [33], регламентирующий правила построения диаграмм в ЕСКД. Как правило, существующие программные средства позволяют достаточно легко придерживаться этих правил.

4.6 Гистограммы

Гистограмма это геометрическое изображение эмпирической функции плотности вероятности случайной величины, построенное по выборке, полученной в результате эксперимента (измерения в реальной системе или имитационной модели).

Правила построения гистограммы следующие.

Пусть в результате эксперимента получена выборка $X = \{x_1, x_2, \ldots, x_n\}$.

1. Диапазон значений, которые принимает величина *X,* разбивается на несколько *k* интервалов (карманов). Чаще всего размеры этих интервалов Δx выбираются равными, хотя в общем случае, они могут быть различны. Например, $\Delta x = [\max(X) - \min(X)]/k$.

2. Интервалы Δx откладываются по горизонтальной оси.

3. Над каждым интервалом строится прямоугольник, площадь которого пропорциональна числу элементов выборки, попадающих в данный интервал значений. Если все интервалы одинаковы, то высота прямоугольника пропорциональна числу элементов выборки, попадающих в данный интервал значений.

Системы Средства имитационного моделирования, как правило, имеют средства построения гистограмм. Если требуется построить гистограмму другими средствами, то для этого можно использовать различные средства, такие как Matlab, Mathcad, Statistica, Excel, и др. Далее приведем пример построения гистограммы в MS Excel.

Приведем пример гистограммы для выборки из n=1000 случайных чисел, подчиняющихся нормальному закону распределения со средним значением 10 и стандартным отклонением 5.

Пример построения гистограммы приведен на рисунок 4.1.

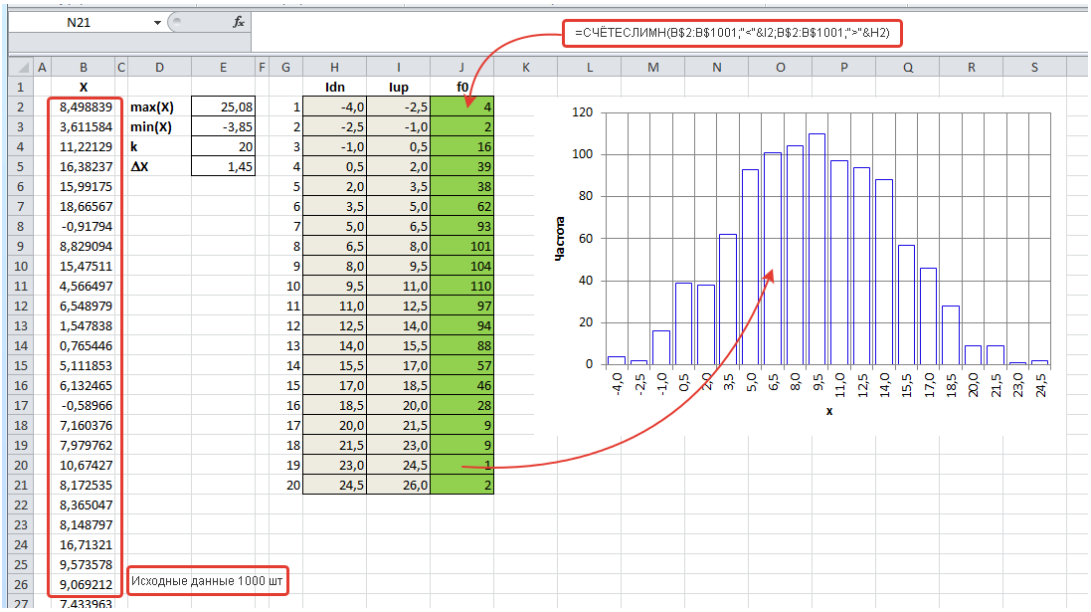


Рисунок 4.1 - Пример построения гистограммы в MS Excel

На листе приведены исходные данные (столбец «B»), они занимают диапазон ячеек B2:B1001. В столбце «E» получены максимальное и минимальное значения данных, задано количество интервалов k=20, и вычислена ширина интервала 1,45. Для простоты ширина интервала выбрана равной 1,5. В столбцах «H» и «I» приведены нижние и верхние границы интервалов. В столбце «J» подсчитывается частота попадания величины в соответствующий интервал (количество попаданий). Для подсчета количества попаданий в интервал используется функция «СЧЁТЕСЛИМН» MS Excel.

Не существует общего строгого правила выбора количества интервалов k (существуют различные подходы, например [20]), поэтому оно выбирается

произвольным образом, как правило, в пределах от 10 до 40. Поскольку гистограмма строится для визуального представления, то количество интервалов должно, в первую очередь, обеспечивать ее наглядность. При выборе k следует учитывать диапазон значений случайной величины, требуемую (желаемую) точность представления значений в границах этого диапазона, количество исходных данных, количество данных, попадающих в каждый интервал. Если имеются интервалы с нулевым значением частоты, то имеет смысл либо увеличить ширину интервалов, либо выбрать интервалы различной ширины.

Следует отметить, что на практике редко удается, как в приведенном выше примере, достичь желаемого результата. Часто эмпирическое распределение имеет «длинный хвост», который мешает наглядному представлению данных.

4.7 Распределения случайных величин, критерии согласия

Построение гистограммы дает представление о форме плотности вероятности случайной величины. Однако, во многих задачах требуется не только продемонстрировать, но и описать полученное эмпирическое распределение. Для описания, если это возможно, используют один из известных законов распределения случайных величин. Задачей анализа в таком случае является проверка гипотезы о возможности описания эмпирического распределения теоретической (аналитической) моделью. Разумеется, первым из критериев выбора аналитической модели является внешнее сходство полученной гистограммы и аналитической модели. При выборе модели полезно руководствоваться следующими свойствами:

-вид случайной величины (дискретное или непрерывное распределение);

-диапазон изменения случайной величины (бесконечное или конечное распределение);

-только положительные значения или положительные или отрицательные значения).

Например, количество пакетов, поступающих в узел связи за интервал времени является дискретной случайной величиной и для его описания следует выбирать дискретное распределение.

Время доставки пакета, если оно может принимать любые значения, может быть описано непрерывным распределением. Время может принимать только положительные значения, поэтому выбираемая модель также должна описывать положительные значения.

Для проверки возможности использования выбранной аналитической модели для описания эмпирического закона. Используют статистические методы называемые критериями согласия [20]. Наиболее часто, на практике, применяются критерии Пирсона (Хи-квадрат) и Колмогорова-Смирнова.

Критерий согласия позволяет отвергнуть или не отвергнуть с заданной доверительной вероятностью гипотезу о том, что эмпирическое распределение может быть описано выбранной аналитической моделью.

Рассмотрим применение критерия Пирсона для полученного выше эмпирического распределение. В качестве гипотезы выберем нормальный закон распределения со средним значением 10 и стандартным отклонением 5.

В данном примере мы заранее знаем, какие параметры распределения нужно выбрать (среднее 10, стандартное отклонение 5). В практических случаях эти значения сначала нужно оценить по выборке.

Для применения критерия вычисляется статистика

$$\chi^2 = \sum_{i=1}^{k} \frac{(n_i - p_i n)^2}{p_i n} \qquad (4.11)$$

где n_i - экспериментальная частота попадания в i-й интервал;

n - общее число наблюдений;

p_i - теоретическая вероятность попадания величины в i-й интервал;

k - количество интервалов.

Вычисленное значение сравнивается со значением распределения χ^2 с n-1 степенью свободы, для требуемой доверительной вероятности α.

Выберем α=0,95, число степеней свободы для нашего примера равно 20-1=19. В MS Excel значение критерия χ^2 получим с помощью функции ХИ2.ОБР(0,95; 19). Это значение равно 30,14. Полученное согласно (11) значение статистики 26,5<30,14. Следовательно, принятая нами гипотеза не отвергается. Применение критерия в MS Excel приведено на рисунке 4.2.

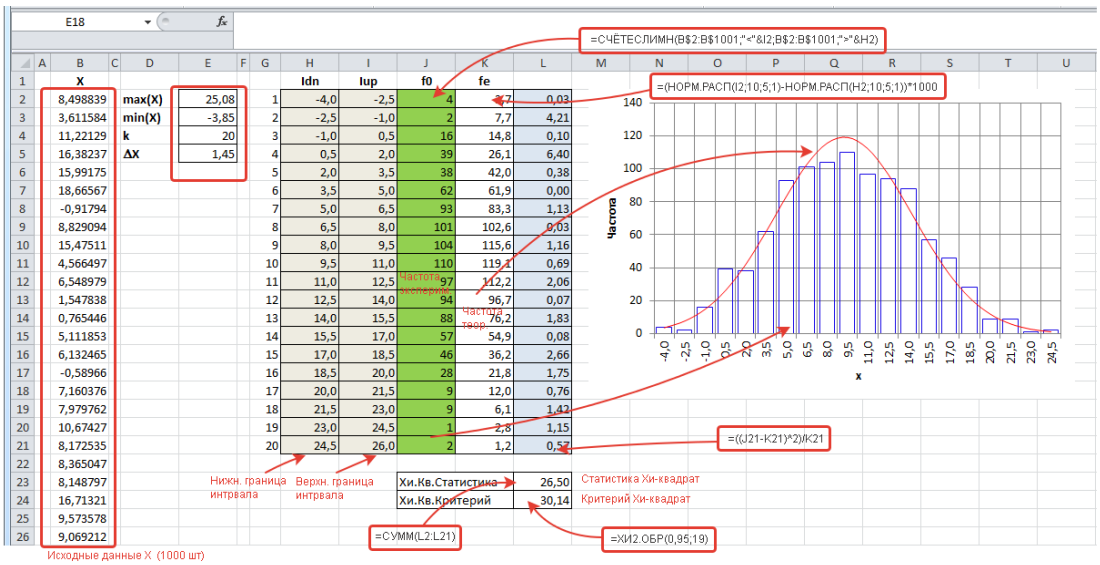


Рисунок 4.2 - Применение критерия Пирсона

Значение теоретической вероятности p_i в формуле (11) вычисляется через значения теоретической (гипотетической) функции распределения как

$$p_i = F(x_{up}, \mu, \sigma) - F(x_{dn}, \mu, \sigma) = \Phi(x_{up}, \mu, \sigma) - \Phi(x_{dn}, \mu, \sigma) \qquad (4.12)$$

где $F(x_{up},\mu,\sigma)$ - значение функции распределения в верхней границе *i*-го интервала (в данном случае, нормальное распределение $\Phi(x_{up},\mu,\sigma)$);

$F(x_{dn},\mu,\sigma)$ - значение функции распределения в нижней границе *i*-го интервала (в данном случае, нормальное распределение $\Phi(x_{dn},\mu,\sigma)$).

Как было отмечено выше, при построении гистограмм и применении критериев согласия следует стремиться к тому, чтобы во все интервалы попадало достаточно много значений (по некоторым мнениям более 10). Однако, этого не всегда можно добиться, изменяя длину интервалов. Примером этого может служить распределения с длинным «хвостом». Например, случайная величина (случайный интервал времени) *X* принимает значения от 0 до 3 с, при этом 90% значений этой величины находятся в интервале от 0 до 0,01 с и только 10% выходят за эти границы. При попытке построить такую гистограмму описанным способом она окажется совершенно не наглядной. Все 90% значений попадут в первый интервал и только 10% в остальные интервалы. В этом случае целесообразно «растянуть» начальный участок, выбрав ширину интервалов исходя не из максимального значения 3 с, а из значения 0,01 с. При этом следует сделать замечание, что на гистограмме приведены только 90% значений, а оставшиеся 10% не отображены и максимальное наблюдаемое значение составило 3с. Сравнивать такую гистограмму с аналитической моделью не следует, так как при этом не будут учтены исключенные 10%, которые могут быть также важны для описания процесса в целом.

5 Измерения параметров трафика

5.1 Описание измерений

Точка измерения

В данном исследовании выполняется анализ клиентского трафика на интерфейсе пользователь сеть. Модельная сеть представляет собой компьютер, включенный в локальную сеть передачи данных и имеющий доступ к сети Интернет, либо непосредственно включенный в сеть провайдера услуг доступа в Интернет.

Средства измерений

Для измерений параметров трафика используется свободно распространяемое программное обеспечение Wireshark.

Подготовка к проведению измерений

Скачать с сайта https://www.wireshark.org/download.html и установить, если ранее не было установлено программное обеспечение Wireshark.

Выбор услуги

Анализ трафика выполняется для услуги потокового видео. Услуга предоставляется в сети Интернет адрес сайта 1tv.ru.

5.2 Проведение измерений и подготовка данных

Открываем в браузере указанный сайт и выбираем онлайн трансляцию. В окне браузера отображается видео трансляции.

Запускаем Wireshark и выбираем локальную сеть (Ethernet карту), запускаем захват пакетов.

Наблюдаем в окне Wireshark непрерывно пополняемый список пакетов регистрируемых на выбранном интерфейсе (рисунок 5.1).

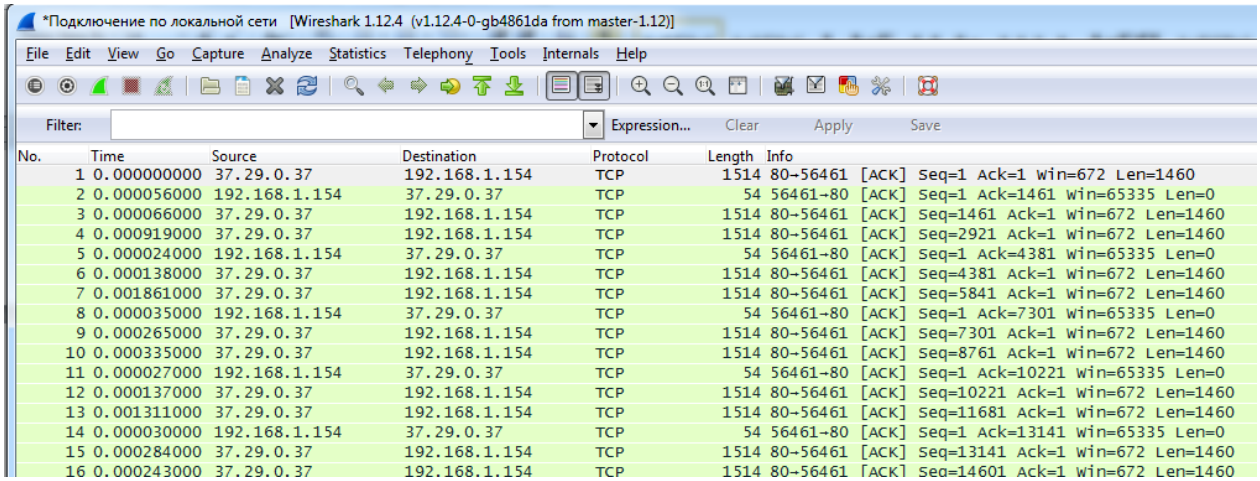


Рисунок 5.1 – Результат регистрации потока пакетов

Продолжаем процесс, пока число захваченных пакетов не достигнет, ориентировочно 20 000, после чего останавливаем процесс захвата пакетов.

Полученные данные, при необходимости, можно сохранить в файл для дальнейшей обработки.

2.2 Подготовка данных для анализа распределения

Среди захваченных пакетов выделяем пакеты потока видео. Для этого находим среди них IP адрес источника пакетов.

Введя в поле «Filter» строку ip.src==XX.XX.XX.XX, где XX.XX.XX.XX IP адрес источника и нажав Enter отфильтровываем только пакеты, поступившие от выбранного источника (рисунок 5.2).

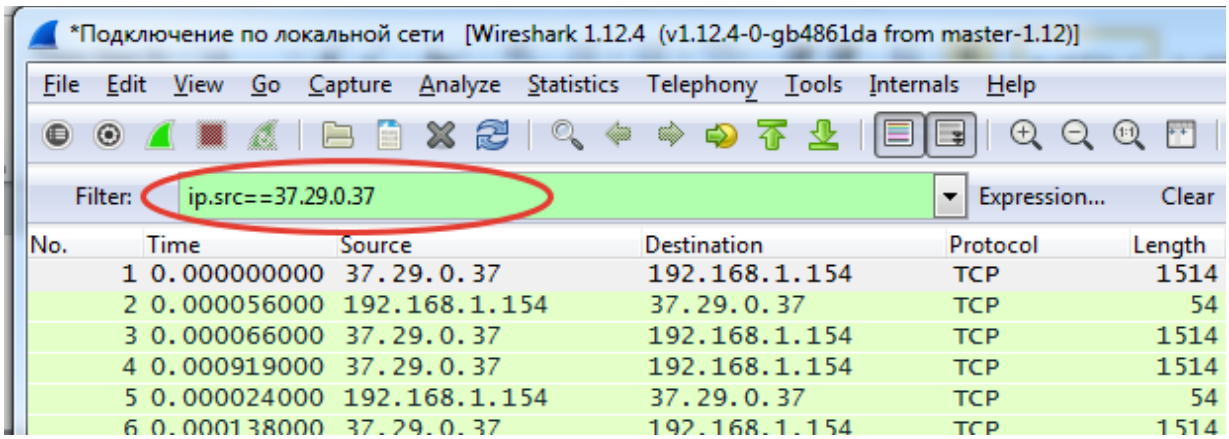
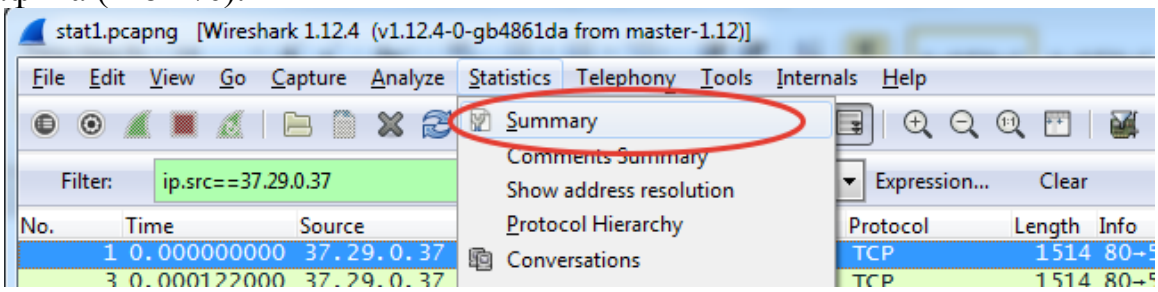


Рисунок 5.2 – Фильтрация потока

5.3 Оценка параметров трафика

В меню Wireshark выбираем пункт Statistics->Summary (рисунок 5.3). В открывшемся окне находим такие параметры как: интенсивность пакетов (среднее число пакетов в секунду), средний размер пакета, интенсивность трафика (Мбит/с).



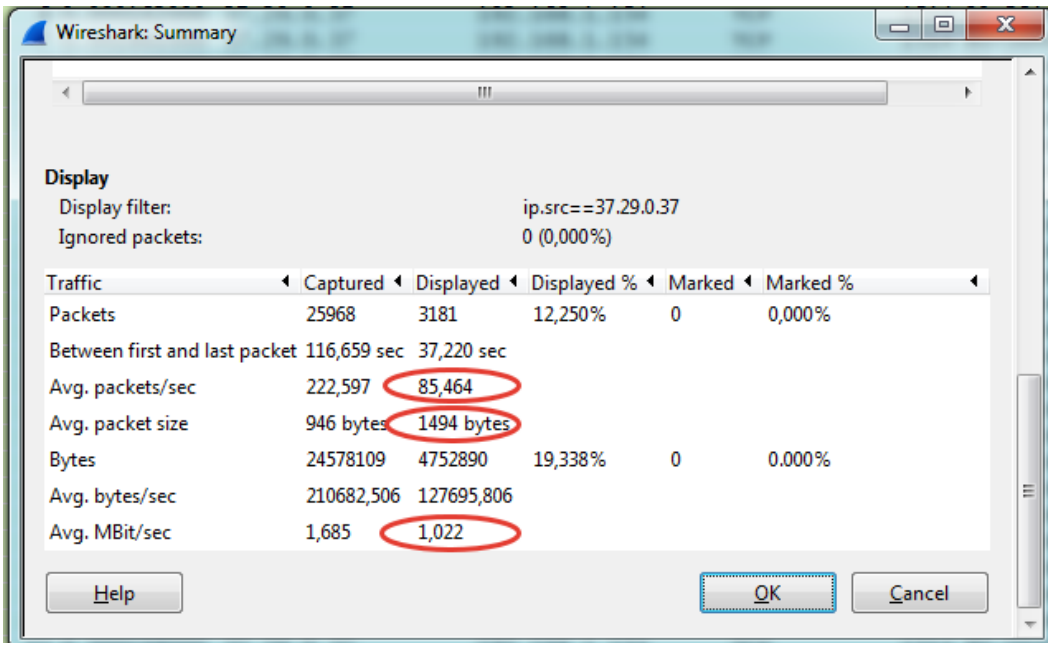


Рисунок 5.3 – Выбор оценок параметров трафика

Согласно данным программы, интенсивность пакетов λ=85,464 пакета/с, средний размер пакета L=1494 байта, интенсивность трафика a=1,022 Мбит/с.

5.4 Доверительные интервалы для полученных оценок

1. Доверительный интервал для интенсивности пакетов

Выберем меню Statistics->IO Graph. В полк «Filter» вводим IP адрес источника пакетов и нажимfем Enter. Проверяем значения в полях, выделенных на рисунке 5.4.

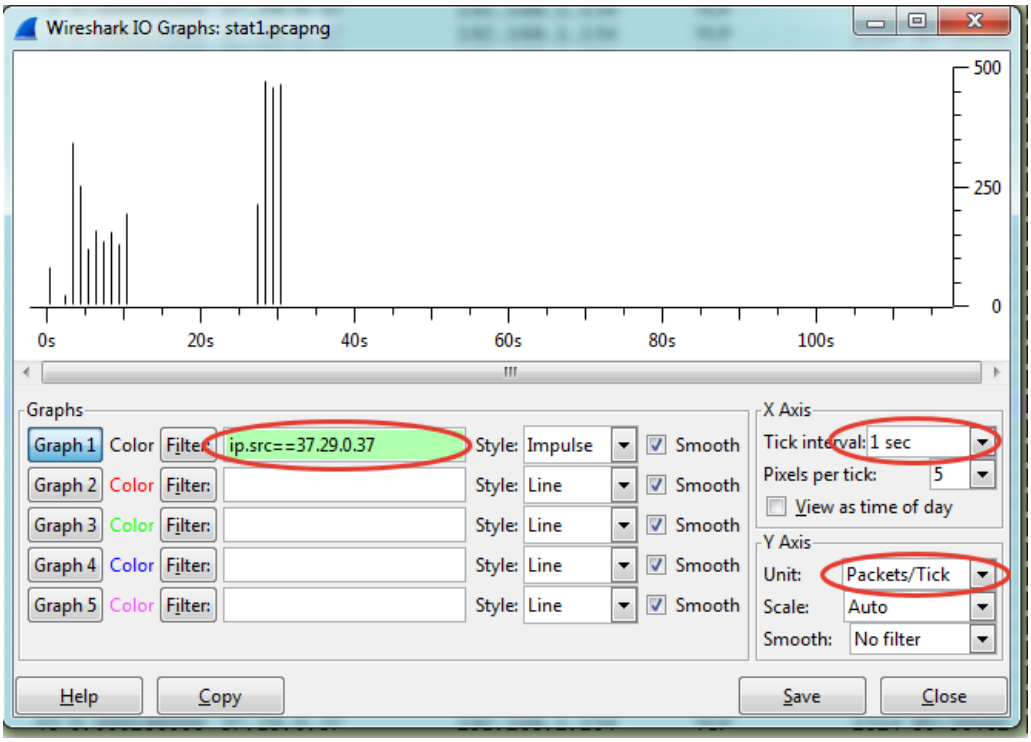


Рисунок 5.4 – Получение значений за секундные интервалы

Нажимаем кнопку «Copy». Открываем табличный процессор (в данном примере MS Excel) и вставляем данные через «мастер вставки», как текст с разделителями, выбрав в качестве разделителя запятую. В результате получаем таблицу приведенную на рисунке 5.5.

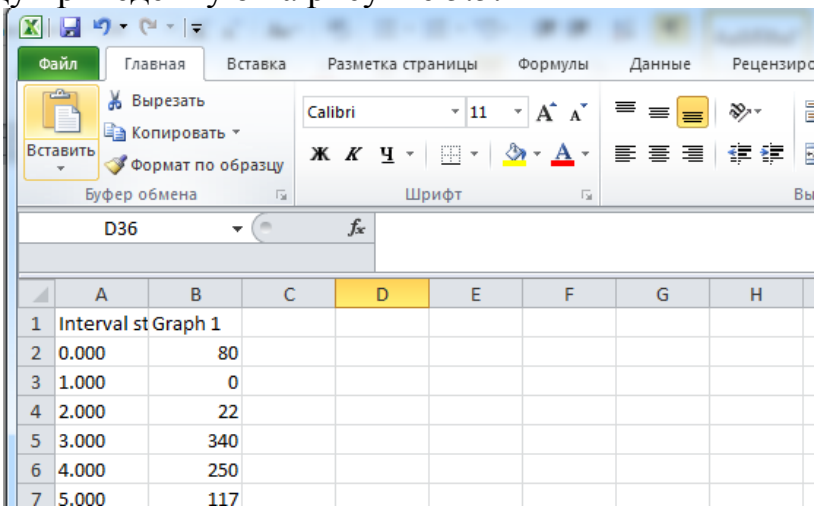


Рисунок 5.5 – Экспорт данных в табличный процессор

В таблице число строк равно числу секунд. В первом столбце номер секунды, а во втором столбце число зарегистрированных пакетов.

Для проверки своих действий вычислим интенсивность пакетов просуммировав число пакетов во втором столбце и разделив его на время регистрации пакетов (Рисунок 3 – 37,220 с), рисунок 5.6.

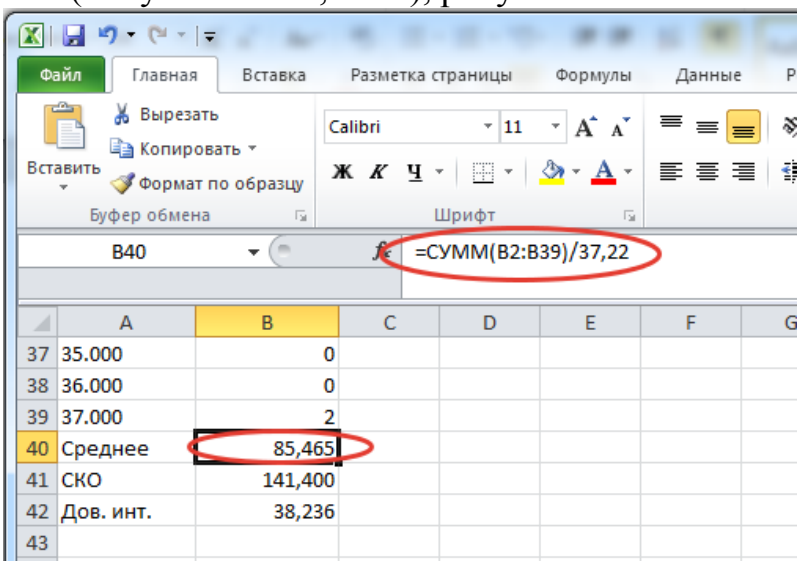


Рисунок 5.6 - Вычисление среднего значения

В результате получили 85,465 (на рисунке 3 – 85,464). Разница результатов в 0,001 обусловлена операциями округления, будем считать ее незначительной.

Далее для данного столбца вычислим среднеквадратическое отклонение (СКО) и полуширину доверительного интервала, рисунок 5.7.

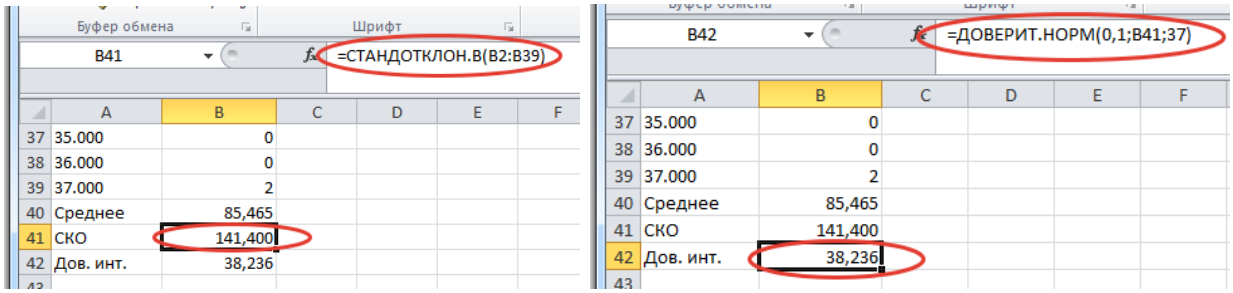


Рисунок 5.7 - Вычисление доверительного интервала для интенсивности пакетов

При вычислении доверительного интервала выберем доверительную вероятность 0,9.

Таким образом, результат оценки интенсивности пакетов можно записать как

$\lambda = 85 \pm 38$ пакетов/с.

(Как видим, относительная погрешность полученной оценки составляет около 45%)

Примечание. Если в задании требуется более высокая точность, то следует увеличить объем выборки, продолжив измерения. Необходимый объем выборки можно будет определить как

$$N = \left(g_{1-\frac{\alpha}{2}} \frac{\sigma}{\bar{\lambda}\,\delta_0} \right)^2$$

Например, при допустимой относительной погрешности $\delta_0 = 0{,}1$ (10%) *и доверительной вероятности 0,9 требуемое количество измерений составит*

$$N = 1{,}64 \frac{141.400}{38{,}236 \cdot 0{,}1} \approx 3678.$$

2. Доверительный интервал для интенсивности трафика

Доверительный интервал для интенсивности трафика вычисляется аналогичным образом с той разницей, что в начале (Рисунок 5) в поле «Unit» выбирается «Bits/Tick».

Последующие операции аналогичны приведенным выше. В результате получаем (Рисунок 5.8).

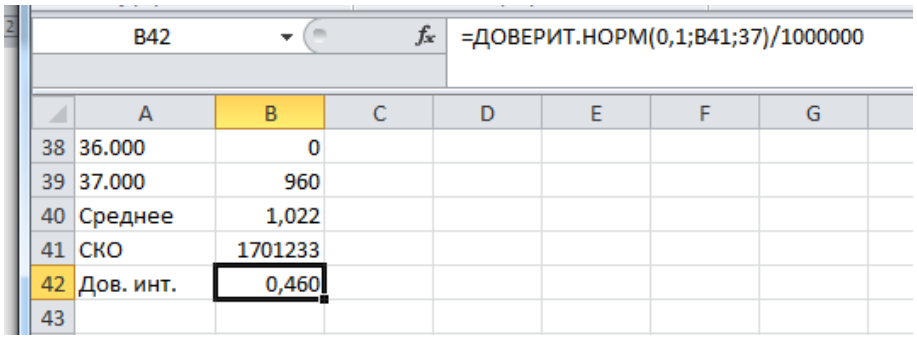


Рисунок 5.8 - Вычисление доверительного интервала для интенсивности трафика

Таким образом, результат оценки интенсивности трафика можно записать как

$a = 1{,}00 \pm 0{,}46$ Мбит/с.

При доверительной вероятности 0,9.

3. Доверительный интервал для размера пакета

Из основного окна Wireshark сохраняем отфильтрованные пакеты в файл, рисунок 5.9.

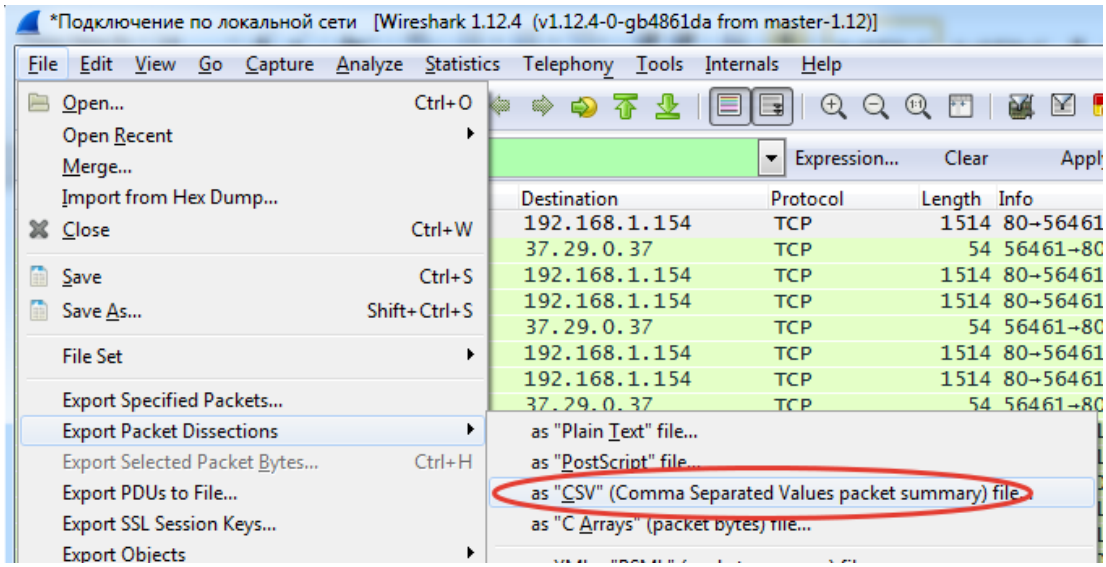


Рисунок 5.9 – Сохранение данных в текстовый файл

Открываем сохраненный файл из текстового процессора (в данном примере Excel) с использованием мастера импорта, в результате чего получаем таблицу, рисунок 5.10.

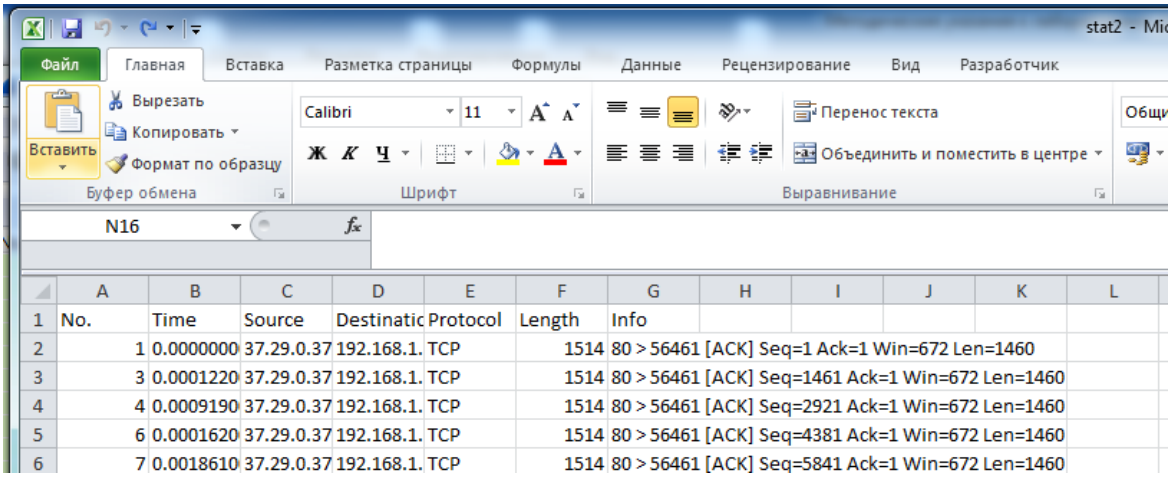


Рисунок 5.10 – Результат экспорта данных

Удаляем лишние столбцы, оставляя только «Time» и «Length».

Для столбца «Length» вычисляем среднее значение, СКО и доверительный интервал, как это было сделано ранее (рисунок 5.11).

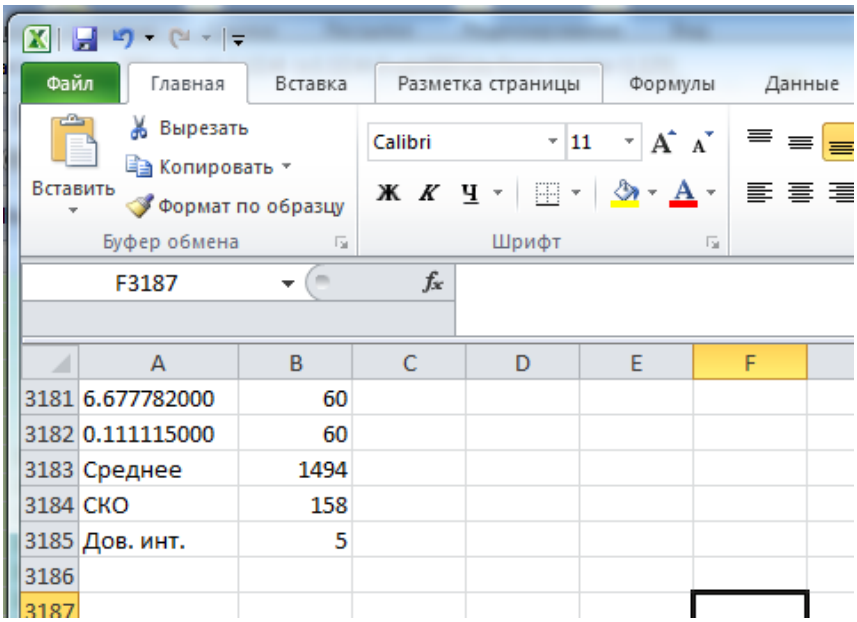


Рисунок 5.11 – Результат оценки

Таким образом, результат оценки среднего размера пакета можно записать как

$L = 1494 \pm 5$ байт.

(Как видим, относительная погрешность полученной оценки составляет около 0,3%)

Результаты оценки параметров трафика

Таблица 5.1 – Результаты оценки параметров трафика

№	Параметр	Ед. измерения	Значение
1	Интенсивность пакетов	пакетов/с	85±38
2	Интенсивность трафика	Мбит/с	1,00±0,46
3	Размер пакета	Байт	1494±5

Характеристики эксперимента

Таблица 5.2 – Параметры эксперимента

№	Параметр	Ед. измерения	Значение
1	Продолжительность	с	37,220
2	Число пакетов	шт.	3181

Литература

1. Б.С. Гольдштейт, Н.А. Соколов, Г.Г. Яновский Сети связи. СПб. «БХВ-Петербург, 2010 г.

2. Б.С. Лившиц, А.П. Пшеничников, А.Д. Харкевич Теория телетрафика. М. «Связь», 1979 г.

3. Приказ №113 Минкомсвязи РФ от 27.09.2007 г.

4. ГОСТ Р 53111- 2008 «Устойчивость функционирования сети связи общего пользования».

5. М.О. Асанов, В.А. Баранский, В.В. Расин Дискретная математика: графы, матроиды, алгоритмы. М. «РХД», 2001 г.

6. Рекомендация ITU-T Y.1540

7. Рекомендация ITU-T Y.1541

8. Клейнрок, Л. Теория массового обслуживания / Л. Клейнрок. – М.: Машиностроение, 1979. – 432 с.

9. Н.Б. Зелигер, О.С. Чугреев, Г.Г. Яновский Проектирование сетей и систем передачи дискретных сообщений. М.: «Радио и связь», 1984 г. – С. 177.

10. РД 45.120-2000 Нормы технологического проектирования. Городские и сельские телефонные сети.

11. А.Е. Кучерявый, А.И. Парамонов, Е.А. Кучерявый. Сети связи общего пользования. Тенденции развития и методы расчёта – М.: ФГУП ЦНИИС, 2008. – 290 с.

11. М.А. Шнепс, Численные методы теории телетрафика. М. :Связь. – 1974. – 232 с.

12. М.А. Шнепс–Шнеппе, Системы распределения информации. Методы расчета. М. : Связь. 1979. – 344 с.

13. С.Н. Степанов Основы телетрафика мультисервисных сетей. М. «Экотрендз», 2010 г.

14. Н.А. Соколов Эволюция местных телефонных сетей. Пермь. «Книга», 1994 г.

15. Н.А. Соколов Задачи планирования сетей электросвязи. СПб. «Техника связи», 2012 г.

16. О.И. Шелухин, А.В. Осин, С.М. Смольский. Самоподобие и фракталы. Телекоммуникационные приложения М.: Физматлит. – 2008. – 368 с.

17. Цыбаков, Б.С. Модель телетрафика на основе самоподобного случайного процесса / Б.С. Цыбаков. – М.: Радиотехника. 1999. №5. – С. 24 – 31.

18. Кучерявый, А.Е. Пакетная сеть связи общего пользования / А.Е. Кучерявый, Л.З. Гильченок, А.Ю. Иванов. С.-Петербург: Наука и техника, 2004. – 272 с.

19. Кучерявый, А.Е. Сети связи следующего поколения / А.Е. Кучерявый, А.Л. Цуприков. – М.: Центральный научно-исследовательский институт связи (ЦНИИС), 2006. – 278 с.

20. Е.С. Вентцель, Л.А. Овчаров Теория вероятностей. М. «Наука», 1969 г.

21. Ю.Н. Тюрин, А.А. Макаров Статистический анализ данных на компьютере. М.: «ИНФРА», 1998 г.

22. Н. Джонсон, Ф. Лион Статистика и планирование эксперимента в технике и науке. Методы обработки данных. М. Мир.-1980.-610с.

23. Р. А. Фишер Статистические методы для исследователей. М. Гссвязьиздат.1958.-267с.

24. Парамонов, А.И. Проблемы развития инфокоммуникационных услуг и их влияние на перераспределение трафика / Парамонов А.И., Сенькина Н.С. // СПбГУТ. Информационные технологии и телекоммуникации.-2016. Т. 4. № 1.-С.46-54.

25. Парамонов, А.И. Модели потоков трафика для сетей M2M / Парамонов А.И. // М. Электросвязь. 2014.-№ 4.-С. 11-16.

26. Парамонов, А.И. Управление трафиком машина-машина на основе расписания / Парамонов А.И. // М. Системы управления и информационные технологии.-2014.-Т. 56.-№ 2.-С. 84-88.

27. Парамонов, А.И. Миграция речевого трафика в современных сетях связи / Парамонов А.И., Кучерявый А.Е. // Электросвязь. 2007.-№ 12.-С.20-22.

28. Кучерявый, А.Е. Сети связи с малыми задержками / Кучерявый А.Е., Парамонов А.И., Аль-Наггар Я.М. //М. Электросвязь.-2013.-№ 12.-С.15-19.

29. Тарасов, Д.В. Особенности видеотрафика для сетей связи следующего поколения / Тарасов Д.В., Парамонов А.И., Кучерявый А.Е. // Электросвязь. 2010.-№ 2.-С.37-43.

30. Кучерявый, А.Е. Реструктуризация трафика сетей связи и новые подходы к прогнозированию их развития / Кучерявый А.Е., Ревелова З.Б., Парамонов А.И. // Электросвязь. 2003.-№ 2.-С.9-12.

31. Киричек, Р.В. Эволюция исследований в области беспроводных сенсорных сетей / Киричек Р.В., Парамонов А.И., Прокопьев А.В., Кучерявый А.Е. // СПбГУТ Информационные технологии и телекоммуникации. 2014.-№ 4 (8).-С.29-41.

32. Стандарт СЭВ СТ СЭВ 543-77 "Числа. Правила записи и округления" (введен в действие постановлением Государственного комитета стандартов Совета Министров СССР от 30 декабря 1977 г. N 3157).

33. Р 50-77-88 Единая система конструкторской документации. Правила выполнения диаграмм. М. Государственный комитет СССР по стандартам. 1989.

Цель университета – удовлетворение потребностей личности в интеллектуальном, культурном и нравственном развитии посредством получения образования различных уровней и направленности; удовлетворение потребностей общества и государства в высококвалифицированных кадрах.

КАФЕДРА СЕТЕЙ СВЯЗИ И ПЕРЕДАЧИ ДАННЫХ:

http://www.seti.sut.ru/

Кафедра сетей связи и передачи данных (СС и ПД) Кафедра сетей связи была образована в 1998 году. В связи с оптимизацией структуры вуза произошло слияние кафедры сетей связи и кафедры многоканальных систем передачи в единую структуру, также было проведено объединение с кафедрой основ передачи дискретных сообщений (ОПДС). Это позволило создать мощное подразделение вуза — кафедру сетей связи и передачи данных — охватывающее вопросы архитектуры, оптимизации и проектирования сетей связи. В настоящее время на кафедре работает 6 докторов наук и 24 кандидата наук.

Среди курсов кафедры особое внимание уделяется вопросам построения сетей связи, студентами изучаются телекоммуникационные технологии, современные методы моделирования, проектирования и расчета сетей, разработке ориентированного на сетевые вопросы программного обеспечения. Курсы постоянно обновляются в соответствии с новейшими разработками лидеров в области телекоммуникаций.

Сегодня на кафедре изучаются принципы построения самоорганизующихся сетей, прогнозы развития Всемирной сети связи в направлении реализации концепции Интернета Вещей, сетей автомобильного транспорта и Интеллектуальной Транспортной Системы, системы е-здоровья, медицинских сетей. Также изучаются сверхплотные сети, сети 4G (частично), 5G.

На кафедре студенты имеют возможность принимать участие в научных конференциях, в том числе и международных, и в научно-исследовательской работе, связанной с проектами для ведущих российских и мировых производителей телекоммуникационного оборудования и операторов связи. Студенты принимают активное участие в международных хакатонах по Интернету Вещей и дополненной реальности, традиционно занимая призовые места.

Выпускники кафедры обладают компетенциями:

-знаниями основ философии и методологии науки и методы научных исследований, методов проектирования распределенных информационных систем, их компонентов и протоколов их взаимодействия, знанием технических характеристик телекоммуникационных систем, нормативных требований;

-умением: формулировать цели и задачи исследования, действовать в нестандартных ситуациях, проектировать системы с параллельной обработкой данных и высокопроизводительные системы, и их компоненты, оценивать технические возможности и вырабатывать рекомендации по построению систем и сетей передачи, применять современные методы исследования, оценивать и представлять результаты выполненной работы;

-владением: способностью к саморазвитию и самореализации, способностью к абстрактному мышлению обобщению, анализу, систематизации и прогнозированию, методами постановки научных экспериментов, способностью проектировать телекоммуникационные системы и проводить анализ проектных решений по обеспечению безопасности телекоммуникационных систем, иностранным языком в научной и деловой сфере, способность производить оценку технических характеристик телекоммуникационных систем.

Трудоустройство выпускников кафедры возможно на любых предприятиях, занимающееся разработками в сфере телекоммуникаций.

Студенты могут проходить практику в компаниях, занимающихся разработками в сфере телекоммуникаций, в Санкт-Петербурге и зарубежных вузах.

Мы готовим квалифицированных бакалавров и магистров в области инфокоммуникационных технологий с новыми знаниями, образом мышления и способностями быстрой адаптации к современным условиям труда.

Парамонов А.И., Маколкина М.А., Киричёк Р.В., Выборнова А.И.

Моделирование и оптимизация в системах и сетях электросвязи
Раздел Математическое моделирование

Учебное пособие

В авторской редакции
Редакционно-издательский отдел СПбГУТ
Зав. РИО
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
СПбГУТ
197101, Санкт-Петербург, пр. Большевиков, 22