«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»
(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова)

ДОПУСКАЕТСЯ К ЗАЩИТЕ:

Факультет И

Выпускающая кафедра И9

Группа И942

Заведующий кафедрой И9
шифр кафедры

Матвеев С.А.
Фамилия И.О.

подпись

« 6 » июня 2018 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Барского Никиты Олеговича
Фамилия, имя, отчество обучающегося

На тему «Реализация курса практических работ по предмету «Сети ЭВМ и телекоммуникации» на свободном программном обеспечении»

Направление подготовки 09.03.02 Информационные системы и технологии
индекс направления / полное наименование направления

Консультант:
при необходимости

подпись

ученая степень, ученое звание

Фамилия ИО

« » 201__ г.

Руководитель:

подпись

Бондарев Е.С.
ученая степень, ученое звание / Фамилия ИО

« 6 » июня 2018 г.

Обучающийся:

подпись

Барский Н.О.
Фамилия ИО

« 6 » июня 2018 г.

САНКТ-ПЕТЕРБУРГ
2018 г.

РЕФЕРАТ

Пояснительная записка 60 с., 31 рис., 19 источников

КОМПЬЮТЕРНАЯ СЕТЬ, МОДЕЛИРОВАНИЕ КОМПЬЮТЕРНЫХ СЕТЕЙ, СЕТЕВОЙ СИМУЛЯТОР, ПРАКТИЧЕСКИЕ РАБОТЫ, OMNeT++

Объектом исследования в рамках данной работы являются компьютерные сети, а предметом исследования имитационное моделирование компьютерных сетей с помощью сетевых симуляторов.

Выпускная работа нацелена на реализацию курса практических работ по предмету «Сети ЭВМ и телекоммуникации» на свободном программном обеспечении.

В ходе выполнения выпускной квалификационной работы был произведен анализ предметной области, произведен обзор средств моделирования сети, определены требования пользователей, на основании которых выбрано средство реализации. Был реализован курс практических работ на выбранном сетевом симуляторе.

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

API – программный интерфейс

ASIC – интегральная схема специального назначения

BDPU – кадр протокола STP

CLI – интерфейс командной строки

CRC – алгоритм нахождения контрольной суммы

ID – уникальный идентификатор

IDE – интегрированная среда разработки

IEEE – институт инженеров электротехники и электроники

IP – протокол сетевого уровня

LLC – подуровень канального уровня управления логической связью

MAC – подуровень канального уровня управления доступа к среде

NED – язык описания топологии сети в сетевом симуляторе OMNeT++

OSI – эталонная сетевая модель

RSTP – улучшенная версия протокола STP

STP – протокол канального уровня

TCP/IP – набор протоколов

ПК – персональный компьютер

ПО – программное обеспечение

ЭВМ – электронно-вычислительная машина

ВВЕДЕНИЕ

В настоящее время уровень развития компьютерных технологий стремительно растет, вследствие чего, компьютерные сети стали одним из главных средств коммуникации.

Вычислительная (компьютерная) сеть – взаимосвязанная совокупность территориально рассредоточенных систем обработки данных, средств и (или) систем связи и передачи данных, обеспечивающая пользователям дистанционный доступ к ее ресурсам и коллективное использование этих ресурсов [1]. Использование компьютерных сетей позволяет осуществлять связь и обмен данными и информацией между вычислительными устройствами. В результате, при подготовке кадров технических специальностей, изучение компьютерных сетей является важной частью образовательной программ. Но для изучения практической части (создание топологии сети, настройка сетевых устройств, работа сетевых протоколов) необходимо дорогое сетевое оборудование и место для его размещения, а также создание специальных лабораторий. Проблему решает программное обеспечение, моделирующее работу компьютерной сети.

На сегодняшний день создано большое множество программных продуктов, моделирующих работу сети, дающие ощущение работы с реальной компьютерной сетью, что позволяет использовать их как полноценную замену сетевому оборудованию в учебном процессе.

Актуальность создания практических работ с использованием среды сетевого моделирования заключается в том, что требуется значительно меньше экономических затрат на организацию специальных лабораторий, при этом создавая возможность освоить все необходимые навыки работы с сетевым оборудованием, используя только компьютер. Помимо этого, моделируемую сеть намного проще изменять, это позволит проводить различные эксперименты, ускоряющие обучение.

Объектом исследования в рамках данной работы являются компьютерные сети, а предметом исследования имитационное моделирование компьютерных сетей с помощью сетевых симуляторов.

Из всего вышесказанного, цель выпускной квалификационной работы – реализация курса практических работ по предмету «Сети ЭВМ и телекоммуникации» на свободно распространяемом программном обеспечении, моделирующем работу сети.

Для достижения поставленной цели необходимо решить следующие задачи:

1) произвести анализ предметной области;

2) определить требования пользователей к программному продукту;

3) на основании требований выбрать программный продукт;

4) реализовать практические работы на выбранном программном продукте.

Выпускная квалификационная работа содержит введение, три раздела и заключение. В первом разделе производится анализ предметной области, во втором разделе определяются методы и процессы моделирования компьютерных сетей, требования пользователей и производится анализ программ моделирования сетей, постановка задачи, в третьем разделе описывается ход реализации практических работ.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Целью данной главы является постановка задачи, изучение теоретических положений компьютерных сетей, обоснование выбора симулятора. Во-первых, создавая имитационную модель в сетевом симуляторе, необходимо понимать топологию сети, ее сетевую архитектуру и используемую сетевую модель [2]. Во-вторых, главной задачей лабораторных работ является приобретение знаний о принципах построения сетей, работы сетевых протоколов, поэтому при выборе сетевого симулятора первостепенную роль играют требования пользователей.

В различных источниках можно встретить большое множество классификаций компьютерных сетей, однако чаще всего используют классификацию по территориальной распространенности. С ее помощью разделяют локальные, региональные и глобальные. В рамках лабораторных работ будут использоваться локальные сети – сети, устройства которых расположены на небольшом расстоянии друг от друга. Такие сети объединяют компьютеры, размещенные внутри одного здания или в нескольких рядом расположенных зданиях.

При связывании в сеть более, чем два компьютера, возникает вопрос выбора топологии сети – конфигурации физических и информационных связей между устройствами.

1.1 Топологии компьютерных сетей

Разделяют физическую и логическую топологии. Физическая топология описывает физическое расположение устройств и каналов связей между ними, логическая описывает маршруты и способы передачи данных. Стоит отметить, что физическая и логическая топологии могут совпадать.

От выбора физической топологии зависят такие характеристики сети, такие как, надёжность, расширяемость, стоимость [2].

Физические топологии делятся на полносвязные и неполносвязные. Полносвязные соответствуют сети, в которой каждый компьютер напрямую связан с каждым. Ее схема представлена на рисунке 1.

Рисунок 1 – Полносвязная топология [2]

Все другие варианты основаны на неполносвязных топологиях, когда для обмена данными между двумя компьютерами может потребоваться транзитная передача данных через другие узлы сети [2].

Неполносвязные физические топологии разделяют на:

– топология «шина»;

– топология «кольцо»;

– топология «звезда»;

– смешанная топология.

1.1.1 Шинная топология

Сети с шинной топологией, схема которой представлена на рисунке 2, используют линейный моноканал передачи данных, на концах которого устанавливаются оконечные сопротивления (терминаторы). Каждый компьютер подключается к коаксиальному кабелю с помощью Т-образного разъема. Данные от передающего узла сети передаются по шине в обе стороны, отражаясь от оконечных терминаторов.

Информация поступает на все узлы, но принимается только тем узлом, которому она предназначается. В топологии логическая шина среда передачи данных используются совместно и одновременно всеми ПК сети, а сигналы от ПК распространяются одновременно во все направления по среде передачи. Так как передача сигналов в топологии физическая шина является широковещательной, то логическая топология данной локальной сети является логической шиной.

Рисунок 2 – Шинная топология [3]

Данная топология применяется в локальных сетях с архитектурой Ethernet (классы 10Base-5 и 10Base-2 для толстого и тонкого коаксиального кабеля соответственно) [3].

Достоинства:

– сеть устойчива к неисправностям отдельных узлов;

– сеть легко настраивать и конфигурировать;

Недостатки:

– повреждение кабеля может повлиять на работу всей сети;

– ограниченная длина кабеля и количество рабочих станций;

– сложность определения дефектов соединений.

1.1.2 Топология «звезда»

В сети, построенной по топологии типа «звезда» каждая рабочая станция подсоединяется кабелем (витой парой) к концентратору (Hub). Концентратор обеспечивает параллельное соединение, как изображено на рисунке 3.

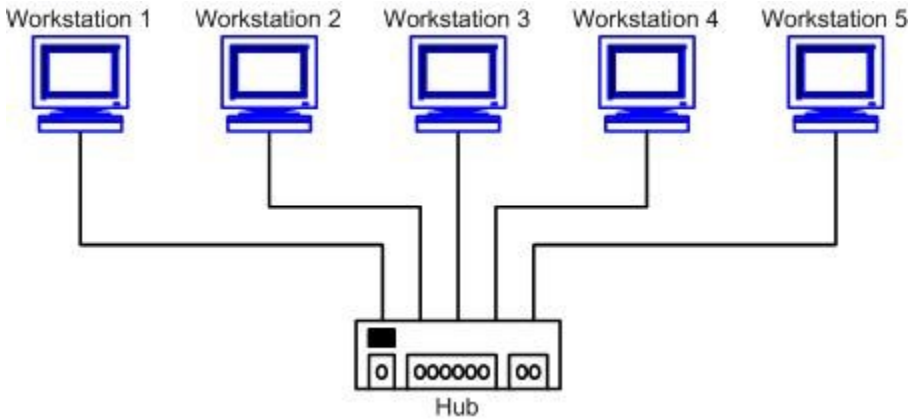

Рисунок 3 – Топология «звезда» [3]

Данные от передающей станции сети передаются через концентратор по всем линиям связи всем ПК. Информация поступает на все рабочие станции, но принимается только теми станциями, которым она предназначается. Так как передача сигналов в топологии физическая звезда является широковещательной, то логическая топология данной локальной сети является логической шиной.

Данная топология применяется в локальных сетях с архитектурой 10Base-T Ethernet [3].

Преимущества:

– расширяемость;

– возможность централизованного управления;

– сеть устойчива к неисправностям отдельных узлов.

Недостатки:

– неисправность концентратора влияет на работу всей сети;

– большой расход кабеля.

1.1.3 Топология «кольцо»

В сети с топологией кольцо, схема которой представлена на рисунке 4, все узлы соединены каналами связи в неразрывное кольцо (необязательно окружность), по которому передаются данные.

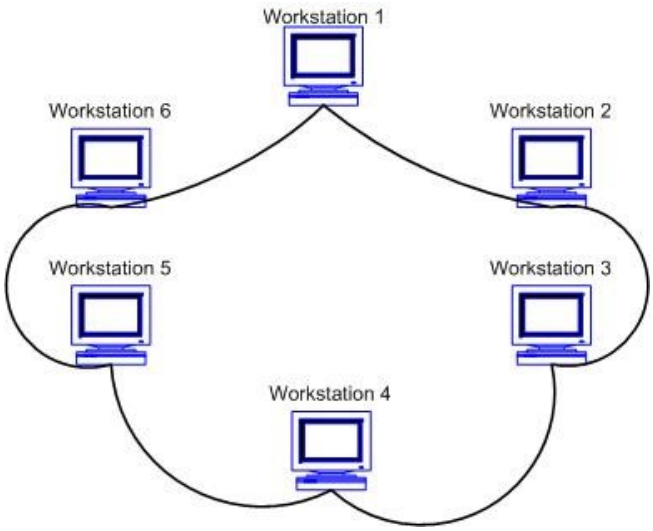

Рисунок 4 – Топология «кольцо» [3]

Выход одного узла соединяется со входом другого. Начав движение из одной точки, данные, в конечном счете, попадают на его начало. Данные в кольце всегда движутся в одном и том же направлении. Принимающая рабочая станция распознает и получает только адресованное ей сообщение.

В сети с топологией типа физическое кольцо используется маркерный доступ, который предоставляет станции право на использование кольца в определенном порядке и исключает возможность возникновения коллизий – наложения наложение двух и более кадров от станций, пытающихся передать кадр в один и тот же момент времени. Логическая топология данной сети – логическое кольцо.

Преимущества:

– простота установки;

– расширяемость;

– возможность устойчивой работы без существенного падения скорости передачи данных при интенсивной загрузке сети.

Недостатки:

– неисправность узла или линии связи влияет на работу всей сети;

– добавление или удаление узла требует остановки сети;

– сложность определения дефектов соединений.

1.1.4 Смешанная топология

В то время как небольшие сети, как правило, имеют типовую топологию, для крупных сетей характерно наличие произвольных связей между компьютерами. Это связано с тем, что созданная на определенном этапе развития сеть с течением времени перестает удовлетворять потребности всех пользователей, и тогда встает проблема расширения ее функциональных возможностей [4]. Проблема расширения конфигурации сети может быть решена как в пределах ограниченного пространства, так и с выходом во внешнюю среду. В таких сетях можно выделить отдельные произвольные подсети, имеющие типовую топологию, поэтому их называют сетями со смешанной топологией.

1.2 Модели компьютерных сетей

В компьютерных сетях для решения задач сетевого взаимодействия используется принцип декомпозиции, то есть разбиения одной сложной задачи на несколько более простых. Архитектура сети отражает декомпозицию общей задачи взаимодействия компьютеров на отдельные подзадачи, которые должны решаться отдельными компонентами сети – конечными узлами (компьютерами) и промежуточными узлами (коммутаторами и маршрутизаторами) [2]. В результате декомпозиции задача организации взаимодействия компьютеров в сети представляется в виде иерархически организованного набора протоколов, который называется стеком протоколов.

Главные сетевые модели компьютерных сетей на сегодняшний день – модель OSI и модель TCP/IP. Они являются многоуровневыми, каждому уровню четко приписаны функции, которые в пределах одной модели не повторяются.

Таким образом, модель OSI, схема взаимодействия компьютеров в которой представлена на рисунке 5, включает в себя семь уровней:

1) физический уровень;
2) канальный уровень;
3) сетевой уровень;
4) транспортный уровень;
5) сеансовый уровень;
6) уровень представления;
7) прикладной уровень.

Модель TCP/IP включает в себя четыре уровня:

1) канальный уровень;
2) сетевой уровень;
3) транспортный уровень;
4) прикладной уровень.

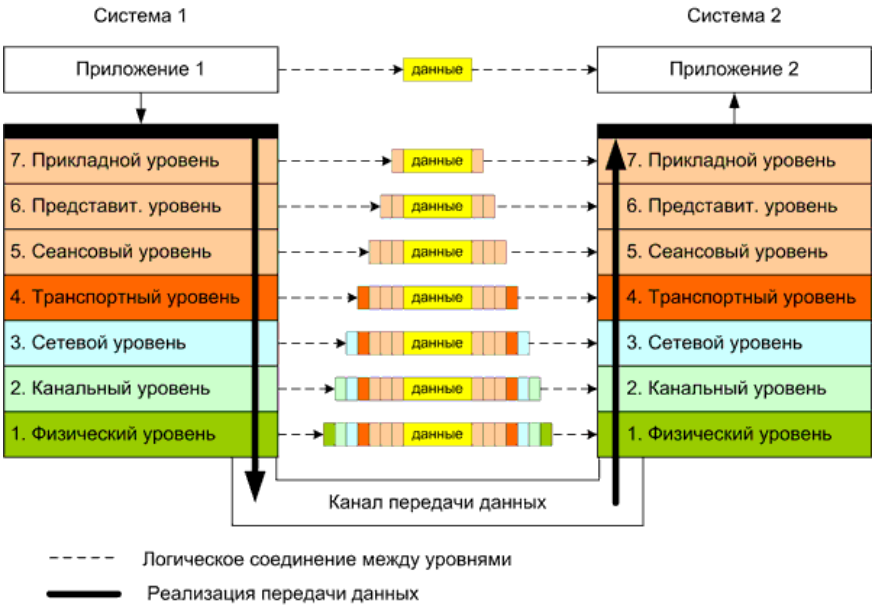

Рисунок 5 – Схема взаимодействия компьютеров в модели OSI [5]

На рисунке 6 представлена схема соответствия уровней моделей OSI и TCP/IP.

Рисунок 6 – Схема соответствия уровней моделей OSI и TCP/IP [6]

Рассмотрев две сетевые модели, заметно, что часть терминов из документации для OSI была заимствована для документации по TCP/IP, вследствие чего OSI стала считаться эталонной сетевой моделью. Наиболее распространенными протоколами являются протоколы модели TCP/IP.

У этих моделей имеется и ряд отличий. Для модели OSI центральными являются три концепции: службы (сервисы), интерфейсы и протоколы. Сервис определяет, что именно делает уровень, интерфейс уровня определяет способ доступа к данному уровню, а протоколы, применяемые в уровне, являются деталями его внутренней реализации. В модели TCP/IP изначально не было четкого разделения между службами, интерфейсом и протоколами, хотя и производились попытки изменить это, чтобы сделать ее более похожей на модель OSI. В результате в модели OSI протоколы скрыты лучше, чем в модели TCP/IP, и при изменении технологии они могут быть относительно легко заменены. Возможность проводить подобные изменения, не затрагивая другие уровни, является одной из главных целей многоуровневых протоколов [7].

Таким образом, используя модель OSI удобно описывать абстрактную структуру компьютерных сетей. С другой стороны, модель TCP/IP включает в себя протоколы, широко используемые уже длительное время. В рамках лабораторных работ для описания общей архитектуры сети решено использовать термины модели OSI, а в реализации протоколов опираться на модель TCP/IP. В настоящее время все устройства, осуществляющие передачу данных по сети, используют сетевую модель TCP/IP.

Для проектирования сетей необходимо рассмотреть принципы работы хотя бы первых трех уровней модели OSI.

1.2.1 Физический уровень

Физический уровень имеет дело с передачей информации по физическим каналам связи. Функции физического уровня реализуются на всех устройствах, подключенных к сети. Для физического уровня не имеет значения смысл информации, которую он передает. Для него эта информация представляет собой однородный поток битов, которые нужно доставить без искажений и в соответствии с заданной тактовой частотой (интервалом между соседними битами) [7].

Все физические носители информации можно разделить на две категории: проводные и беспроводные. К проводниковым носителям относятся коаксиальные кабели, витые пары, линии электропитания и волоконная оптика. К беспроводным носителям – радиосвязь, связь в микроволновом диапазоне, связь в инфракрасном диапазоне, связь в видимом диапазоне [7].

1.2.2. Канальный уровень

Канальный уровень обеспечивает прозрачность соединения для сетевого уровня. Для этого он предоставляет ему следующие услуги:

– установление логического соединения между взаимодействующими узлами;

– согласование в рамках соединения скоростей передатчика и приемника информации;

– обеспечение надежной передачи, обнаружение и исправление ошибок.

Задача установления логического соединения между устройствами решается с помощью механизма коммутации. Коммутацией называется соединение конечных узлов через сеть транзитных узлов. Последовательность узлов, лежащих на пути от отправителя к получателю, образует маршрут. Через один транзитный узел может проходить несколько маршрутов, поэтому он должен уметь распознавать поступающие на него потоки данных, для того чтобы обеспечивать передачу каждого из них именно на тот свой интерфейс, который ведет к нужному узлу.

Информационным потоком, или потоком данных, называют непрерывную последовательность данных, объединенных набором общих признаков, выделяющих эти данные из общего сетевого трафика. Признаки потока могут иметь глобальное или локальное значение. Пара адресов конечных узлов для идентификации потока – это пример глобального признака. Примером признака, локально определяющего поток в пределах устройства, может служить идентификатор интерфейса, на который поступили данные [2].

Выделяют два основных подхода к решению задачи коммутации: коммутация каналов и коммутацию пакетов. Коммутация каналов требует предварительного установления непрерывной связи между передающим и принимающим узлами, что не позволяет передавать данные по маршрутам, имеющим общие участки.

При коммутации пакетов все передаваемые данные разбиваются в исходном узле на части, называемые пакетами. Затем из пакетов формируются кадры, состоящие из поля данных и заголовка. Пакет помещается в поле данных одного или нескольких кадров, а заголовок кадра заполняет служебной информацией. Такой подход позволяет пересылать пакеты без установления постоянного соединения, обеспечивая возможность прохождения по одному физическому участку сети нескольких информационных потоков [2].

Для установления определенных норм и правил работы с сетями были разработаны стандарты сетевых технологий. Созданный в 1973 Ethernet до сих пор является одним из основополагающих стандартов. По нему определяют проводные соединения и электрические сигналы на физическом уровне, формат кадров и протоколы управления доступом к среде – на канальном уровне.

Формат кадра, применяемый для отправки данных, показан на рисунке 7. В начале кадра расположено поле преамбула (заголовок) длиной 8 байт, последний байт называют разделителем начала кадра.

Затем следуют два адреса: получателя и отправителя. Каждый занимает по 6 байт. При проектировании стандарта Ethernet было предусмотрено, что каждая сетевая карта (сетевой интерфейс) должна иметь уникальный шестибайтный номер (MAC-адрес), записанный в ней при изготовлении. Этот номер используется для идентификации отправителя и получателя кадра.

Затем следует поле данных, включающее в себя поля идентификации, поле, содержащие код протокола, поле Type, описывающее тип передаваемых в кадре данных, и сами данные. Размер поля данных ограничен 1500 байт.

Последнее поле кадра стандарта Ethernet – контрольная сумма, представляющая 32-битный код CRC, который позволяет обнаруживать ошибки [7].

Рисунок 7 - Структура кадра Ethernet

Ethernet разделяет функции канального уровня на два различных подуровня: подуровень управления логическим каналом (LLC) и подуровень управления доступом к среде (MAC).

Функции, описанные в модели OSI для канального уровня, назначаются подуровням MAC и LLC. Использование этих подуровней значительно способствует совместимости между разнообразными конечными устройствами. Для Ethernet стандарт IEEE 802.2 описывает функции подуровня LLC, а стандарт 802.3 описывают функции подуровня MAC и функции физического уровня. Управление логическим каналом обрабатывает передачу между верхними уровнями и сетевым программным обеспечением, и нижними уровнями, обычно аппаратными средствами.

В сетях Ethernet используется следующий механизм коммутации пакетов. Коммутация выполняется специальными устройствами – коммутаторами. Коммутаторы отдают кадры только на те порты, для которых эти кадры предназначены. Когда на порт коммутатора приходит кадр, коммутатор проверяет MAC-адреса и узнает, на какой порт этот кадр нужно отдать. Для этого используется специальная таблица коммутации, сопоставляющая MAC-адреса и номера портов. Далее коммутатор пересылает кадр на порт получателя. Порт получателя затем отправляет кадр станции назначения по соединяющей их линии связи [7].

Для построения таблицы коммутации используется алгоритм прозрачного моста. Коммутатор строит свою адресную таблицу на основании пассивного наблюдения за трафиком, циркулирующим в подключенных к его портам сегментах. При этом коммутатор учитывает адреса источников кадров, поступающих на его порты. По адресу источника кадра мост делает вывод о принадлежности узла-источника тому или иному сегменту сети.

Каждый порт коммутатора работает, как конечный узел своего сегмента, за одним исключением – порт может не иметь собственного MAC-адреса, так как он работают в режиме неразборчивого захвата кадров, когда все поступающие на порт кадры, независимо от их адреса назначения, запоминаются на время в буферной памяти коммутатора [2]. Серьезным ограничением функциональных возможностей коммутаторов является отсутствие поддержки петлеобразных конфигураций сети. Последствиями наличия петель являются:

– «размножение» кадра, то есть появление нескольких его копий;

– бесконечная циркуляция копий кадра по петле в противоположных направлениях, а значит, засорение сети ненужным трафиком;

– постоянная перестройка мостами своих адресных таблиц.

Избыточные связи необходимо блокировать, то есть переводить их в неактивное состояние. В сетях с простой топологией эта задача решается вручную путем блокирования соответствующих портов коммутаторов. В больших сетях со сложными связями используются алгоритмы, которые позволяют решать задачу обнаружения петель автоматически. Наиболее известными из них являются алгоритмы STP и RSTP, использующие принцип остовного дерева.

Еще одним вариантом применения избыточных связей является агрегирование линий связи. Отличие техники агрегирования линий связи от алгоритма покрывающего дерева достаточно принципиально:

– протоколы STP и RSTP переводят избыточные связи в горячий резерв, оставляя в рабочем состоянии только минимальный набор линий,

необходимых для связности сегментов сети. В этом случае повышается надежность сети, но не ее производительность;

– при агрегировании физических каналов все избыточные связи остаются в рабочем состоянии, в результате повышается как надежность сети, так и ее производительность, так как несколько физических каналов объединяются в один логический, пропускная способность которого будет суммой объединенных каналов связи [2].

1.2.3. Сетевой уровень

Сетевой уровень отвечает за разработку маршрутов доставки пакетов от отправителя до получателя. Чтобы добраться до пункта назначения, пакету может потребоваться преодолеть несколько транзитных участков между маршрутизаторами. Сетевой уровень оказывается последним уровнем, который имеет дело с передачей данных по всему пути от одного конца до другого.

Для достижения этих целей сетевой уровень должен обладать информацией о топологии сети (то есть о множестве всех маршрутизаторов и связей) и выбирать нужный путь по этой сети, даже если она достаточно крупная. При выборе маршрутизаторов он должен также заботиться о том, чтобы нагрузка на маршрутизаторы и линии связи была, по возможности, более равномерной [7].

Чтобы связать между собой сети, построенные на основе отличающихся технологий, нужны дополнительные средства, и такие средства предоставляет сетевой уровень. Функции сетевого уровня реализуются группой сетевых протоколов и маршрутизаторами.

Одной из функций маршрутизатора является физическое соединение сетей. Маршрутизатор имеет несколько сетевых интерфейсов, подобных интерфейсам компьютера, к каждому из которых может быть подключена одна сеть. Таким образом, все интерфейсы маршрутизатора можно считать узлами разных сетей.

Определение маршрута является важной задачей сетевого уровня. Маршрут описывается последовательностью сетей (или маршрутизаторов), через которые должен пройти пакет, чтобы попасть к адресату. Маршрутизатор собирает информацию о топологии связей между сетями и на основе этой информации строит таблицы адресации, которые в данном случае носят специальное название таблиц маршрутизации.

Для того чтобы передать пакет через очередную сеть, сетевой уровень помещает его в поле данных кадра соответствующей канальной технологии, указывая в заголовке кадра канальный адрес интерфейса следующего маршрутизатора. Сеть, используя свою канальную технологию, доставляет кадр с инкапсулированным в него пакетом по заданному адресу. Маршрутизатор извлекает пакет из прибывшего кадра и после необходимой обработки передает пакет для дальнейшей транспортировки в следующую сеть. Таким образом, сетевой уровень играет роль координатора, организующего совместную работу сетей [2].

Определение маршрута является сложной задачей, особенно когда конфигурация сети такова, что между парой взаимодействующих сетевых интерфейсов существует множество путей. Чаще всего выбор останавливают на одном оптимальном по некоторому критерию маршруте. Критерий оптимальности также называют метрикой. Так, для измерения длины маршрута могут быть использованы разные метрики – количество транзитных узлов, линейная протяженность маршрута и даже его стоимость в денежном выражении. Для построения метрики, учитывающей пропускную способность, часто используют следующий прием: длину каждого участка характеризуют величиной, обратной его пропускной способности. Поскольку топология и состав информационных потоков могут меняться (отказы узлов или появление новых промежуточных узлов, изменение адресов или определение новых потоков), решение задач определения и задания маршрутов предполагает постоянный анализ состояния сети и обновление маршрутов и таблиц маршрутизации в соответствии с изменением метрик [2].

Сетевой уровень реализуется в соответствии с протоколом IP.

Изучив основные термины и понятия, необходимые для построения компьютерных сетей, можно переходить к изучению способов сетевого моделирования, выполнять обзор и анализ существующих программных продуктов, определять требования пользователей, производить постановку задачи.

2 ВЫБОР СРЕДСТВ РЕАЛИЗАЦИИ

Моделирование можно рассматривать как замещение исследуемого объекта его математическим описанием или другим объектом, именуемым моделью и обеспечивающим близкое к оригиналу поведение в пределах погрешностей, установленных в процессе моделирования. Моделирование обычно выполняется с целью познания свойств оригинала путем исследования его модели, а не самого объекта. Моделирование оправдано в том случае, когда оно проще создания самого оригинала. В случае моделирования компьютерных сетей, моделирование полностью оправдано по причинам, описанным ранее.

Программы, моделирующие компьютерные сети, делят на эмуляторы и симуляторы.

Эмуляция – имитация функционирования одного устройства посредством другого устройства или устройств вычислительной машины, при которой имитирующее устройство воспринимает те же данные, выполняет ту же программу и достигает того же результата, что и имитируемое [8].

Симуляция – комплекс программных и аппаратных средств, направленный на создание имитационной модели системы, воспроизводящая необходимые для решения задачи свойства и принципы работы [9].

Главным отличием между симуляторами и эмуляторами заключается в том, что симуляторы оперируют полностью виртуальными программными моделями сетевых устройств и протоколов, в то время как эмуляторы нацелены на взаимодействие с реальным оборудованием и приложениями [7].

Эмуляторы компьютерных сетей разделяются на использующие для воссоздания технологию виртуализации и использующие инструменты ядра операционной системы. Первый тип эмуляторов использует виртуальные машины, имитирующие физическое оборудование, что обеспечивает взаимодействие с реальными сетевыми устройствами. Второй тип эмуляторов реализуется в виде расширений ядра операционной системы таким образом, что каждое виртуальное устройство запускается в отдельном процессе,

который использует встроенное в операционную систему сетевое программное обеспечение.

Симуляторы разделяются на работающие в реальном времени и на дискретно-событийные симуляторы. Симуляторы реального времени имитируют работу сетевых устройств в реальном времени, что позволяет имитировать реальные сетевые протоколы. Дискретно-событийные симуляторы используют математическую модель компьютерной сети. Это позволяет выполнять расчет потоков трафика и нагрузки на линии связи, однако при этом отсутствует возможность тонкой настройки индивидуальных сетевых устройств.

Подводя итоги, эмуляторы, в отличие от симуляторов, дают возможность для взаимодействия с реальными сетями, сетевыми протоколами и устройствами, но виртуализация дает большую нагрузку на систему, снижая ее производительность, а симуляторы потребляют значительно меньше ресурсов системы и ограничения по размеру сети устанавливается параметрами симулятора, а не вычислительной мощностью компьютера, а также позволяют моделировать тестовые сценарии перед внедрением их в реальную сеть.

2.1 Обзор программных продуктов моделирования сетей

2.1.1. Эмуляторы

1. ENSP [10]

ENSP (Enterprise Network Simulator Platform) – бесплатный графический эмулятор сети, разработанный компанией Huawei.

Эмулятор моделирует работу большого количества сетевых устройств, включая маршрутизаторы серии AR и коммутаторы x7 компании Huawei, имитирует функции конфигурирования сетевых устройств, использует реальные сетевые карты в процессе эмуляции [11]. ENSP обладает собственным CLI, используемом на оборудовании Huawei.

Клиентская часть эмулятора представляет собой удобный и интуитивно понятный графический интерфейс, позволяющий конфигурировать сети с

различными топологиями. Рабочее окно программы представлено на рисунке 8.

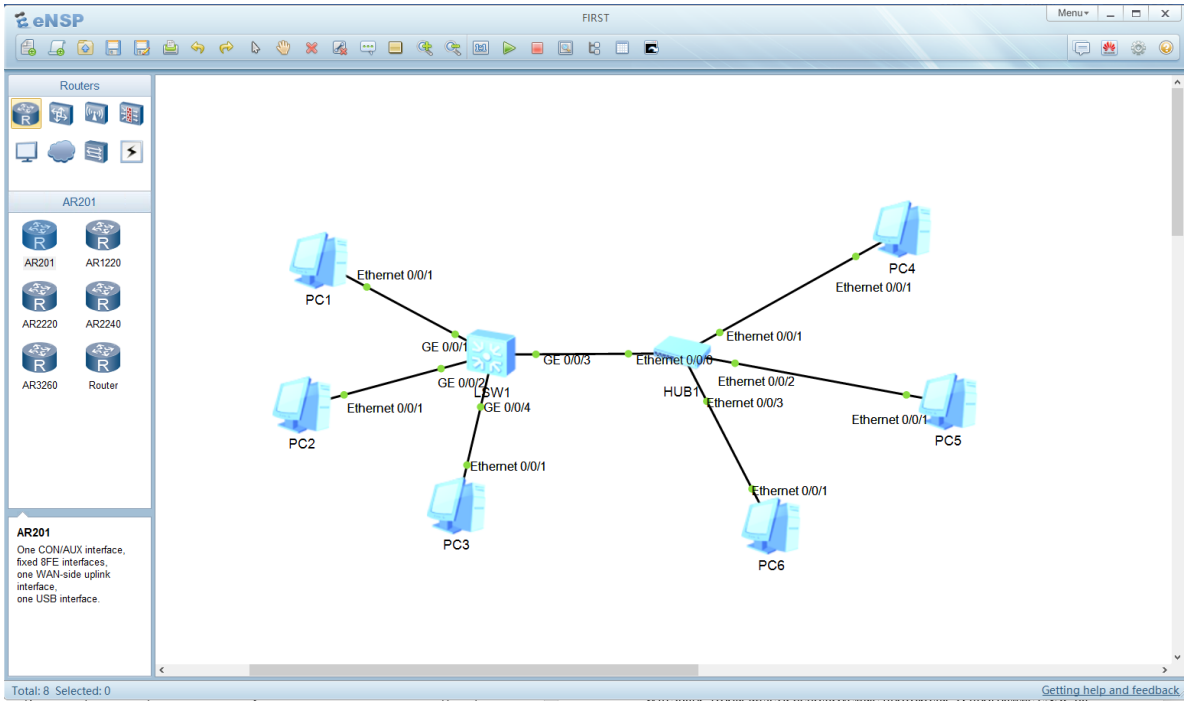

Рисунок 8 – Графический интерфейс ENSP

Серверная часть эмулятора использует VirtualBox [12] для виртуализации устройств, что позволяет работать с реальными устройствами и устройствами других эмуляторов. Возможность построения сетей с использованием оборудования разных организаций значительно расширяет сферу использования эмулятора, потому как редко встречаются сети, где всё оборудование одного производителя. Существует возможность добавления новых устройств при скачивании дополнительного модуля.

Распространяется ENSP бесплатно для некоммерческого использования, для скачивания программы достаточно зарегистрироваться на сайте Huawei. Также, ENSP имеет свою систему тестирования для обучающихся с возможностью получения сертификата после сдачи экзамена.

Достоинства:

– возможность взаимодействия с реальными сетевыми устройствами и их ПО;

– бесплатное использование;

– удобный графический интерфейс, позволяющий быстро освоить функционал программы;

– программа тестирования;

– возможность добавления сетевых устройств.

Недостатки:

– высокое потребление ресурсов при проектировании крупных сетей;

– требует регистрации на сайте.

2. GNS3 [13]

GNS3(Graphic Network Simulator) – бесплатный графический эмулятор сети с открытым исходным кодом.

GNS3 состоит из двух программных компонентов: GNS3-all-in-one software и GNS3 virtual machine.

GNS3-all-in-one является клиентской частью системы GNS3 и представляет собой графический интерфейс пользователя, который представлен на рисунке 9.

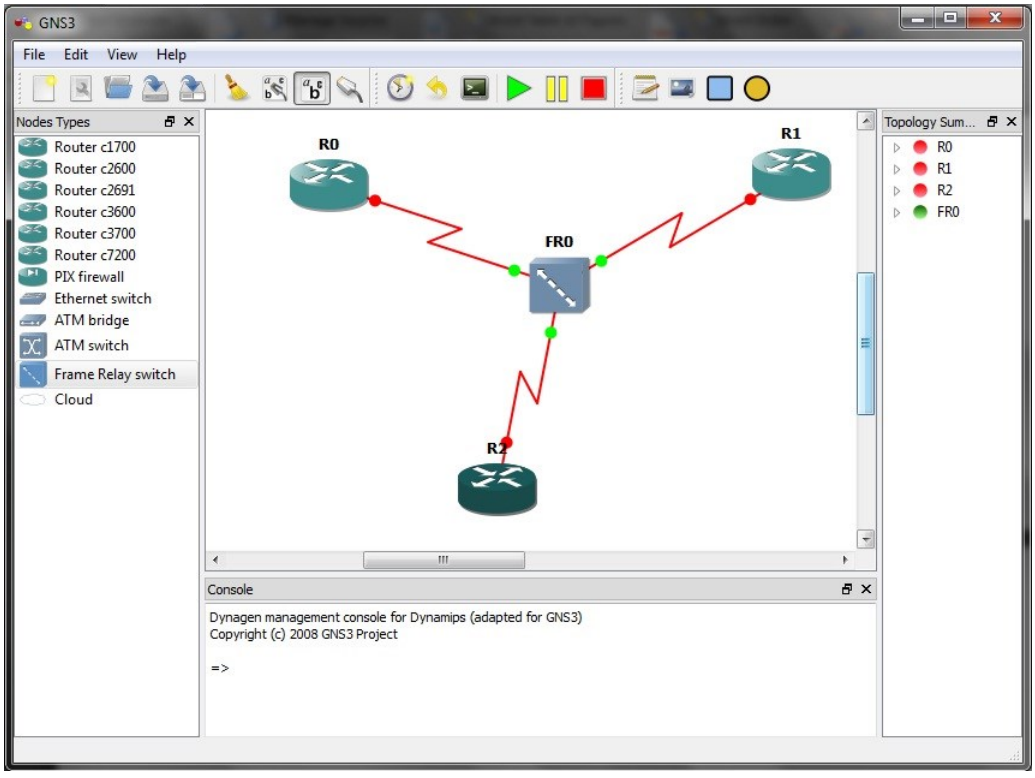

Рисунок 9 – Графический интерфейс GNS3

GNS3 VM представляет собой серверную часть системы GNS3. Она может быть развернута как на локальном компьютере с помощью средств виртуализации VMware Workstation или Virtualbox, так и удаленно на сервере с технологией VMware ESXi. Имеется возможность развертывания на облачных серверах.

GNS3 VM поддерживает как эмуляцию, так и симуляцию сетевого оборудования, что позволяет менять режим работы в зависимости от задач. Эмуляция поддерживает запуск реальных образов программного обеспечения сетевых устройств, но образ должен быть загружен пользователем.

Достоинства:

– лицензия Open-Source;

– наличие режима симуляции;

– возможность использовать программное обеспечение реальных сетевых устройств;

– поддержка широкого круга средств виртуализации.

Недостатки:

– необходимость установки программного обеспечения при добавлении устройств;

– невозможность эмулировать коммутаторы, использующие ASIC-микросхемы;

– высокое потребление ресурсов системы.

2.1.2. Симуляторы

1. Cisco Packet Tracer [14]

Cisco Packet Tracer – симулятор сети передачи данных компании Cisco Systems. Позволяет создавать работоспособные модели сети, настраивать маршрутизаторы и коммутаторы, имеет свой CLI Cisco IOS. В симуляторе реализованы серии маршрутизаторов Cisco 800, 1800, 1900, 2600, 2800, 2900 и коммутаторов Cisco Catalyst 2950, 2960, 3560, а также межсетевой экран ASA 5505. Вид симулятора представлен на рисунке 10.

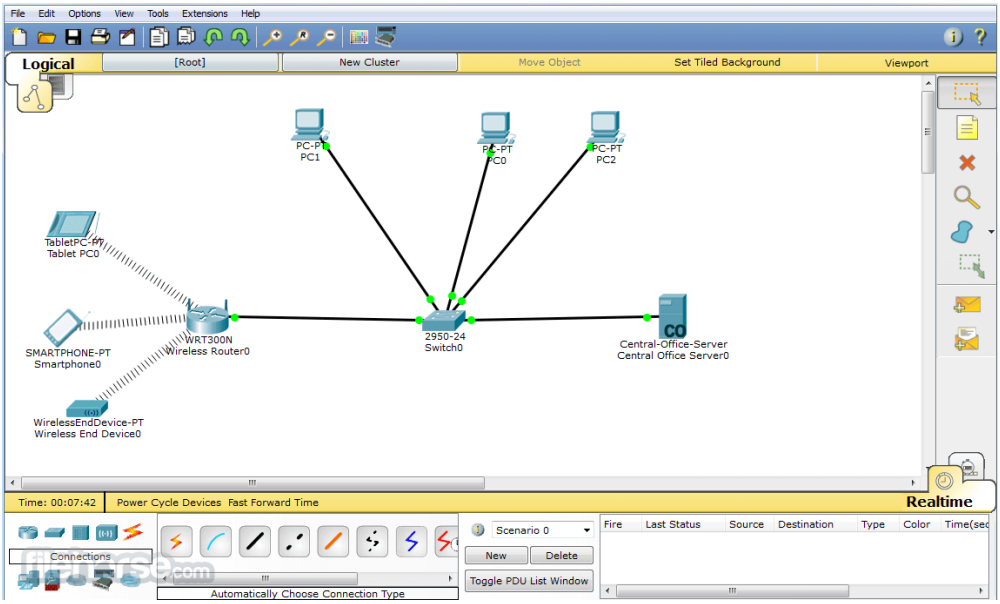

Рисунок 10 – Cisco Packet Tracer

Симулятор имеет логичный и понятный интерфейс, способствующий обучению, а также поддерживает два режима работы: дискретный и в режиме реального времени. Используя оба режима работы, можно отследить маршрут пакетов и более подробно рассмотреть работу сети.

Главный недостаток заключается в том, что Cisco Packet Tracer является частью комплексной среды обучения сетевой академии Cisco Academy и распространяется бесплатно только для ее участников.

Достоинства:

– поддержка дискретного режима и режима реального времени;

– наличие системы тестирования и сертифицирования;

– кроссплатформенность;

– интуитивно понятный графический интерфейс.

Недостатки:

– работа возможна только в Cisco IOS;

– распространяется бесплатно только для участников Cisco Academy.

2. NS-3 [15]

NS-3 – свободно распространяемый сетевой симулятор, работающий в дискретном режиме. NS-3 разработан в основном на языке C++ с использованием языка Python.

NS3 является очень гибким и мощным средством моделирования, используя в качестве встроенных языков моделирования C++, как и свой предшественник NS-2, и Python. Оба языка в симуляторе равноправны и принимаются для описания моделей.

Благодаря очень обширному и гибкому API, а также благодаря подробной документации программных интерфейсов, пользователь имеет возможность строить новые модели или изменять и дополнять существующие, входящие в ПО.

В NS3 разработаны модели беспроводных типов сетей, позволяющие проводить моделирование с движущимися объектами в трёхмерном пространстве, присутствует реализация различных типов Mesh-сетей на основе стека протоколов 802.11s, разработан фреймворк под названием FlowMonitor, предоставляющий очень гибкие методы сбора самых различных показаний с моделируемых активных сетевых устройств и каналов связи. Но симулятор не имеет собственного графического интерфейса, поэтому для визуализации моделей необходимо использовать стороннее ПО.

Достоинства:

– лицензия Open-Source;

– широкие возможности построения и изменения моделей;

– низкое потребление ресурсов.

Недостатки:

– отсутствие графического интерфейса.

3. Riverbed Modeler [16]

Riverbed Modeler (ранее, OPNET Modeler) – сетевой симулятор, работающий в дискретном режиме, разработан на языках C и C++, имеет несколько модульных библиотек для различных функций.

Приложение имеет удобный и широкий графический интерфейс, позволяющий помимо отображения топологии сети, визуализировать результаты моделирования с помощью таблиц, диаграмм и графиков.

Главное отличие от других симуляторов заключается в том, что Riverbed Modeler разделяет моделирование, симуляцию и анализ. Это значительно ускоряет процесс моделирования. Соответственно, модель тоже разделяется на три составляющие: модель сети, модель узлов и модель процессов, редактирование каждой из которых производится в отдельном рабочем окне.

Однако, для использования симулятора необходимо приобретать лицензию, что ограничивает область применения.

Достоинства:

– высокая скорость моделирования;

– широкие возможности для анализа результатов моделирования и отладки;

– возможность использования внешних моделей.

Недостатки:

– высокая стоимость лицензии;

4. Omnet++ [17]

OMNeT++ (Objective Modular Network Tested in C++) – свободно распространяемая, расширяемая, модульная C++ библиотека моделирования на основе компонентов, предназначенная для создания сетевых симуляторов.

Хотя OMNeT++ не является сетевым симулятором, этот продукт приобрел широкую популярность в качестве платформы сетевого моделирования в научном сообществе. OMNeT++ обеспечивает компонентную, иерархическую, модульную и расширяемую архитектуру.

В основе всей программной конструкции лежит высокоуровневый специализированный язык NED. Это язык описания топологии модели (или проще – язык описания соединения модулей). Процесс написания NED-программ происходит за удобным графическим интерфейсом, с помощью которого пользователь оперирует графическим представлением модулей (пиктограммами) и их иерархий, а NED-программы генерируются автоматически.

OMNeT++ может быть расширен с помощью специальных библиотек моделирования. Наиболее известной и распространенной является INET Framework Библиотека INET Framework – это комплект модулей с открытым исходным кодом, позволяющих моделировать узлы и протоколы проводных и беспроводных сетей, включает модели различных протоколов Интернета. В комплект также входят различные реалистичные примеры использования этих протоколов.

Для проектирования больших сетей, OMNeT++ поддерживает параллельное моделирование, ускоряя процесс симуляции. Следует понимать, что для ускорения используемая модель должна обладать параллелизмом, иначе параллельное моделирование только замедлит процесс.

Компоненты OMNeT++:

– библиотека ядра моделирования;

– OMNeT ++ IDE на базе платформы Eclipse;

– графический интерфейс выполняемого моделирования, ссылки на исполняемый файл (Tkenv);

– пользовательский интерфейс командной строки для выполнения моделирования (Cmdenv);

– документация, примеры.

Достоинства:

– лицензия Open-Source;

– графический интерфейс, упрощающий отслеживание сетевого трафика;

– расширяемость библиотеки;

– возможность параллельного моделирования.

Недостатки:

– использование требует знания языка C++;

– без расширений поддерживает малое количество протоколов.

2.2 Типы лицензий программного обеспечения

Основным документом, который определяет права и обязанности пользователя программного обеспечения, является лицензионное соглашение, которое прилагается к приобретенному продукту либо в виде бумажного документа, либо в электронном виде. Соглашение определяет правила использования программного обеспечения.

В основном программы делятся на две большие группы – свободного использования (бесплатная и открытая лицензия) и несвободного (коммерческая лицензия), а также между ними существуют условно-бесплатные программы, которые можно отнести к двум группам пополам, такие программы можно скачать и использовать, но пока ее не оплатить у вас могут возникнуть некоторые проблемы или ограничения. Типы лицензий представлены на рисунке 11.

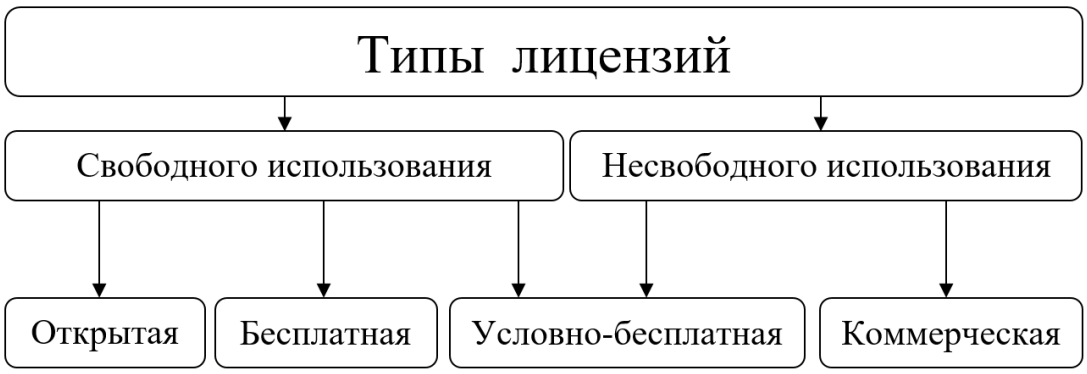

Рисунок 11 – Типы лицензий программного обеспечения [18]

Рассмотрим типы лицензий выбранных ранее программных продуктов.

К программным продуктам с коммерческим типом лицензий относятся Cisco Packet Tracer и Riverbed Modeler, не подходящие по требованиям пользователей.

Лицензии продуктов NS-3, Omnet++ и GNS-3 относятся к открытому типу, а ENSP является программным обеспечением с бесплатным типом лицензии.

Лицензии свободного использования предполагают отсутствие оплаты за использование, однако, по бесплатной лицензии пользователи не имею права распространять программы, дарить их, копировать, модифицировать,

давать право другим лицам на их распространение, а также выполнять ряд других действий, которые допускаются при использовании программ с открытой лицензией.

2.3 Постановка задачи

В рамках выпускной квалификационной работы требуется реализовать курс практических работ по предмету «Сети ЭВМ и телекоммуникаций», используя свободно распространяемую среду моделирования.

Основной задачей разработки лабораторных работ по предмету «Сети ЭВМ и телекоммуникаций» является закрепление теоретических и получение практических навыков обучающихся.

Главными требованиями к среде моделирования являются:

– открытая лицензия;

– наличие службы поддержки программного продукта;

– удобный и интуитивно понятный графический интерфейс;

– поддержка операционных систем Windows и Linux.

Рассмотрев выше представленные программные продукты, решено использовать платформу сетевого моделирования OMNeT++.

Симулятор OMNeT++ обладает необходимыми функциональными возможностями, позволяющими реализовать курс лабораторных работ, имеет дружелюбный и логичный интерфейс, что ускорит процесс обучения. На официальном сайте компании доступны все версии программы с документациями.

Симулятор позволяет создавать различные модели, используя либо графический интерфейс, либо используя высокоуровневый язык NED. Широкие возможности данной программы делают ее одним из ведущих свободно распространяемых симуляторов в мире.

Мощный графический интерфейс программы позволяет анализировать данные, собранные в процессе симуляции. Для удобства визуализации результатов есть возможность строить различные графики и диаграммы.

Для начинающих в программе есть раздел обучение с пошаговыми инструкциями, и есть примеры топологий различных сетей, принципы работы устройств и протоколов.

3 ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

Перед выполнением практических работ, необходимо корректно установить OMNeT++, используя выбранную операционную систему. OMNeT++ имеет дополнительные надстройки, которые рекомендуется устанавливать для обеспечения полностью работоспособной среды.

OMNeT++ недоступен на веб-сайте сообщества в качестве исполняемой программы, доступен только его исходный код. Это означает, что для запуска OMNeT++ необходимо компиляция кода.

Сначала необходимо перейти на страницу загрузки OMNeT++ по следующему URL-адресу: http://www.omnetpp.org/omnetpp. Есть два варианта загрузки:

1. OMNeT ++ x.x.x win32 (исходный код + IDE + MINGW, zip)
2. OMNeT ++ x.x.x (исходный код + IDE, tgz)

Первый вариант для пользователей Windows, второй для пользователей Linux.

OMNeT++ включает в себя три дополнительных пакета, которые были установлены:

— MPI: библиотека функций, предназначенная для поддержки работы параллельных процессов в терминах передачи сообщений.

– PCAP: библиотека функций, позволяющая создавать программы анализа сетевых данных, поступающих на сетевую карту компьютера. Примером программного обеспечения, использующего библиотеку Pcap, служит программа Wireshark.

– Akaroa: библиотека, отвечающая за сбор данных во время выполнения моделирования, самостоятельно останавливает симуляцию, как только достаточное количество данных для анализа было собрано. Полезный инструмент для отладки сети, для повышения ее эффективности и надежности.

После скачивания, для корректной установки и запуска следует изучить руководство Install Guide.pdf, доступное на веб-сайте сообщества в разделе документации.

Варианты лабораторных работ можно создавать, изменяя:

– топологию сети;

– параметры файла конфигурации сети;

– адресацию в сети;

– сценарий симуляции.

3.1 Практическая работа №1

Цель: ознакомление с системой моделирования, ее основными возможностями и настройками.

Краткая теоретическая выкладка:

Модель OMNeT++ состоит из модулей, которые взаимодействуют путем передачи сообщений. Простые модули написаны на C ++, используя библиотеку классов моделирования. Простые модули могут быть сгруппированы в составные модули, при чем количество уровней иерархии не ограничено. Вся модель, называемая сетью в OMNeT++, сама является составным модулем.

На рисунке 12 в блоках представлены и составные модули. Стрелки, соединяющие их, представляют собой соединения и интерфейсы.

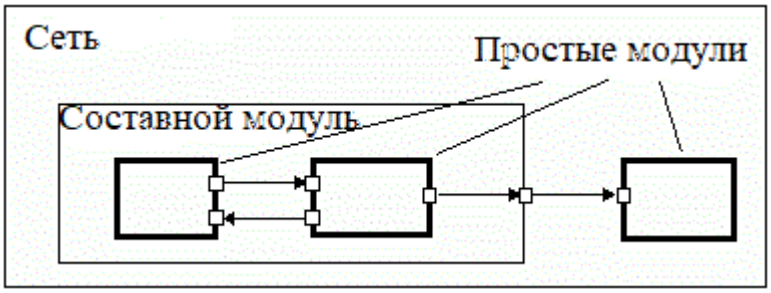

Рисунок 12 – Простые и составные модули [19]

Модули обмениваются сообщениями, которые могут содержать произвольные данные, в дополнение к обычным атрибутам, таким как временная метка. Соединения создаются в пределах одного уровня иерархии

модулей; в составном модуле могут быть соединены соответствующие интерфейсы двух подмодулей или интерфейс одного подмодуля и интерфейс составного модуля. Соединения, охватывающие несколько уровней иерархии, не разрешены, так как они будут препятствовать повторному использованию модели. Из-за иерархической структуры модели сообщения обычно проходят через цепочку соединений, поступая в простые модули. Для соединений могут быть назначены такие параметры, как задержка распространения, скорость передачи данных и скорость передачи битов. Можно также определить типы соединений со специфическими свойствами (называемыми каналами) и повторно использовать их в нескольких местах.

Модули могут иметь параметры. Параметры используются главным образом для передачи конфигурации простым модулям и для определения топологии модели. Параметры могут быть назначены либо в файлах NED, либо в файле конфигурации omnetpp.ini. Параметры могут принимать строковые, числовые, логические значения или даже быть XML-файлом. Поскольку параметры представлены как объекты в программе, они могут быть как постоянными значениями, так и случайными числами, подчиняющимися законам распределения.

Итак, модель OMNeT ++ состоит из следующих частей:

– описание топологии NED (.ned файлы), которые описывают структуру модуля. Файлы NED могут быть написаны с использованием любого текстового редактора, но среда OMNeT ++ обеспечивает отличную поддержку как графического и текстового редактирования.

– сообщения (файлы .msg), которые позволяют определять типы сообщений и добавлять к ним поля данных. OMNeT ++ преобразует определения сообщений в полноценные классы C ++.

– исходный код простых модулей. Это файлы C ++ с расширением .h или .cc.

Система моделирования содержит следующие компоненты:

– ядро симуляции, которое содержит код, управляющее имитацией и библиотекой классов моделирования. Написано на C ++, скомпилировано в общую или статическую библиотеку.

– пользовательские интерфейсы, используются для выполнения моделирования, для облегчения отладки, демонстрации выполнения имитаций. Написаны на C ++, скомпилированы в библиотеки.

Когда запускается моделирование, сначала считываются файлы NED, а затем файл конфигурации. Файл конфигурации содержит параметры, которые определяют, как выполняется имитация, значения параметров модели и т.д. Результат моделирования записывается в файлы: выходные векторные файлы, и скалярные файлы и, возможно, собственные выходные файлы пользователя. OMNeT++ содержит интегрированную среду разработки, которая обеспечивает широкие возможности для анализа этих файлов. Выходные файлы представляют собой текстовые файлы, которые позволяют обрабатывать их с помощью множества инструментов и языков программирования.

Задание:

– спроектировать простейшую сеть;

– обеспечить пересылку сообщения между устройствами сети.

Ход выполнения практической работы:

Шаг 1. Создается новый пустой проект OMNeT++ (File->New-> OMNeT++ Project). Затем создается новый NED файл, используя шаблон «NED file with one item» и ему назначается тип «Network». На рисунке 13 изображено диалоговое окно создания NED-файла. Открывшийся файл представляет собой файл топологии сети, изменять которую можно либо в графическом редакторе, добавляя модули и соединения, либо описывать топологию сети на языке NED, переключившись на режим «Source». При переключении режимов редактор автоматически синхронизирует топологию сети.

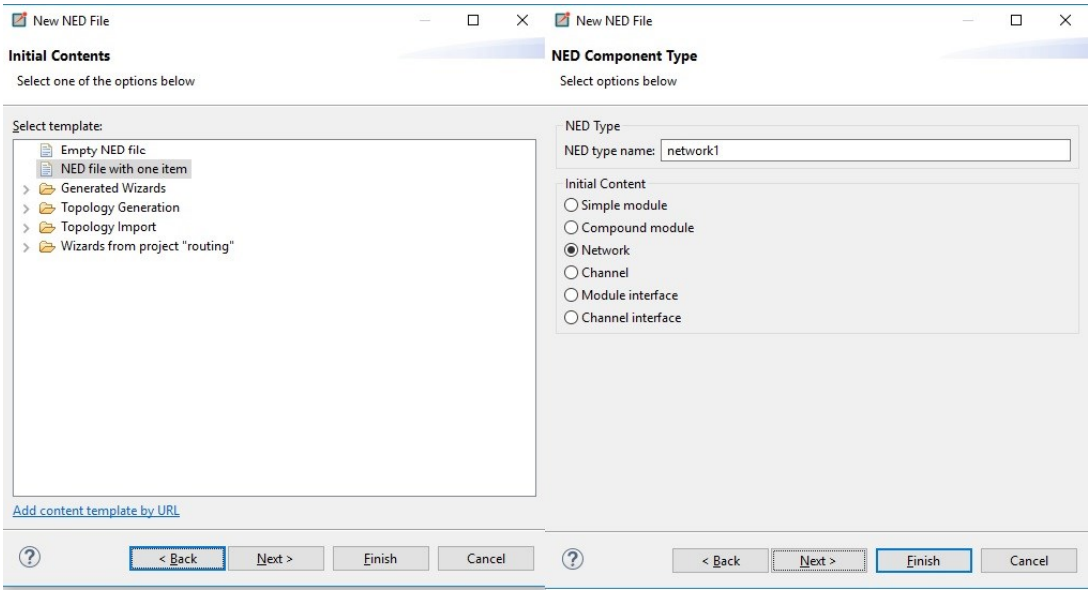

Рисунок 13 – Создание NED-файла сети

Шаг 2. Далее, создав простой модуль PC (File->New->Simple module), в текстовом редакторе необходимо описать его интерфейсы и добавить в сеть два таких модуля, переименовав их в PC1 и PC2. Топология сети представлена на рисунке 14

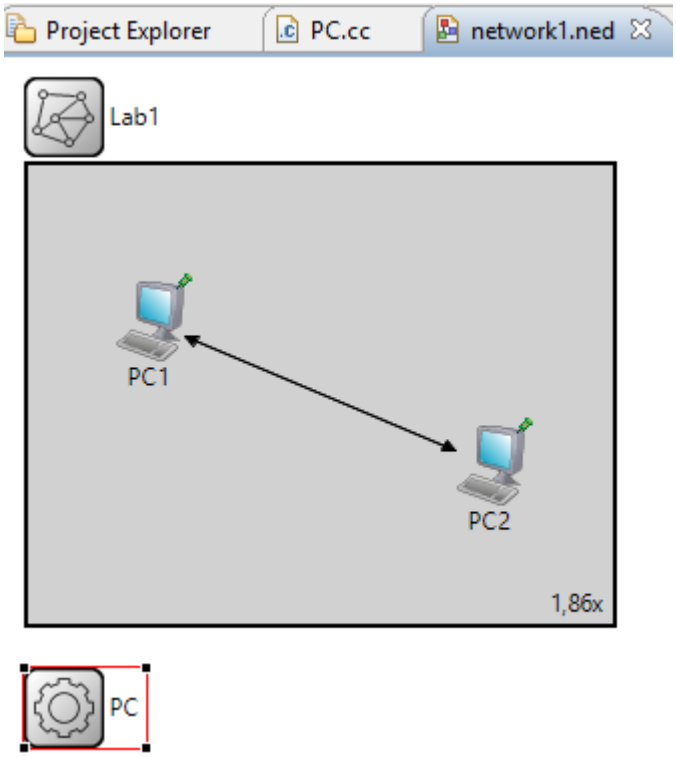

Рисунок 14 – Топология сети

Шаг 3. Теперь необходимо описать функциональность созданного простого модуля PC в C++. Создается файл исходного кода (File->New->Other-

>C/C++->Source code), где будет описываться класс PC, которым представлен простой модуль PC. Класс PC, исходный код которого представлен на рисунке 15, должен быть подклассом класса OMNeT ++ cSimpleModule и должен быть зарегистрирован в OMNeT ++ с помощью макроса Define_Module(). В классе переопределены два метода из cSimpleModule: initialize() и handleMessage(). Они вызываются из ядра моделирования: первый - только один раз, а второй - каждый раз, когда сообщение приходит в модуль. В initialize() создается объект сообщения (cMessage) и отправляется на выход из строя. Поскольку этот шлюз подключен к входному интерфейсу другого модуля, ядро моделирования доставляет это сообщение другому модулю в аргументе handleMessage() после задержки отправки на 100 мс, назначенной в файле NED. Другой модуль отправляет его обратно с такой же задержкой. Сообщения (пакеты, кадры, задания) и события (таймеры, тайм-ауты) представлены объектами cMessage (или его подклассами) в OMNeT ++. После планирования отправки или после самой отправки, они будут удерживаться ядром моделирования в списке «запланированные события» или «будущие события», пока не наступит их время, после чего они будут доставлены в модули через handleMessage ().

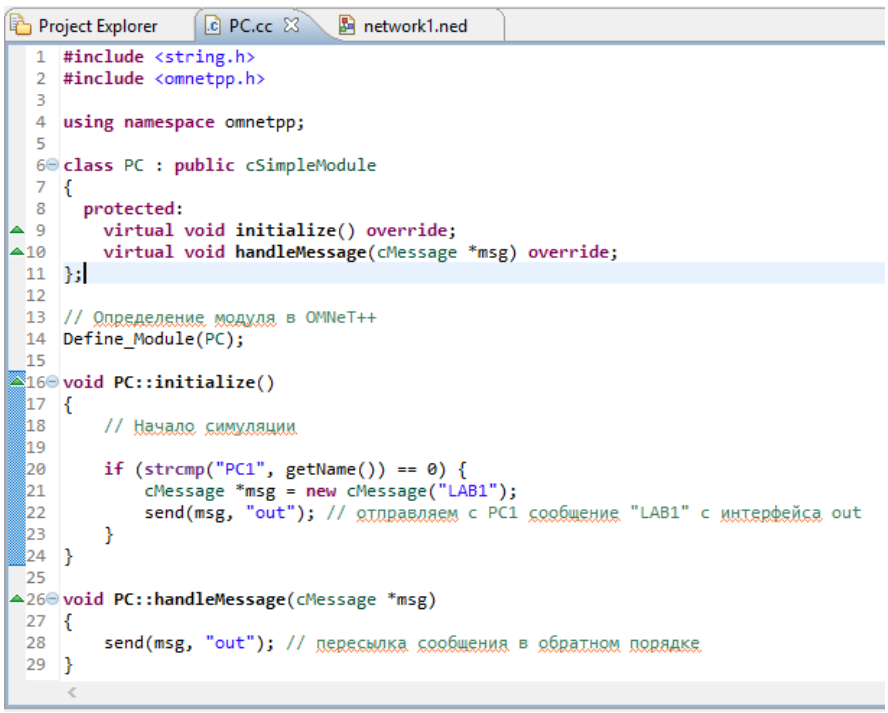

Рисунок 15 – Исходный код класса

Шаг 4. Для запуска симуляции, следует создать файл конфигурации (File -> New -> Initialization file (INI)). Новый файл откроется в двухрежимном редакторе. В режиме «Source» указываются параметры сети, как представлено на рисунке 16, в данной работе параметрами будут являться название и ограничение симуляции по времени.

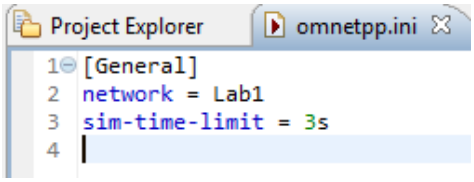

Рисунок 16 – Файл конфигурации

Шаг 5. Необходимые файлы созданы и настроены, можно запускать симуляцию, проект будет скомпилирован и запущен. В режиме симуляции для симуляции в реальном времени нужно нажать кнопку Run. По истечении времени симуляции, пользователь получит сообщение о завершении симуляции. На рисунке 17, в окне трафика сообщений видно, что сообщение передавалось корректно, с задержкой 100 мс после каждой отправки.

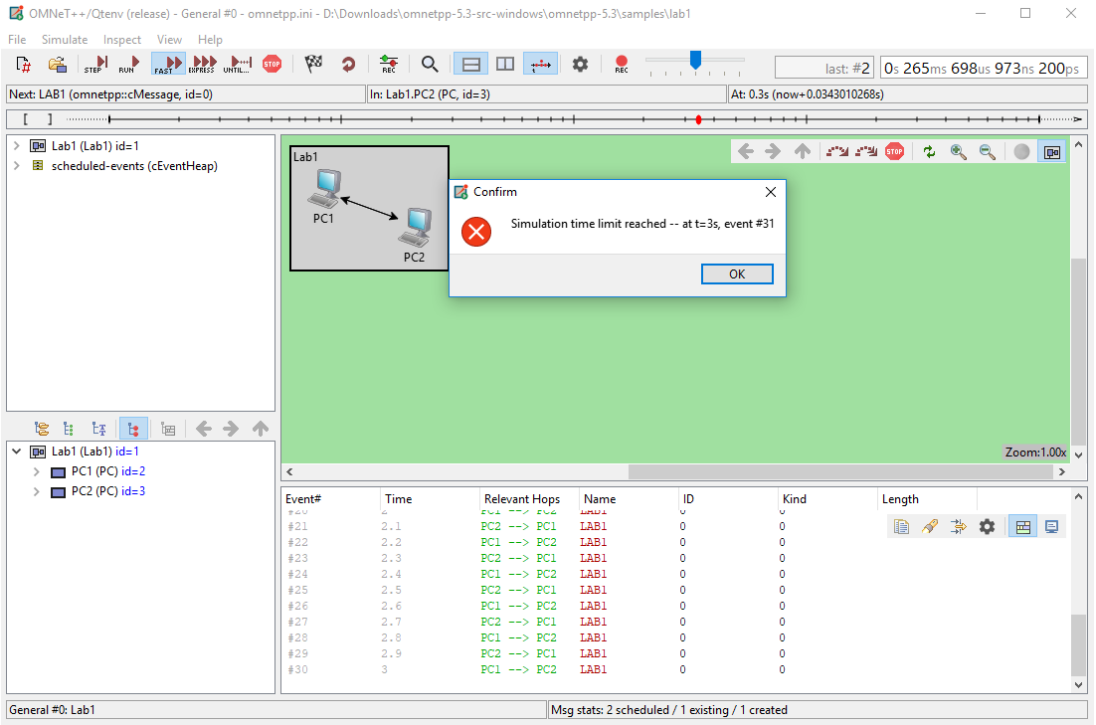

Рисунок 17 – Режим симуляции

3.2 Практическая работа №2

Цель: изучить принцип работы сети на канальном уровне, исследовать работу протокола STP.

Краткая теоретическая выкладка:

Устройства канального уровня выполняют физическую адресацию. Работа на этом уровне выполняется с кадрами (фреймами). На этом уровне устройство идентифицирует получателя и отправителя только по MAC-адресу и передает кадры между ними. Такие устройства как правило называют коммутаторами, иногда уточняя, что это «коммутатор уровня L2».

Сетевой коммутатор – это устройство, используемое в сетях передачи пакетов, предназначенное для объединения нескольких сегментов.

Коммутатор хранит в памяти таблицу коммутации, в которой указывается соответствие MAC-адреса узла порту коммутатора. При включении коммутатора эта таблица пуста, и он работает в режиме обучения. В этом режиме поступающие на какой-либо порт данные передаются на все остальные порты коммутатора. При этом коммутатор анализирует фреймы и, определив MAC-адрес хоста-отправителя, заносит его в таблицу на некоторое время. Впоследствии, если на один из портов коммутатора поступит кадр, предназначенный для хоста, MAC-адрес которого уже есть в таблице, то этот кадр будет передан только через порт, указанный в таблице. Если MAC-адрес хоста-получателя не ассоциирован с каким-либо портом коммутатора, то кадр будет отправлен на все порты, за исключением того порта, с которого он был получен. Со временем коммутатор строит таблицу для всех активных MAC-адресов, в результате трафик локализуется.

STP (Spanning Tree Protocol) — сетевой протокол, предназначенный для автоматического удаления петель коммутации из топологии на канальном уровне в Ethernet-сетях. Первоначальный протокол STP описан в стандарте IEEE 802.1D.

STP позволяет делать топологию избыточной на физическом уровне, но при этом логически блокировать петли. Достигается это с помощью того, что

STP отправляет сообщения BPDU (пакет включает в себя значения BridgeID, RootID и RootPathCost) и обнаруживает фактическую топологию сети. А затем, определяя роли коммутаторов и портов, часть портов блокирует так, чтобы в итоге получить топологию без петель.

Для того чтобы определить, какие порты заблокировать, а какие будут передавать данные, STP выполняет следующее:

1. Выбор корневого моста (Root Bridge).

Корневым становится коммутатор с наименьшим идентификатором моста (Bridge ID). Bridge ID состоит из приоритета моста и его MAC-адреса, размер 8 байт.

2. Определение корневых портов (Root Port).

Порт коммутатора, который имеет кратчайший путь к корневому коммутатору, называется корневым портом. У любого не корневого коммутатора может быть только один корневой порт. Корневой порт выбирается на основе меньшего Root Path Cost – это общее значение стоимости всех линков до корневого коммутатора. Если стоимость линков до корневого коммутатора совпадает, то выбор корневого порта происходит на основе меньшего Bridge ID коммутатора.

3. Определение выделенных портов (Designated Port)

Коммутатор в сегменте сети, имеющий наименьшее расстояние до корневого коммутатора, называется назначенным коммутатором (мостом). Порт этого коммутатора, который подключен к рассматриваемому сегменту сети называется назначенным портом, выбирается по принципу выбора корневого порта.

4. Блокировка портов

Далее, во всех сегментах, с которыми соединено более одного порта моста, мосты блокируют все порты, не являющиеся корневыми и назначенными. В итоге получается древовидная структура с вершиной в виде корневого коммутатора.

Задание:

– построить сеть, используя хосты и коммутаторы;

– протестировать работу протокола STP.

Ход выполнения практической работы:

Шаг 1. Необходимо создать папку проекта в папке inet (INET Framework) для возможности дальнейшего использования подмодулей этой библиотеки, и в этой папке создать NED файл с типом «Network».

Шаг 2. Используя подмодули библиотеки INET, нужно создать сеть, как показано на рисунке 18. Коммутаторы – модули EtherSwitch, компьютеры – модули StandartHost. Модуль L2NetworkConfigurator вычисляет сетевую конфигурацию канального уровня узлов сети. Результат вычисления сохраняется только в сетевом конфигураторе.

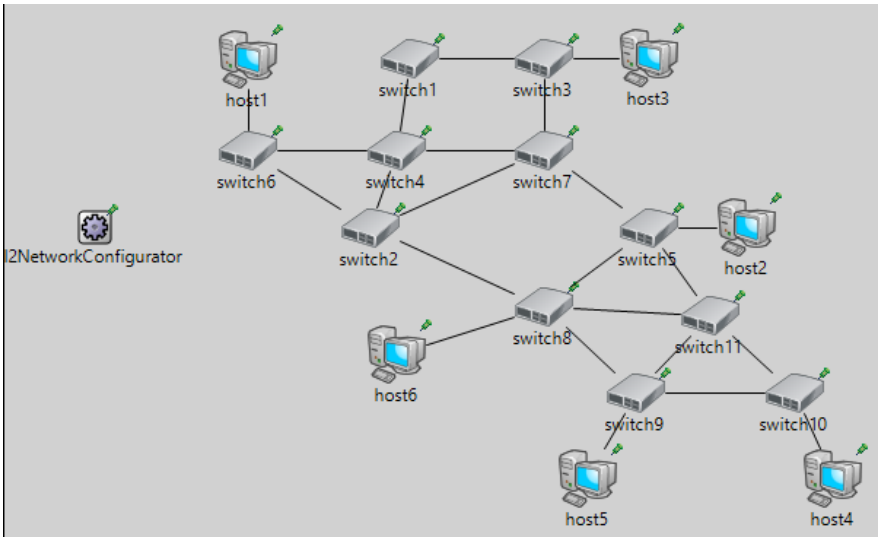

Рисунок 18 – Топология сети

Шаг 3. В файле конфигурации важно описать генератор трафика для проверки доступности узлов, ограничение по времени, присвоить коммутаторам MAC-адреса вида AAAAAA00000x, где x – номер коммутатора в шестнадцатеричной системе исчисления.

Шаг 4. Запуская режим симуляции, необходимо включить запись log-файла. После окончания симуляции следует убедиться в работоспособности сети, используя log-файл для анализа. Выбрав один из сгенерированных трафиков из файла конфигурации, например, трафик между Host2 и Host1, в первую очередь, проверяется доставка до адреса назначения, затем

оценивается рациональность маршрута. Для удобства отображения, рекомендуется отфильтровать результаты по имени сообщения и по именам сетевых устройств, расположив сетевые устройства в порядке следования пакета, пример представлен на рисунке 19. Ось абсцисс показывает время симуляции, на оси ординат отмечены выбранные устройства, стрелки обозначают отправку пакета одним устройством и прием вторым устройством. Надпись рядом со стрелкой обозначает сформированное автоматически имя пакета.

После проверки, в случае полной работоспособности сети, можно начинать тестирование протокола STP.

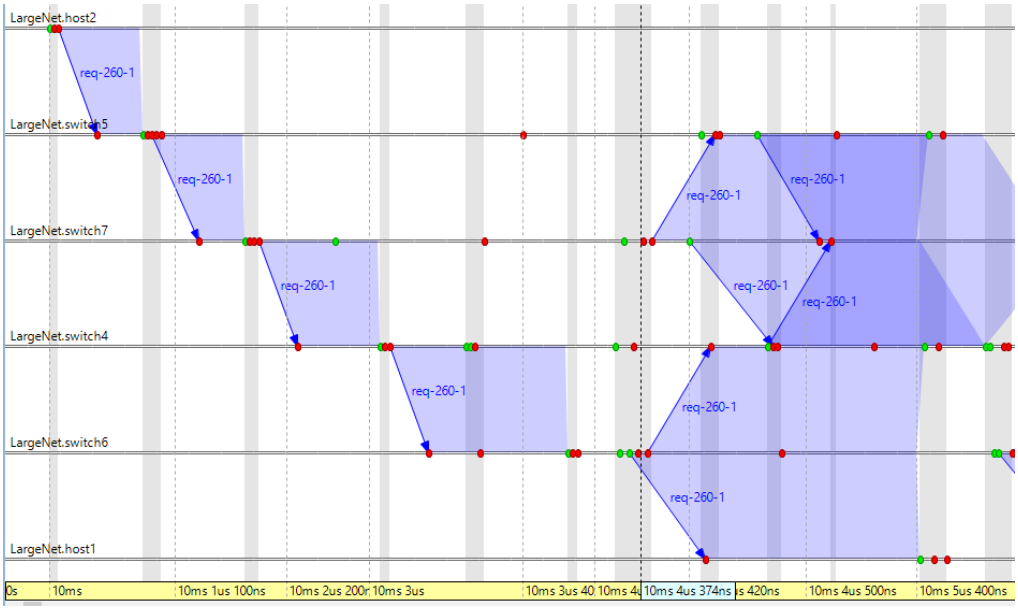

Рисунок 19 – Log-файл симуляции

Шаг 5. Для тестирования протокола STP нужно добавить в NED-файл созданной сети модуль STPTester, который извлекает сетевую топологию (сетевые узлы, помеченные свойством NED), только для разрешенных линков. Затем модуль анализирует полученный граф на связность и наличие петель, используя модифицированный поиск в глубину. Также необходимо добавить в файл конфигурации значение параметра используемого протокола **.spanningTreeProtocol = "STP".

Шаг 6. Подготовка к тестированию завершена, следует переходить в режим симуляции. В результате работы протокола все избыточные соединения были устранены, получена топология исходной сети без петель, изображенная на рисунке 20. Значения параметров пакета BDPU можно посмотреть, сначала переключившись с журнала событий на журнал сообщений/пакетов, а затем выбрав необходимый пакет, параметры которого отобразятся в рабочей области в нижнем левом углу экрана.

Таким образом, имеется возможность пошаговой проверки работы протокола.

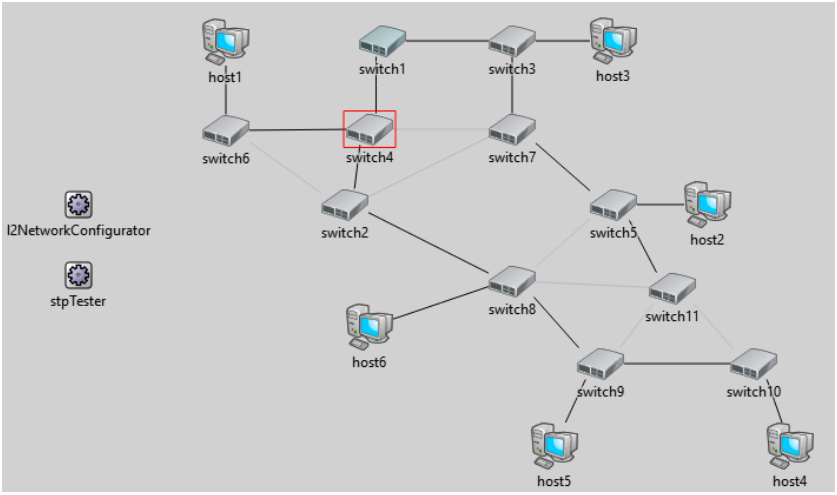

Рисунок 20 – Результат работы протокола STP

Значения параметров пакета BDPU устройства Switch 1 показаны на рисунке 21.

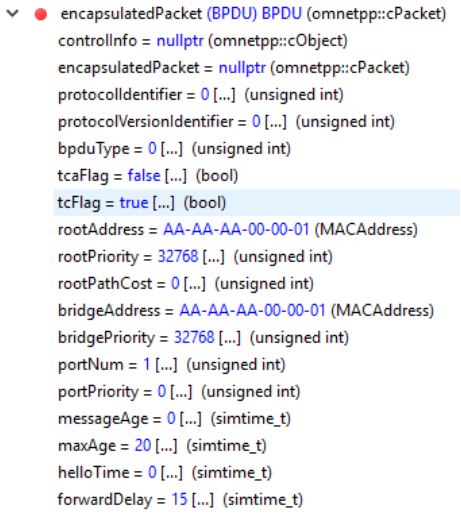

Рисунок 21 – Значения параметров BDPU для устройства Switch1

3.3 Практическая работа №3

Цель: изучить возможности статической маршрутизации в OMNeT++.

Краткая теоретическая выкладка:

Маршрутизация – это процесс определения пути следования информации в сетях связи. Маршрутизация служит для приема пакета от одного устройства и передаче его другому устройству через другие сети. Маршрутизатором или шлюзом называется узел сети с несколькими интерфейсами, каждый из которых имеет свой MAC-адрес и IP адрес.

Другим важным понятием является таблица маршрутизации. Таблица маршрутизации – это база данных, хранящаяся на маршрутизаторе, которая описывает соответствие между адресами назначения и интерфейсами, через которые следует отправить пакет данных до следующего узла. Таблица маршрутизации содержит: адрес узла назначения, маску сети назначения, адрес шлюза (обозначающий адрес маршрутизатора в сети на который необходимо отправить пакет, следующий до указанного адреса назначения), интерфейс (физический порт через который передается пакет), метрика (числовой показатель, задающий приоритет маршрута).

Размещение записей в таблице маршрутизации может производиться тремя различными способами. Первый способ предполагает применение прямого соединения, при котором маршрутизатор сам определяет подключенную подсеть. Прямой маршрут — это маршрут, который является локальным по отношению к маршрутизатору. Если один из интерфейсов маршрутизатора соединен с какой-либо сетью напрямую, то при получении пакета, адресованного такой подсети, маршрутизатор сразу отправляет пакет на интерфейс, к которому она подключена. Прямое соединение является наиболее достоверным способом маршрутизации.

Второй способ предполагает занесение маршрутов вручную. В данном случае имеет место статическая маршрутизация. Статический маршрут определяет IP-адрес следующего соседнего маршрутизатора или локальный выходной интерфейс, который используется для направления трафика к

определенной подсети-получателю. Статические маршруты должны быть заданы на обеих концах канала связи между маршрутизаторами, иначе удаленный маршрутизатор не будет знать маршрута, по которому нужно отправлять ответные пакеты и будет организована лишь односторонняя связь.

И третий способ подразумевает автоматическое размещение записей с помощью протоколов маршрутизации. Данным способ называется динамической маршрутизацией. Протоколы динамической маршрутизации могут автоматически отслеживать изменения в топологии сети. Успешное функционирование динамической маршрутизации зависит от выполнения маршрутизатором двух основных функций:

1. поддержка своих таблиц маршрутизации в актуальном состоянии;
2. своевременное распространение информации об известных им сетях и маршрутах среди остальных маршрутизаторов.

Задание:

– построить сеть, состоящую из нескольких подсетей;

– назначить IP-адреса и маски узлам;

– настроить маршрутизаторы, используя статическую маршрутизацию;

– визуализировать результаты, используя встроенный инструмент OMNeT++.

Ход выполнения работы:

Шаг 1. Необходимо создать папку проекта в папке inet (INET Framework) для возможности дальнейшего использования подмодулей этой библиотеки, и в этой папке создать NED файл с типом «Network».

Шаг 2. Используя подмодули библиотеки INET, нужно создать сеть, состоящую из 3 локальных сетей, соединенных между собой маршрутизаторами, как представлено на рисунке 22. Маршрутизаторы – модули Router, коммутаторы – модули EtherSwitch, компьютеры – модули StandartHost. Модуль IPv4NetworkConfigurator позволяет назначать IP-адреса узлам сети и настраивать таблицы маршрутизации, модуль Visualizer помогает визуализировать сети.

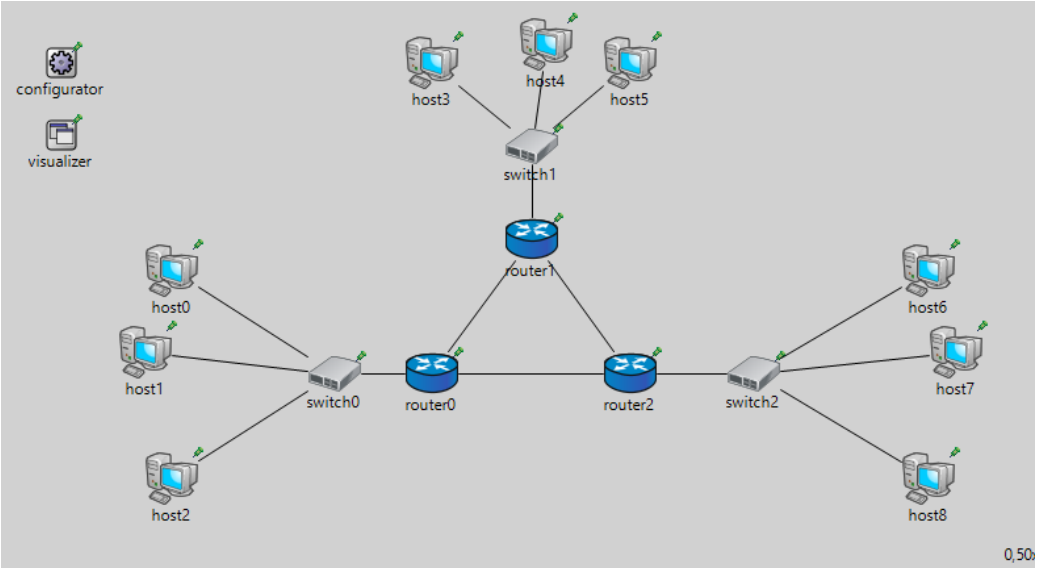

Рисунок 22 – Топология сети

Шаг 3. Используя модуль IPv4NetworkConfigurator будем назначать IP-адреса и маски узлам сети. Модуль зачастую может правильно настроить сеть, используя только настройки по умолчанию. Конфигуратор имеет несколько параметров, которые относятся к назначению IP-адреса:

– assignAddresses = default (true) определяет, должен ли конфигуратор автоматически назначать IP-адреса интерфейсам. Подробности могут быть указаны в конфигурации XML.

– assignDisjunctSubnetAddresses = default (true) определяет, должен ли конфигуратор назначать разные префиксы адресов и сетевые маски для узлов на разных каналах связи. (Узлы считаются одним каналом связи, если они соединены напрямую или только через устройства канального уровня).

Когда пользователь не указывает конфигурацию XML (например, на этом этапе), конфигуратор назначает IP-адреса для всех интерфейсов хостов из диапазона адресов 10.0.0.0 - 10.255.255.255 и диапазона сетевой маски 255.0.0.0 - 255.255.255.255.

Шаг 4. В файле конфигурации необходимо описать визуализацию IP-адресов и масок интерфейсов через параметры модуля Visualizer. Для

назначения и отображения IP-адресов и масок узлов, нужно запустить режим симуляции.

Результатом симуляции является сеть с распределенными для всех интерфейсов IP-адресами и масками, изображенная на рисунке 23. Сеть разбита на шесть подсетей, используя метод бесклассовой адресации. Использование различных масок в различных подсетях позволяет экономно использовать ограниченный ресурс IP-адресов. Адрес до косой черты у каждого интерфейса обозначает его IP-адрес в своей подсети. Число после косой черты называют длиной префикса, оно обозначает количество единичных разрядов маски подсети.

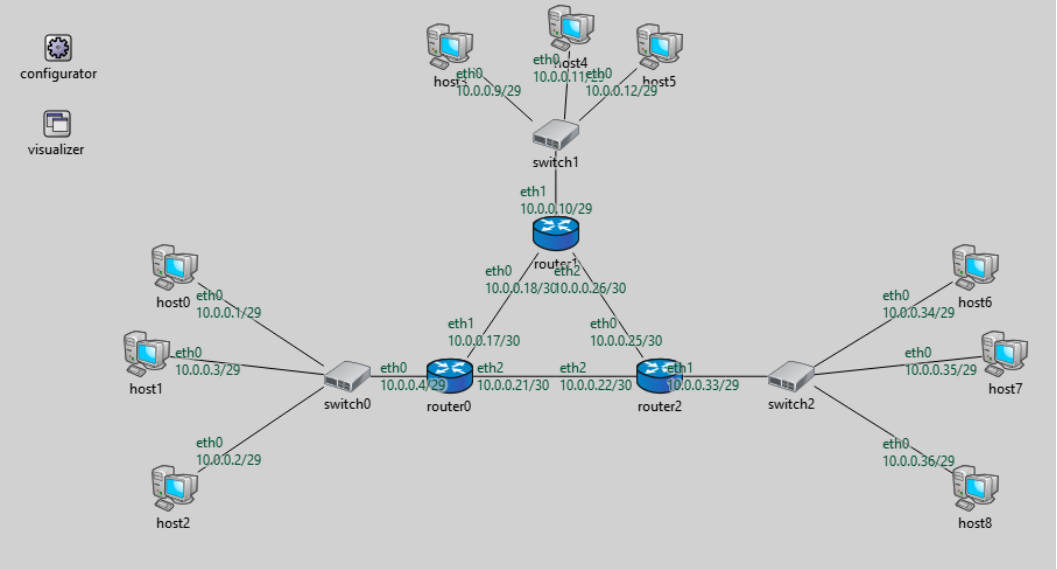

Рисунок 23 – Назначенные конфигуратором IP-адреса и маски

Шаг 5. Приступим к заполнению таблиц маршрутизации. Как и в случае с IP-адресами, во многих случаях конфигуратор правильно настраивает маршруты в сети без какого-либо ввода пользователем. Этот шаг демонстрирует конфигурацию по умолчанию для статической маршрутизации.

Параметры конфигуратора, касающиеся статической маршрутизации, следующие:

– addStaticRoutes = default(true). Данный параметр включает автоматическое заполнение таблиц маршрутизации. Когда параметр отключен, добавляются только маршруты, указанные в конфигурации XML, и игнорируются все следующие параметры.

– addDefaultRoutes = default(true). Добавляет маршрут по умолчанию, если все маршруты от узла проходят через один и тот же интерфейс.

– addSubnetRoutes = default(true). Оптимизирует таблицы маршрутизации, добавляя маршруты в направлении к подсетям вместо отдельных интерфейсов.

– optimizeRoutes = default(true). Оптимизирует таблицы маршрутизации путем объединения записей, где это возможно.

Для проверки доступности узлов сети, можно использовать утилиту ping. В качестве примера, отправляется ping-запрос с узла Host1 к узлу Host7. Для отображения пути, нужно описать в файле конфигурации визуализацию маршрута до host7. Визуализация активируется установкой параметра displayRoutingTables параметра RoutingTableVisualizer в значение true. Набор маршрутов для визуализации выбирается с параметрами destinationFilter и nodeFilter визуализатора. Все маршруты, ведущие к назначенному узлу, обозначены стрелками, первый адрес над стрелкой указывает адрес сети назначения, второй указывает интерфейс, на который должен быть передан пакет. Стрелки не проходят через коммутаторы, коммутаторы являются устройствами канального уровня. Можно запускать режим симуляции. На рисунке 24 показан результат симуляции, на котором, используя подключенные средства визуализации, зеленым цветом подписаны назначенные конфигуратором IP-адреса и маски узлов сети, а стрелками, сконфигурированные маршруты, ведущие к узлу назначения Host7. Также, при запуске симуляции, наблюдается анимированный маршрут ping-запроса, отмеченный красным цветом, с узла Host1 к узлу Host7.

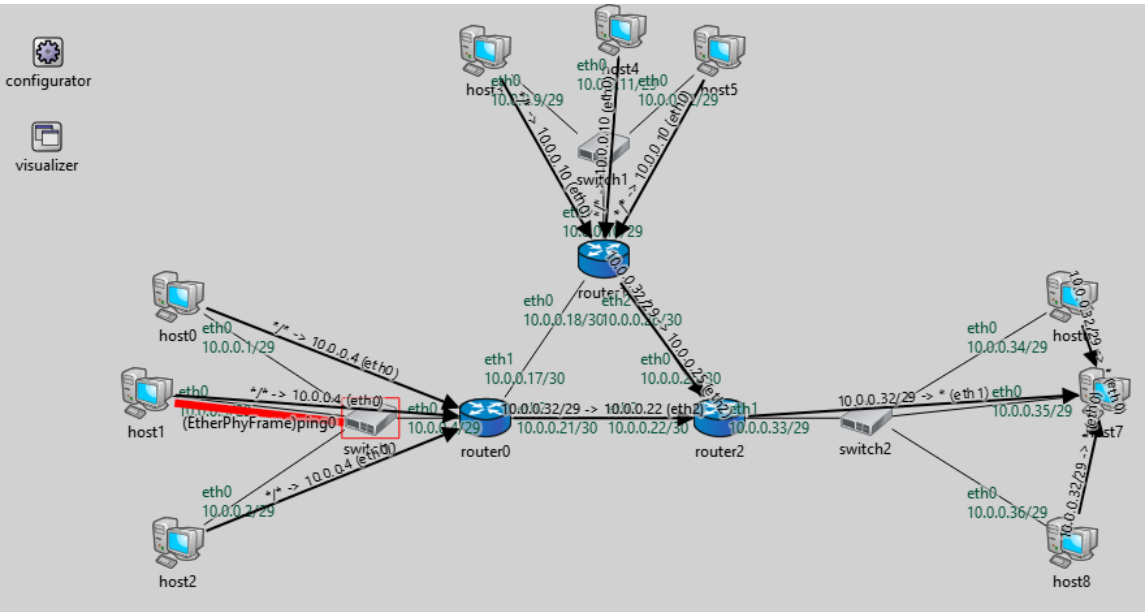

Рисунок 24 – Сеть с маршрутами до Host7

3.4 Практическая работа №4

Цель: исследовать работу протокола OSPF.

Краткая теоретическая выкладка:

OSPF – протокол динамической маршрутизации, основанный на технологии отслеживания состояния канала (link-state technology) и использующий для нахождения кратчайшего пути Алгоритм Дейкстры.

Использование данного протокола дает следующие преимущества:

– отказоустойчивость. В случае выхода из строя любого из маршрутизаторов, обмен информацией мгновенно переключается на другой маршрут, что предотвращает простои в работе.

– экономия. Связь между узлами надежно резервируется, а изменение структуры не требует больших трудозатрат. Таким образом не нужно содержать много персонала, который будет обслуживать систему.

– снижение рисков. Использование данной технологии значительно снижает риски простоя, а также риск зависимости функционирования системы от обслуживающего персонала.

Описание работы протокола:

1. Маршрутизаторы обмениваются hello-пакетами через все интерфейсы, на которых активирован OSPF. Маршрутизаторы,

разделяющие общий канал передачи данных, становятся соседями, когда они приходят к договоренности об определённых параметрах, указанных в их hello-пакетах.

2. На следующем этапе работы протокола маршрутизаторы будут пытаться перейти в состояние смежности со своими соседями. Переход в состояние смежности определяется типом маршрутизаторов, обменивающихся hello-пакетами, и типом сети, по которой передаются hello-пакеты. Пара маршрутизаторов, находящихся в состоянии смежности, синхронизирует между собой базу данных состояния каналов.

3. Каждый маршрутизатор посылает объявления о состоянии канала маршрутизаторам, с которыми он находится в состоянии смежности.

4. Каждый маршрутизатор, получивший объявление от смежного маршрутизатора, записывает передаваемую в нём информацию в базу данных состояния каналов маршрутизатора и рассылает копию объявления всем другим смежным с ним маршрутизаторам.

5. Рассылая объявления внутри одной OSPF-зоны, все маршрутизаторы строят идентичную базу данных состояния каналов маршрутизатора.

6. Когда база данных построена, каждый маршрутизатор использует алгоритм Дейкстры для вычисления графа без потерь, который будет описывать кратчайший путь к каждому известному пункту назначения с собой в качестве корня. Этот граф — дерево кратчайших путей.

7. Каждый маршрутизатор строит таблицу маршрутизации из своего дерева кратчайших путей.

В случае, если у одного из маршрутизаторов пропадает связь с соседом:

1. Маршрутизатор рассылает новые объявления о состоянии канала.

2. Получившие объявления маршрутизаторы строят новую карту сети.

3. Пересчитывают кратчайшие маршруты во все сети.

4. Обновляют свою таблицу маршрутизации.

Задание:

– построить сеть, используя 4 маршрутизатора и 2 хоста;

– создать сценарий симуляции, изменяющий кратчайший маршрут между двумя хостами;

– проанализировать трафик пакетов, отследить алгоритм работы протокола OSPF.

Ход выполнения практической работы:

Шаг 1. Необходимо создать папку проекта в папке inet (INET Framework) для возможности дальнейшего использования подмодулей этой библиотеки и в этой папке создать NED файл с типом «Network».

Шаг 2. Используя подмодули библиотеки INET нужно создать сеть, как представлено на рисунке 25. Маршрутизаторы – модули OSPFRouter, компьютеры – модули StandartHost, модуль IPv4NetworkConfigurator. Модуль LifecycleController управляет операциями включения, выключения, перезагрузки, сбоя и восстановления работы узлов сети. Модуль ScenarioManager дает возможность изменять параметры симуляции по определенному сценарию.

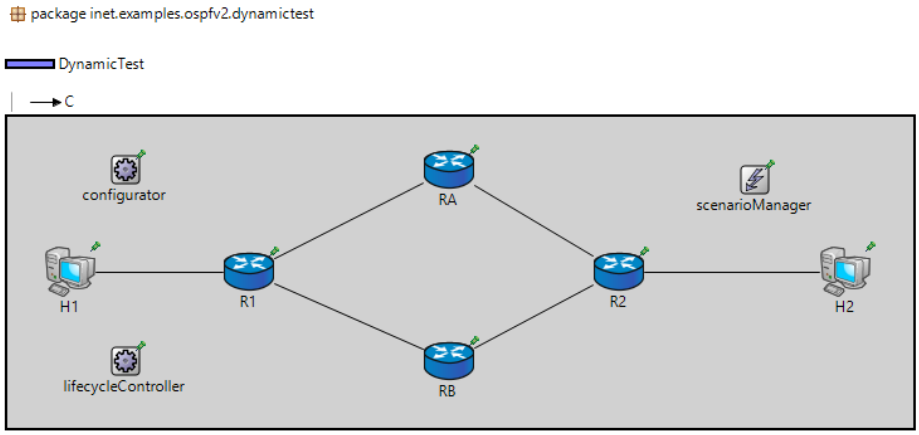

Рисунок 25 – Топология сети

Шаг 3. Необходимо назначить IP адреса интерфейсов и шлюзы по умолчанию узлов сети. Рассмотрим способ назначения IP-адресов вручную, не используя конфигурацию модуля по умолчанию. Можно создать отдельный XML-файл с конфигурацией и добавить его в проект или добавить

конфигурацию через параметры модуля Configurator, как показано на рисунке 26.

Рисунок 26 – Конфигурация сети

Шаг 4. После настройки интерфейсов необходимо сконфигурировать работу протокола OSPF в сети. Для этого требуется создать XML-файл, в котором необходимо включить протокол на маршрутизаторах, определить зоны работы протокола и стоимость соединения(метрику). Маршрут через RA установлен маршрутом с меньшей метрикой.

Шаг 5. Для изменения параметров сети во время симуляции, надо создать XML-файл сценария, исходный код которого представлен на рисунке 27, в котором, например, в момент времени t=200c происходит обрыв соединения между RA и R1, в момент времени t=400c восстановление соединения.

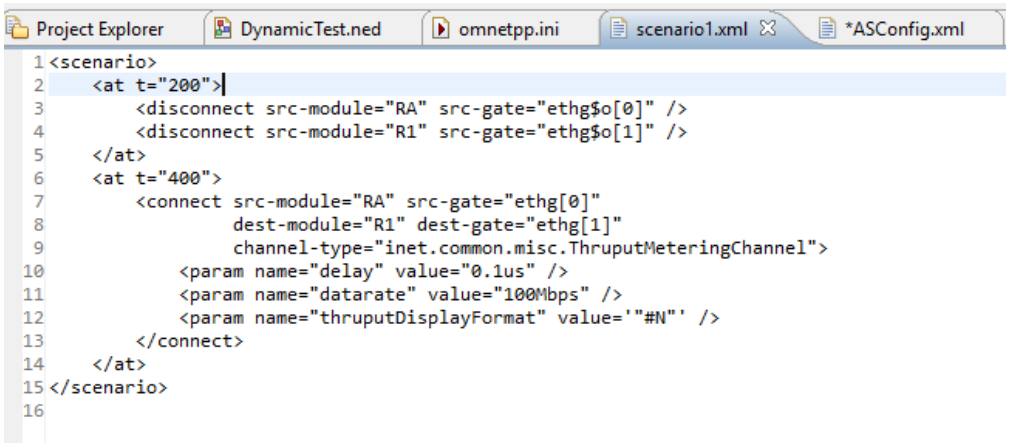

Рисунок 27 – Файл сценария

Шаг 6. Необходимо создать файл конфигурации, в нём должно быть описано: ограничение времени симуляции до 600с, организация сетевого трафика между H1 и H2, подключение файла сценария и файла конфигурации протокола OSPF.

Шаг 7. Можем переходить в режим симуляции, перед запуском симуляции нужно включить запись log-файла. Запустив симуляцию, заметно, что после установки соединения и до момента времени t=200с, практически весь трафик идет кратчайшим маршрутом, то есть через роутер RA, что видно на рисунке 28.

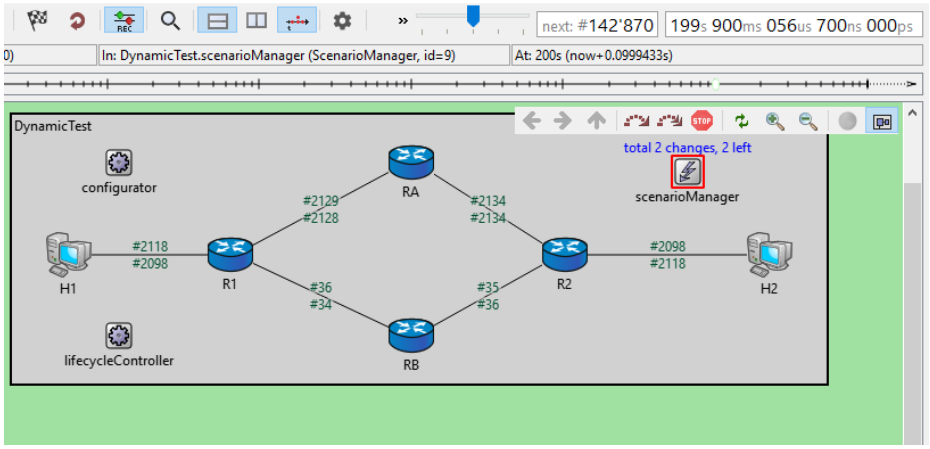

Рисунок 28 – Режим симуляции сети до обрыва соединения

В момент времени t=200с происходит обрыв соединения между R1 и RA, после чего маршрутизаторы начинают объявлять о потере соединения, происходит обновление таблиц маршрутизации, пакеты начинают идти через роутер RB. На рисунке 29 представлен момент времени перед восстановлением связи между RA и R1.

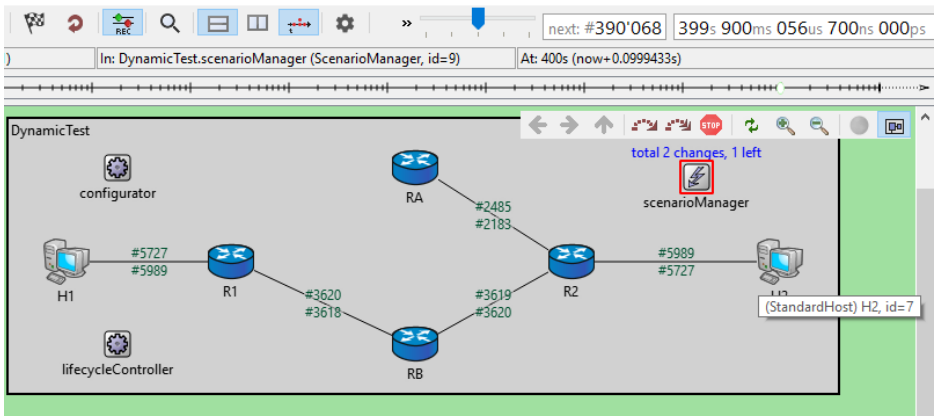

Рисунок 29 – Режим симуляции сети перед восстановлением

После восстановления связи и переопределения таблиц маршрутизации, пакеты снова идут по маршруту с наименьшей метрикой до окончания симуляции, что видно на рисунке 30.

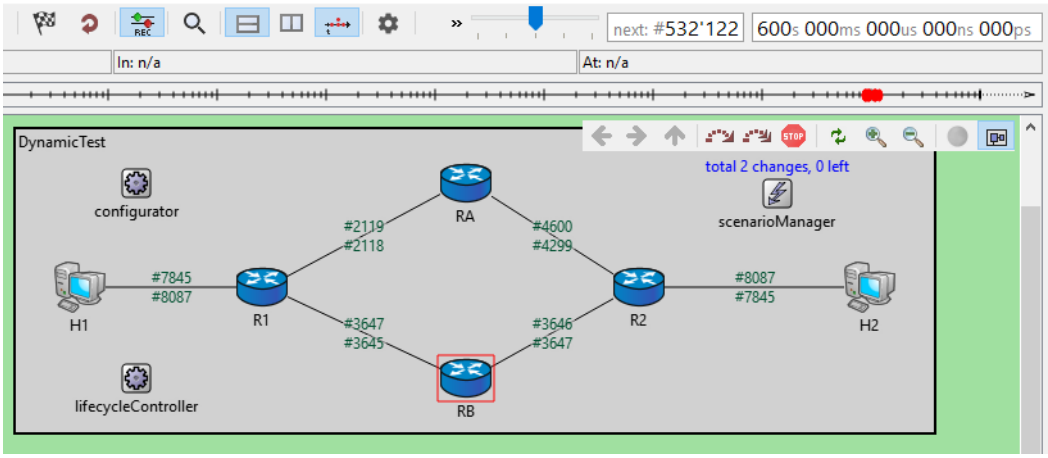

Рисунок 30 – Конец симуляции

Шаг 7. После окончания симуляции, необходимо провести анализ собранных данных. Открыв log-файл, для удобства представления рекомендуется отфильтровать интересующую нас информацию по имени устройства. После применения фильтра, необходимо отследить передачу пакетов протокола OSPF. На рисунке 31 представлена часть log-файла симуляции.

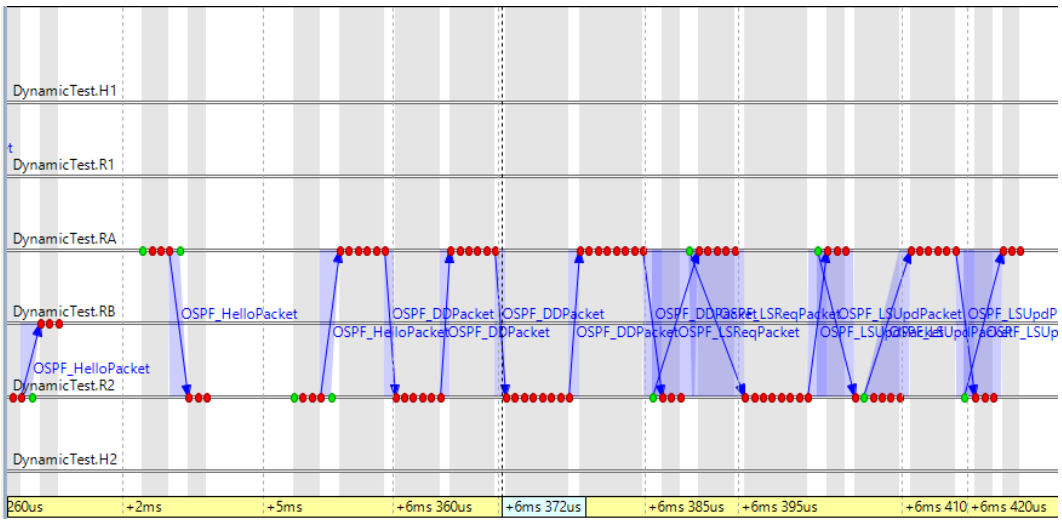

Рисунок 31 – Log-файл симуляции

OSPF_HelloPacket – hello-пакет;

OSPF_DDPacket – пакет, описывающий содержание базы данных состояний каналов;

OSPF_LSReqPacket – пакет, с помощью которого запрашивается полная информация о состоянии каналов, которых недостает в базе данных состояния каналов маршрутизатора;

OSPF_LSUpdPacket – пакет, который передают полную информацию, которая содержится в объявлении о состоянии канала;

OSPF_LSAckPacket – пакет, с помощью которого подтверждается получение других пакетов.

ЗАКЛЮЧЕНИЕ

В результате выпускной квалификационной работы был реализован курс практических работ по предмету «Сети ЭВМ и телекоммуникаций» на свободном программном обеспечении.

В ходе работы был проведен подробный анализ предметной области, изучены необходимые теоретические аспекты моделирования компьютерных сетей.

Следующим этапом проводился обзор средств сетевого моделирования, определялись требования к программному продукту, исходя из которых был выбран наиболее подходящий сетевой симулятор.

Выбрав свободно распространяемый сетевой симулятор, был реализован курс лабораторных работ, с помощью которого пользователь может научиться работать в среде моделирования OMNeT++, строить сети на языке NED, изучить внутреннее устройство различных сетевых устройств, алгоритмы работы этих устройств и рассмотреть процессы соединения благодаря модульной структуре и точности среды моделирования. Используя широкие графические возможности OMNeT++, может изучить алгоритмы работы некоторых протоколов канального и сетевого уровня.

Все поставленные задачи были выполнены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 24402-88. Телеобработка данных и вычислительные сети. Термины и определения, – М.: Стандартинформ,1989. – 12с.

2. Олифер, В.Г., Олифер, Н.А. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. – 5-е изд. – СПб.: Питер, 2016. – 996с.

3. Образовательный портал «Обучение в интернет». – URL: http://www.lessons-tva.info/edu/telecom-loc/m1t4_3loc.html (дата обращения 30.03.2018).

4. Борисова Л.Ф. Повышение эффективности функционирования локальных вычислительных сетей // Вестник МГТУ – 2000. – №1. – С. 45-54.

5. Курс «Вычислительные системы, сети и телекоммуникации». – URL: https://v1.savant.pro/community/3/~h/item/zwuvRXHc (дата обращения 04.04.2018).

6. Онлайн-курс научного открытого университета «Построение сетей на базе коммутаторов и маршрутизаторов». – URL: https://www.intuit.ru/studies/courses/636/492/lecture/11116?page=2 (дата обращения 04.04.2018).

7. Таненбаум Э., Уэзеролл Д. Компьютерные сети. 5-е изд. – СПб.: Питер, 2012. – 960 с.

8. ГОСТ 15971-90. Системы обработки информации. Термины и определения, – М.: Издательство стандартов, 1991. – 11с.

9. Бондаревский А.С., Лебедев А.В. Имитационное моделирование: Определение, применяемость и техническая реализация// Фундаментальные исследования. – 2011. – № 12-3. – С. 535-541.

10. Официальный сайт компании Huawei. Документы и ПО компании Huawei. – URL: http://support.huawei.com/enterpriseproduct/docTypeNewOffering?pid=9017384&lang=ru (дата обращения 26.04.2018).

11. Официальный представитель компании Huawei в России. Инструмент управления ENSP. – URL: http://huawei-russia.ru/stati/upravlenie-setju/instrumenty-upravlenija/ensp/ (дата обращения 26.04.2018).

12. Официальный сайт программы виртуализации Virtualbox. – URL: https://www.virtualbox.org/ (дата обращения 26.04.2018).

13. Официальный сайт программы моделирования GNS3. – URL: https://www.gns3.com/ (дата обращения 29.04.2018).

14. Официальный сайт компании Cisco. Cisco Packet Tracer. – URL: https://www.cisco.com/c/ru_ua/training-events/netacad/training-courses/cisco-packet-tracer.html (дата обращения 29.04.2018).

15. Официальный сайт сетевого симулятора NS-3. – URL: https://www.nsnam.org/ (дата обращения 29.04.2018).

16. Официальный сайт компании Riverbed. Riverbed Modeler – URL: https://www.riverbed.com/gb/products/steelcentral/steelcentral-riverbed-modeler.html (дата обращения 05.05.2018).

17. Официальный сайт модульной библиотеки OMNeT++. – URL: https://www.omnetpp.org/ (дата обращения 05.05.2018).

18. Компьютерный портал «Линчакин». Основные виды лицензии программ. – URL: https://linchakin.com/posts/post/29/ (дата обращения 15.05.2018).

19. OMNeT++ Simulation Manual. – URL: https://www.omnetpp.org/doc/omnetpp/manual/ (дата обращения 15.05.2018).