Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В.Непомнящий
подпись инициалы, фамилия
« ___» _____ 2018 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Моделирование работы сети произвольной топологии, работающей по протоколу RSTP, и оценка ее устойчивости средствами OmNet++
тема

Руководитель	_____	доцент, к.т.н.	О.А. Русанова
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		О.Ю. Анищенко
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____	доцент, к.т.н.	В.И. Иванов
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Как известно, грамотное построение отказоустойчивой сети подразумевает использование резервирования каналов. Наиболее подходящей для этого является кольцевая топология. Однако, в случае построения сети по технологии Ethernet в чистом виде, множественные связи между узлами не предусмотрены и могут привести к полной неработоспособности всей конструкции. Например, в случае попадания в кольцо широковещательного пакета, он будет передаваться активным оборудованием по кругу бесконечно, обеспечивая предельную загрузку. Для предотвращения подобных ситуаций был создан специальный протокол Rapid spanning tree protocol(STP).

Основной задачей RSTP является приведение сети Ethernet с множественными связями к древовидной топологии, исключающей циклы пакетов. Происходит это путем автоматического блокирования ненужных в данный момент для полной связности портов.

Моделирование данной сети будет происходить в среде OMNeT++ – модель позволяет оценить поведение будущей системы в различных случаях и ее статистические характеристики.

Система OMNeT++ является совокупностью программных библиотек, в которых хранятся функции для работы с программами моделирования, позволяющие выполнить реализацию собственных модулей, компиляцию и сборку этих программ.

Включает собственный высокоуровневый язык описания моделей и сетей: NED (язык описания топологии модели). NED - программа пишется с помощью графического интерфейса (с представлением модулей в виде пиктограмм и их иерархий), и NED-программы генерируются автоматически. Файлы формата NED не определяют поведение работы сети. Их поведение определяется дополнительным файлом исходного кода на C++ (файлы расширения .CC или .CPP). После создания файла формата NED и соответствующего управляющего файла C++ сеть можно моделировать.

OMNeT++ является бесплатным только для некоммерческого пользования со средой разработки, основанной на пользовательском интерфейсе Eclipse, с разновидностью интерфейса Graphical user interface (GUI), в котором элементы исполнены в виде графических изображений; модульное ядро дискретно - событийного симулятора содержит несколько сетевых симуляторов.

1 Анализ предметной области

1.1 Обзор протокола RSTP

Протокол RSTP (Rapid Spanning Tree Protocol) - сетевой протокол второго уровня, данная технология изобретена Радьей Перельманом. Данный протокол используется при соединении между собой двух и более коммутаторов дублирующими соединениями или при построении сети по топологии «mesh» (связь всех со всеми).

Протокол RSTP позволяет анализировать сеть и устранять петли (forwardingloops) между коммутаторами, создавая граф передачи пакетов между коммутаторами. При нормальном функционировании сети Ethernet, между двумя коммутаторами может быть активно лишь одно соединение, другие не будут использоваться. Так как несколько активных соединений между коммутаторами, создают петлю, RSTP блокирует одно или несколько таких соединений. Для обеспечения возможности резервирования соединений, протокол RSTP определяет граф (дерево) из сетевых коммутаторов.

Протокол RSTP, вычислив дублирующийся пути для данных, блокирует их, переводя в режим ожидания. В случае, если сегмент сети становится недоступным, или меняется вес пути графа RSTP (например, при изменении скорости соединения), алгоритм RSTP перестраивает дерево и если требуется, активирует резервные соединения. Работа протокола RSTP незаметна для оконечных узлов в сети, и неважно, подключены ли они к одному сегменту.

1.1.1 Алгоритм работы протокола RSTP

Использует алгоритм STA (SpanningTreeAlgorithm), результатом работы которого является граф в виде дерева. Для обмена информацией между собой коммутаторы используют специальные пакеты - BPDU (Bridge Protocol Data Units). BPDU бывают двух видов: конфигурационные (ConfigurationBPDU) и TCN (TopologyChangeNotificationBPDU) которые извещают все устройства в сети о том, что топология поменялась. Первые регулярно рассылаются корневым коммутатором (и ретранслируются остальными) и используются для построения топологии, вторые отсылаются в случае изменения топологии сети (при подключении\отключении коммутатора). Конфигурационные BPDU содержат несколько полей: идентификатор отправителя (Bridge ID), идентификатор корневого коммутатора (Root Bridge ID), идентификатор порта, из которого отправлен данный пакет (Port ID), стоимость маршрута до корневого коммутатора (Root PathCost). Так как устройства не знают своих соседей, никаких отношений (смежности/соседства) они друг с другом не устанавливают.

Они шлют BPDU из всех работающих портов на мульти кастовый Ethernet-адрес 01-80-c2-00-00-00 (по умолчанию каждые 2 секунды), который прослушивают все коммутаторы с включенным RSTP. Формирование топологии без петель. Для устранения петель в сети выбирается корневой мост/коммутатор(root bridge). Это устройство, которое RSTP считается точкой отсчета, центром сети; все дерево RSTP сходится к нему. Выбор базируется на таком понятии, как идентификатор коммутатора (Bridge ID). Bridge ID это число длиной 8 байт, которое состоит из Bridge Priority (приоритет, от 0 до 65535, по умолчанию 32768+номер Vlan или Instants MSTP, в зависимости от реализации протокола), и MAC-адреса устройства. В начале выборов каждый коммутатор считает себя корневым, о чем и заявляет всем остальным с помощью BPDU, в котором представляет свой идентификатор как ID корневого коммутатора. При этом, если он получает BPDU с меньшим Bridge

ID, он начинает анонсировать полученный Bridge ID в качестве корневого. В итоге, корневым оказывается тот коммутатор, чей Bridge ID меньше всех. Статусы портов в протоколе Rapid Spanning Tree protocol. Когда топология сети еще не определена и сеть находится в неопределенном статусе, порт коммутатора, передавая данные, создает временные петли пакетов. Порты коммутаторов должны подождать некоторое время, пока не получат информацию о топологии сети, чтобы начать передавать пакеты по сети LAN. Также, они должны подождать, пока не истечет время жизни пакетов старой топологии. Каждый порт коммутатора, который использует протокол RSTP, может находиться в одном из пяти состояний: Порт находится в заблокированном состоянии. В данном состоянии, порт не передает пакеты, но получает и отвечает на служебные сообщения. Порт сканирует сеть. В данном состоянии, порт не имеет MAC адреса и не передает других MAC адресов, порт принимает и передает пакеты BPDU, а также получает и отвечает на служебные сообщения. Порт изучает топологию сети. В данном состоянии, порт не передает никаких пакетов, кроме пакетов BPDU. Порт находится в режиме заполнения таблицы MAC адресов, и готовится к началу передачи данных, а также он получает и отвечает на служебные сообщения.- Порт находится в режиме передачи. В данном состоянии, порт принимает и передает все пакеты сети.- (Выключен - обычно переведен в такое состоянии вручную). Отключенный порт не принимает и получает никаких типов пакетов в сети. После перезагрузки коммутатора или изменения топологии сети, порт, использующий RSTP, четырежды меняет свое состояния. Коммутаторы с резервными соединениями, и правильно настроенные коммутаторы, также могут иметь порты в статусах blocking или Forwarding.

1.1.2 Недостатки протокола RSTP

Такой подход таит в себе довольно серьезную проблему. При равных значениях Priority (а они равные, если не менять ничего) корневым

выбирается самый старый коммутатор, так как мак адреса прописываются на производстве последовательно, соответственно, чем MAC-адрес меньше, тем устройство старше (если у нас все оборудование одного производителя). Это может привести к падению производительности сети, так как старое устройство, как правило, имеет худшие характеристики. Подобное поведение протокола можно изменить, выставляя значение приоритета на желаемом корневом коммутаторе вручную.

1.2 Моделирование в среде OMNET++

В настоящее время существует широкий спектр различных программных средств моделирования и исследования сетей и систем связи: OPNET Modeler; OMNeT++; Ns-2; Ns-3; COMNET III; BONes Designer; QualNet. В перечисленных системах имеются возможности моделирования сетей, но в разной степени осуществлена их поддержка.

Для реализации предложенного способа, моделирования и проверки всех утверждений и предположений используется фреймворк OMNeT++ версии 5.2. [3]

OMNeT++ – расширяемая, модульная C++ библиотека моделирования на основе компонентов, предназначенная для создания сетевых симуляторов, разработчик – Андраш Варга (Andras Varga). Это продукт с открытым исходным кодом, является бесплатным только для академического и некоммерческого использования. Здесь сеть понимается в более широком смысле, включая в себя проводные и беспроводные сети связи, чипы, которые построены по архитектуре сеть-на-чипе (подразумевается, что каждое вычислительное ядро связано непосредственно только с ближайшими ядрами), сети массового обслуживания и т. д. Предметно-ориентированная функциональность (поддержка сенсорных сетей, беспроводных ad-hoc-сетей или беспроводных динамических (самоорганизующихся) сетей, интернет-протоколов, моделирования производительности, фотонных сетей и т. д.)

обеспечивается модельными фреймворками, разработанными в качестве самостоятельных проектов. OMNeT++ предоставляет IDE на основе Eclipse, графическую среду выполнения, а также множество других инструментов. Есть расширения для моделирования в реальном времени, эмуляции сети, интеграции с базами данных, с SystemC и ряд других функций.

Хотя OMNeT++ не является сетевым симулятором, этот продукт приобрел широкую популярность в качестве платформы сетевого моделирования в научном сообществе, а также в промышленных установках. OMNeT++ предоставляет компонентную архитектуру для моделей. Компоненты (модули) запрограммированы в C++, затем собраны в более крупные компоненты и модели с использованием языка высокого уровня (NED). Данный инструмент имеет расширенную поддержку графического интерфейса пользователя, и благодаря своей модульной архитектуре ядро моделирования (и модели) может быть легко встроено в пользовательские приложения.

OMNeT++ работает на Windows, Linux, Mac OS X и других UNIX-подобных системах и состоит из следующих компонентов:

- библиотека ядра моделирования;
- язык описания топологии NED (англ. NEtwork Description);
- IDE на базе платформы Eclipse;
- GUI для выполнения моделирования Tkenv, линкуется в исполняемый файл симуляции;
- интерфейс командной строки для выполнения моделирования (Cmdenv);
- утилиты (инструмент создания makefile и т. д.);
- документация, примеры моделей и др.

OMNeT++ предоставляет эффективные инструменты для пользователя, чтобы описать структуру реальной системы. Некоторые из главных особенностей:

- иерархически вложенные модули;
- модули являются экземплярами типов модулей;

- модули связываются сообщениями по каналам;

- гибкие параметры модуля;

- язык описания топологии.

Модель OMNeT++ состоит из следующих частей:

- описания топологий на языке NED (файлы с расширением .ned), которые описывают структуру модуля с параметрами, шлюзами и т. д. Такие файлы могут быть созданы с помощью любого текстового редактора, но IDE OMNeT++ обеспечивает поддержку двустороннего графического и текстового редактирования;

- определения сообщений (.msg файлы). Можно определить различные типы сообщений и добавить поля данных в них. OMNeT++ автоматически переведет определения сообщений в полноценные классы C++;

- исходные коды модуля представляют собой C++ файлы с расширением .h или .cc.

Система моделирования предоставляет следующие компоненты:

- ядро моделирования, которое управляет моделированием и классовыми библиотеками моделирования, написано на C++, скомпилировано в общие или статические библиотеки;

- пользовательские интерфейсы, используются при выполнении моделирования, чтобы облегчить отладку, демонстрацию или пакетное выполнение симуляций, написаны на C++, скомпилированы в библиотеки.

Программы моделирования построены из вышеуказанных компонентов. Сначала .msg файлы преобразуются в код на C++ с использованием компилятора сообщений `opp_msgc`. После все исходники на C++ компилируются и линкуются с ядром моделирования и библиотекой пользовательского интерфейса для формирования исполняемого файла моделирования или общей библиотеки. NED-файлы загружаются динамически в исходных текстовых формах при запуске программы моделирования. Программа моделирования может быть скомпилирована как отдельный исполняемый файл так, что она может быть запущена на других компьютерах, где OMNeT++ не установлена.

Или симуляция может быть создана как общая библиотека, тогда общие библиотеки OMNeT++ должны присутствовать в системе.

После запуска программа считывает все файлы NED, потом – конфигурационный файл (обычно называется `omnetpp.ini`). Этот файл содержит параметры, которые определяют, как выполняется моделирование, значения параметров модели и т. д. В файле конфигурации также можно задать несколько выполнений моделирования; в простейшем случае они будут выполнены программой моделирования один за другим.

Выходные данные моделирования записываются в файлы результатов: выходные файлы векторов (запись данных во время выполнения моделирования), выходные файлы скаляров (итоговые результаты, вычисленные во время моделирования и записанные после завершения моделирования), выходные файлы, определенные пользователем. OMNeT++ содержит интегрированную среду разработки, которая обеспечивает богатые возможности для анализа этих файлов. Выходные файлы – это линейно-ориентированные текстовые файлы, что дает возможность обрабатывать их с помощью различных инструментов и языков программирования (в т. ч. MATLAB, GNU R, Perl, Python и программы для работы с электронными таблицами).

OMNeT++ может быть расширен с помощью специальных библиотек моделирования и новых экзотических сценариев.

1.3 Критерии оценки

В соответствии с ГОСТР 53111-2008 [9] функционирование сетей электросвязи в условиях воздействия дестабилизирующих факторов физического или технологического характера (далее - дестабилизирующие факторы) определяется свойством сети, называемым устойчивостью. Обеспечение устойчивости заключается в сохранении функционирования сетей электросвязи в условиях дестабилизирующих факторов.

При функционировании сети в условиях дестабилизирующих информационно-технических воздействий особую значимость приобретает такое свойство сети, как устойчивость к внешним дестабилизирующим факторам, воздействующим на информацию. Устойчивость информационной системы (англ. resilience) трактуется как способность системы противостоять действиям, влияющим на ее работоспособность, а также восстанавливать исходное операционное состояние после сбоя в работе, вызванного любым инцидентом.

Основными дестабилизирующими факторами, воздействующими на сеть и нарушающими устойчивость ее функционирования, являются:

— Случайные отказы и сбои сетевого и серверного оборудования;

— не выявленные ошибки программного обеспечения;

— ошибки исходных данных программ;

— некорректные действия пользователей сети;

— стихийные воздействия среды, приводящие к повреждению или уничтожению элементов сети;

— случайные и преднамеренные помехи, приводящие к искажению информации, циркулирующей в сети;

— скачкообразно изменяющийся трафик от абонентов системы;

— воздействие программных вирусов и другого вредоносного кода;

— информационно-технические (программно-аппаратные и радиоэлектронные) воздействия на объекты сети.

Понятие устойчивости относится не к самой системе, а к какому-либо свойству её функционирования. Различные виды возмущающих факторов определяют частные виды устойчивости, поскольку может наблюдаться устойчивость по отношению к одним возмущениям и неустойчивость по отношению к другим возмущениям в смысле какого-либо одного определения устойчивости. В отношении сложной системы можно рассматривать следующие виды устойчивости:

— устойчивость к отказам техники (отказоустойчивость);

— устойчивость к ошибкам программ (ошибкоустойчивость);

— устойчивость (стойкость) к внешним климатическим факторам;

— устойчивость к повреждению элементов системы (живучесть);

— устойчивость к механическим воздействиям;

— устойчивость к радиоэлектронным воздействиям (помехоустойчивость, помехозащищенность);

— устойчивость к программно-аппаратным воздействиям, к сетевым атакам, вредоносным программам;

— устойчивость к резкому увеличению трафика, рабочей нагрузки (трафикоустойчивость, нагрузоустойчивость).

Среда OMNeT++ не предоставляет возможность смоделировать взаимодействие на сеть каких-либо внешних физических факторов, а также не имеет в своем инструментарии возможности воссоздавать программные ошибки и вредоносные программы. Вследствие выше сказанного, возможно оценить устойчивость сети по следующим характеристикам:

— устойчивость к отказам техники (отказоустойчивость);

— устойчивость к резкому увеличению трафика, рабочей нагрузки (трафикоустойчивость, нагрузоустойчивость).

2 Постановка эксперимента

2.1 Описание модели

Для моделирования в среде OmNet++ была построена сеть произвольной топологии, работающая по протоколу RSTP. Представленная на рисунке 1.

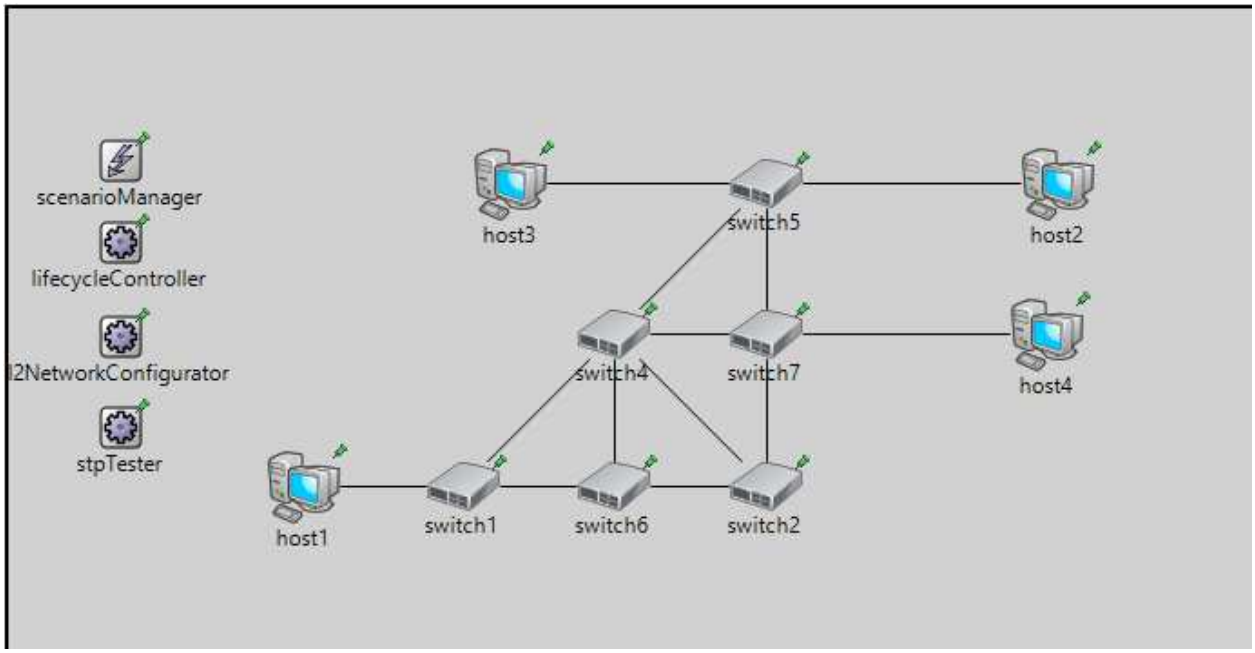

Рисунок 1 – топология сети

Данная модель описана кодом networks.ned и представлена в приложении А.

В состав сети входят 7 сетевых коммутаторов switch 1 – switch 7, типа EtherSwitch, конечные устройства host 1- host 4, типа Etherhost. Host1 и host2 постоянно обмениваются пакетами, а host3 и host4 используются в экспериментах с нагрузкой сети. Так же на схеме присутствует модуль scenarioManager, управляющий имитационными экспериментами и дающий возможность планировать определенные события в определенное время, модуль lifecycleController, управляет операциями выключение/перезагрузка, остановка/возобновление, сбой/восстановление и другими операциями над узлами, модуль stpTester, обеспечивает моделирование протокола RSTP, модуль L2NetworkConfigurator.

Рассмотрим создание сетевого коммутатора switch 1.

```
switch1: EtherSwitch {
    parameters:
        @display("p=333,78");
    gates:
        ethg[];
}
```

Рисунок 2 – создание switch1

Создавая сетевой коммутатор, ему присваивается готовый класс EtherSwitch, который уже включен в OMNeT++, а именно в модуль INET, где прописана вся структура роутера, принадлежащего к данному классу. Затем, в секции parameters указывается значение для @display, которая отвечает за точку размещения роутера на картинке сети. После секции parameters следует секция gates, где необходимо обозначить количество интерфейсов маршрутизатора.

Создание остальных маршрутизаторов и хостов имеет такой же вид, с той лишь разницей, что у каждого своего значения для @display и свое количество задействованных интерфейсов.

У каждого элемента, созданного нами имеется своя структура (рисунок – 3,4).

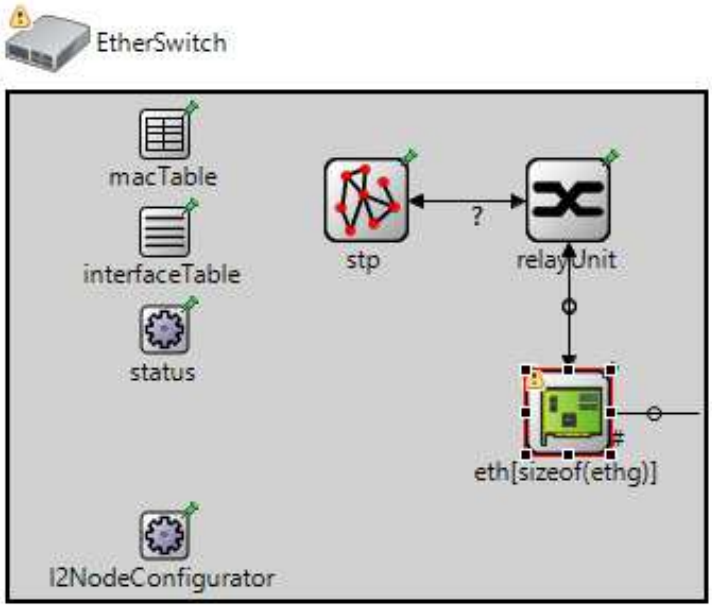

Рисунок 3 – структура сетевого коммутатора

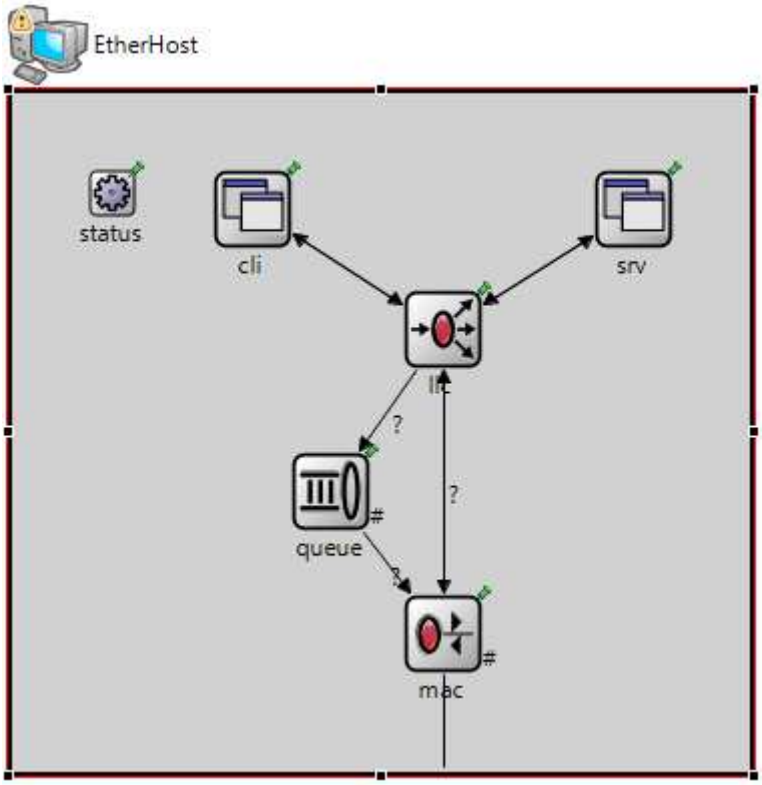

Рисунок 4 – структура конечного устройства

Между двумя компонентами устанавливается соответствие. При установлении соответствия необходимо указать тип канала, который соединяет два узла, в данном случае «Eth1G».

```
connections:
    switch1.ethg++ <--> Eth1G <--> switch4.ethg++;
    switch1.ethg++ <--> Eth1G <--> switch6.ethg++;
    switch2.ethg++ <--> Eth1G <--> switch4.ethg++;
    switch2.ethg++ <--> Eth1G <--> switch6.ethg++;
    switch2.ethg++ <--> Eth1G <--> switch7.ethg++;
    //  switch3.ethg++ <--> Eth1G <--> switch7.ethg++;
    switch4.ethg++ <--> Eth1G <--> switch5.ethg++;
    switch4.ethg++ <--> Eth1G <--> switch6.ethg++;
    switch4.ethg++ <--> Eth1G <--> switch7.ethg++;
    switch5.ethg++ <--> Eth1G <--> switch7.ethg++;
```

Рисунок 5 – создание каналов связи

2.2 Эксперимент

При запуске моделирования проекта Nedworks происходит сбора всех входящих в него файлов и их компиляция. После компиляции открывается окно моделирование, где происходит визуализация всех проходящих процессов и можно наблюдать, как коммутаторы пошагово обмениваются пакетами данных, как происходит коммутация. Пример полученного окна моделирования представлена на рисунке 6.

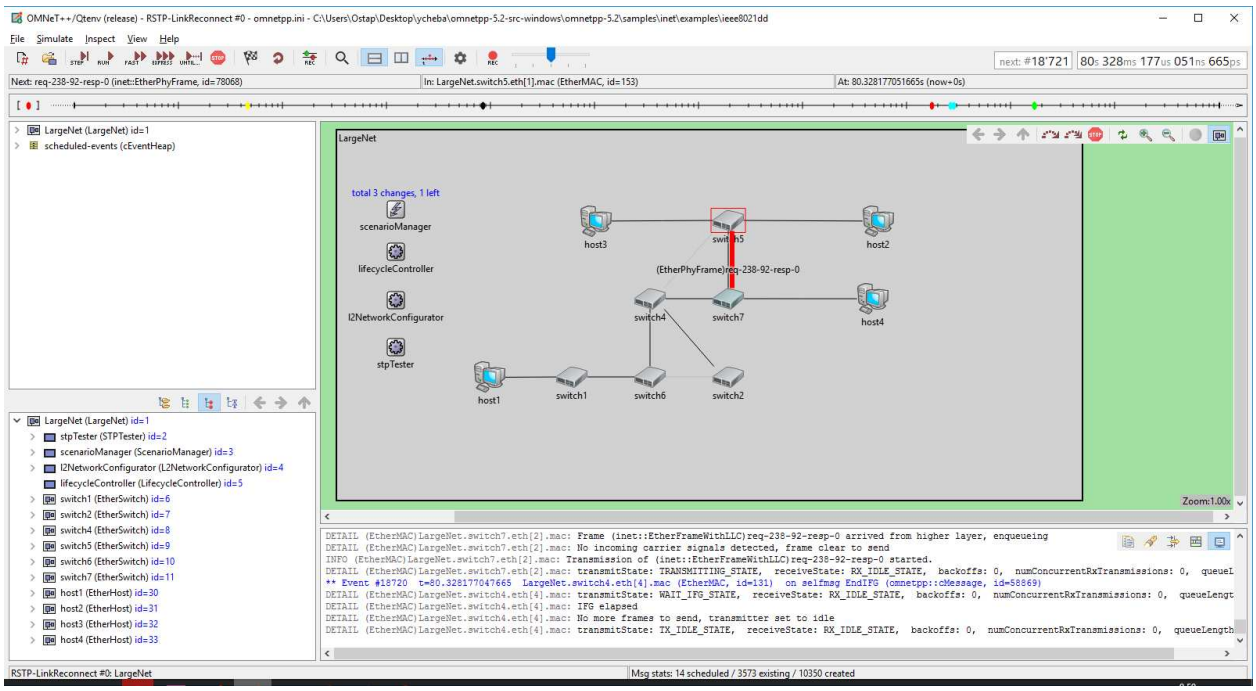

Рисунок 6 – пример окна моделирования

В центре экрана изображена моделируемая сеть. На экране присутствует элемент с названием «scenaioManager», который располагается в левом верхнем углу. Данный модуль запускает наш сценарий, который заранее был написан, для моделирования той ситуации, которая нужна в эксперименте.

```
<script>
    <at t="59s">
        <disconnect src-module="switch1" src-gate="ethg[0]" />
    </at>
    <at t="120s">
        <connect src-module="switch1" src-gate="ethg[0]"
                 dest-module="switch4" dest-gate="ethg[0]"
                 channel-type="inet.node.ethernet.Eth1G">
        </connect>
    </at>
</script>
```

Рисунок 7 – скрипт сценария

Запустив через горизонтальное меню симуляцию работы сети, можно наблюдать процесс настройки сети и ее работы. С течением времени окно со списком событий будет постепенно пополняться новыми событиями и

информацией о них. Данное окно располагается внизу окна моделирования, под окном с отображением топологии сети.

Над окном с топологией сети располагается временная шкала, занимающая всю ширину окна программы. На временной шкале отображаются события, которые произойдут в течение следующих 100 секунд.

При включении самой низкой скорости симуляции протекание каждого события сопровождается анимацией в окне моделирования. Можно наблюдать инициализацию коммутаторов в начале симуляции, отправку Hello-пакетов и назначение корневых портов.

По окончанию симуляции в папке проекта был создан файл results, в который были помещены все данные, собранные программой в ходе моделирования. Собранные данные могут быть использованы для получения графиков работы сети. В зависимости от выбора, график может быть выведен в виде скаляра или вектора после выбора необходимых параметров.

Для создаваемого графика был выбран вариант вывода данных в виде вектора.

В качестве параметров вывода было выбрано, что на график отправки пакетов с конечного устройства host2 и график получение пакетов на конечном устройстве host1.

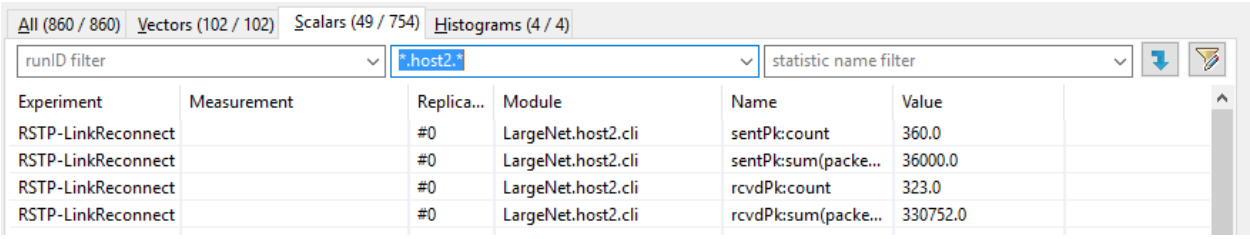

Рисунок 8 – Разница отправленных пакетов и полученных

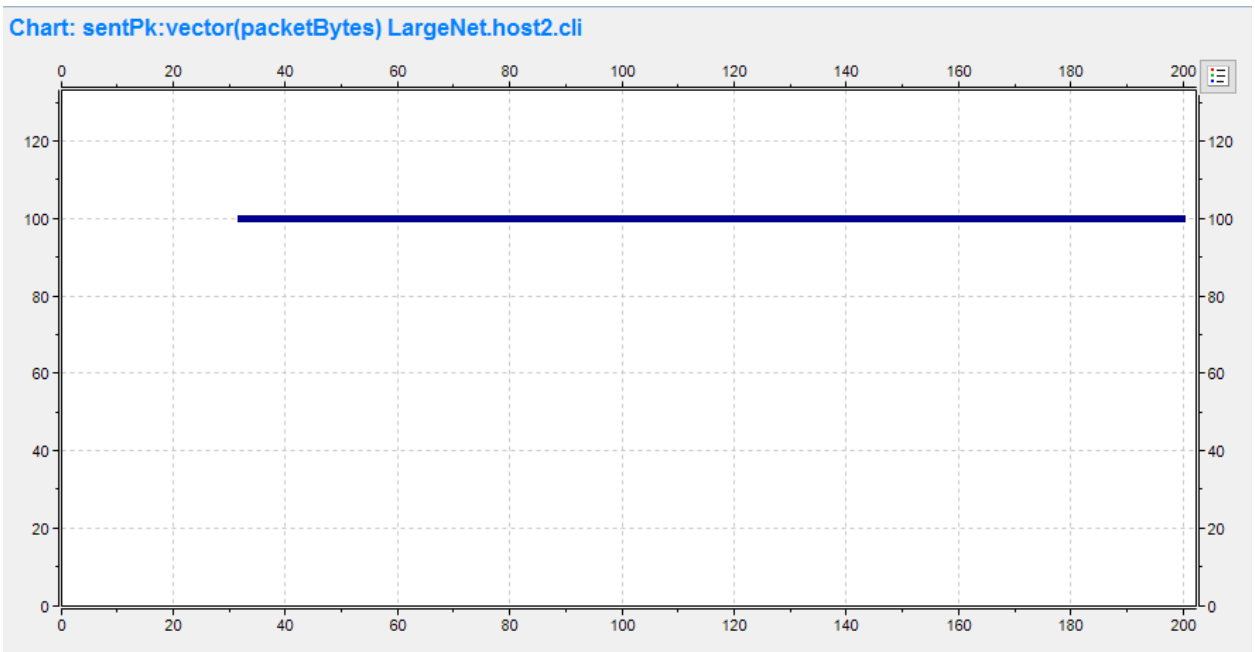

Рисунок 9 – график отправки пакетов

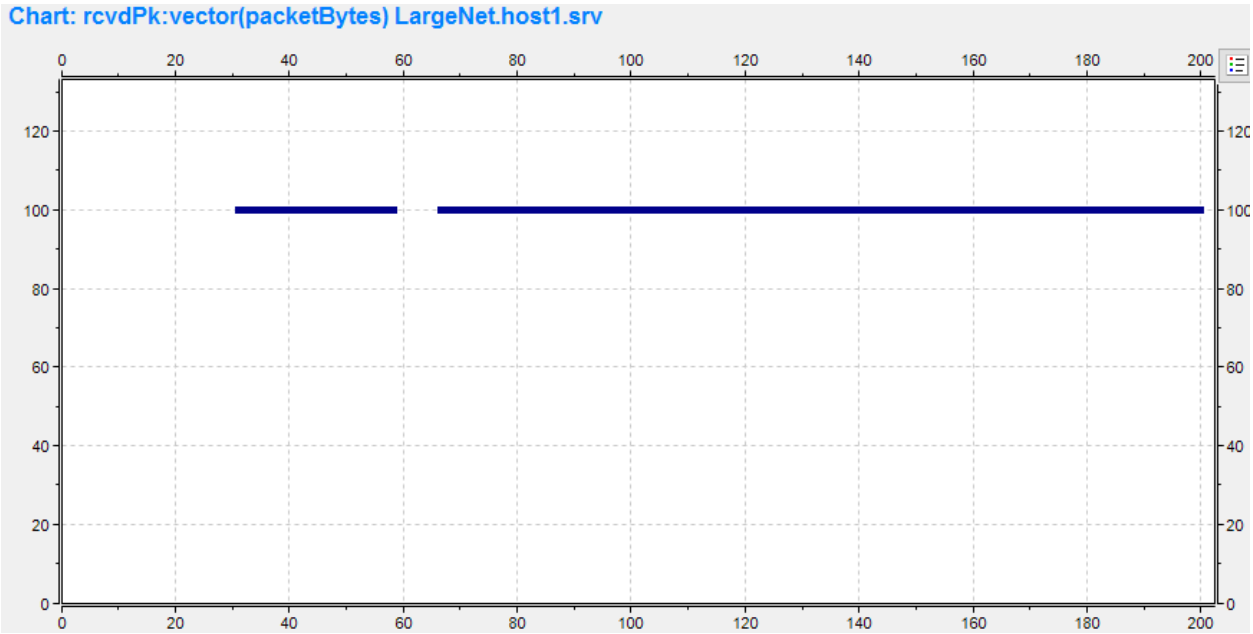

Рисунок 10 – график получения пакетов

В ходе экспериментов было проведено множество тестов, с отключением разных каналов связи между коммутаторами, а так же с нагрузочным трафиком по сети. Целью серии экспериментов является сбор данных и анализ полученных результатов. Описание экспериментов можно увидеть в таблице 1.

Таблица 1 – описание экспериментов

Номер эксперимента	Описание
1	Отключение корневого порта между switch5 и switch7.
2	Отключение корневого порта между switch2 и switch7, нагрузка канала связи между switch5 и switch7.
3	Отключение корневого порта между switch2 и switch6, нагрузка канала связи между switch5 и switch7.
4	Отключение назначенного порта между switch1 и switch6, появление новой связи коммутаторами switch1 и switch4, нагрузка канала связи между switch5 и switch7.

3 Результаты экспериментов и их анализ

Таблица 2 – результаты, полученные в ходе экспериментов без нагрузки

№ эксперимента	Пакетов отправлено	Потеряно пакетов	Процент Потерь, %	Время восстановления, сек
1	360	39	10,8	6
2	360	37	10,2	6
3	360	37	10,2	6
4	360	63	17,5	12

Таблица 3 – результаты, полученные в ходе экспериментов с нагрузкой сети

Номер эксперимент	Пакетов отправлено	Потеряно пакетов	Процент Потерь, %	Время восстановления, се
2	354	38	10,7	6
3	352	37	10,5	6
4	353	63	17,8	12

В результате экспериментов мы видим, что количество потерянных пакетов не меняется в зависимости от нагрузки сети, уменьшается количество пройденных пакетов через узлы, а так же немного уменьшается скорость передачи пакетов, но всё это не влияет на потерю пакетов.

Количество потерянных пакетов напрямую зависит от сходимости сети, так как протоколу RSTP нужно три “Hello” пакета, что бы понять, что данная связь больше недоступна, а “Hello” пакеты отправляются каждые 2 секунды, то мы можем сделать вывод, что максимальное время потери пакетов может составить 7,59 секунд, при обрыве, потери данного соединения. Но это время не является максимальным для времени восстановления сети, при потере одного соединения и одновременно подключении другого, которого ранее не было, время восстановления сети может составлять 12 секунд, за которое происходит большая потеря пакетов.

ЗАКЛЮЧЕНИЕ

В данной работе была смоделирована произвольная топология сети, работающая по протоколу RSTP. Разобрана работа протокола RSTP. Выбраны такие критерии оценки устойчивости работы сети, как время восстановления канала связи, после отключения основного канала. Так же были проанализированы потери пакетов данных, связанные с отключение основного канала связи, с определенными факторами, а именно нагрузкой сети другими хостами, передающими сообщения между собой. Отсюда можно сделать вывод, что сеть является устойчива к отказам техники и увеличению трафика.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Олифер, В. Г. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов / В. Г. Олифер, Н. А. Олифер. Изд. 4-е – Санкт-Петербург : Питер, 2012. – 944 с.
2. OmNet++ [Электронный курс] : OMNeT++ Discrete Event Simulator. - Режим доступа: https://www.omnetpp.org/
3. Олифер, Н. А. Средства анализа и оптимизации локальных сетей [Электронный ресурс] / Н. А. Олифер, В. Г. Олифер. // Центр Информационных Технологий. – 1998. – Режим доступа: http://citforum.ru/nets/optimize/index.shtml
4. Таненбаум, Э. Компьютерные сети / Э. Таненбаум. - 4-е изд.: Питер, 2013. - 992 с.
5. Хант, К. TCP/IP. Сетевое администрирование. К. Хант. - 1е изд.: Символ-Плюс, 2010. - 816 с.
6. Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности. СТО 4.2-07-2014. Красноярск.: ИПЦ СФУ, 2014 – 60 с.
7. Сайт о программировании litl-admin.ru [Электронный ресурс] – Режим доступа: https://litl-admin.ru/cisco/otlichiya-v-rabote-stp-i-rstp.html.
8. Сайт о работе протокола RSTP habr.com [Электронный ресурс] – Режим доступа: https://habr.com/company/cbs/blog/330358.
9. Официальный сайт OMNeT++ [Электронный ресурс] – Режим доступа: https://omnetpp.org/models.
10. ГОСТ Р 53111-2008 Устойчивость функционирования сети связи общего пользования. Требования и методы проверки.

ПРИЛОЖЕНИЕ А
Код файла Networks.ned

```
package inet.examples.ieee8021dd;

import inet.common.lifecycle.LifecycleController;
import inet.common.scenario.ScenarioManager;
import inet.linklayer.configurator.L2NetworkConfigurator;
import inet.linklayer.ieee8021d.tester.STPTester;
import inet.node.ethernet.Eth1G;
import inet.node.ethernet.EtherHost;
import inet.node.ethernet.EtherSwitch;


network SwitchNetwork
{
    @display("bgb=689,368");
    submodules:
        stpTester: STPTester {
            @display("p=75,277");
        }
        scenarioManager: ScenarioManager {
            @display("p=75,101");
        }
        l2NetworkConfigurator: L2NetworkConfigurator {
            @display("p=75,217");
        }
        lifecycleController: LifecycleController {
            @display("p=75,155");
        }

        switch1: EtherSwitch {
            parameters:
                @display("p=333,78");
            gates:
                ethg[];
        }

        switch2: EtherSwitch {
            parameters:
                @display("p=301,217");
            gates:
                ethg[];
        }

      //  switch3: EtherSwitch {
      //      parameters:
      //          @display("p=443,78");
      //      gates:
      //          ethg[];
      //  }

        switch4: EtherSwitch {
            parameters:
                @display("p=322,153");
            gates:
                ethg[];
        }

        switch5: EtherSwitch {
```

```
            parameters:
                @display("p=529,217");
            gates:
                ethg[];
        }

        switch6: EtherSwitch {
            parameters:
                @display("p=200,154");
            gates:
                ethg[];
        }

        switch7: EtherSwitch {
            parameters:
                @display("p=443,153");
            gates:
                ethg[];
        }

    connections:
        switch1.ethg++ <--> Eth1G <--> switch4.ethg++;
        switch1.ethg++ <--> Eth1G <--> switch6.ethg++;
        switch2.ethg++ <--> Eth1G <--> switch4.ethg++;
        switch2.ethg++ <--> Eth1G <--> switch6.ethg++;
        switch2.ethg++ <--> Eth1G <--> switch7.ethg++;
    //  switch3.ethg++ <--> Eth1G <--> switch7.ethg++;
        switch4.ethg++ <--> Eth1G <--> switch5.ethg++;
        switch4.ethg++ <--> Eth1G <--> switch6.ethg++;
        switch4.ethg++ <--> Eth1G <--> switch7.ethg++;
        switch5.ethg++ <--> Eth1G <--> switch7.ethg++;
}

network LargeNet extends SwitchNetwork
{
    @display("bgb=952,472");
    submodules:
        host1: EtherHost {
            @display("p=199.565,39.435");
        }

        host2: EtherHost {
            @display("p=609,217");
        }

   //     host3: EtherHost {
   //         @display("p=529,78");
   //     }
   //     host4: EtherHost {
   //         @display("p=442.15002,295.165");
    //    }
    connections:
   //     switch5.ethg++ <--> Eth1G <--> host3.ethg;
        switch5.ethg++ <--> Eth1G <--> host2.ethg;
        switch1.ethg++ <--> Eth1G <--> host1.ethg;
    //    host4.ethg <--> Eth1G <--> switch7.ethg++;

}
```

Код файла omnetpp.ini

```
[General]
sim-time-limit = 200s
tkenv-plugin-path = ../../../etc/plugins
**.vector-recording = true
**.visualize = true
**.agingTime = 1s
**.connectionColoring = false
network = SwitchNetwork
**.switch1.**.address="AAAAAA000001"
**.switch2.**.address="AAAAAA000002"
**.switch3.**.address="AAAAAA000003"
**.switch4.**.address="AAAAAA000004"
**.switch5.**.address="AAAAAA000005"
**.switch6.**.address="AAAAAA000006"
**.switch7.**.address="AAAAAA000007"

[Config LargeNet]
network = LargeNet
**.switch8.**.address="AAAAAA000008"
**.switch9.**.address="AAAAAA000009"
**.switch10.**.address="AAAAAA00000A"
**.switch11.**.address="AAAAAA00000B"
**.host2.cli.destAddress = "LargeNet.host1"
**.host3.cli.destAddress = "LargeNet.host4"
**.host5.cli.destAddress = "LargeNet.host3"
**.host6.cli.destAddress = "LargeNet.host3"
**.host*.cli.startTime = 17s
```

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В.Непомнящий
подпись инициалы, фамилия
« 15 » 06 2018 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Моделирование работы сети произвольной топологии, работающей по протоколу RSTP, и оценка ее устойчивости средствами OmNet++
тема

Руководитель	_____ подпись, дата	доцент, канд. техн. наук должность, ученая степень	О. А. Русанова инициалы, фамилия
Выпускник	_____ подпись, дата		О. Ю. Анищенко инициалы, фамилия
Нормоконтролер	_____ подпись, дата	8.06.18г	В. И. Иванов инициалы, фамилия

Красноярск 2018