Санкт-Петербургский государственный университет
Математическое обеспечение и администрирование информационных систем

Кафедра Системного Программирования

Деркунский Виктор Артурович

Реализация SDN/NFV

Дипломная работа

Научный руководитель:
ст. преп. Зеленчук И. В.

Рецензент:
преп. Ханов А. Р.

Санкт-Петербург
2019

Оглавление

1. Введение

С середины прошлого века архитектура компьютерных сетей претерпела не так много изменений. В последнее время наблюдается стремительный рост развития информационных технологий, что выразилось в появлении крупных дата-центров и облачных структур.
В настоящее время при настройке компьютерных сетей необходимо работать с каждым сетевым устройством индивидуально, при этом для настройки используется обширный спектр протоколов.
Программно-конфигурируемые сети (ПКС) — это новый взгляд на архитектуру сети, где основополагающими правилами являются:

1. Централизованное управление сетевыми устройствами.

2. Упрощение функциональности сетевых устройств, заключающееся в том, что на них остается только задача физической передачи трафика.

Рассматривая случай сети, имеющей не менее 10000 узлов, провести настройку сети является достаточно сложной задачей, практически невыполнимой без какой-либо оптимизации процесса.
Почти каждая крупная IT компания имеет собственное решений данной задачи. Однако это приводит к негативным последствиям, таким как зависимость от производителя сетевого оборудования и невозможность использования дополнительного оборудования другого бренда, поскольку зачастую они оказываются несовместимы.
В связи с тем, что все устройства, входящие в ПКС-сеть, являются программируемыми, то появляется возможность вынести функцию управления сетью на выделенное устройство, называемое контроллером, который в свою очередь общается с другими устройствами при помощи специального стандартизованного протокола OpenFlow. Таким образом, при помощи контроллера появляется возможность эффективно решать следующие задачи:

1. Добавление и настройка нового коммутатора.

2. Анализ поведения трафика.

3. Изменение конфигурации системы.

Более того, общие затраты на оборудование сети снижаются, что связано с рядом причин, одной из которых является упрощение функциональности сетевых устройств.

2. Постановка задачи

Целью работы является создание SDN стенда, демонстрирующего возможности данной архитектуры. Для её достижения были поставлены следующие задачи:

- Изучение предметной области.
- Моделирование сети с помощью эмулятора сети mininet.
- Реализация тестового стенда на свитчах Zodiac GX.
- Проведение ряда экспериментов на реализованном стенде.

3. Программно-конфигурируемые сети

3.1. Необходимые понятия

Для более четкого понимания описания архитектуры ПКС-сетей, а также протокола OpenFlow, вводится ряд понятий, облегчающих понимание.

- Пакет данных (Packet) — кадр Ethernet, который состоит из заголовка и полезной нагрузки.
- Конвейер (Pipeline) — совокупность таблиц, отвечающих за проверку заголовка, а также передачу и изменение пакетов в OpenFlow-коммутаторе.
- Порт OpenFlow представляет собой интерфейс, через который производится переадресация данных логическим переключателем OpenFlow. Порт Openflow может соответствовать физическому порту физического или виртуального переключателя.
- Поле проверки (Match Field) — это рассматриваемая часть пакета, включающая в себя его заголовок, входной порт и значение метаданных.
- Метаданные (Metadata) — это маскируемое значение, используемое для отправки пакета от одной таблицы к другой.
- Инструкция (Instruction) — некоторая операция, которая либо добавляется в набор действий (Action set), либо состоит из списка действий немедленно применяемых к потоку. Инструкция может изменять процесс обработки пакетов на конвейере.
- Набор действий (Action set) — это накапливаемые действия в момент прохождения пакета через группу таблиц, которые выполняются над пакетом после его выхода из конвейера.
- Контейнер действий (Action Bucket) — это множество действий и относящиеся к ним параметры, определённые для группы пакетов.

3.2. Архитектура ПКС-сетей

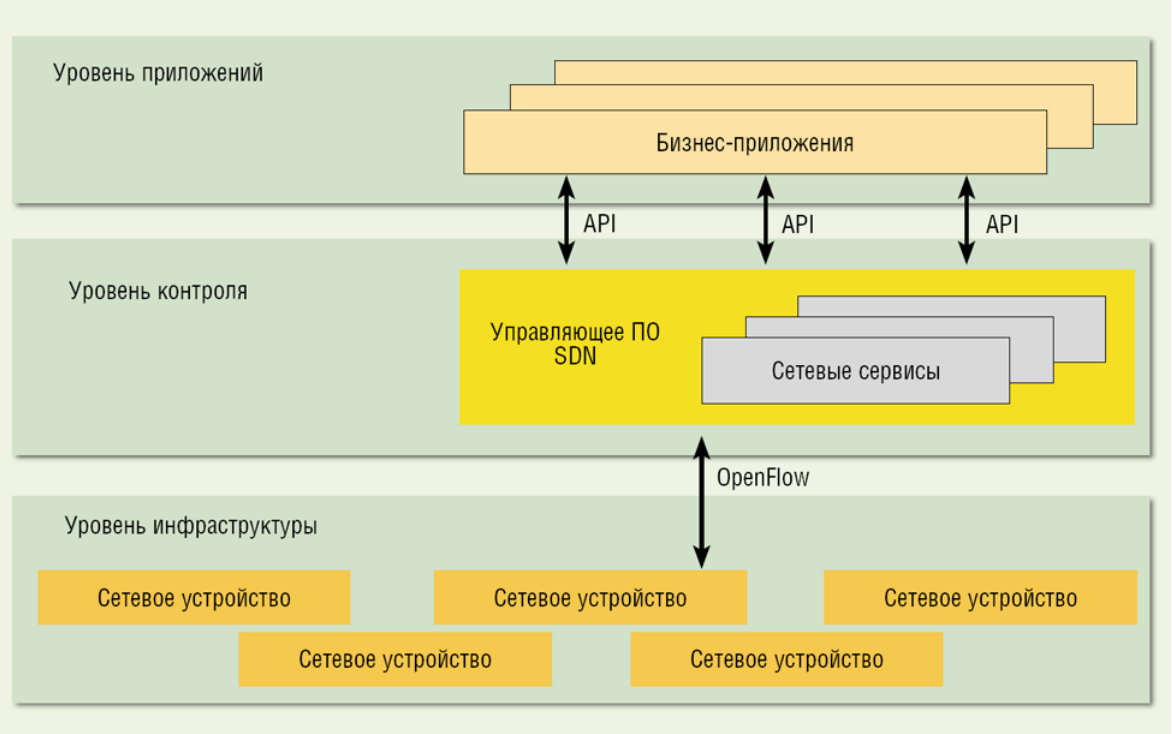

Рис. 1: https://www.osp.ru/iz/rusnet/articles/13033012 (запрос 20.12.2018 14:52)

В работе [17] авторы говорят о том, что традиционные сети в скором времени станут тяжело управляемыми и конфигурируемыми, так как достигнут слишком больших размеров. Все эти проблемы предлагаются решать с помощью использования технологии Software-Defined Networks. Также в статье предполагают, что SDN — это будущее компьютерных сетей.

Суть SDN состоит в централизованном управлении сетью. Данный подход помогает эффективно обрабатывать крупные потоки данных, а также упрощает контроль и настройку большого количества сетевого оборудования.

Идеи SDN:

1. Разделение процессов передачи данных и управление ими.
2. Консолидация управления сетью.
3. Виртуализация физических ресурсов.

4. Единый протокол OpenFlow для взаимодействия плоскостей управления и передачи данных.

Архитектура SDN логически состоит из 3 уровней (рис. 1):

1. Уровень приложений — функции управления сетью, такие как мониторинг трафика, безопасность, управление потоками данных и другие.

2. Уровень контроля — отслеживание топологии сети, API для сетевых приложений.

3. Уровень инфраструктуры — сетевые устройства и каналы передачи данных.

3.3. Протокол OpenFlow для взаимодействия контроллера с сетевыми устройствами

OpenFlow - это один из протоколов взаимодействия, а ПКС - это целая архитектура. На рисунке 2 отображено взаимодействие контроллера ПКС с сетвыми узлами.

OpenFlow — это открытый протокол, главная цель которого в консолидации управления оборудованием различных производителей.
Openflow с момента появления постоянно разивается и усовершенствуется, то есть добавляется функциональность, исправляются ошибки прошлых стандартов, мешающие получить полную выгоду от использования ПКС-сетей.

3.3.1. OpenFlow v1.0

В первой версии, вышедшей в 2009 году, выпущенной компанией Open Networking Foundation, поток определялся как множество, состоящее из пакетов, которые можно описать определенным набором полей. Все записи о потоках находились в таблице потоков. В первое время она состояла из поля заголовка (header fields), счетчиков (counter) и действий (actions). В заголовке располагались поля, по которым можно

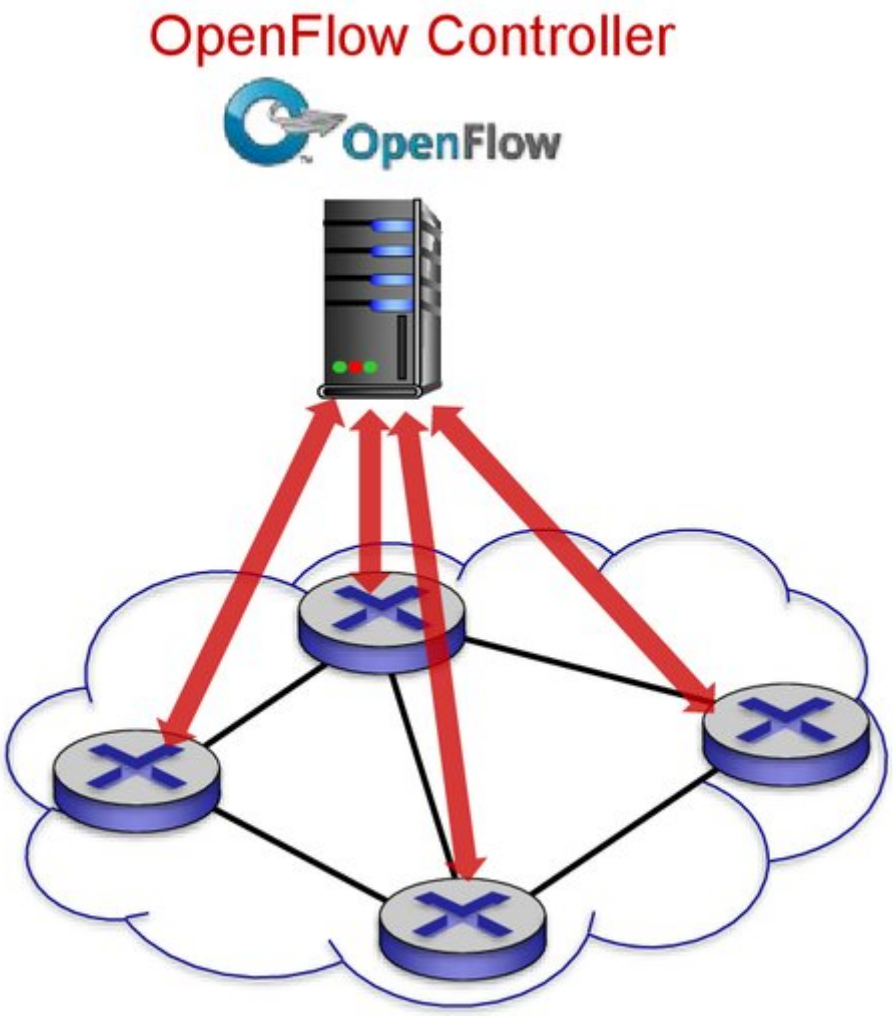

Рис. 2: https://en.ppt-online.org/346708 (запрос 20.12.2018 14:52)

назначить поток. Первая версия описывала 12 параметров, из которых складывался заголовок. Их иллюстрация приводится в таблице 1.

Когда происходит совпадение полей, то увеличивается соответвующее значение счетчика. Потоки, как правило, затрагивают несколько полей в таблице потоков. Счетчики определяются как для таблицы (per table), так и для потока (per flow), порта (per port) или очереди (per queue). Все виды счетчиков отражены в таблицах 2-5.

Поле	Размер, бит	Тип
Ingress Port	32	все пакеты
Ethernet Source	48	на активных портах
Ethernet Destination	48	на активных портах
Ether type	16	на активных портах
VLAN id	12	Все пакеты с Ethernet type 0x8100
VLAN priority	3	Все пакеты с Ethernet type 0x8100
IP Source	32	Все IP и ARP пакеты
IP Destination	32	Все IP и ARP пакеты
IP Protocol	8	Все IP и ARP пакеты
IP Tos bits	6	Все IP пакеты
TCP/UDP source port	16	Все TCP, UDP, ICMP
TCP/UDP dest. port	16	Все TCP, UDP, ICMP

Таблица 1:

Кроме этого существуют поля действий, и когда поля совпадают, то выполняются соответвующие действия. Эти действия (actions) делятся на две группы — обязательные (required) и опционные (optional). Обязательные действия выполняют следующие функции:

1. Передачи пакетов на физические или виртуальные порты.
   - All — происходит посылка пакета со всех портов за исключением того, на который пакет пришел.
   - Local — посылка пакета на локальный порт.
   - Controller — посылка пакета на контроллер.
2. Сбрасывание пакета при отсутствии каких-либо применяемых действий.

Счетчик	бит
Reference Count	32
Packet Lookups	64
Packet Matches	64

Таблица 2: Счетчики по таблицам

Счетчик	бит
Received Packets	64
Received Bytes	64
Duration (seconds)	32
Duration (nanoseconds)	32

Таблица 3: Счетчики по потокам

Счетчик	бит
Received Packets	64
Transmitted Packets	64
Received Bytes	64
Transmitted Bytes	64
Receive Drops	64
Transmit Drops	64
Receive Errors	64
Transmit Errors	64
Receive CRC Errors	64
Collisions	64

Таблица 4: Счетчики по портам

Счетчик	бит
Transmit Packets	64
Transmit Bytes	64
Transmit Overrun Errors	64

Таблица 5: Счетчики по очереди

К опционным действиям относятся:

1. Посылка пакетов на виртуальные порты.
2. Enqueue — отправка пакета через очередь на порт для обеспечения обслуживания QoS.
3. Modify Field — позволяет выполнять большое множество действий, отображенных в таблице 6.

Действия	Данные, бит	Описание
Set VLAN ID	12	Добавляет новый заголовок, содержащий VLAN ID и приоритет 0 или замещает существующий заголовок с VLAN ID
Set VLAN priority	3	Добавляет новый заголовок, содержащий значение поля приоритет и VLAN ID 0 или замещает в существующем заголовке поле приоритета
Strip VLAN header	-	Отбрасывает заголовок VLAN
Modify Ethernet source MAC address	48	Замещает существующий MAC-адрес источника на указанное значение
Modify Ethernet destination MAC address	48	Замещает существующий MAC-адрес получателя на указанное значение
Modify IPv4 source address	32	Заменяет IPv4-адрес источника новым значением и обновляет контрольную
Modify IPv4 destination address	32	Заменяет IPv4-адрес получателя новым значением и обновляет контрольную сумму
Modify IPv4 ToS bits	6	Заменяет значение в поле ToS, только для IPv4
Modify TCP/UDP source port	16	Заменяет TCP/UDP-порт источника новым значением и обновляет контрольную сумму
Modify TCP/UDP destination port	16	Заменяет TCP/UDP-порт получателя новым значением и обновляет контрольную сумму

Таблица 6: Действия по модификации полей в протоколе OpenFlow

Производители стали активно использовать протокол OpenFLOW версии 1.0. Но при эксплуатации нашлись недостатки, которые не давали использовать возможности ПКС в полной мере. Например, над пакетом возможно было провести только одно действие, при этом память TCAM в недорогих коммутаторах была рассчитана на 1-2 тыс. записей, что являлось очень сильным ограничением. Все это привело к усовершенствованию протокола и появлению его новых версий.

3.3.2. Эволюция протокола OpenFlow v1 - v5

В каждой последующей версии происходило расширение функциональности. Ключевые изменение показаны в таблице 7.

OpenFlow 1.0	Одна таблица содержит 12 полей
OpenFlow 1.1	Multi-table, group table, поддержка VLAN, MPLS
OpenFlow 1.2	Поддержка IPv6, увеличение полей match, возможность смены роли контроллера в кластере
OpenFlow 1.3	Meters table для обеспечения Qos, увеличение гибкости обработки пакетов, для которых не найдено совпадений
OpenFlow 1.4	Оптические порты, мониторинг потоков, bundle-группировка связанных изменений состояний
OpenFlow 1.5	Scheduled bundle - вводит время выполнения

Таблица 7: Эволюция протокола OpenFlow

Интересный факт состоит в том, что, несмотря на то, что каждая следующая версия содержала немаловажные изменения, версии 1.1 и 1.2 не были поддержаны производителями. В них были такие новинки как Group-table, дающая возможность провести над пакетом несколько действий, поддержка VLAN и MPLS [5], а также поддержка IPv6 и

возможность смены роли контроллера в кластере [6]. Только при появлении версии 1.3 производители заинтересовались данным стандартом. Согласно этой версии Meters table расширяли функционал Qos [7]. После этого в версии 1.4 стала возможна предварительная проверка сообщений при отправке на множество коммутаторов [8]. В последней версии число полей уже увеличилось почти в 2 раза по сравнению с версией 1.0, были добавлены такие полезные поля, как приоритет, время жизни и cookie [9].
В каждой из версий была возможность отправки сообщений контроллер-коммутатор, которые начинались на контроллере для управления коммутатором, установки на нем нужной конфигурации, сбора статистики и другое.

3.4. OpenFlow-коммутатор

Рассмотрим коммутатор с поддержкой OpenFlow версии 1.3 [20], так как эта версия наиболее распространенная и поддерживаемая производителями. Основные компоненты отражены на рисунке 3.

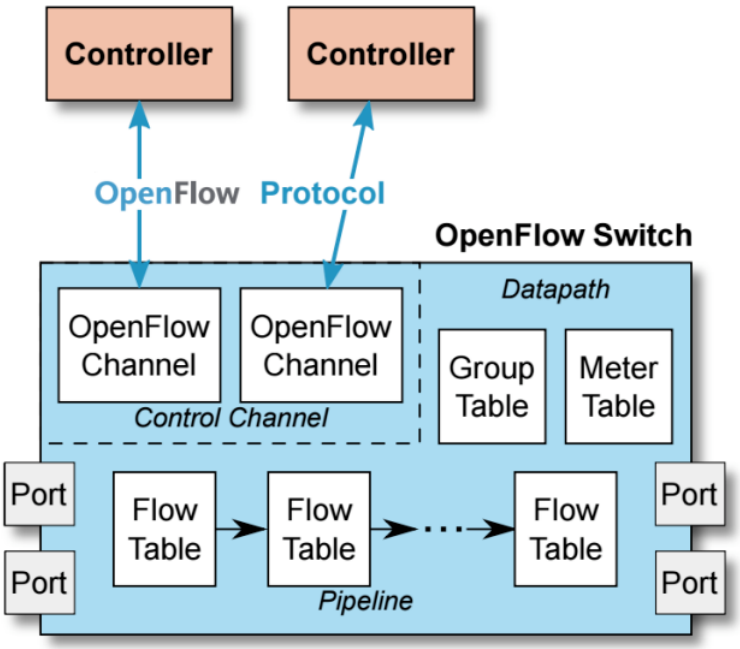

Рис. 3: Структура маршрутизатоа Openflow

Openflow коммутатор состоит из одной или нескольких таблиц потоков (Flow table), групповой таблицы (Group-table), таблицы метрик, портов, а также OpenFlow каналов, которые обеспечивают подключение к ПКС-контроллеру и получение от него управляющих команд и данных [3]. Порты, таблицы потоков, групп, метрик являются элементами data plane. Над таблицей потоков (Flow entries) по протоколу OpenFlow можно совершать ряд действий: удалить запись, обновить запись или добавить новую запись. Каждая запись потока состоит из полей соответствия, счетчиков и действий или инструкций по обработке пакета.

Также есть возможность использовать широковещательную рассылку, используя Group-table. Пакеты в этом случае отправляются на дополнительную обработку в группу. Это позволяет более эффективно получать выходные действия записей. Таблица группы состоит из записей группы, при этом каждая группа состоит из множества действий с определенным описанием, зависящим от ее типа. Когда пакет приходит на обработку в группу, он обрабатывается действиями из одного или нескольких множеств.

Коммутаторы с поддержкой Openflow делятся на 2 типа: OpenFlow-only и OpenFlow-hybrid. В первом случае происходит только конвейерная обработка по протоколу OpenFlow, все другие при этом не поддерживаются. Второй тип поддерживает также классическую обработку коммутатора Ethernet.

4. Обзор SDN-контроллеров

	Язык Прог.	Архитектура	Northbound API	Southbound API	Платформы
Beacon	Java	Centralized	ad-hoc	OpenFlow 1.0	Linux, MacOS, Windows
Floodlight	Java	Centralized	REST, Java RPC, Quantum	OpenFlow 1.0, 1.3	Linux, MacOS, Windows
Maestro	Java	Centralized	ad-hoc	OpenFlow 1.0	Linux, MacOS, Windows
NOX	C++	Centralized	ad-hoc	OpenFlow 1.0	Linux
Onix	C++	Distributed Flat	Onix API	OpenFlow 1.0	-
ONOS	Java	Distributed Flat	REST, Neutron	OpenFlow 1.0, 1.3	Linux, MacOS, Windows
OpenDaylight	Java	Distributed Flat	REST, RESTCONF, XMPP, NETCONF	OpenFlow 1.0, 1.3	Linux, MacOS, Windows
OpenIRIS	Java	Distributed Flat	REST	OpenFlow 1.0-1.3	Linux
OpenMul	C	Centralized	REST	OpenFlow 1.0, 1.3,	Linux
POX	Python	Centralized	ad-hoc	OpenFlow 1.0	Linux, MacOS, Windows
RunOS	C++	Distributed Flat	REST	OpenFlow 1.3	Linux
Ryu	Python	Centralized	REST	OpenFlow 1.0-1.5	Linux, MacOS

Таблица 8: Особенности контроллера 1) — 5)

Контроллер ПКС выполняет управление сетью, опираясь на полученные правила, поэтому ялвяется наиважнейшим местом программно-конфигурируемых сетей [2]. Кроме этого контроллер управляет рядом других специализированных приложений, например, служб, эмулирующих работу протоколов маршрутизации. В итоге результат работы контроллера пересылается по протоколу OpenFlow в виде множества правил на Flow-таблицы, которые, в свою очередь, осуществляют передачу трафика. Контроллер увеличивает гибкость управления и упро-

	Интерфейс	Лицензия	Многопоточность	Модульность	Документация
Beacon	CLI, Web UI	GPL 2.0	+	+	-
Floodlight	CLI, Web UI	Apache 2.0	+	+	+
Maestro	Web UI	LGPL 2.1	+	+	-
NOX	CLI, Web UI	GPL 3.0	+	+	-
Onix	-	Proprietary	+	+	-
ONOS	CLI, Web UI	Apache 2.0	+	+	+
OpenDaylight	CLI, Web UI	EPL 1.0	+	+	+
OpenIRIS	CLI, Web UI	Apache 2.0	+	+	+
OpenMul	CLI	GPL 2.0	+	+	+
POX	CLI, GUI	Apache 2.0	+	+	-
RunOS	CLI, Web UI	Apache 2.0	+	+	-
Ryu	CLI	Apache 2.0	+	+	+

Таблица 9: Особенности контроллера 6) — 10)

щает процесс администрирования.

Для выявления наиболее подходящего контроллера были выбраны следующие: Beacon, Floodlight, Maestro, NOX, Onix, ONOS, OpenDayLight, OpenIris, OpenMul, POX, RunOS, Ryu. Они все обладают открытым API (Application programming interface). Они все являются расширяемыми и с их помощью ведется активная разработка ПКС. Для сравнения были выбраны следующие критерии:

1. Язык программирования.
2. Архитектура.
3. Поддерживаемые ”северные” протоколы.

4. Поддерживаемые "южные" протоколы.
5. Поддерживаемые платформы.
6. Наличие графического или сетевого интерфейса для администрирования.
7. Лицензия, определяющая правила использования.
8. Поддержка многопоточности.
9. Модульность.
10. Наличие подробной документации.

опираясь на работу [1], были составлены таблицы 8 и 9, показывающие обзор характеристик контроллеров. Наиболее подходящие контроллеры должны быть кроссплатформенными, иметь лицензию, позволяющую свободное использование кода, интерфейс для администрирования, поддержку многопоточности, высокую модульность и обширную документацию для ознакомления. По выбранным критериям подходит целый ряд контрллеров.

В работе [11] методологией cbench (наиболее популярный инстурмент тестирования ПКС-контроллеров) обозреваются контроллеры ONOS, POX, NOX, OpenMul, Beacon, Maestro, Ryu и OpenIris. RunOS не поддерживает инстурмент проверки cbench, поэтому он не рассматривался в этом анализе. Также из-за того, что OpenDaylight имел проблемы при тестировании, использовался другой иструмент Wcbench, который является расширенной версией cbench.

В этой статье проводился ряд экспериментов по изменению эффективности работы контроллера при увеличении количества коммутаторов, а также при изменении количества потоков. Результаты заключались в том, что лучшую производительность показали контроллеры, написанные на языке C, например OpenMul. Немного хуже себя показали контроллеры, написанные на java: Beacon, Iris и Maestro.

Ещё одно замечание от авторов этой статьи заключается в том, что контроллеры, написанные на java и C, повышают производительность при

увеличения числа нитей, тогда как контроллеры, которые используют Python, почти не реагируют на это.
Однако авторами утверждается, что нельзя ориентироваться только на то, как показывают себя контроллеры на их тестах. Они выделяют OpenDaylight и ONOS в связи с их высокой модульностью, которая способствует эффективной разработке приложений. По их мнению данные контроллеры будут широко использоваться.
В работе [1] также проводится тестирование более двадцати SDN-контроллеров, используя тестирование cbench, после которого предлагают для рассмотрения два контроллера: OpenDayLight и ONOS. Авторы уделяют внимание их кроссплатформенности и самой высокой модульности среди всех тестируемых контроллеров.
Также благодаря языку, на котором они были написаны, есть возможность использовать контейнер OSGI, который позволяет догружать пакеты прямо во время выполнения. Это является ключевым фактором для выбора контроллера OpenDayLight в статье [13]. Отмечается хороший графический интерфейс для администрирования.
Контроллеры OpenDayLight и ONOS являются самыми популярными по мнению авторов контроллерами, поэтому они находятся в партнерстве с большим числом сетевых провайдеров, что помогает им иметь стратегию развития и подробную расширяемую документацию.
Распределенная схема делает возможным развертывание SDN в реальном масштабе. OpenDayLight выделяется отдельно как обладатель наибольшего числа различных северных и южных протоколов, что облегчает его использование. Выбор между этими двумя контроллерами авторы предлагают осуществлять в зависимости от поставленных задач.
Более подробно эти контроллеры тестируются в работе [15], где строятся кластеры на их основе и обозначаются ряд тенденций:

- Размер кластера влияет на скорость установки потока.
- Время отработки отказа контроллера зависит от количества устройств.
- Задержка оказывает значительное влияние на крупномасштабную WAN.

Однако в работе [14], где авторы используют для тестирования The Analytic Hierarchy Process (AHP), суть которого в автоматическом выборе контроллера исходя из выбранных критериев. Данный процесс иллюстрирован на рисунке 4. В результате тестирования авторы ставят OpenDayLight лишь на 3 место после Ryu и Floodlight. Рассматривае-

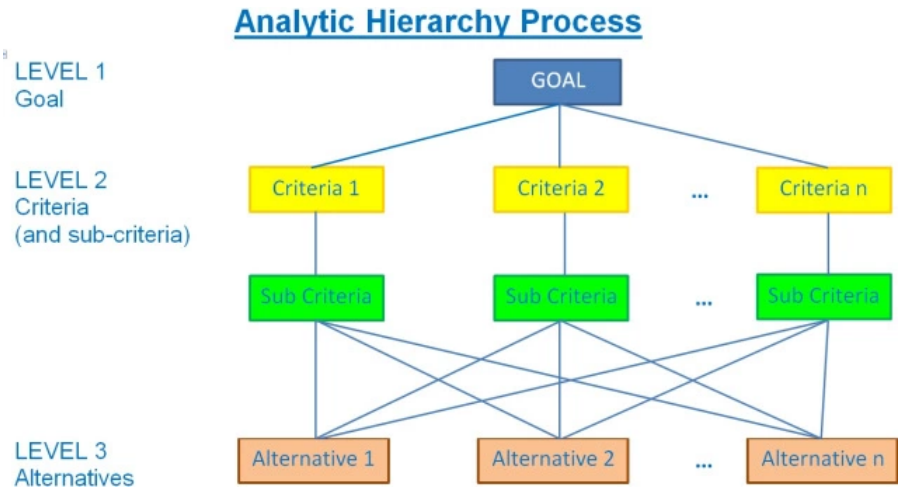

Рис. 4: The Analytic Hierarchy Process

мые критерии: наличие подробной документации, удобный интерфейс, модульность, производительность и ряд других. В результате работы алгоритма составлялась матрица весов, по которой каждому контроллеру выставлялась окончательная оценка. Стоит отметить, что все три первых места не сильно отличаются друг от друга, тогда как оценки POX и Trema намного ниже остальных.

Большая часть рассмотренных работ рекомендует к выбору OpenDayLight и даже в тех случаях, когда он не показывает лучшие результаты при тестировании, его выделяют как наиболее перспективный. Данная платформа обладает поддержкой наибольшего числа провайдеров и имеет большее количество коммерческих применений. Также обладает модульностью, удобным графическим интерфейсом и кроссплатформенностью и рядом других ценных качеств. Именно поэтому для дальнейшей работы был выбран OpenDayLight.

4.1. Более подробно про OpenDayLight

OpenDaylight (ODL) — это открытая SDN платформа, которая обеспечивает сетевые сервисы, позволяющие пользователю управлять приложениями, протоколами и плагинами, обеспечивающими связь между провайдерами и потребителями.
Это полностью открытая платформа, процесс разработки открыт для всех. Новый релиз выходит каждые 6 месяцев, каждый раз исправляя часть прошлых ошибок и добавляя новые интерфейсы взаимодействия с большим количеством SDN приложений.
Политика проекта:

- Максимальный набор поддерживаемых протоколов — от NETCONF до OpenFlow.
- Открытая модель для разработки новых приложений.
- Сильный акцент на безопасность, масштабируемость, стабильность.

4.2. Коммерческие контроллеры на основе ODL

В работе [16] говорится, что больше трех четвертей опрошенных компаний предпочитают коммерческие версии open-source
ПКС-контроллеров, так как они объединяют как положительные стороны ПО с открытым кодом, так и реализуют техническую поддержку продукта от компании.
В статье [15] приведены компании, использовавшие OpenDayLight в качестве основы для своих коммерческих проектов. Опираясь на данную статью, была составлена таблица 10. В данной таблице все производители попадают в одну из трех колонок:

1. Участники OpenDayLight Foundation — это те, которые поддерживают проект будучи спонсорами и имеют свои контроллеры на основе OpenDayLight.
2. Лояльные — это компании, поддерживающие OpenDayLight Foundation, но также предлагают свои разработки.

Участники ODL Foundation	Лояльные	Конкурирующие
Brocade	Cyan	Big Switch
Cisco	HP	Juniper
Ericsson	NEC	Plexxi
Extreme networks	Nuage Networks	PLUMgrid
Hewlett Packard Enterprise		Pluribus
Huawei		Sonus
		VMware NSX

Таблица 10: Производители коммерческих контроллеров

3. Конкурирующие — компании, предлагающие контроллеры на основе своих исследований.

Большое количество компаний используют OpenDayLight как основу для своих контроллеров. Это говорит о том, что модульность ODL позволяет им создавать контроллеры под бизнес-требования компании. Коммерческие контроллеры различаются между собой по количеству поддерживаемых протоколов и ряду других характеристик. В связи с такой заинтересованностью производетелей в адоптации open-source контроллера ODL, в проекте есть уверенность, что их ПО будет иметь место на рынке. Также стоит отметить, что почти всегда приложения работают как на открытой версии, так и на коммерческом продукте. Это все помогает продолжать исследования на базе указанных контроллеров.

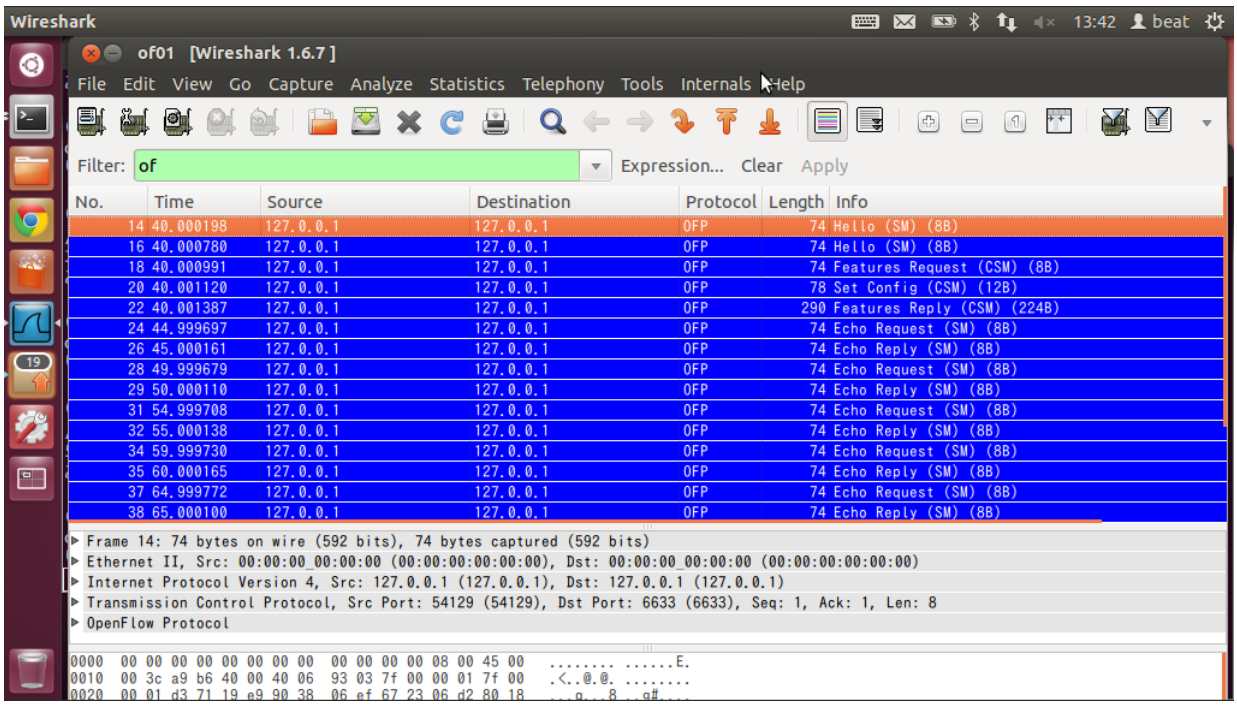

Рис. 5:

5. Эмуляция виртуальной сети

5.1. Wireshark

Wireshark — программа для анализа трафика компьютерных сетей технологии Ethernet и некоторых других, имеющая удобный графический пользовательский интерфейс, а также ряд возможностей по сортировке и фильтрации информации. Программа позволяет захватывать весь трафик как в режиме реального времени, так и открывать лог-файлы. Wireshark позволяет мониторить пакеты различных протоколов, в том числе и OpenFlow (Рис. 5). Также отображается необходимая структура пакетов этих протоколов.

5.2. Mininet

В старой документации к OpenFlow на сайте разработчиков есть ссылка на образ диска VirtualBox (.dvi), где уже установлены необходимые для работы с OpenFlow утилиты Linux и ряд других сетевых утилит. Также на этом диске непосредственное внимание уделяется двум

программным продуктам:

1. Mininet — платформа для эмуляции сетей, позволяющая создавать различные сетевые устройства, такие как свитчи, узлы и соединения на одном компьютере.

2. Dpctl — утилита для общения согласно стандарту OpenFlow.

Mininet даёт возможность строить сети различной топологии и размера не выходя за пределы одной виртуальной машины. Более того, все это делается через очень простой и удобный API. Над всеми хостами сети можно предпринимать различные действия, по сути использовать все утилиты Linux из терминала, также на коммутаторе можно изменять правила маршрутизации и выполнять ряд других действий.

5.3. Алгоритм обработки ARP-запросов

В работе [10] описывается анализ служебного трафика ПКС. Сначала необходимо создать модель сети с помощью среды mininet и OpenDayLight контроллера. Схематичное изображение взаимодействия всех программ показано на рисунке 6. Коммутатор взаимодействует с контроллером по средством протокола 1.3. Далее если воспользоваться утилитой ping в mininet для определения доступности хостов с помощью Wireshark можно анализировать пакеты OpenFlow-трафика. На

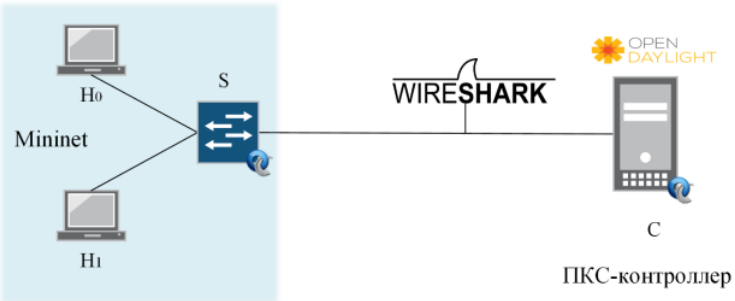

Рис. 6: Из работы [10]

диаграмме в рисунке 7 отображены все действия, которые произошли во время исполнения утилиты ping. Представленный процесс на рисунке 7 состоит из двух этапов:

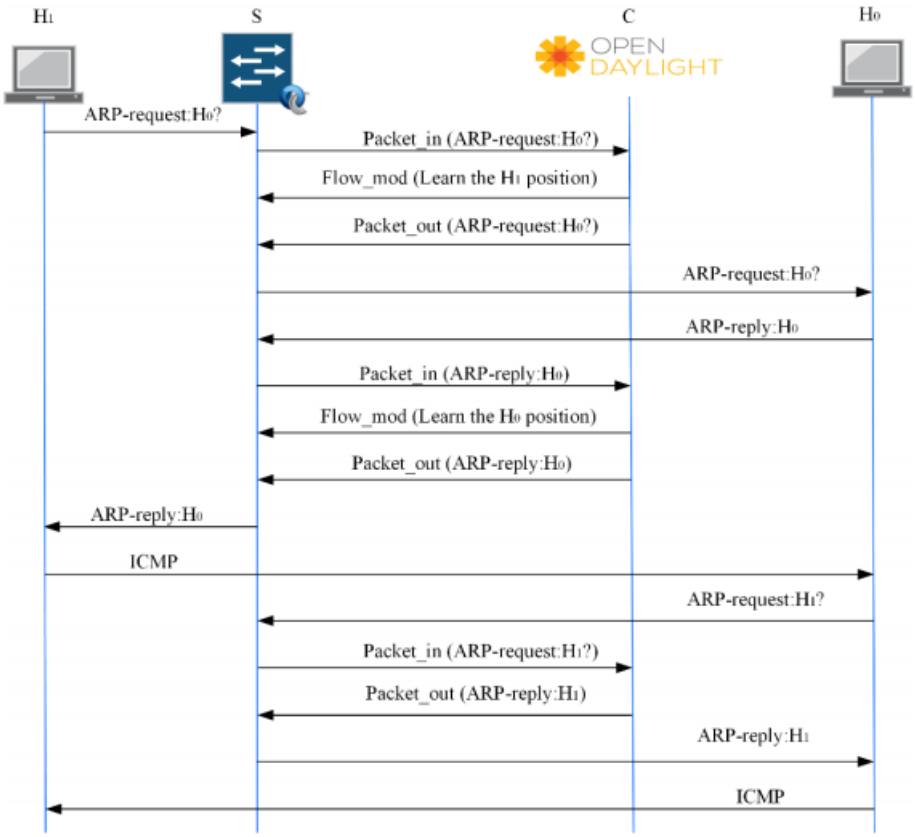

Рис. 7: Из работы [10]

1. Этап установления соединения от корневого сетевого элемента H1 до искомого H0.

2. Этап установления соединения от искомого сетевого элемента H0 до корневого H1.

5.4. Создание тестовой сети

Для реализации задачи зеркалирования портов на физическом стенде была спроектирована топология, схематично описанная на рисунке 8.

Для создания такой тестовой сети необходимо предпринять ряд действий:

- Создать 4 хоста.

- Соединить все хосты к коммутатору.

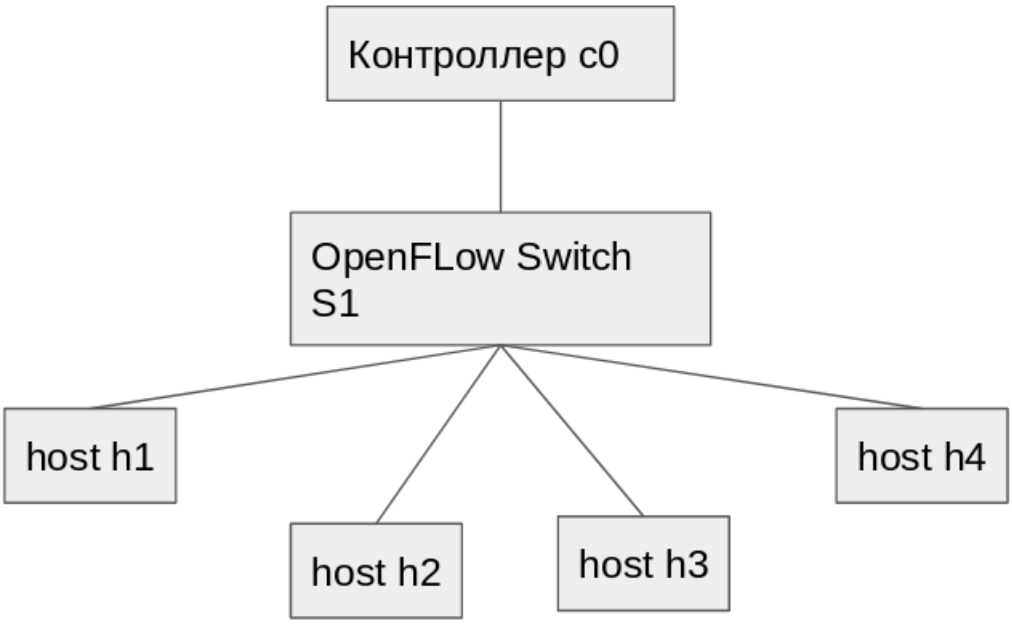

Рис. 8:

- Установить MAC-адрес для каждого хоста.

Ниже показаны команды запуска этой сети (рис. 9).

После создания сети возникает необходимость проверки связи хостов с контроллером и оценка скорости передачи трафика, для этого воспользуемся утилитой ping (рис. 10). Также оценим пропускную способность утилитой iperf (рис. 11).

Время первой передачи пакетов намного превышает время передачи остальных (рис. 10). Это связано с тем, что изначально все таблицы маршрутизации пустые и хосты постоянно обращаются к контроллеру для добавления новой записи в таблицу.

После того как в сети пошел трафик, встает вопрос о её безопасности [4]. В работах https://github.com/mauromoura/opendaylight-firewall (online; accessed:26.05.2019) и https://github.com/icarocamelo/OpenDaylight-Firewall (online; accessed:26.05.2019) приведены межсетевые экраны для OpenDayLight контроллера. Добавление правил посредством использования restconfig приводит к полной блокировки трафика по указанным потокам. В рамках более сложного сценария моделирования была выбрана установка

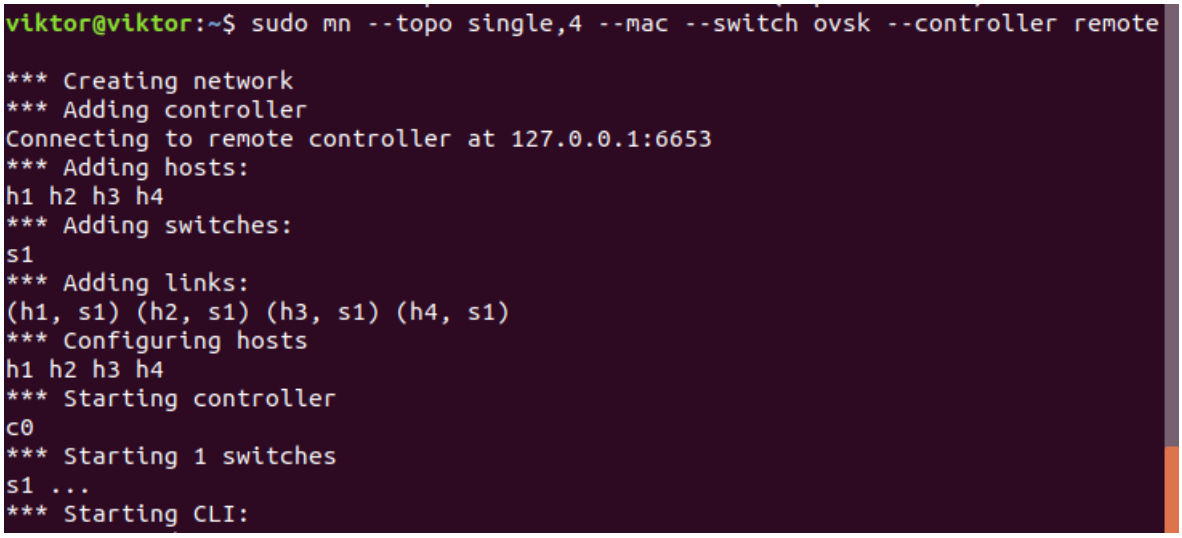

Рис. 9:

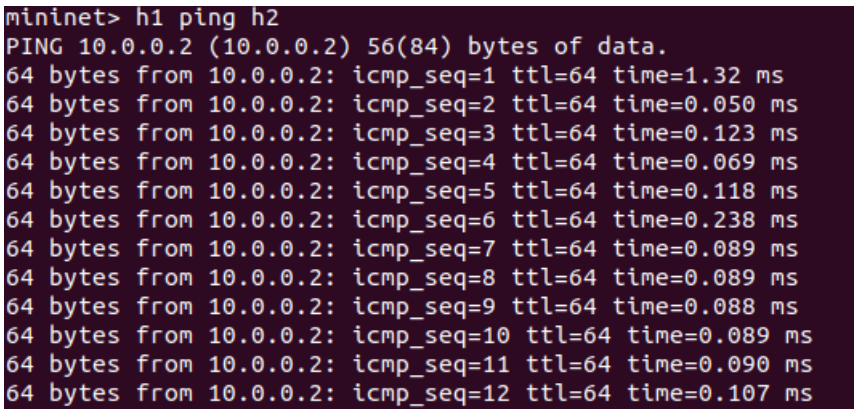

Рис. 10:

Рис. 11:

своего межсетевого экрана, после применения которого к потоку между вторым и четвертым хостом был получен аналогичный изученным работам результат (рис. 12).

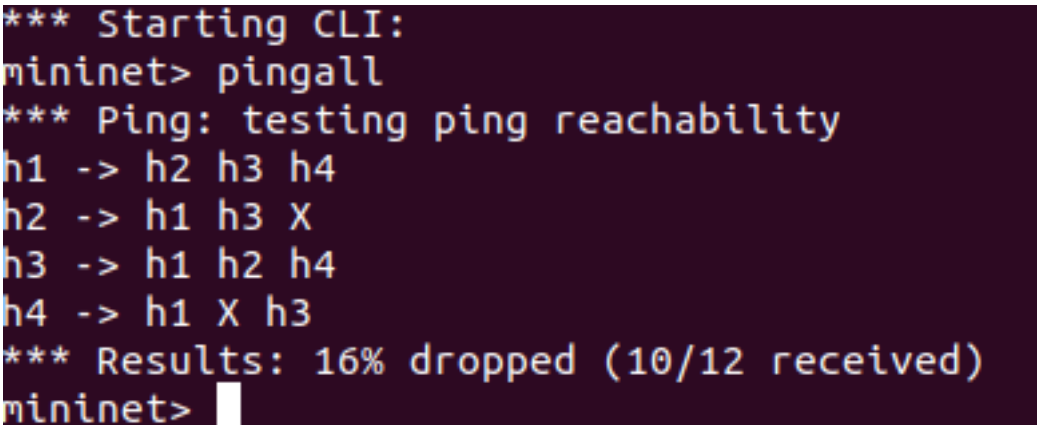

Рис. 12:

6. Тестовый стенд

У брендов IT индустрии, таких как Cisco, Huawei, и HP, есть коммутаторы с поддержкой OpenFlow, однако их цена очень высока — в среднем 1000 долларов и выше. Однако за основу они взяли ODL платформу и имеют платиновый уровень поддержки этого проекта.
Также есть возможность использовать открытый программный продукт OpenVSwitch, специально предназначенный для работы с системами виртуализации. Некоторые его возможности: контроль трафика с помощью sFlow и NetFlow, способность создания VLAN, поддержка OpenFlow, политики QoS и другие.
Компания Northbound Networks решила сделать коммутатор для ознакомления с ПКС. Его главные принципы: простота, дешевизна, открытость. Результат их деятельности — это коммутаторы Zodiac GX, с которыми и ведется дальнейшая работа.
В качестве начальных подходов к изучению возможностей ПКС сетей при помощи коммутатора Zodiac GX, 4 хостов и контроллера ODL были выбраны следующие ”кейсы”:

1. Реализация физического стенда и проверка корректности его работы.

2. Конфигурация стенда с помощью выбранного приложения для управления.

6.1. Проверка корректности работы тестового стенда

Одним из главных преимуществ SDN является автоматизация процесса настройки нового оборудования при подключении его к уже существующей сети. Поэтому было решено измерить время необходимое на настройку коммутатора при добавлении его в сеть. Были рассмотрены два случая: подключение первого устройства к контроллеру (подключение коммутатора к пустой сети) и подключение устройства к сети, в

которой уже находится один коммутатор.

Для определения времени, необходимого на настройку, использовалась программа-анализатор трафика Wireshark. В качестве точки начала отсчета было выбрано время, когда на контроллер поступает сообщение OPFT_HELLO от подключенного коммутатора. Данное сообщение является симметричным и отправляется сразу после установки TCP(или TLS) соединения для согласования версий протокола. После этого контроллер запрашивает доступные функции коммутатора и модифицирует таблицы потоков с помощью сообщения FLOW_MOD. Стоит отметить, что данное сообщение не предусматривает ответа при положительном завершении операции. Поэтому концом отсчета был выбран момент отправки последнего сообщения FLOW_MOD. Измерения были проведены десять раз для каждого случая. Результаты показаны на рисунке 13. Таким образом, на настройку нового коммутатора в среднем ушло 2.098 и 2.112 секунд соответственно Также были измерена

№	Случай 1	Случай 2
1	2.089 с	2.136 с
2	2.101 с	2.115 с
3	2.113 с	2.113 с
4	2.111 с	2.114 с
5	2.094 с	2.115 с
6	2.098 с	2.112 с
7	2.094 с	2.106 с
8	2.094 с	2.102 с
9	2.094 с	2.108 с
10	2.091 с	2.103 с

Рис. 13:

пропускная способность стенда. Коммутатор Zodiac GX имеет 5 гигабитных портов. В первом тесте были измерена скорость между двумя хостами, подключенными к одному коммутатору, во втором — к двум последовательно соединенным коммутаторам. Результаты тестов — 677 Мбит/с и 638 Мбит/с соответственно.

6.2. Приложение для конфигурирования

Есть ряд интерфейсов для администрирования ПКС-сетью. Но их функциональность была ограничена, поэтому было выбрано приложение https://github.com/ZhekehZ/OFUtils (online; accessed:26.05.2019), которое работает на более низком уровне. Оно состоит из графического интерфейса, состоящего из топологии сети, где напротив каждого сетевого узла есть ячейки для постановки определенного VLAN, а также таблицы потоков, конфигурации которых представлены в json формате. Их можно перезаписывать и получать некую статистику.

6.2.1. VLAN

Чтобы определить узлу VLAN необходимо запустить графическое приложение, выбрать интересующий узел и нажатием правой кнопки вызвать всплывающее меню, которое предложит поменять VLAN, что показано на рисунке 14.

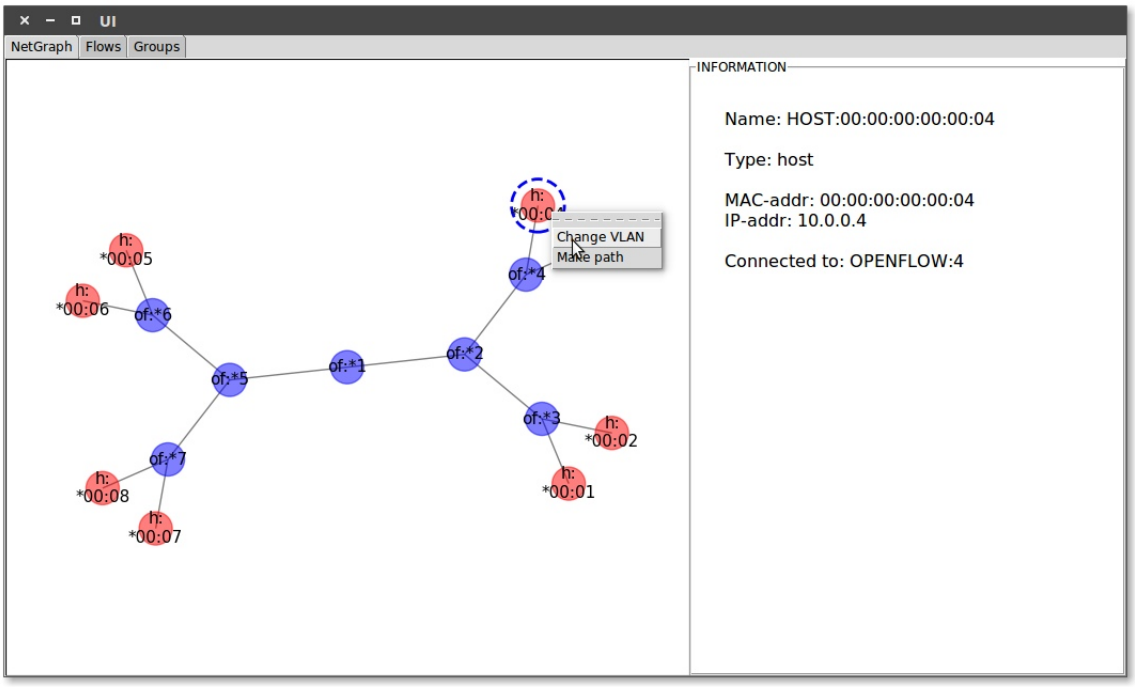

Рис. 14:

6.3. Зеркалирование

6.3.1. Первый подход

Есть два подхода к решению этой проблемы, первый заключается в том, чтобы в выбранном приложении для управления открыть конфигурацию нужного потока и изменить её на ту, что находится в приложении А. Как это выглядит показано на рисунке 15.

Рис. 15:

6.3.2. Второй подход

Второй заключается в том, чтобы использовать любое приложение для общения через Rest API, например Postman. Более подробное описание как в этом случае быть описано в этом пункте.

OpenFlow 1.3 — совместимые коммутаторы позволяют организовать многоадресную рассылку (Рис. 16). Для этого используется групповая таблица (Group Table). Каждая запись в таблице содержит группу наборов правил.

В зависимости от типа группы пакет может обрабатываться следующими способами:

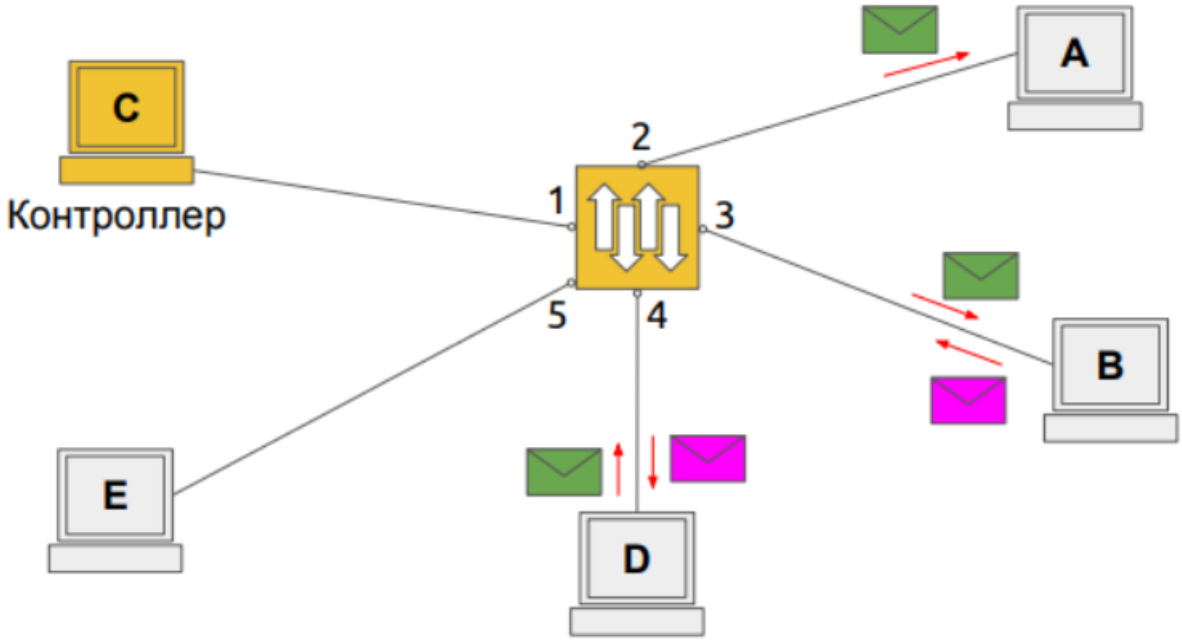

Рис. 16:

1. ALL — пакет клонируется, после чего к каждому пакету применяется индивидуальный набор правил.

2. SELECT — к пакету применяется какой-то один набор правил.

3. INDIRECT — пакету применяется один определенный набор правил.

4. FAST FAILOVER — применяется первый набор правил, ассоциированный с ”живым” портом.

Для примера была создана группа типа ALL. Для проверки использовалась сеть с единственным коммутатором и четырьмя хостами. Подробная конфигурация группы описана в приложении А. После применения данной группы к пакету, его копии будут отправлены на порт 3 со значением TTL, равным 239 и на порт 2 с TTL = 13. Для загрузки конфигурации на контроллер использовался Rest API. В приложении представлена соответствующая команда для утилиты curl, в которой 10.0.1.8

— ip адрес контроллера, а 75668639618598 — номер коммутатора, определенный контроллером. Далее необходимо добавить поток, использующий ее. Конфигурация потока описана в приложении Б. После добавления все пакеты, пришедшие на порт 1 коммутатору 75668639618598, будут обработаны в соответствии с группой, описанной выше.

7. Заключение

- Проведена работа по изучению текущих решений по SDN и сделан обзор, по результатам которого был выбран лучший контроллер.
- Была спроектирована виртуальная сеть, средой виртуализации послужил VirtualBox и программа Mininet. Был установлен межсетевой экран.
- Был собран тестовый физический стенд на свитчах Zodiac GX и установлена его корректная работа.
- С помощью приложения для администрирования сети была решена задача по установке VLAN, произведено зеркалирование портов.

Литература

1. Семеновых А. А., Лапонина О.Р. Сравнительный анализ SDN-контроллеров. International Journal of Open Information Technologies vol. 6, no.7, 2018

2. Лапонина О.Р., Сизов М.Р. Лабораторный стенд для тестирования возможностей интеграции ПКС-сетей и традиционных сетей. International Journal of Open Information Technologies vol. 5, no.9, 2017.

3. Open Networking Foundation (ONF). OpenFlow Switch Specification [Text], 2015 — 283.стр

4. Open Networking Foundation (ONF). Principles and Practices for Securing Software Defined Networks [Text], 2015 — 27.стр

5. ONF TS-002 OpenFlow Switch Specification version 1.1.0 (wire protocol 0x02) [Электронный ресурс] // Open Network Foundation. – URL: 144 https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wpconte spec-v1.1.0.pdf (дата обращения: 04.05.2018)

6. ONF TS-003 OpenFlow Switch Specification version 1.2. (wire protocol 0x03) [Электронный ресурс] // Open Network Foundation. – URL: https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wpcontent/u spec-v1.2.pdf (дата обращения: 04.05.2018)

7. ONF TS-006 OpenFlow Switch Specification version 1.3.0 (wire protocol 0x04) [Электронный ресурс] // Open Network Foundation. – URL: https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wpcontent/u spec-v1.3.0.pdf (дата обращения: 04.05.2018).

8. ONF TS-012 OpenFlow Switch Specification version 1.4.0 (wire protocol 0x05) [Электронный ресурс] // Open Network Foundation. – URL: https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wpcontent/u spec-v1.4.0.pdf (дата обращения: 04.05.2018).

9. ONF TS-025 OpenFlow Switch Specification version 1.5.1 (wire protocol 0x06) [Электронный ресурс] // Open Network Foundation. – URL: https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wpcontent/u switch-v1.5.1.pdf (дата обращения: 04.05.2018).

10. Галич С. В. "Исследования анализа задержки обработки трафика управления в программно-конфигурируемых сетях", диссертация в Волгоградском государственном университете, 2018 стр.97–99

11. O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "SDN controllers: A comparative study," in Mediterranean Electrotechnical Conference, 2016, pp. 1–6.

12.] D. Suh, S. Jang, S. Han et al., "Toward Highly Available and Scalable Software Defined Networks for Service Providers," IEEE Communications Magazine, vol. 55, no. 4, pp. 100–107, 2017

13. "Сравнительный анализ существующих контроллеров для программно-конфигурируемых сетей" Чернов В.И., Шиховцов С.Ю., Полежаев П.Н. ФГБОУ ВО "Оренбургский государственный университет"

14. R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Feature-based comparison and selection of Software Defined Networking (SDN) controllers," in World Congress on Computer Applications and Information Systems, 2014, pp. 1–7.

15. SDN Controllers Report 2015 Edition [Электронный ресурс] // SDNCentral, LLC. – 2015. – URL: https://www.sdxcentral.com/reports/sdncontrollers-report-2015/ (дата обращения: 25.04.2018).

16. Leary M., SDN, NFV, and open source: the operator's view [Электронный ресурс] // Gigaom Research. – 2014. – URL: http://ftp.tiaonline.org 20.04.2018).

ПРИЛОЖЕНИЕ А

Команда:

```
curl --user ''admin'':''admin'' -H ''Content-type: application/json'' -X P
opendaylight-inventory:nodes/node/openflow:75668639618598/group/13
```

group.json:

```
{
  ''flow-node-inventory:group'': [
    {
      ''group-id'': 13,
      ''group-name'': ''GROUP-1'',
      ''group-type'': ''group-all'',
      ''barrier'': false,
      ''buckets'': {
        ''bucket'': [
          {
            ''bucket-id'': 1,
            ''action'': [
              {
                ''order'': 0,
                ''output-action'': {
                  ''output-node-connector'': 2
                }},
              {
                ''order'': 1,
                ''set-nw-ttl-action'': {
                  ''nw-ttl'': 13
                }}]
          },
          {
            ''bucket-id'': 2,
            ''action'': [
```

```
                {
                  ''order'': 1,
                  ''output-action'': {
                    ''output-node-connector'': 3
                  }
                },
                {
                  ''order'': 2,
                  ''set-nw-ttl-action'': {
                    ''nw-ttl'': 239
                  }}]
              }]}}]
}
```

ПРИЛОЖЕНИЕ Б

Команда:

```
curl --user ''admin'':''admin'' -H ''Content-type: application/json'' -X P
opendaylight-inventory:nodes/node/openflow:75668639618598/table/0/
flow/13
```

flow.json:

```
{
  ''flow'': [
    {
      ''table_id'': 0,
      ''id'': ''13'',
      ''priority'': 10,
      ''match'': {
        ''in-port'': ''openflow:75668639618598:1''
      },
      ''instructions'': {
        ''instruction'': [
          {
            ''order'': 0,
            ''apply-actions'': {
              ''action'': [
                {
                  ''order'': 0,
                  ''group-action'': {
                    ''group'': ''GROUP-1'',
                    ''group-id'': 13
                  }}]}}]}}]
}
```