

Санкт-Петербургский Государственный Университет
Математическое обеспечение и администрирование информационных
систем

Кафедра Системного Программирования

Кравченко Евгений Артурович

Реализация тестового стенда SDN

Дипломная работа

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

Рецензент:
д-р техн. наук, доцент Киричѐк Р. В.

Санкт-Петербург
2019

SAINT-PETERSBURG STATE UNIVERSITY
Software and Administration of Information Systems

Software Engineering Chair

Eugene Kravchenko

Implementation of the SDN test bench

Graduation Thesis

Scientific supervisor:
Senior lecturer Ilya Zelenchuk

Reviewer:
Doctor of Engineering Sciences, Associate Professor Ruslan Kirichek

Saint-Petersburg
2019

Оглавление

Введение	4
1. Постановка задачи	6
2. Обзор	7
2.1. Архитектура SDN	7
2.2. OpenFlow	8
2.3. Контроллер	9
2.4. Коммутаторы	11
2.5. SDN-Стенд	12
2.6. Приложения для настройки сети	12
3. Реализация	14
4. Дополнительный функционал	17
Заключение	19
Список литературы	20
ПРИЛОЖЕНИЕ А	22

Введение

Объем интернет трафика экспоненциально растет с каждым годом. В последние годы встал вопрос об изменении основных концептов построения и управления компьютерными сетями. Проблема в том, что сопровождение традиционных сетей, где управление осуществляется посредством стека TCP/IP, представляет собой трудоемкий процесс, требующий знаний большого количества протоколов. Для настройки сети, системному администратору требуется вручную настроить каждое входящее в нее устройство, в то время как количество устройств может достигать нескольких тысяч. Несмотря на то, что ведущие мировые производители оборудования имеют собственные решения для контроля и настройки сети, данные решения, как правило, несовместимы между собой [3, 6].

Software defined networking (SDN, программно-конфигурируемая сеть) — это новый принцип построения компьютерных сетей, в котором плоскости передачи данных и управления разделены и физически удалены друг от друга. Все физические устройства, входящие в инфраструктуру сети, являются программируемыми, за их настройку отвечает программа-контроллер. Конфигурирование производится посредством стандартизированного протокола (OpenFlow).

Такой подход имеет множество преимуществ. Во-первых, снижение степени зависимости от вендора сетевого оборудования, поскольку единственным требованием к оборудованию является поддержка стандартного протокола. Во-вторых, поскольку SDN-сеть полностью управляется контроллером, добавление в нее и настройка нового коммутатора является автоматизированным процессом и не требует больших усилий. Также наличие централизованного контроля позволяет гибко контролировать трафик, и легко определять состояние сети. Еще одним преимуществом при переходе на SDN является потенциальное снижение стоимости сети. С одной стороны, аппаратная часть сетевых устройств сильно упрощается, так как все задачи, связанные с маршрутизацией переносятся на контроллер. С другой, каждое устройство теперь может

выполнять различные функции в зависимости от загруженной на него конфигурации. Таким образом, пропадает необходимость в приобретении большого количества различного специализированного оборудования, так как их функции могут быть решены программно.

SDN активно развивается и используется крупнейшими ИТ-компаниями, такими как Google, Deutsche Telecom [1] и многие другие организация, присоединившиеся к Open Networking Foundation¹.

Однако, несмотря на высокие темпы развития, не существует приложения, которое позволило бы человеку, начинающему изучать данную область, провести низкоуровневую настройку сети, не ограничивая множество возможных конфигураций. Поэтому для экспериментирования с различными настройками сети зачастую приходится использовать API контроллера напрямую, что отнимает немало времени.

¹<https://www.opennetworking.org/>

1. Постановка задачи

Исходя из вышесказанного, была поставлена цель создать решение для изучения возможностей SDN, позволяющее гибко настраивать конфигурацию сети.

Поскольку реализация приложения зависит от северного API SDN-контроллера, было решено реализовать тестовый стенд, возможности которого можно будет изучить с помощью реализованного решения. В связи с этим были поставлены следующие задачи:

- Сделать обзор предметной области;
- Развернуть тестовый стенд SDN;
- Изучить существующие решения для манипулирования настройками сети на низком уровне и выявить их недостатки;
- Реализовать решение.

2. Обзор

2.1. Архитектура SDN

Основной идеей SDN является разделение в сети уровней передачи данных и управления. Таким образом, логически структура сети может быть представлена тремя слоями: слой приложений, управления и передачи данных (Рис. 1). На первом уровне реализуются все необходимые бизнес-приложения, которые выполняют настройку сети. Доступ к сети осуществляется через SDN-контроллер со второго уровня. Передача информации между контроллером и приложениями происходит посредством северного API контроллера, в качестве которого в большинстве выступает Rest API. Контроллер выполняет непосредственную загрузку конфигураций на сетевые устройства. Для этого используется южный протокол, как правило, OpenFlow.

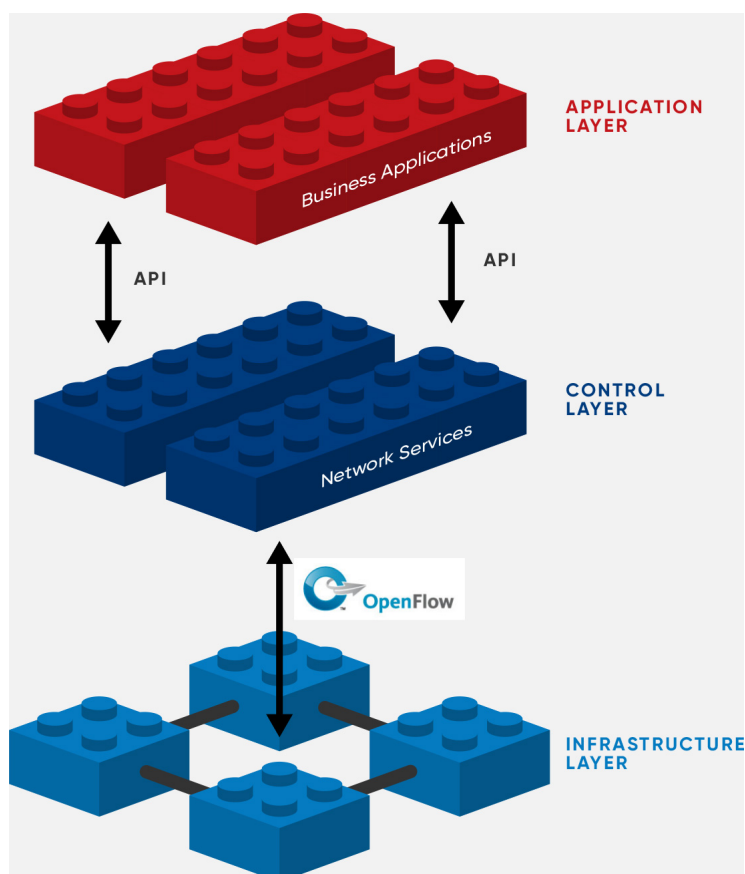


Рис. 1: Архитектура SDN-сети, <https://www.opennetworking.org/sdn-definition> (дата обращения: 14.12.18)

2.2. OpenFlow

OpenFlow(OF) — это стандартизированный протокол для управления устройствами в сети SDN, за разработку спецификаций которого отвечает организация Open Networking Foundation[5]. Существуют и другие "северные" протоколы, такие как OVSDB, NETCONF, SNMP. На данный момент не существует решения, единогласно принятого всеми производителями, однако OpenFlow занимает лидирующую позицию в данном вопросе.

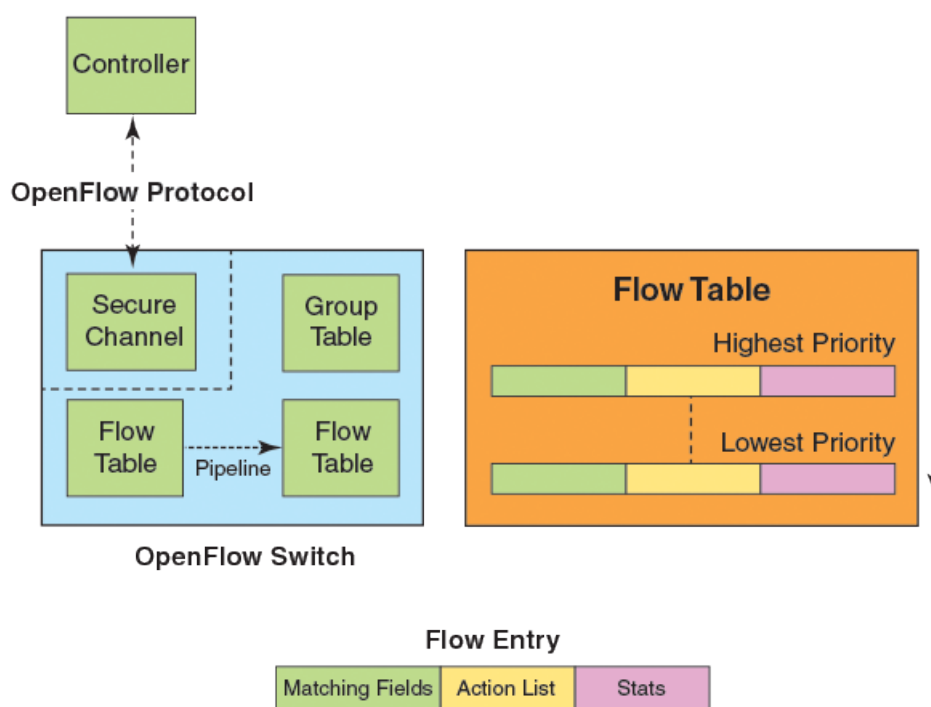


Рис. 2: Архитектура OpenFlow 1.3, <http://docs.ruckuswireless.com/fastiron/08.0.61/fastiron-08061-sdnguide/GUID-031030CA-62EC-4009-A516-5510238EF8F4.html> (дата обращения: 10.04.19)

В OF-совместимых коммутаторах пересылка пакетов осуществляется на основе содержимого их таблиц потоков (Flow tables) [2, 5]. Каждая запись в таблице содержит поля сравнения (Matching Fields), счетчиков (Stats) и инструкций (Action List). Заголовки входящих пакетов последовательно сопоставляются с полями сравнений в порядке приоритета записей. Сравнение начинается с нулевой таблицы. Если совпадение найдено, выполняются соответствующие инструкции. Инструк-

ции описывают правила передачи пакета на физический и модификации пакета, передачи его на обработку другой таблице (Рис. 2) и другие[5]. Помимо таблиц потоков существует также групповая таблица, которая позволяет использовать дополнительный уровень абстракции для многопутевой рассылки пакета. Пакет, попадающий на обработку в группу(Рис. 3), дублируется необходимое количество раз, после чего к каждой копии применяется определенный блок инструкций. То, какие блоки инструкций будут применены, определяется типом группы.

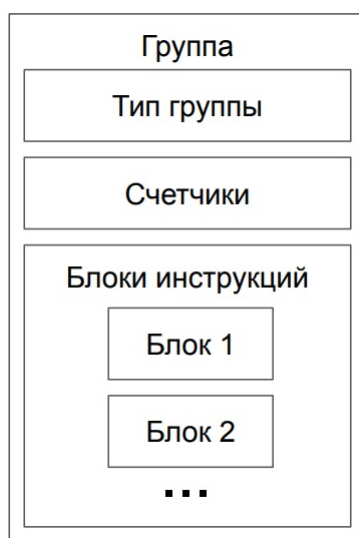


Рис. 3: Структура групповой таблицы

Поля счетчиков, как в таблицах потоков, так и в групповой таблице, используются для сбора статистики о том, сколько пакетов и какое количество байт было обработано данным потоком.

Протокол OpenFlow определяет ключевые понятия, которыми необходимо оперировать при конфигурировании сети. Поэтому решение для настройки сети должно позволять модифицировать таблицы потоков и групповые таблицы.

2.3. Контроллер

От выбора контроллера напрямую зависит реализация решения для настройки сети, так как конфигурирование сети будет производиться посредством его северного API. Поэтому было решено выбрать откры-

тый контроллер, который на данный момент активно развивается и не уступает в большинстве характеристик остальным.

Существует множество работ, посвященных сравнению различных контроллеров. Самыми популярными на данный момент являются ONOS, OpenDayLight, NOX, POX, Ryu, Beacon. В [4] Сравниваются POX, Ryu, ONOS и OpenDaylight. Контроллеры POX и Ryu реализованы на языке Python, из-за чего обладают меньшей производительностью по сравнению с ONOS и OpenDayLight, которые написаны на Java. Однако, авторы отмечают простоту написания программ для POX, что является причиной его использования во многих исследовательских проектах. В [4] было проведено сравнение контроллеров по производительности, и скорости настройки таблиц потоков. Тестирование было проведено на виртуальной сети, созданной в эмуляторе Mininet, также использовались утилиты ping (для замера времени настройки таблиц) и iperf (для измерения производительности). В результате было выявлено, что время, требуемое для настройки таблиц различается не сильно, в то время как на тесте производительности лучший результат показал ONOS. Контроллеры OpenDayLight и Ryu показали схожие результаты, несильно уступая первому месту, в то время как производительность POX оказалась значительно ниже.

В статье [9] сравниваются более десяти различных контроллеров, при этом, в отличие от [4], помимо производительности уделяется внимание таким характеристикам, как кроссплатформенность, надежность, масштабируемость, а также поддержка различных "северных" и "южных" интерфейсов. Авторы отмечают, что контроллеры ONOS и OpenDayLight обладают наиболее обширной функциональностью. К тому же, они являются кроссплатформенными и имеют высокую степень модульности, а используемый в них контейнер OSGI позволяет добавлять новую функциональность без необходимости перезагрузки программы. С другой стороны, контроллер Ryu является неплохим решением для проведения исследований и реализации небольших бизнес-приложений. Тесты производительности были проведены с помощью утилиты Cbench. В результате было выявлено, что наилучшей произво-

дительностью в тесте пропускной способности обладают контроллеры Mul и Libfluid, реализованные на C/C++, в то время как наименьшей задержкой обладает Maestro, написанный на Java.

Для реализации тестового стенда был выбран OpenDayLight, так как он показал хорошие результаты в тестах производительности, а также обладает рядом других преимуществ, отмеченных в [9]. Кроме того, данный контроллер имеет подробную документацию, в отличие от NOX, POX и Veason и на его основе спроектированы многие коммерческие решения.

2.4. Коммутаторы

Для реализации SDN-стенда необходимо было выбрать коммутаторы, поддерживающие протокол OpenFlow. На данный момент большинство существующих аппаратных реализаций SDN-коммутаторов разработаны для использования в промышленных масштабах, из-за чего имеют довольно высокую стоимость. В качестве альтернативы существуют программные реализации OF-совместимых коммутаторов.

Одной из таких реализаций является Open vSwitch - виртуальный коммутатор с открытым исходным кодом, который имеет поддержку протокола OpenFlow и позволяет привязать виртуальные порты к физическим портам устройства. Таким образом, в качестве SDN-коммутатора может выступать любое устройство, на котором возможна работа Open vSwitch.

Zodiac GX² является одним из готовых решений, реализованных данным способом. На сегодняшний день данный коммутатор является самым доступным среди аналогичных предложений, поэтому для реализации тестового стенда были приобретены два таких устройства.

²<https://northboundnetworks.com/products/zodiac-gx>

2.5. SDN-Стенд

Из указанных выше коммутаторов, с использованием контроллера OpenDaylight, был собран тестовый стенд. Работоспособность стенда была проверена утилитой ping. При этом на контроллер был установлен плагин odl-l2switch-switch-ui[7], который конфигурирует потоки так, чтобы коммутаторы выполняли роль L2-коммутатора, то есть автоматически добавляли новые потоки, выполняющие коммутацию на основе MAC-адресов.

Для возможности настройки таблицы потоков, в контроллер были также установлена группа плагинов odl-openflowplugin-*, позволяющая модифицировать OpenFlow-таблицы, собирать статистику и информацию о подключенных устройствах и плагин odl-restconf-all, добавляющий возможность конфигурировать настройки посредством REST API.

2.6. Приложения для настройки сети

В процессе изучения возможностей протокола OpenFlow на практике были использованы различные способы модифицирования OF-таблиц. В результате выяснилось, что на данный момент для настройки SDN под управлением контроллера OpenDaylight на уровне конфигурирования таблиц OpenFlow существуют два популярных решения. Первое заключается в непосредственном использовании REST API контроллера с помощью утилит, позволяющих совершать различные HTTP-запросы, такие, как curl или Postman. При этом Postman неоднократно упоминается в документации OpenDaylight и рекомендуется к использованию. Однако, стоит заметить, что поскольку данные решения предназначены для широкого круга применения, они требуют дополнительных действий при модификации таблиц, таких как заполнение заголовков и выполнения вспомогательных запросов для получения текущей конфигурации.

Альтернативным способом настройки конфигурации на уровне логики работы протокола OpenFlow является открытое приложение Cisco

OpenDaylight Openflow Manager³(OFM). Данное приложение имеет графический web-интерфейс и позволяет модифицировать таблицы потоков, собирать и агрегировать статистику, а также предоставляет визуализацию топологии сети.

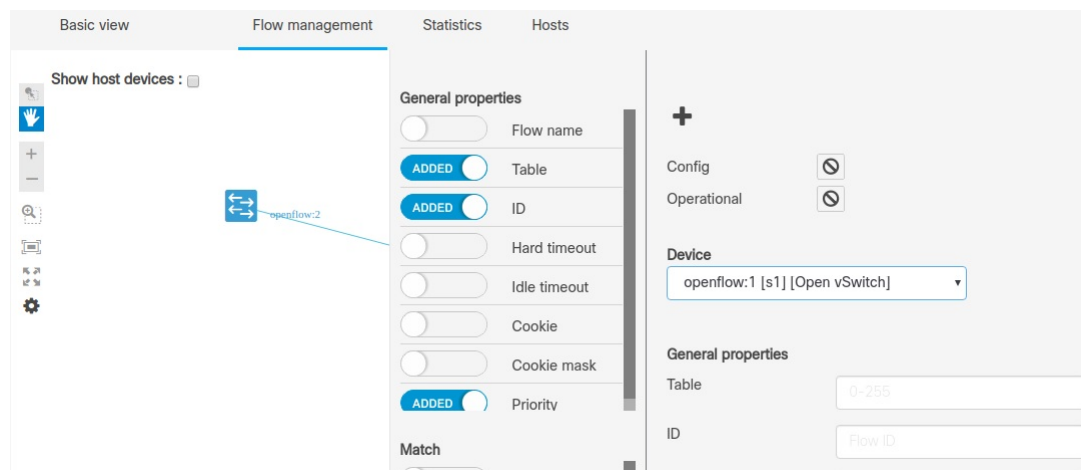


Рис. 4: OpenFlow Manager

Для модификации таблиц потоков в OFM предусмотрен графический интерфейс (Рис. 4), в котором изменение полей сравнений и инструкций потока осуществляется с помощью соответствующего переключателя и поля ввода.

Данный графический интерфейс сильно упрощает процесс настройки, однако ограничивает общее число конфигураций, которые возможно создать используя Rest API напрямую. Кроме того, OFM не предусматривает возможности модифицирования и просмотра групповых таблиц, что не позволяет создавать потоки с многопутевой рассылкой пакета. Пример конфигурации в формате json, которую невозможно задать посредством OFM представлен в приложении А.

В связи с вышеописанным, было принято решение создать средство для конфигурирования SDN-сети, управляемой контроллером OpenDaylight, использующее более низкий уровень абстракции для модификации потоков и групповых таблиц, чем OFM и имеющее удобный графический интерфейс для обозревания текущих конфигураций.

³<https://github.com/CiscoDevNet/OpenDaylight-Openflow-App>

3. Реализация

Решение было реализовано на языке Python. Было решено создать инструмент, обладающий следующими особенностями:

- Наличие окна с визуализатором топологии сети;
- Наличие окна с редактором конфигурации таблиц потоков и групповых таблиц;
- Автоматическое обновление данных о топологии и конфигурации при ее изменении.

Для автоматического обновления информации о топологии и настройках сети используется возможность OpenDaylight подписки на уведомления об изменении конфигурации. За получение уведомлений отвечает отдельный класс (DU), при вызове метода (start) которого происходит вызов удаленной функции OpenDaylight, возвращающей имя нотификатора, после чего совершается get-запрос для получения URI, представляющую адрес, по которому находится WebSocket-сервер, уведомляющий об изменении конфигурации. В случае корректного получения адреса, создается отдельный поток, в котором запускается клиент, слушающий по данному адресу. При создании клиента использовались библиотеки websockets и threading. После выполнения вышеперечисленных действий, узнать, произошли ли какие-либо изменения, можно с помощью специальных методов, возвращающих информацию об изменении конфигурации OpenFlow-таблиц (getInvChanged) и топологии сети (getTopoChanged).

За получение и обработку информации о потоках отвечает класс FlowManager. Метод requestData совершает запрос к хранилищу opendaylight-inventory, которое содержит всю информацию о коммутаторах и подключенных к ним устройствах, и к хранилищам, содержащим всю информацию о потоках — config и operational[8]. Первое отвечает за конфигурации, добавленные через REST API, второе расширяет данный набор конфигурациями, созданными самим контроллером и статистикой.

После получения информации об оборудовании происходит визуализация топологии сети с помощью библиотек Matplotlib и NetworkX. Для создания оконного интерфейса была использована библиотека tkinter. На рисунке 5 представлено изображение окна с визуализатором сетевого графа. При нажатии на вершину графа, обозначающую коммутатор, на панели справа выводится информация о том, какие устройства подключены к портам коммутатора.

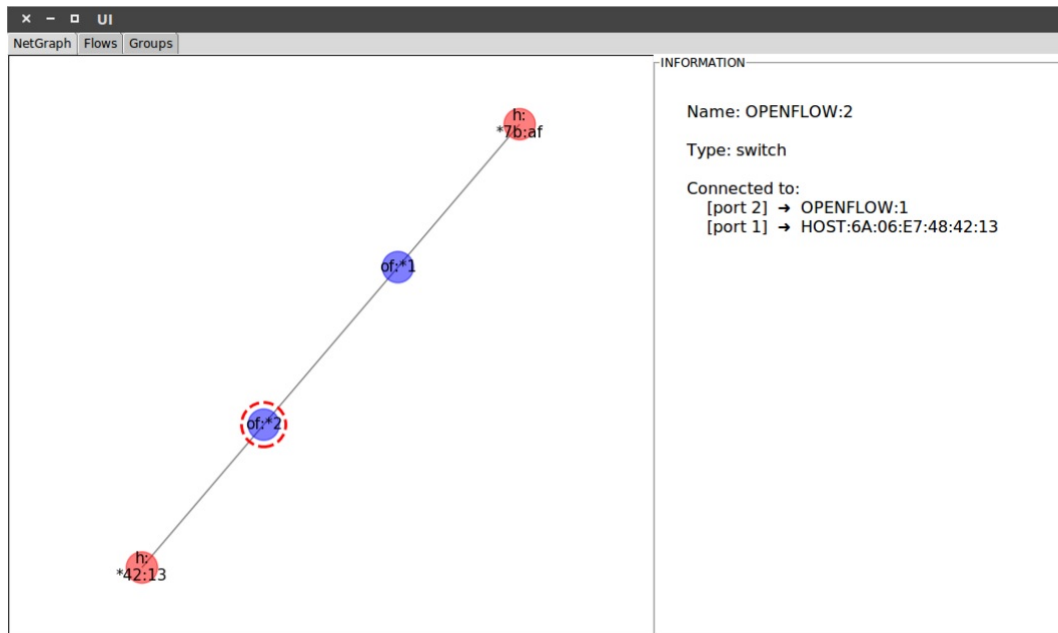


Рис. 5: Граф сети

Соседние две вкладки (Flows и Groups) предоставляют интерфейс для редактирования таблиц потоков и групповой таблицы (рис.6). Модификация потока осуществляется посредством редактирования его структуры, представленной в формате json в правой части окна. Слева находится панель со списком потоков, загруженных на выбранный коммутатор и кнопочный интерфейс для создания (Add), удаления (Delete) и отправки модифицированной версии потока (Send). При нажатии на кнопку Stats, в отдельном окне выводится краткая статистика о количестве пакетов и байт, обработанных данным потоком, которую можно использовать для определения, какие потоки участвуют в передаче трафика.

Исходный код приложения, а также инструкции по запуску нахо-



Рис. 6: Редактор потоков

дятся в открытом доступе: <https://github.com/ZhekehZ/OFUtils/>.

4. Дополнительный функционал

Для расширения функционала полученного инструмента было принято решение реализовать поддержку функции перемещения устройства в отдельную виртуальную сеть. Пример, демонстрирующий работу данной функции изображен на рисунке 7. Требуется "переместить" одного из хостов, находящихся в виртуальной сети 99 в сеть 44. При этом не должна изменяться конфигурация самого устройства, то есть перемещение должно осуществляться исключительно возможностями протокола OpenFlow. Данная задача может найти применение в области безопасности — при выявлении подозрительной активности одного из участника сети, его можно изолировать в отдельную виртуальную сеть.

Решение данной задачи состоит в просмотре списка потоков с выявлением принимающих или перенаправляющих пакет на порт, к которому подключено устройство, после чего данные потоки модифицируются путем добавления действий по изменению VLAN_ID в список правил.

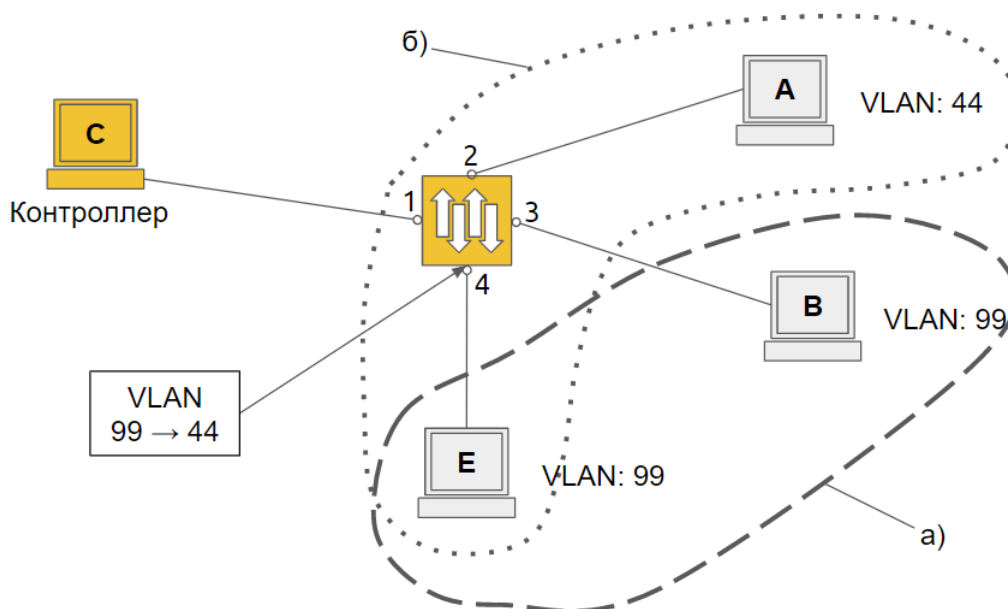
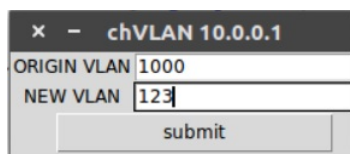


Рис. 7: Перемещение устройства E из VLAN 99 в VLAN 44; а) VLAN 99 до внесения изменений; б) VLAN 44 после изменений

Данная функция была интегрирована в графический интерфейс: при выборе вершины, обозначающей хоста на сетевом графе, в выпа-

дающем меню необходимо выбрать опцию "change VLAN", после чего в появившемся окне ввести исходный и новый номер VLAN(рис. 8).



The image shows a terminal window with the title bar "x - chVLAN 10.0.0.1". Inside the window, there is a form with two input fields. The first field is labeled "ORIGIN VLAN" and contains the value "1000". The second field is labeled "NEW VLAN" and contains the value "123". Below these fields is a button labeled "submit".

Рис. 8: Окно опции "change VLAN"

Заключение

В ходе данной работы было выполнено следующее.

1. Изучены принципы функционирования программно-конфигурируемых сетей;
2. Развернут тестовый стенд SDN;
3. Изучены существующие инструменты для настройки SDN под управлением контроллера OpenDaylight;
4. Изучены существующие инструменты для настройки SDN под управлением контроллера OpenDaylight и выявлены их ограничения;
5. Реализовано решение для настройки SDN, не обладающее ограничениями по созданию конфигураций сети.

Также функционал реализованного решения был расширен путем добавления функции перемещения устройства в отдельную VLAN.

Список литературы

- [1] А.А. Терешкевич, А.Н. Зубалов. ОБЗОР ТЕХНОЛОГИИ «ПРОГРАММНО-КОНФИГУРИРУЕМЫЕ СЕТИ ПКС/SDN». — ГАЗОВАЯ ПРОМЫШЛЕННОСТЬ № 12, 2016.
- [2] Ю.А. Семенов. Сетевая технология OpenFlow (SDN). — Режим доступа: <http://book.itep.ru/4/41/openflow.htm> (дата обращения: 15.11.2018).
- [3] Т.Смелянский. ПЕРЕХОД НА SDN: предпосылки, подходы, перспективы. — ПЕРВАЯ МИЛЯ 2(63)54-61, 2017.
- [4] A Comparison between Several Software Defined Networking Controllers / Stancu A. L., Halunga S., Vulpe A. et al. — 12th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS), 2015.
- [5] Foundation Open Networking. OpenFlow® Switch Specification. — 2015. — Режим доступа: <https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf> (дата обращения: 17.10.2018).
- [6] Kim H., Feamster N. Improving Network Management with Software Defined Networking. — IEEE Communications Magazine 51(2)114-119, 2013.
- [7] Project OpenDaylight. OpenDaylight documentation: L2 Switch. — Режим доступа: https://wiki.opendaylight.org/view/OpenDaylight_Controller:MD-SAL:L2_Switch (дата обращения: 21.01.2019).
- [8] Project OpenDaylight. OpenDaylight documentation: MD-SAL. — Режим доступа: <https://docs.opendaylight.org/projects/mdsal/en/latest/index.html> (дата обращения: 21.01.2019).

- [9] SDN controllers: A comparative study / Salman O., Elhajj I. H., Kayssi A., Chehab A. — 18th Mediterranean Electrotechnical Conference (MELECON), 2016.

ПРИЛОЖЕНИЕ А

Конфигурация таблицы 0:

```
{
  "id": 0,
  "flow": [
    {
      "id": "0",
      "priority": 0,
      "table_id": 0,
      "match": {},
      "instructions": {
        "instruction": [
          {
            "order": 0,
            "go-to-table": {
              "table_id": 1
            }
          }
        ]
      }
    }
  ]
}

{
  "id": "2",
  "priority": 1000,
  "table_id": 0,
  "match": {
    "ethernet-match": {
      "ethernet-type": {
        "type": 2048
      },
      "ethernet-destination": {
        "address": "30:f9:ed:c7:62:46"
      }
    },
    "vlan-match": {
      "vlan-id": {
        "vlan-id": 99,
        "vlan-id-present": true
      }
    }
  },
  "instructions": {
    "instruction": [
      {
```

```

        "order": 0,
        "apply-actions": {
            "action": [
                {
                    "order": 0,
                    "set-field": {
                        "vlan-match": {
                            "vlan-id": {
                                "vlan-id": 44
                            }}}}],
        {
            "order": 1,
            "go-to-table": {
                "table_id": 1
            }
        }
    ],
    {
        "id": "1",
        "priority": 1000,
        "table_id": 0,
        "match": {
            "ethernet-match": {
                "ethernet-source": {
                    "address": "30:f9:ed:c7:62:46"
                },
                "ethernet-type": {
                    "type": 2048
                }
            },
            "vlan-match": {
                "vlan-id": {
                    "vlan-id": 44
                }
            },
            "instructions": {
                "instruction": [
                    {
                        "order": 0,
                        "apply-actions": {
                            "action": [
                                {
                                    "order": 0,

```

```

        "set-field": {
            "vlan-match": {
                "vlan-id": {
                    "vlan-id": 99
                }}}}],
    {
        "order": 1,
        "go-to-table": {
            "table_id": 1
        }}}
    }
]
}

```

Конфигурация таблицы 1:

```

{
  "id": 1,
  "flow": [
    {
      "id": "2f",
      "priority": 1,
      "table_id": 1,
      "match": {
        "ethernet-match": {
          "ethernet-type": {
            "type": 2048
          },
          "ethernet-destination": {
            "address": "28:92:4a:3f:ce:2f"
          }}}},
      "instructions": {
        "instruction": [
          {
            "order": 0,
            "apply-actions": {
              "action": [
                {
                  "order": 0,
                  "output-action": {
                    "max-length": 0,

```



```

        "output-node-connector": "5"
    ]]]]]}
},
{
    "id": "46",
    "priority": 1,
    "table_id": 1,
    "match": {
        "ethernet-match": {
            "ethernet-type": {
                "type": 2048
            },
            "ethernet-destination": {
                "address": "30:f9:ed:c7:62:46"
            }
        },
        "instructions": {
            "instruction": [
                {
                    "order": 0,
                    "apply-actions": {
                        "action": [
                            {
                                "order": 0,
                                "output-action": {
                                    "max-length": 0,
                                    "output-node-connector": "3"
                                }
                            ]
                        ]
                    }
                ]
            ]
        }
    ],
    "id": "0",
    "priority": 0,
    "table_id": 1,
    "match": {},
    "instructions": {
        "instruction": [
            {
                "order": 0,
                "apply-actions": {
                    "action": [
                        {

```

```

        "order": 0,
        "output-action": {
            "max-length": 0,
            "output-node-connector": "NORMAL"
        }
    ]
},
{
    "id": "d2",
    "priority": 1,
    "table_id": 1,
    "match": {
        "ethernet-match": {
            "ethernet-type": {
                "type": 2048
            },
            "ethernet-destination": {
                "address": "d8:97:ba:08:4f:d2"
            }
        },
        "instructions": {
            "instruction": [
                {
                    "order": 0,
                    "apply-actions": {
                        "action": [
                            {
                                "order": 0,
                                "output-action": {
                                    "max-length": 0,
                                    "output-node-connector": "2"
                                }
                            }
                        ]
                    }
                }
            ]
        }
    ]
}

```