На правах рукописи

Тарасов Виталий Сергеевич

Анализ методов и алгоритмов обеспечения показателей надежности программно-конфигурируемых сетей

Направление подготовки
11.04.02 «Инфокоммуникационные технологии и системы связи»
Программа магистратуры – Сети, системы и устройства телекоммуникаций

АВТОРЕФЕРАТ
магистерской диссертации
на соискание квалификации (степени) магистра

Екатеринбург 2021

Работа выполнена в Федеральном государственном бюджетном образовательном учреждении высшего образования «Сибирский государственный университет телекоммуникаций и информатики» Уральский технический институт связи и информатики (филиал) в г. Екатеринбурге (УрТИСИ СибГУТИ)

Научный руководитель:
кандидат технических наук, доцент Н.В. Будылдина

Рецензент: Е.В. Букрина

Защита состоится «28» июня 2021г. в 09:00 часов в Федеральном государственном бюджетном образовательном учреждении высшего образования «Сибирский государственный университет телекоммуникаций и информатики» Уральский технический институт связи и информатики (филиал)в г. Екатеринбурге (УрТИСИ СибГУТИ), г. Екатеринбург, ул. Репина, д. 15.

Секретарь государственной аттестационной комиссии: О.А. Шумилова

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы исследования и степень ее разработанности.

Телекоммуникационные сети связи имеют как контроль, так и плоскости данных, объединенные в сетевой узел. Традиционная сеть поддерживает несколько протоколов надежности, которые варьируются от многопутевой маршрутизации до отказа определенного элемента (например, обнаружение и обработка механизмов). Однако разные типы сетевых коммуникационных технологий имеют разные стандарты и со временем все больше развиваются, что затрудняет управление ядром традиционной сети, например, требование использовать низкоуровневые команды, зависящие от вендора, для обновления каждого сетевого устройства. Такая сложность традиционной сети препятствует росту трафика данных и надежности услуг.

С другой стороны, программно-конфигурируемые сети (SDN) разделяют плоскости управления и данных. Контроллер SDN предлагает централизованную точку управления, которая может собирать информацию и определять наилучшую политику пересылки. Он может обнаруживать и обрабатывать отказы элементов, а затем предоставлять альтернативные маршруты. Это свойство делает SDN отличным кандидатом для поддержки надежности сети. Однако, в отличие от традиционной сети, SDN создает новую сетевую плоскость, которая соединяет разделенные плоскости данных и управления, а именно, сеть путей управления. Многодоменная архитектура SDN еще больше усложняет управление надежностью сети. Эти проблемы надежности одинаково применимы как к устаревшим системам, так и к системам на основе SDN.

Надежность сети измеряет, в какой степени сеть может оставаться в рабочем состоянии при минимальном уровне требований в условиях эксплуатации.

Актуальность задачи исследования – это возможность создать телекоммуникационную сеть с использованием виртуального контроллера программно-конфигурируемых сетей, для увеличения надежности телекоммуникационной сети и уменьшения эксплуатационных затрат. Для этого необходимо сделать анализ методов и алгоритмов надежности в программно-конфигурируемых сетях.

Объект исследования – надежность в программно-конфигурируемых сетях.

Предмет исследования – методы и алгоритмы повышения надежности в программно-конфигурируемых сетях.

Целью работы является анализ методов и алгоритмов обеспечение показателей надежности программно-конфигурируемых сетей.

Для достижения означенной цели необходимо решить следующие задачи:

1) проведение анализа публикаций по теме исследования;

2) анализ показателей надежности в телекоммуникационных сетях;

3) описание методов повышения надежности в телекоммуникационных сетях;

4) сравнение надежности телекоммуникационных и виртуальных сетей;

5) описание архитектуры программно-конфигурируемых сетей и её инструментов;

6) исследование алгоритмов обеспечения надежности в программно-конфигурируемых сетях;

7) исследование методов обеспечения надежности в программно-конфигурируемых сетях;

8) моделирование программно-конфигурируемых сетей с использованием контроллера.

Теоретическая значимость исследования обоснована следующим:

- исследованы основные показатели надежности и их свойства;
- исследована архитектура программно-конфигурируемых сетей и её инструментов;
- исследованы методы и алгоритмы для обеспечения надежности в программно-конфигурируемых сетях;

Практическая значимость.

Практическая значимость работы заключается в разработке модели программно-конфигурируемых сетей и внедрение результатов моделирования в учебный процесс.

Методология и методы исследования. Решение поставленных задач осуществлялось с использованием программной среды Hyper-V, в которой можно создавать виртуальные машины, которые могут использоваться в виде контроллера программно-конфигурируемых сетей. Также применялся анализ схем топологий сетей.

Положения, выносимые на защиту:

- методы повышения надежности;
- основные понятия архитектуры программно-конфигурируемых сетей;
- исследования методов и алгоритмов обеспечения надежности
- выводы на основе исследования.

Степень достоверности результатов исследования обусловлена следующими фактами:

- теория построена на известных и проверяемых данных с использованием методов теории моделирования, теории эксперимента и согласуется с экспериментальными данными по теме диссертации;
- для экспериментальных работ результаты получены в специальной программной среде для исследования надежности программно-конфигурируемых сетей при разных методах и алгоритмах;
- использованы современные средства сбора и обработки данных.

Апробация результатов.

Основные результаты диссертации были получены и использованы в рамках научно-исследовательских работ (НИР), в том числе:

В рамках сборника научных трудов Международной научно-практической конференции «Инфокоммуникационные технологии:

актуальные вопросы цифровой экономики» (Екатеринбург, 2021), которая проходила 17-18 февраля 2021 года.

По теме диссертации опубликовано 1 научные работа, в том числе 1 статьи в научных журналах, рекомендованных РИНЦ.

Диссертационная работа включает введение, четыре главы, заключение, список литературы из 27 наименований. Объем диссертации 91 страниц, включены также 54 рисунка и 1 таблица.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во **введении** изложена общая характеристика диссертационной работы: показана ее актуальность, сформулирована цель работы, отражена практическая ценность. Также описаны основные вопросы и проблемы связанные с темой диссертации.

В первой главе рассмотрены статьи (публикации) по теме исследования.

Представлены основные понятия программно-конфигурируемых сетей и виртуализации и отмечено преимущества использования данных технологий.

Далее рассмотрены основные моменты по каждой из публикаций.

Сначала рассматривается основная теория, затем преимущества использования и различие между SDN и NFV в конце приводятся выводы на основе публикаций.

Software Defined Network стала технологической тенденцией, которая привлекла поставщиков услуг, исследователей, производителей оборудования, разработчиков программного обеспечения и пользователей.

Изменения в сетевой инфраструктуре, такие как добавление оконечных систем и добавление новых сервисов, трудно обрабатывать в обычных сетях связи. SDN известен тем, что разделяет данные и функции управления сетевых устройств, таких как маршрутизаторы и коммутаторы, взаимодействуя через интерфейс прикладного программирования (англ. API – Application Programming Interface) между данными и функциями управления.

Рассмотренные различия между SDN и NFV касаются общих сфер, которые отвечают за управление сетью и стандартов, определяющих архитектурное и функциональное развитие. SDN определяет общие аспекты всей сети – тип инфраструктуры, доступных сервисов и приложений, определяет сетевые политики, которые регулируют доставку и использование сетевых ресурсов. Гипервизор организует и контролирует сетевые функции нижнего уровня. NFV – предоставляет широкий спектр конкретных функций, которые должны выполняться на всех уровнях и этапах сети - на периферии, границе и ядре - под контролем гипервизора.

Так же SDN и NFV имеют разные стандарты. SDN имеет стандарт Open Network Foundation, которая стремится разработать «различные открытые стандарты, а также стандарты, не зависящие от поставщика, для интерфейса связи, определенного между уровнями управления и пересылки в архитектуре SDN». NFV имеет стандарт Европейского института телекоммуникационных стандартов (ETSI), которые определяют и поддерживают «глобально применимые стандарты информационных и телекоммуникационных технологий в отношении NFV».

Далее были рассмотрены основные понятия надежности и их показатели.

Надежность является элементом, которая будет выполнять предназначенную функцию в течении определенного периода времени или в определенной среде без сбоев.

Приведены основные показатели надежности на рисунке 1.

Рисунок 1 – Основные показатели надежности

Далее были рассмотрены методы повышения надежности телекоммуникационных сетей.

Были описаны пять основных подхода повышения надежности. А также по результатам исследований было выявлено, что многие ИТ-организации развертывают серверы с сервисами, которые обеспечивают надежность в телекоммуникационной сети, но сервера работают только на небольшую часть своей мощности, часто потому, что они выделяют свой физический сервер для определенного приложения. Обычно это неэффективный механизм, поскольку имеется избыточная мощность, которая не используется, что приводит к более высоким эксплуатационным расходам и затратам на оборудование. В целях повышения эффективности использования емкости и снижения затрат была создана виртуализация. Одна из форм технологий виртуализации является NFV.

Далее было рассмотрено сравнение виртуализации по сравнению с традиционной сетью.

Самая большая разница между виртуализацией и традиционной сетью заключается в том, что виртуализация является программным средством, а традиционные сети обычно основаны на оборудовании. Программная поддержка виртуализации обеспечивает масштабируемость и гибкость, а также предоставляет пользователям больший контроль и более простое управление ресурсами, позволяя пользователям виртуально управлять ресурсами с помощью плоскости управления.

В отличие от виртуализации, традиционные сети используют маршрутизаторы, коммутаторы и другую аппаратную и физическую инфраструктуру для создания подключений и работы сетей.

По сравнению с традиционной сетью, виртуализация может лучше взаимодействовать с оборудованием, использующим сеть. Вместо использования физической инфраструктуры виртуализация позволяет пользователям применять программное обеспечение для подготовки новых устройств и позволяет ИТ-администраторам строить сетевые узлы и сетевые службы.

Во второй главе описаны основные положения программно-конфигурируемых сетей и протокола OpenFlow. Рассмотрены инструменты

для эмуляции сети SDN с протоколом OpenFlow, а также рассмотрен инструмент для конфигурации коммутаторов и маршрутизаторов Postman.

Ключевые принципы – разделение процессов передачи и управления данными, централизация управления сетью при помощи унифицированных программных средств, виртуализация физических сетевых ресурсов.

Для SDN виртуализация превращает плоскость управления из физического оборудования в программную плоскость. Виртуализация позволяет получить доступ к плоскости управления через подключенное устройство и дает ИТ-администраторам больше возможностей для управления потоком трафика с помощью централизованного пользовательского интерфейса (англ. UI – User Interface).

Рисунок 2 описывает общую архитектуру SDN в соответствии с сетевыми плоскостями и взаимодействиями между ними. Сети SDN разделены на уровень приложений, уровень управления и уровень данных.

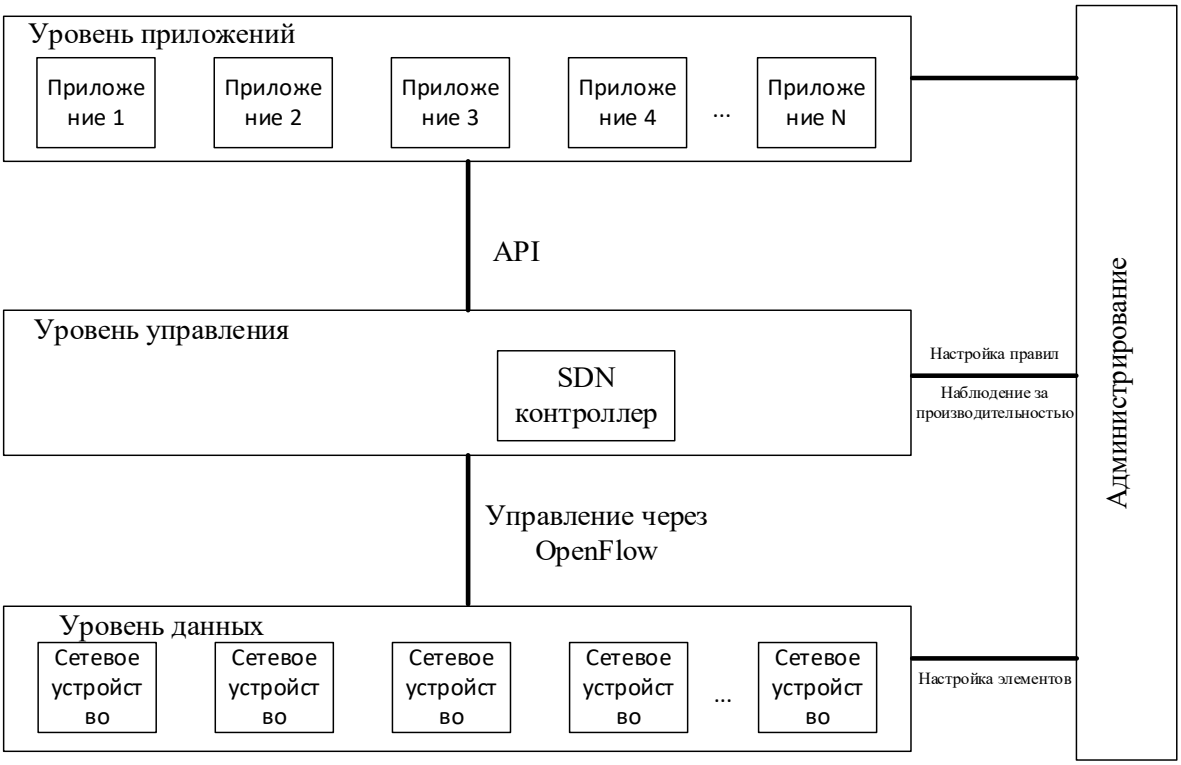

Рисунок 2 – Архитектура SDN

Далее рассматривается набор сетевых элементов и инструментов программно-конфигурируемых сетей.

На основе протокола OpenFlow можно отделить основное существующие оборудование от управления этим оборудованием за счет создания специального программного обеспечения, которое может работать на отдельном компьютере и находится под контролем администратора сети. Это позволяет упростить набор сетевых элементов плоскости передачи данных и логическую централизацию управления сетью.

Контроллеры SDN направляют трафик в соответствии с учетом политики пересылки, которые устанавливает сетевой оператор, тем самым сводя к минимуму ручную настройку отдельных сетевых устройств.

Коммутатор OpenFlow состоит из одной или более таблиц потоков и таблицы групп, которые осуществляют операции поиска и пересылки пакетов, а также канала OpenFlow, обеспечивающего соединение с внешним контроллером.

Mininet позволяет создавать топологии очень большого размера до тысяч узлов и очень легко проводить на них тестирование. Mininet имеет очень простые инструменты командной строки и API. Mininet позволяет пользователю легко создавать, настраивать, совместно использовать и тестировать сети SDN.

Были представлены топологии, которые используются в эмуляторе Mininet. Исходя из архитектуры программно-конфигурируемой сети, можно сделать вывод, древовидная топология обеспечивает наиболее высокую надежность системы передачи и качества предоставления услуг.

Также был рассмотрен инструмент Postman.Postman разработан, чтобы ускорить и упростить разработку API, так же он используется для тестирования API, которые создается в проекте или программном продукте.

В третьей главе проанализированы алгоритмы и методы надежности в программно-конфигурируемых сетях.

С помощью алгоритма Дейкстры можно найти кратчайший путь между узлами графа. В частности, можно найти кратчайший путь от узла (называемого «исходным узлом») до всех других узлов в графе, создавая дерево кратчайших путей. Кроме того, обеспечивает равномерное распределение трафика по всем участкам сети, что снижает нагрузку на сеть и увеличивает скорость передачи данных, тем самым улучшая надежность и качество передачи.

Алгоритм Йена вычисляет количество кратчайших путей между двумя узлами. Этот алгоритм часто называют алгоритмом k-кратчайшего пути Йена, где k - количество кратчайших путей для вычисления. Алгоритм поддерживает взвешенные графы с положительными весом. Он также учитывает параллельные отношения между одними и теми же двумя узлами при вычислении нескольких кратчайших путей.

Проанализировав алгоритмы, был сделан вывод, что для повышения надежности лучше использовать алгоритм Дейкстра, так как этот алгоритм работает в сетях любой сложности и является более надежным в случае высокой нагрузки на сети и равномерно распределяет трафик.

Далее рассматриваются методы повышения надежности.

Для реализации обеспечения надежности используются контроллеры, они могут быть как реальные, так и виртуальные. Виртуальные коммутаторы или маршрутизаторы представляют собой наименьший общий знаменатель в сетевой среде и, как правило, способны к меньшему количеству переадресованных записей, чем их выделенные аппаратно-ориентированные коммутаторы. Были рассмотрены три контроллера программно-

конфигурируемых сетей. Платформа OpenDayLight – программный комплекс, претендующий на универсальное применение в различных SDN-сетях, а не только в сетях OpenFlow.

Ryu Controller – это открытый программно-конфигурируемый сетевой контроллер (SDN), разработанный для повышения гибкости сети за счет упрощения управления и адаптации способов обработки трафика.

Контроллер POX – это контроллер с открытым исходным кодом для разработки приложений SDN. Контроллер POX обеспечивает эффективный способ реализации протокола OpenFlow, который фактически является протоколом связи между контроллерами и коммутаторами.

В таблице 1 показано сравнение контроллеров.

Таблица 1 – Сравнительная таблица контроллеров

	OpendayLight	Ryu	POX
Поддерживаемый язык программирования	Java	Python	Python
Поддержка OpenFlow	V1.0 V1.1 V1.2 V1.3 V1.4	V1.0 V1.1 V1.2 V1.3 V1.4	V1.0 V1.1 V1.2
Открытый исходный код	Да	Да	Да
Веб-интерфейс	Да	Да, но нельзя запускать вместе с другими приложениями.	Да, но нельзя запускать вместе с другими приложениями.
REST API	Да	Да	Нет
Поддерживаемая операционная система	Linux, MAC, Windows	Linux	Linux, MAC, Windows

Проанализировав методы обеспечения надежности на базе контроллеров , был сделан вывод, что контроллер Ryu, является наилучшим вариантом, так как у него обширная библиотека приложений и сервисов, возможность реализовывать сервисы с помощью языка программирования Python, а также поддерживает протокол OpenFlow последний версии. Контроллер OpenDayLight имеет тоже свои преимущества в виде веб-интерфейса. Что упрощает работу в самом контроллере, но имеет меньшее количество поддерживаемых приложений, а также поддерживает только язык программирования Java-JSON, что для неопытного сетевого администратора является трудным решением.

В четвертой главе изложены результаты по исследованию и моделированию программно-конфигурируемых сетей с использованием контроллеров Ryu и OpenDayLight.

Смоделирована схема организаций связи с помощью технологий виртуализации с использованием гипервизора под названием «Hyper-V». Схема организации связи корпоративной сети изображена на рисунке 3.

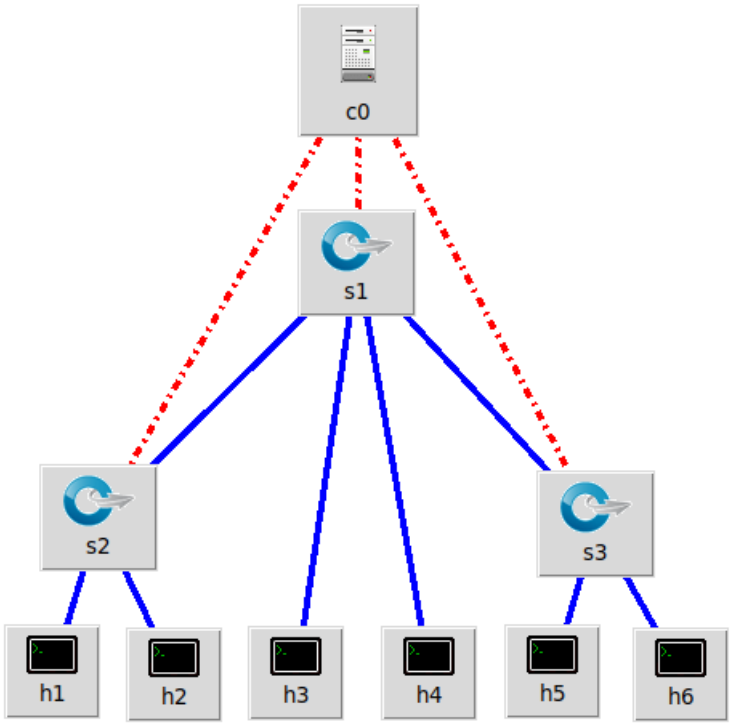

Рисунок 3 – Схема организации связи корпоративной сети

Далее была выбрана операционная система Linux в качестве платформы, где будет установлен контроллер SDN. Linux в отличии от других операционных систем обладает высокой надежностью и максимальной производительностью, а также имеет открытый исходный код. В качестве дистрибутива был выбран Ubuntu, так как он является простым для начинающего сетевого администратора и имеет постоянную поддержку со стороны разработчика.

Затем был проделан ряд настроек и установлен набор компонентов для дальнейшей работы системы с контроллером программно-конфигурируемой сети. Набор компонентов изображен на рисунке 4.

```
sdn@sdn-mininet: $ sudo apt install net-tools -y && sudo apt install git -y & sudo apt install unzip -y && sudo apt
 install curl -y && sudo apt install python -y && sudo apt install python3-pip -y && sudo apt install mininet -y
```

Рисунок 4 – Набор установки программ в Ubuntu

Далее создали собственную топологию в Mininet согласно рисунку 3. Создать топологию можно двумя способами. Первый способ – это ввести

команду «sudo mn» и задать предложенные параметры, которые поддерживает Mininet. Этот способ является быстрым, так как не нужно указывать IP-адреса хостов или контроллера. Главным минусом данного способа является соединение вышестоящих устройств с нижестоящими устройствами, поэтому перейдем к следующему способу. Следующий способ заключается в том, что топология будет написана на языке программирования Python, это позволит менять структурно топологию сети под свои нужны.

Далее был загружен контроллер Ryu. После чего, были загружены файлы для запуска алгоритма Дейкстры на контроллер Ryu и топология корпоративной сети согласно рисунку 4. Результат работы алгоритма Дейкстры с использованием контроллера Ryu показывает, что выбор происходит всех возможных путей от хоста «h1» до всех возможных хостов и выбран самый первый маршрут. (Пример показан на рисунке 5).

```
All the shortest from 10.0.0.1 to 10.0.0.6 are:
10.0.0.1 -> [2, 3] -> 10.0.0.6      delay: 418
Shortest path from 10.0.0.1 to 10.0.0.6is:
10.0.0.1 -> 2 -> 3 -> 10.0.0.6
All the shortest from 10.0.0.6 to 10.0.0.1 are:
10.0.0.6 -> [3, 2] -> 10.0.0.1      delay: 418
Shortest path from 10.0.0.6 to 10.0.0.1is:
10.0.0.6 -> 3 -> 2 -> 10.0.0.1
All the shortest from 10.0.0.1 to 10.0.0.5 are:
10.0.0.1 -> [2, 3] -> 10.0.0.5      delay: 418
Shortest path from 10.0.0.1 to 10.0.0.5is:
10.0.0.1 -> 2 -> 3 -> 10.0.0.5
All the shortest from 10.0.0.1 to 10.0.0.3 are:
10.0.0.1 -> [2, 1] -> 10.0.0.3      delay: 456
Shortest path from 10.0.0.1 to 10.0.0.3is:
10.0.0.1 -> 2 -> 1 -> 10.0.0.3
All the shortest from 10.0.0.3 to 10.0.0.1 are:
10.0.0.3 -> [1, 2] -> 10.0.0.1      delay: 456
Shortest path from 10.0.0.3 to 10.0.0.1is:
10.0.0.3 -> 1 -> 2 -> 10.0.0.1
All the shortest from 10.0.0.1 to 10.0.0.4 are:
10.0.0.1 -> [2, 1] -> 10.0.0.4      delay: 456
Shortest path from 10.0.0.1 to 10.0.0.4is:
10.0.0.1 -> 2 -> 1 -> 10.0.0.4
All the shortest from 10.0.0.1 to 10.0.0.2 are:
10.0.0.1 -> [2] -> 10.0.0.2      delay: 0
Shortest path from 10.0.0.1 to 10.0.0.2is:
10.0.0.1 -> 2 -> 10.0.0.2
All the shortest from 10.0.0.6 to 10.0.0.3 are:
10.0.0.6 -> [3, 1] -> 10.0.0.3      delay: 341
Shortest path from 10.0.0.6 to 10.0.0.3is:
10.0.0.6 -> 3 -> 1 -> 10.0.0.3
All the shortest from 10.0.0.3 to 10.0.0.6 are:
10.0.0.3 -> [1, 3] -> 10.0.0.6      delay: 341
Shortest path from 10.0.0.3 to 10.0.0.6is:
10.0.0.3 -> 1 -> 3 -> 10.0.0.6
All the shortest from 10.0.0.6 to 10.0.0.4 are:
10.0.0.6 -> [3, 1] -> 10.0.0.4      delay: 341
Shortest path from 10.0.0.6 to 10.0.0.4is:
10.0.0.6 -> 3 -> 1 -> 10.0.0.4
All the shortest from 10.0.0.6 to 10.0.0.2 are:
10.0.0.6 -> [3, 2] -> 10.0.0.2      delay: 418
Shortest path from 10.0.0.6 to 10.0.0.2is:
10.0.0.6 -> 3 -> 2 -> 10.0.0.2
All the shortest from 10.0.0.3 to 10.0.0.5 are:
10.0.0.3 -> [1, 3] -> 10.0.0.5      delay: 341
Shortest path from 10.0.0.3 to 10.0.0.5is:
10.0.0.3 -> 1 -> 3 -> 10.0.0.5
All the shortest from 10.0.0.3 to 10.0.0.2 are:
10.0.0.3 -> [1, 2] -> 10.0.0.2      delay: 456
Shortest path from 10.0.0.3 to 10.0.0.2is:
10.0.0.3 -> 1 -> 2 -> 10.0.0.2
```

Рисунок 5 – Результат работы алгоритма Дейкстры на контроллере Ryu

Далее был рассмотрен протокол Spanning Tree на контроллере Ryu. Spanning Tree Protocol (STP) используется для устранения петель в сети и это является одним из элементов надежности сети. Для проверки работоспособности контроллера с запущенным протоколом Spanning Tree, необходимо настроить топологию сети так, чтобы образовалась петля коммутации. В данном случае соединим коммутаторы «s2» и «s3» для того, чтобы образовалась петля коммутации. Процесс инициализации пересылки и блокировки порта показан на рисунке 6.

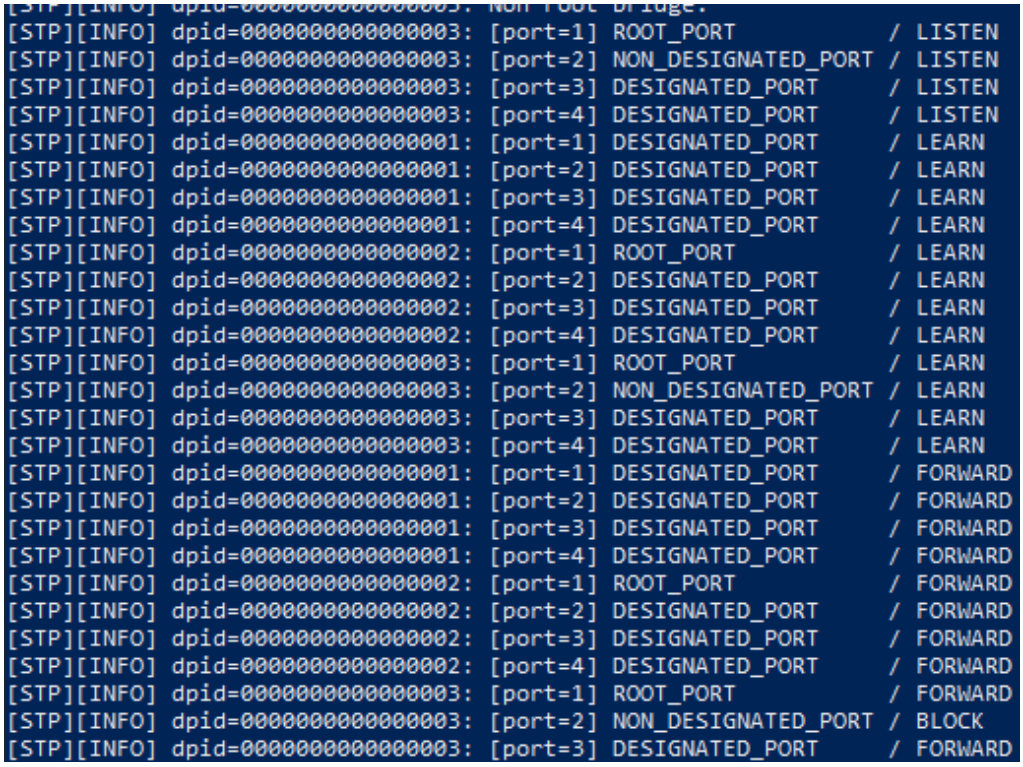

Рисунок 6 – Результат инициализации протокола Spanning Tree на контроллере Ryu

Далее был рассмотрен межсетевой экран на контроллере Ryu. Межсетевой экран (англ. Firewall) используется для защиты от несанкционированного доступа, вредоносных программ или для внутрисетевых политик безопасности. Пример добавления правил для межсетевого экрана показаны на рисунке 7.

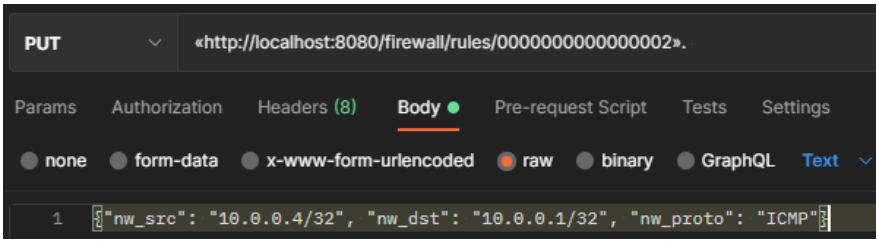

Рисунок 7 – Пример добавления правил для межсетевого экрана на контроллере Ryu

Изначально все хосты не могут общаться между собой, поэтому необходимо включить и добавить правила в межсетевой экран, разрешающие прохождение пакетов протокола ICMP. Необходимо прохождение пакетов ICMP между «h1» и «h4» на коммутаторе «s2» и «s1» соответственно. Для включения межсетевого экрана был создан запрос в инструменте Postman и в этом же запросе было создано правило, которое позволяет проходит пакетам ICMP между хостами «h1» и «h4».

После установки правил на межсетевой экране необходимо проверить достоверность этих правил. Проверить установленные правила можно двумя способами. Первый заключается в том, чтобы отправить запросы ICMP от всех хостов ко всем хостам, для этого напишем команду «pingall». Второй способ – это отправить запросы ICMP с хоста «h1» на хост «h4» с помощью команды «ping». Результаты успешных конфигураций представлено на рисунках 8 и 9.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X h4 X
h6 -> X X X X X
h5 -> X X X X X
h3 -> X X X X X
h4 -> h1 X X X X
h2 -> X X X X X
*** Results: 93% dropped (2/30 received)
mininet>
```

Рисунок 8 – Результат успешной конфигурации межсетевого экрана с помощью «pingall»

```
mininet> h1 ping h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=0.352 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.053 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.063 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.055 ms
```

. Рисунок 9 – Результат успешной конфигурации межсетевого экрана с помощью «ping»

Далее был установлен контроллер OpenDayLight и установлены компоненты оболочки и компоненты для пересылки пакетов между узлами.

Для включения графического интерфейса OpenDayLight, установим компоненты «odl-dlux». Эти компоненты содержат в себе, отображение созданной топологии, конфигурацию и отображение узлов сети, и дополнительные инструменты. Для коммутации и пересылки пакетов между хостами установим компоненты «odl-l2switch». Установка компонентов «odl-

dlux» и «odl-l2switch-all» происходит через команду «feature:install». Список необходимых компонентов представлен на рисунке 10.

```
opendaylight-user@root>feature:uninstall odl-
odl-dlux-core                  (Opendaylight dlux minimal feature)
odl-dluxapps-nodes          (ODL :: dluxapps :: odl-dluxapps-nodes)
odl-dluxapps-yangui        (ODL :: dluxapps :: odl-dluxapps-yangui)
odl-l2switch-switch          (OpenDaylight :: L2Switch :: Switch)
odl-dluxapps-topology (ODL :: dluxapps :: odl-dluxapps-topology)
odl-l2switch-arphandler (OpenDaylight :: L2Switch :: ArpHandler)
```

Рисунок 10 – Список компонентов OpenDayLight

Затем была запущенна ранее созданная топология в Mininet. Топология показана на рисунке 11.

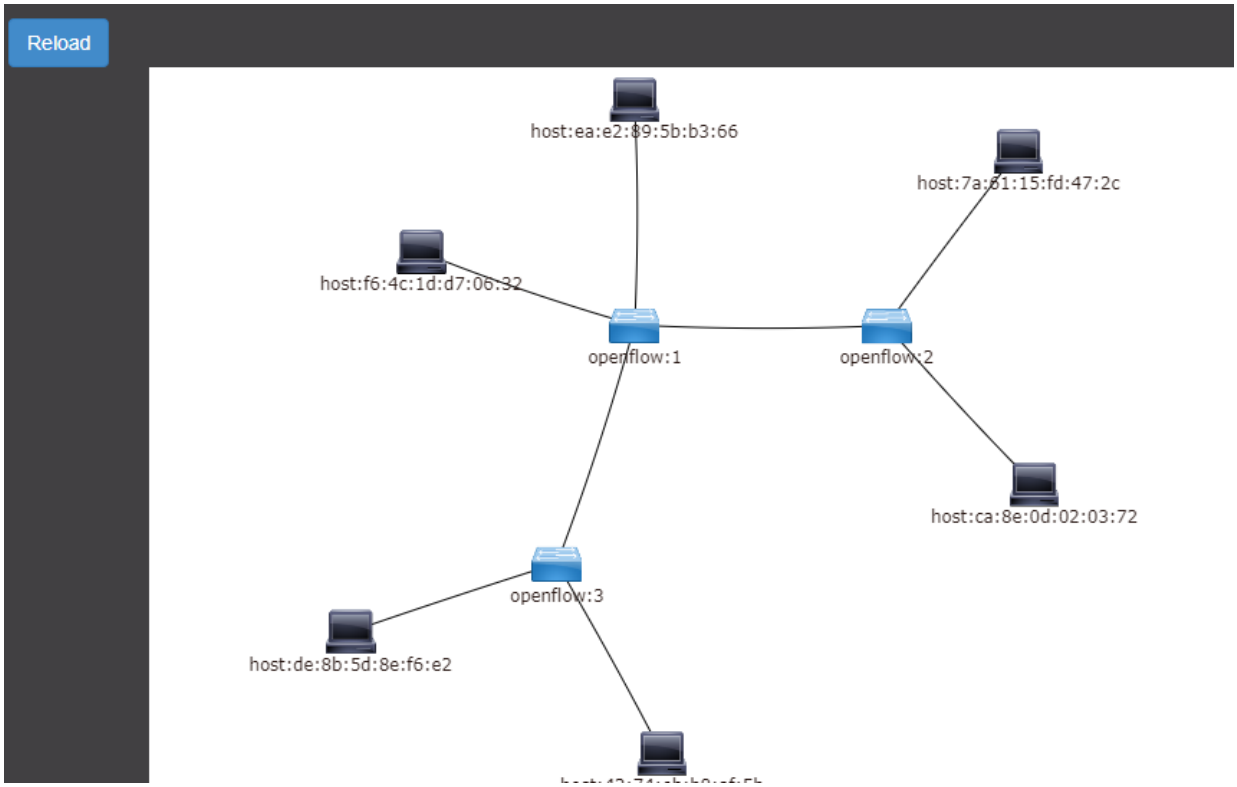

Рисунок 11 – Топология корпоративной сети на контроллер OpenDayLight

Для проверки работоспособности контроллера с запущенным протоколом Spanning Tree, необходимо настроить топологию сети так, чтобы образовалась петля коммутации. Соединим коммутаторы «s2» и «s3» как и в случае с контроллером Ryu. Преимущество контроллера OpenDayLight – это встроенный протокол Spanning Tree, который сразу блокирует порт согласно протоколу Spanning Tree. Топология с использованием протокола Spanning Tree изображена на рисунке 12.

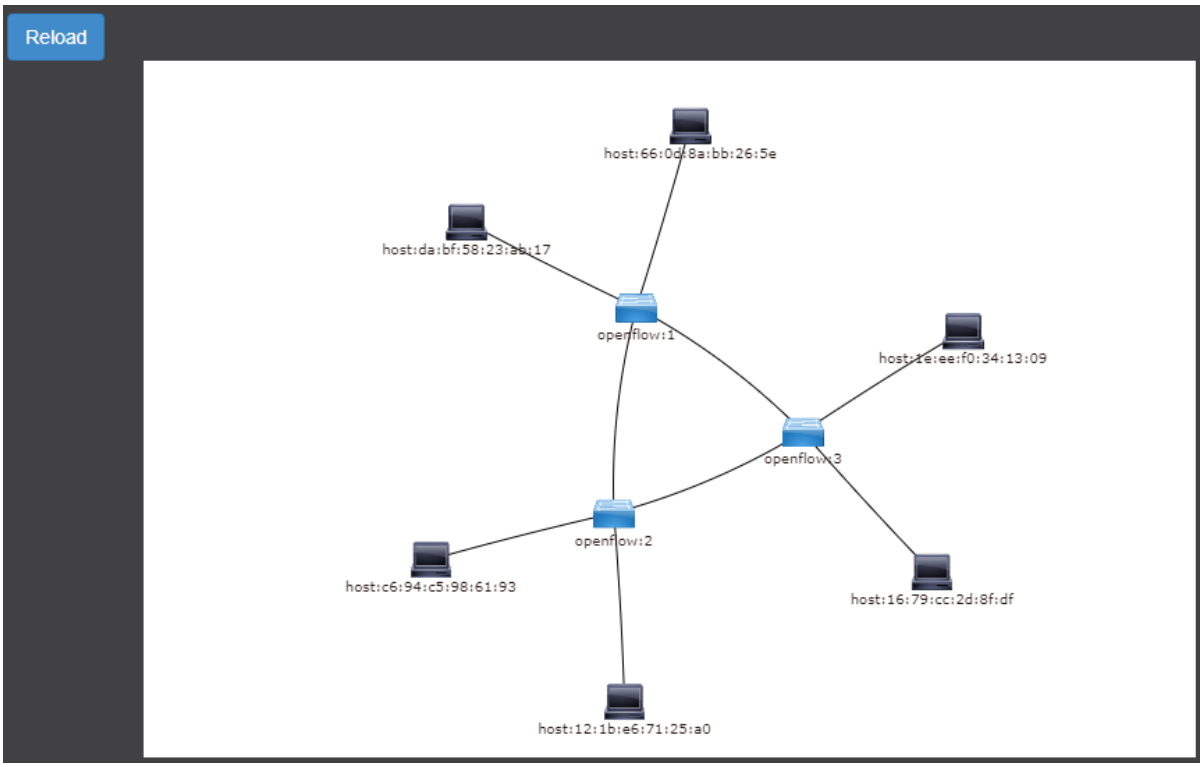

Рисунок 12 – Топология с использованием протокола Spanning Tree

Далее в ранее установленном компоненте «YangUI» узнали какой порт на каком коммутаторе заблокирован. В ходе исследования был сделан вывод, что недостаток данного контроллера с использованием протокола Spanning Tree – это сложность нахождения, какой порт заблокирован. На рисунках 13 и 14 изображены информация о коммутаторах «s2» и «s3». Надпись «discarding» в строке «status» обозначает, что порт отключен согласно протоколу Spanning Tree.

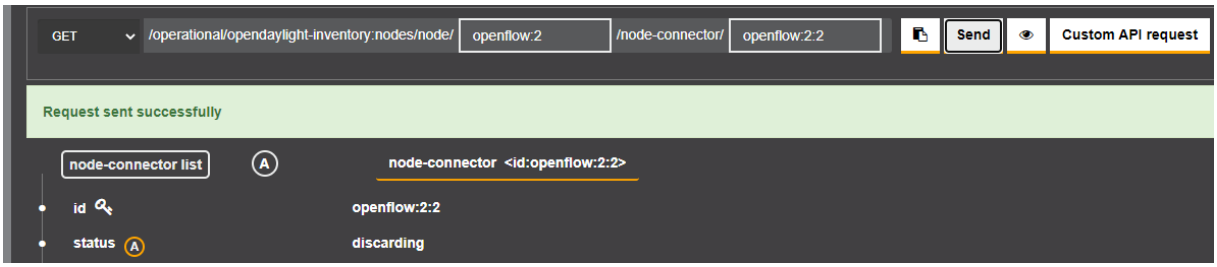

Рисунок 13 – Результат работы протокола Spanning Tree на коммутаторе «s2» на контроллере OpenDayLight

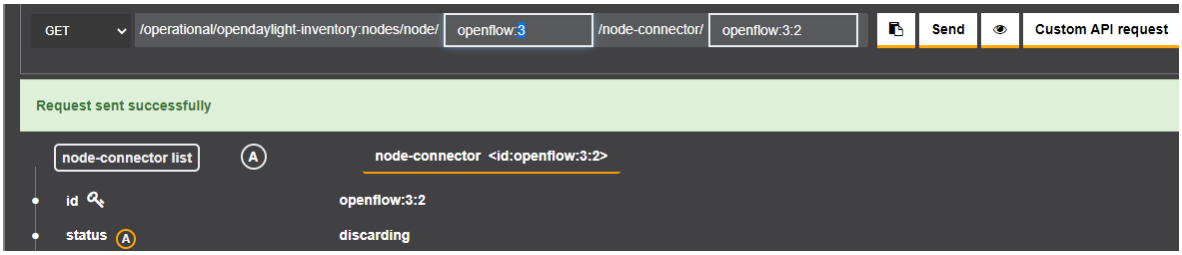

Рисунок 14 – Результат работы протокола Spanning Tree на коммутаторе «s3» на контроллере OpenDayLight

Таким образом, в главе было произведено моделирование программно-конфигурируемых сетей с использованием контроллера Ryu и OpenDayLight, а также смоделирована топология древовидной сети с помощью эмулятора Mininet. На контроллере Ryu показан алгоритм работы Дейкстры, настроен протокол устранения петель в сети под названием Spanning Tree, а также произведена настройка межсетевого экрана. На контроллере OpenDayLight реализована работа протокола Spanning Tree. OpenDayLight имеет множество возможностей использования в виде контроллера программно-конфигурируемых сетей и требуют дальнейшего исследования. На данный момент, метод повышения надежности с использованием контроллера Ryu является более эффективным по сравнению с другими контроллерами.

ЗАКЛЮЧЕНИЕ

В ходе выполнения магистерской диссертации была достигнута цель – проведён анализ методов и алгоритмов обеспечения показателей надежности программно-конфигурируемых сетей. В ходе выполнения работы был поставлен и решен ряд задач.

Первой задачей является анализ публикаций по теме исследования. На основе обзора публикаций проведен анализ состояния области исследования.

В ходе обзора публикаций рассматривались показатели надежности как в программно-конфигурируемых сетях, так и в традиционных телекоммуникационных сетях. В результате анализа, был сделан вывод, что сеть с использованием виртуализации является более надежной и гибкой в управлении, чем традиционная телекоммуникационная сеть. Поэтому, использование SDN является актуальным в настоящие время, так как виртуализация является более эффективным методом обеспечения надежности и качества предоставления услуг, освобождает количество затрачиваемых ресурсов у пользователя.

Следующей задачей был анализ показателей надежности в телекоммуникационных сетях. Надежностью является вероятностью того, что элемент, вещь или услуга будут адекватно выполнять предназначенную функцию в течение определенного периода времени или будут работать в определенной среде без сбоев. На основе анализа и классификации был сделан вывод, что надежность сети измеряет, в какой степени сеть может оставаться в рабочем состоянии при минимальном уровне требований при постоянной работе.

Третьей задачей было описание методов повышения надежности в телекоммуникационных сетях. На основе изучения, был сделан вывод, что для повышения надежности телекоммуникационной сети, в настоящее время актуально использовать виртуализацию, так как существует возможность создать работоспособную копию телекоммуникационной сети, которая будет соблюдать все характеристики надежности.

Четвертой задачей рассматривалось сравнение надежности телекоммуникационных и виртуальных сетей. При сравнении был сделан вывод, что виртуализация по сравнению с традиционной сетью имеет преимущества в виде улучшенного взаимодействия с аппаратным оборудованием. Также виртуализация позволяет пользователям применять программное обеспечение для подготовки новых устройств и позволяет ИТ-администраторам строить сетевые узлы и сетевые службы.

Пятой задачей было описание архитектуры программно-конфигурируемых сетей и её инструментов. Было рассмотрена архитектура программно-конфигурируемой сети, виртуальный централизованный контроллер SDN, протокол OpenFlow, инструмент для моделирования сети Mininet. Так же был сделан вывод, что, древовидная топология обеспечивает наиболее высокую надежность системы передачи и качества предоставления услуг.

В шестой задаче было произведено исследование алгоритмов обеспечения надежности в программно-конфигурируемых сетях. При исследовании был сделан вывод, что алгоритм Дейкстра является более надежным так как, работает в сетях любой сложности и является более надежным в случае высокой нагрузки на сети и равномерно распределяет трафик.

В рамках решения седьмой задачи было произведено исследование методов обеспечения надежности в программно-конфигурируемых сетях. В ходе исследования, был выбран контроллер Ryu, потому что, имеется обширная библиотека приложений и сервисов, возможность реализовывать сервисы с помощью языка программирования Python, а также поддерживает протокол OpenFlow последний версии.

При выполнении восьмой задачи было произведено моделирование программно-конфигурируемых сетей с использованием контроллера. В ходе моделирования на контроллере Ryu показан алгоритм работы Дейкстры, настроен протокол устранения петель в сети и так же произведена настройка межсетевого экрана. На контроллере OpenDayLight реализована работа протокола Spanning Tree. OpenDayLight имеет множество возможностей использования в виде контроллера программно-конфигурируемых сетей и требуют дальнейшего исследования. На данный момент, метод повышения надежности с использованием контроллера Ryu является более эффективным по сравнению с другими контроллерами.

Таким образом, исследование методов и алгоритмов показателей надежности в программно-конфигурируемых сетях является немаловажным фактором повышения уровня знаний студентов. По результатам исследования предложено внедрение в учебный процесс практической работы «Моделирование программно-конфигурируемых сетей с использованием контроллеров» по дисциплине «Теория построение телекоммуникационных сетей и систем».

Основные теоретические и практические результаты работы.

1) Рассмотрены и выявлены основные показатели надежности в телекоммуникационных сетях и методы их повышения.

2) Рассмотрены основные положения программно-конфигурируемых сетей и её инструментов.

3) Проведен анализ методов и алгоритмов обеспечения надежности в программно-конфигурируемых сетях.

4) Разработана модель программно-конфигурируемых сетей с использованием контроллеров.

СПИСОК РАБОТ, ОПУБЛИКОВАННЫХ АВТОРОМ ПО ТЕМЕ ДИССЕРТАЦИИ

1 Тарасов, В.С. Роль программно-конфигурируемых сетей в развитии мобильной связи / В.С. Тарасов, К.И. Брагин, Н.В. Будылдина // Инфокоммуникационные технологии: актуальные вопросы цифровой экономики: Сборник научных трудов I Международной научно-практической конференции, Екатеринбург, 17-18 февраля 2021 года. – Екатеринбург: Сибирский государственный университет телекоммуникаций и информатики, 2021. – С. 118-121.