Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра автоматизации систем вычислительных комплексов

Романов Андрей Романович

Разработка системы обеспечения надежного и масштабируемого виртуального сетевого сервиса в облачной среде

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
к.ф.-м.н.
В.А. Антоненко

Москва, 2016

Аннотация

В данной работе рассматриваются проблемы организации надежной работы и масштабируемости виртуального сетевого сервиса.

В рамках выпускной квалификационной работы рассмотрен высокоуровневый стандарт архитектуры NFV платформ ETSI NFV MANO. Проанализированы существующие решения, решающие задачи восстановления сервиса и расширения его инфраструктуры.

Разработан программный модуль, интегрированный в облачную платформу C2 Platform и соответствующий архитектуре ETSI NFV MANO. Модуль позволяет управлять виртуальными сетевыми сервисами и обеспечивает их отказоустойчивость и масштабируемость автоматическом режиме. Проведены эксперименты, подтверждающие автоматическое восстановление корректной работы сервиса в случае возникновения неполадок или в случае нехватки ресурсов.

Оглавление

Введение **5**

1 Постановка задачи **9**

2 Обзор предметной области **10**

2.1 Общее описание NFV 10

2.2 Архитектура ETSI NFV MANO 11

2.2.1 Менеджер виртуальной инфраструктуры 13

2.2.2 Менеджер виртуальных сетевых функций 14

2.2.3 Оркестратор виртуальных сетевых сервисов 16

3 Обзор существующих NFV платформ **18**

3.1 Open Platform for NFV 19

3.2 Cloudify 20

3.3 OpenStack Tacker 22

3.4 OpenBaton 23

3.5 Проприетарные решения 24

4 Исследование и построение решения задачи **25**

4.1 Анализ результатов обзора 25

4.2 Требования к решению 26

4.3 План построения решения задачи 26

5 Описание практической части **28**

5.1 Облачная платформа C2 Platform 28

5.1.1 Графический интерфейс 29

5.1.2 Модуль виртуализации ресурсов 29

5.1.3 Модуль мониторинга 30

5.1.4 Модуль управления виртуальными сетевыми функциями 30

5.2 Описание реализованного модуля 31

6 Экспериментальные исследования **35**

6.1 Эксперимент healing 35

6.1.1 Описание эксперимента 35

6.1.2 Входные данные 35

6.1.3 Ожидаемая реакция 36

6.1.4 Результаты эксперимента 36

6.2 Эксперимент scaling 37

6.2.1 Описание эксперимента 37

6.2.2 Входные данные 37

6.2.3 Ожидаемая реакция 38

6.2.4 Результаты эксперимента 38

6.3 Выводы из экспериментального исследования 39

Заключение **40**

Введение

В современных сетях функционирует огромное количество сервисов: маршрутизация (routing), трансляция сетевых адресов (NAT), сетевой экран (firewall), туннелирование (VPN), проски-сервер и т.д.. Многие из них реализованы в одном физическом устройстве (например, маршрутизатор). Для эффективной работы сервисов нагрузку распределяют сразу на несколько таких устройств.

При возникновении необходимости в дополнительных функциях требуется приобретать оборудование, часть функциональности которого будет избыточной.

Функции, реализованные в составе отдельных сетевых узлов, зачастую плохо масштабируются, так как при увеличении нагрузки на сеть увеличивается число необходимых физических устройств. При обычных (не пиковых) нагрузках часть устройств простаивает. Следовательно, становится актуальным вопрос динамической масштабируемости сервиса в зависимости от его загрузки.

Одной из проблем современных сетей является зависимость от производителя аппаратных устройств. Оборудование разных производителей может конфликтовать между собой. Со временем производители перестают поддерживать устаревшие устройства.

Таким образом, можно выделить ключевые проблемы организации работы

сетевого сервиса:

- использование оборудования с избыточной функциональностью;
- рассчет производительности сервиса исходя из максимальной возможной нагрузки;
- простаивание оборудования в случае, если нагрузка не является пиковой;
- зависимость от производителя оборудования (техническое обслуживание и устаревание оборудования, невозможность модифицировать сервис без вмешательства производителя);

Для введения понятия концепции виртуальных сетевых функций нам потребуется рассмотреть существующие модели обслуживания облачных вычислений:

- Infrastructure-as-a-Service (IaaS) – инфраструктура как услуга;
- Platform-as-a-Service (PaaS) – платформа как услуга;
- Software-as-a-Service (SaaS) – программное обеспечение как услуга;

IaaS - модель, в которой клиенту предоставляется возможность использования облачной инфраструктуры. Пользователь сам управляет программным обеспечением предоставленных ему ресурсов. Контроль и управление физическими ресурсами облака осуществляется облачным провайдером — поставщиком облачных услуг.

PaaS - модель, в которой клиенту предоставляется возможность использования платформы, предустановленной на облачной инфраструктуре. Пользователь в рамках платформы может сам определять и использовать при-

ложения. Примером такой модели предоставления облачных услуг является платформы Google App Engine.[12]

SaaS - модель, в которой клиенту предоставляется возможность использования программного обеспечения облачного провайдера. Отличие от PaaS заключается в том, что в SaaS клиент не может управлять сервисами облака. Контроль и управление физическими ресурсами облака и предоставляемого програмного обеспечения осуществляется облачным провайдером. Основное преимущество модели SaaS для потребителя услуги состоит в отсутствии затрат, связанных с установкой, обновлением и поддержкой работоспособности оборудования и работающего на нём программного обеспечения. [11]

Виртульные сетевые функции (Network Function Virtualization, NFV) — это концепция, позволяющая виртуализировать сетевые сервисы, которые на данный момент реализованы лишь на физических устройствах. Под виртуализацией сетевых сервисов понимается предоставление сетевых услуг в виде программного обеспечения, функционирующего на одной или нескольких связанных виртуальных машинах. NFV работает в рамках модели SaaS. Свойства концепции NFV:

- масштабируемость – в зависимости от загруженности сервиса будет задействована та часть инфраструктуры, которая необходима для корректной работы сервиса;
- надежность – в случае сбоев в работе сервиса будут предприниматься действия по восстановлению его работы в автоматическом режиме;
- высокая скорость развертки сервиса – виртуализация позволяет быстро внедрять новые сервисы, а также развертывать уже существующие сервисы на новой инфраструктуре.

Концепция NFV отделяет программную составляющую сетевых функций

от аппаратной (вычислительные и сетевые ресурсы). Такой подход подразумевает использование физической инфраструктуры, не зависящей от производителя, что требует стандартизации интерфейсов между различными компонентами системы.

Разработкой высокоуровневой архитектуры ETSI NFV Management and Orchestration (ETSI NFV MANO) для NFV платформ занимается организация ETSI. Главной особенностью архитектуры является оптимальное использование инфраструктуры: она выделяется для каждой функции по запросу. Базовыми блоками, из которых строятся виртуальные сетевые сервисы, являются виртуальные сетевые функции (VNF). Платформа на базе ETSI NFV MANO умеет размещать VNF на подконтрольной инфраструктуре. В результате комбинирования блоков VNF получаются виртуальные сетевые сервисы (VNS), которыми пользуются клиенты платформы.

Целью данной работы является разработка модуля управления виртуальными сетевыми сервисами. Модуль должен обеспечивать отказоустойчивость и масштабируемость сервисов в автоматическом режиме.

В разделах 2.1 и 2.2 подробно рассматривается концепция NFV и архитектура ETSI NFV MANO. Далее в разделе 3 приводится обзор существующих NFV платформ. В разделах 5 и 6 приводится описаниия практической части и экспериментов.

Для исследования разработанного решения были проведены эксперименты, подтверждающие автоматическое срабатывание тригерров для масштабирования инфраструктуры сервисов и восстановление работы сервиса после сбоя.

1 Постановка задачи

Целью данной работы является разработка решения, которое управляет жизненным циклом виртуальных сетевых сервисов и обеспечивает их надежную работу и масштабируемость.

Разрабатываемый модуль должен работать в облачной среде. Это означает, что все клиенты виртуальных сетевых сервисов являются виртуальными машинами. Далее под клиентом виртуального сетевого сервиса будем подразумевать виртуальную машину.

2 Обзор предметной области

2.1 Общее описание NFV

Описание концепции NFV рассматривается на основе стандарта [3].

В традиционных сетях используется специализированное оборудование. Дорогостоящая разработка новых аппаратных решений мешает конкуренции в современных сетях, затрудняя выход на рынок новых кампаний.

NFV предполагает использование стандартных серверов и коммутаторов в качестве виртуализированной инфраструктуры для функционирования услуг. Концепция предлагает использование технологий для виртуализации функций в виде составных элементов, которые могут быть связаны для создания телекоммуникационных сервисов.

Таким образом, виртуальная сетевая функция (VNF) – это описание требуемой инфраструктуры, требуемого программного обеспечения, параметров подключения пользователей к этой услуге и т.д.. Заметим, что программное обеспечение, описанное в VNF должно иметь ограниченную и законченную функциональность. Не следует виртуализировать ненадежное программное обеспечение, которое не рационально использует ресурсы инфраструктуры (например, утечки памяти будут способстовать частым перезагрузкам сервиса).

Виртуальный сетевой сервис (VNS) – это некоторое множество связанных

между собой виртуальных сетевых функций. Это конечная услуга, которая будет предоставляться клиентам. Концепция NFV предполагает внутреннее представление VNS как произвольное непустое множество, состоящее из VNF. При этом VNF как-то связаны друг с другом.

По мнению автора, наиболее интересен случай цепочек виртуальных сетевых функций (VNF chaining). В этом случае можно считать каждую VNS как цепочку сетевых функций. Близкую аналогию можно провести с математическим понятием функции. Пусть x - это входящий трафик некоторого объема. Тогда результатом работы сервиса S, состоящего из последовательной цепочки функций f_1, f_2, f_3 будет трафик y, такой что:

$$y = f_3(f_2(f_1(x))) = S(x) \tag{2.1}$$

В результате трафик x трансформировался в трафик y. S - это суперпозиция функций f_1, f_2, f_3.

2.2 Архитектура ETSI NFV MANO

Европейский Институт Телекоммуникационных Стандартов (ETSI) в 2013 году опубликовал высокоуровневые рекомендации по разработке платформы, управляющей виртуальными сетевыми сервисами Network Function Virtualization Management and Orchestration (NFV MANO). Основная цель документа — стандартизировать интерфейсы каждого модуля в платформе.[2] Как показано на Рис. 2.1, в ETSI NFV MANO имеется 3 основных модуля:

1. менеджер виртуальной инфраструктуры (Virtualized Infrastructure Manager, VIM);

2. менеджер виртуальных сетевых функций (Virtual Network Function

Manager, VNFM);

3. оркестратор виртуальных сетевых сервисов (Network Function Virtualization, NFVO).

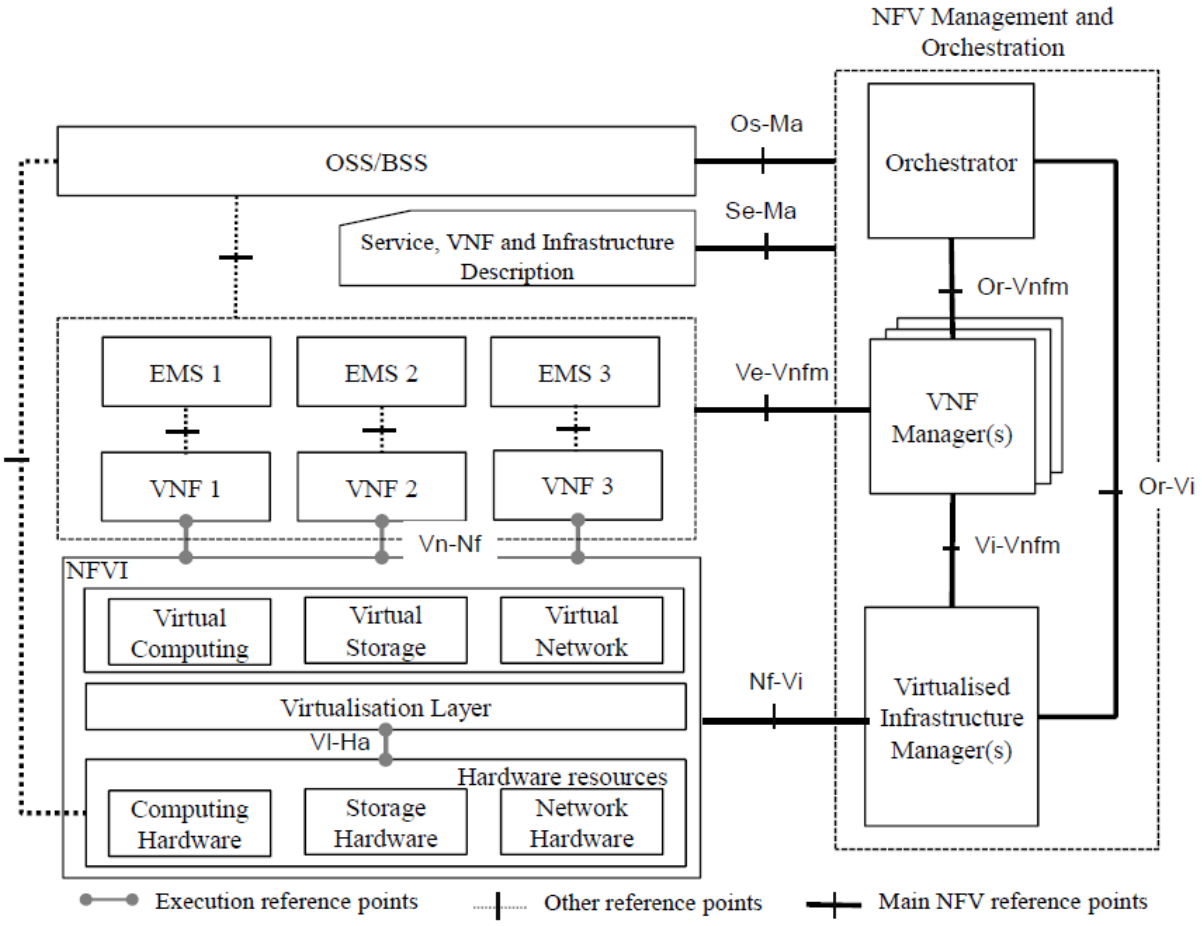

Рис. 2.1: Архитектура NFV Management and Orchestration

Каждый модуль обеспечивает определенный уровень абстракции. VIM занимается виртуализацией физических ресурсов. Менеджер функций предоставляет набор функций, размещенных на виртуальных ресурсах. Оркестратор управляет виртуальными сетевыми сервисами, которые построены на базе виртуальных сетевых функций.

Так же в архитектуре присутствуют неосновные модули:

- описание виртуальных сетевых сервисов, функций и используемой ими

инфраструктуры. В ETSI NFV MANO блоки, которые содержат такие описания, называют каталогами (catalog). В платформах, разрабатываемых на базе ETSI NFV MANO, функции каталогов обычно выполняют менеджеры соответствующего уровня (оркестратор сервисов, менеджер функций, VIM);

- система управления элементами (Element Management System, EMS). Система управляет работой элементов экземпляра виртуальной сетевой функции, отвечает за параметры функции. Данная система взаимодействует с менеджером функций через закрытые интерфейсы. Поэтому в существующих решениях, известных автору, данный модуль включен в состав менеджера функций.

Далее рассмотрим подробнее основные модули архитектуры ETSI NFV MANO.

2.2.1 Менеджер виртуальной инфраструктуры

Менеджер виртуальной инфраструктуры обеспечивает виртуализацию физической инфраструктуры в рамках одного домена. ETSI NFV MANO предполагает использование нескольких менеджеров инфраструктуры в одном домене. В задачи модуля входит:

- управление полным жизненным циклом виртуальных ресурсов в рамках одного домена
  - управление вычислительными ресурсами (computing resource) и хранилищами (storage);
  - управление сетевыми ресурсами (networking resource), то есть управление коммутаторами, роутерами, сетевая настройка;

  - остальные задачи гипервизора (асбтракция физической инфраструктуры, эффективное отображение ресурсов на их виртуальные аналоги и т.д.);

- иметь полную информацию о доступных физических ресурсах и о запущенной виртуальной инфраструктуре;
- мониторинг за состоянием виртуальных ресурсов, обнаружение отказов оборудования, виртуальных машин, програмного обеспечения;
- оповещение остальных модулей о смене состояния виртуальной инфраструктуры;
- предоставление интерфейса для использования виртуальной инфраструктуры и для мониторинга физической инфраструктуры.

Подробнее о функциях VIM, и его спецификациях можно прочитать в стандарте [1].

2.2.2 Менеджер виртуальных сетевых функций

Менеджер виртуальных сетевых функций (VNFM) — это основной модуль архитектуры ETSI NFV MANO, ответственный за полный жизненный цикл виртуальных сетевых функций. Архитектура предполагает возможность наличиия нескольких менеджеров функций.

В ETSI NFV MANO следует отличать два понятия: описание виртуальной сетевой функции (VNF description) и экземпляр виртуальной сетевой функции (VNF instance). Когда говорят про сетевую функцию, чаще всего имеют ввиду ее спецификации, то есть ее описание. Экземпляр VNF — это уже размещенная виртуальная инфраструктура, на которой функционирует программное обеспечение, присутствующее в описании функции. Таким образом,

для каждой VNF существует единственное описание и множество ее экземпляров. Все экземпляры функции независимы друг от друга. В общем случае, они могут быть размещены в разных доменах (в разных VIM).

В задачи модуля входит:

- регистрация и удаление VNF (в случае, если менеджер функций управляет сразу несколькими функциями);
- владение полной информации о спецификациях функции
  - топология инфраструктуры, которую необходимо разместить для работы функции;
  - входные параметры функции;
  - исчерпывающая информация о программном обеспечении виртуальных машин;
  - описание событий, по которым можно судить о состоянии функции (например, в ситуации, когда инфраструктура экземпляра функции недоступна, можно считать, что функция не работает)
  - описание обработчиков на вышеуказанные события (например, перезапустить виртуальную машину в случае, если она не отвечает на команду ping в течении 5 секунд)
- управление жизненным циклом виртуальной функции
- отслеживание неисправностей в работе программного обеспечения функции;
- реакция на неисправности в работе VNF в соответствии с ее описанием (например, применить горизонтальное масштабирование в ответ на событие о недостатке производительности виртуальной машины);

- после размещения виртуальной инфраструктуры инициализировать ее, установить и настроить необходимое для работы функции программное обеспечение;
- обновление программного обеспечения уже размещенных виртуальных сетевых функций;
- оповещение остальных модулей о событиях, связанных с работой функции;

Более подробную спецификацию менеджера виртуальных функций можно посмотреть в стандарте [1].

2.2.3 Оркестратор виртуальных сетевых сервисов

Оркестратор виртуальных сетевых сервисов решает две основные задачи:

- оркестрация ресурсов между несколькими менеджерами инфраструктуры, резервация ресурсов;
- управление жизненным циклом виртуальных сетевых сервисов;

Задача управления виртуальными сервисами нетривиальна. Она включает в себя множество подзадач:

- предоставление интерфейса создания, удаления, изменения, обновления VNS;
- авторизация для управления сетевыми сервисами, разграничение прав доступа;
- синхронизация работы с менеджерами функций;

- отслеживание неисправностей в работе сервисов;
- реакция на события, связанные с изменением состояния сервиса;
- оповещение остальных модулей об изменении состояния сервисов;
- резервация ресурсов под сервисы с помощью модуля VIM;

Более подробный стандарт ETSI NFV MANO архитектуры NFV оркестратора находится в разработке, поэтому информацию о менеджере виртуальных сетевых сервисов можно получить, например, в высокоуровнем стандарте.[1]

3 Обзор существующих NFV платформ

Рассмотрим существующие NFV платформы. Цель обзора — выяснить достоинства и недостатки существующих решений и сформировать требования к разрабатываемому решению. Решения будут сравниваться по следующим критериям:

- соответствие архитектуры платформы стандарту ETSI NFV MANO;
- независимость от платформы виртуализации ресурсов. Данный пункт означает, что решение использует программную прослойку (адаптер) для взаимодествия с платформой виртуализации. При необходимости использовать другую платформу достаточно заменить программную прослойку без переписывания кода основных модулей;
- поддержка работы с несколькими VIM одновременно. Решение, обладающее данным свойством, способно управлять несколькими платформами виртуализации ресурсов одновременно (один модуль VIM для одной платформы виртуализации);
- мониторинг состояния виртуального сетевого сервиса. Поддержка этого свойства позволяет следить за состоянием инфраструктуры виртуальных сетевых сервисов;

- автоматическое срабатывание обработчиков scaling и healing. Scaling – событие, связанное с масштабирование сервиса, healing – событие, связанное с некорректной работой сервиса. Решение, в котором реализован данный пункт, может автоматически запускать обработчики на события scaling и healing, чтобы восстановить работу сервиса. Обработчики на события присутствуют в описании виртуальной сетевой функции.

Выполнение всех критериев, указанных выше, позволит решению выполнять задачи по управлению виртуальными сетевыми сервисами и обеспечивать их отказоустойчивость и масштабируемость в автоматическом режиме.

3.1 Open Platform for NFV

Open Platform for NFV (OPNFV) - это платформа с открытым исходным кодом, на базе которой можно создавать компоненты идеологии NFV. Проект OPNFV фокусируется на разработке менеджера инфраструктуры (NFVI) архитектуры ETSI NFV MANO[6]. Основные цели проекта:

- разработка интегрированной и протестированной открытой платформы, которая может быть использована для построения NFV, ускорения внедрения новых продуктов и сервисов;

- привлечение заинтересованных лиц со стороны конечных заказчиков для удовлетворения требований пользовательского сообщества;

- создание экосистемы NFV решений, основанной на открытых стандартах и программном обеспечении, которое удовлетворяет требованиям конечных пользователей;

- продвигать OPNFV как предпочтительную платформу и сообщество для создания NFV решений с открытым кодом.

OPNFV стремится учавствовать в смежных открытых проектах, которые могут быть использованы в OPNFV, обеспечить целостность, производительность и функциональную совместимость компонентов. OPNFV активно взаимодействует с открытыми проектами: OpenStack, KVM, Open vSwitch, OpenDyalight, ONOS, Open Contrail, ETSI, IETF. Сообщество состоит из более чем 60 компаний, начиная с производителей оборудования и заканчивая поставщиками SDN и NFV решений.[7]

Первый релиз (Arno) состоялся в июня 2015 году и какой-либо функциональности в себе не нес. Вторая версия проекта OPNFV (Brahmaputra) вышла 1 марта 2016 года. По словам сообщества, теперь платформа готова для проведения лабораторных тестов.

Как уже было отмечено, OPNFV - это база для реализации продуктов на базе NFV MANO. В данной платформе разрабатывается лишь модуль NFVI, отвечающий за виртуальные ресурсы. Задачи по масштабируемости и отказоустойчивости здесь выполняются только на уровне виртуальных и физических ресурсов.

3.2 Cloudify

Cloudify - это платформа с открытым исходным кодом. Cloudify архитектурно состоит из основного модуля, называемого Cloudify Manager VM, и Cloudify агентов, установленных на подконтрольных виртуальных машинах.

Cloudify Manager VM исполняет роли сразу двух основных модулей - это VNFM и NFVO. Таким образом, указанный модуль выполняет множество задач:

- регистрация новых виртуальных функций. Описание функций представляется в формате собтсвенной разработки, называемый blueprints.

Он основан на стандарте описания функций TOSCA (формат, основанный на YAML);

- размещение инфраструктуры VNF, используя плагины к существующим платформам виртуализации ресурсов (поддерживаются Openstack, VMware);
- инициализация инфраструктуры функций. Используются программы-агенты на подконтрольных виртуальных машинах;
- мониторинг изменения состояния виртуальных сетевых функций с помощью агентов;
- запуск обработчика события из описания функции;

Cloudify агенты ответственны за выполнения команд Cloudify Manager VM. Различают агентов со стороны Cloudify менеджера (manager side agents) и со стороны виртуальной сетевой функции (application side agents). Агенты менеджера устанавливаются вместе с операционной системой виртуальной машины и выполняют следующие служебные задачи: создание виртуальной машины, привязка внешнего ip-адреса и т.д.. Агенты виртуальной функции являются опцией (устанавливаются, если стоит соответствующая запись в описании функции). Задачи, выполняемые агентами фукнции, должны присутствовать в описании функции.[8]

Изучение платформы Cloudify показало, что в действительности полной автоматизации процесса мониторинга и срабатывание обработчиков событий еще не достигнуто. После размещения функции требуется часть настроек произвести в ручном режиме.

3.3 OpenStack Tacker

Openstack Tacker - это проект с открытым исходным кодом. Является дополнительным модулем для платформы виртуализации Openstack. Использует разработки проекта OPNFV. Основной целью проекта является реализация основных блоков ETSI NFV MANO (VNF-Manager и VNF-Orchestrator) в виде плагина для платформы облачной виртуализации Openstack. Tacker реализует управление виртуальными функциями и оркестрацию сетевых сервисов. Рассмотрим основную функциональность базовых блоков архитектуры ETSI NFV MANO в рамках проекта Openstack Tacker. Основные задачи, выполняемые блоком VNF-Manager:

- хранилище всех виртуальных функций, доступных системе;
- управление полным жизненным циклом каждой виртуальной функции (размещение, инициализация, масштабирование, остановка, удаление);
- мониторинг за размещенными виртуальными функциями. Основные параметры мониторинга: производительность и отказоустойчивость функции;
- автоматическое восстановление работы функции в случае ее полного или частичного отказа в предоставлении услуги по заданным политикам;
- облегчение первоначальной настройки виртуальной сетевой функции;

Задачи, выполняемые блоком VNF-Orchestrator:

- использование шаблонов при управлении сетевыми сервисами, комбинирование различных виртуальных сетевых функций между собой;

- обеспечение эффективного размещения виртуальных функций;
- создание цепочек виртуальных сетевых функций (сетевые сервисы);
- контроль за выделение ресурсов с помощью блока VIM;
- оркестрация виртуальных сетевых функций на множестве различных блоков VIM.

На текущий момент возможности Tacker реализованы только командном интерфейсе и не доступны в графическом интерфейсе Horizon платформы Openstack.[9] Восстановление работы функции и расширение инфраструктуры функции доступно только в ручном режиме. Интеграция с платформой виртуализации Openstack не позволяет использовать другие средства виртуализации инфраструктуры.

3.4 OpenBaton

OpenBaton - проект с открытым исходным кодом, реализующий архитектуру ETSI NFV MANO. Основными модулями платформы являются:

- оркестратор виртуальных сетевых сервисов NFVO;
- менеджер виртуальных сетевых функций VNFM;

Основным модулем, над которым ведется разработка - это NFVO. В нем содержится основная функциональность: размещение функций, слежение за их состоянием, восстановление из аварийного состояния и масштабирование. VNFM - является заменяемым модулем: возможно использование модуля управления функциями собственной разработки. При этом с OpenBaton поставляются библиотеки, позволяющие упростить разработку и интегрирование собственного VNFM с оркестратором.

OpenBaton независима от платформы виртуализации ресурсов. На текущий момент разработан только плагин под платформу Openstack. Разработчиками заявлена поддержка нескольких VIM. В OpenBaton для включения функции мониторинга необходимо дополнительно установить Zabbix сервер (о поддержке других решений по слежению за виртуальными машинами автору не известно).

На момент написания работы в OpenBaton идет разработка следующей функциональности: развертывания дополнительной инфраструктуры и разнообразные улучшения в blueprints. Из этого следует, что о реализации автоматического масштабирования и восстановления функции речи пока не идет.

Для реализации собственных виртуальных сетевых сервисов OpenBaton предлагает либо реализовать менеджер функций собственной разработки, либо привести описании функции через VNFPackage. VNFPackage — это описание функции на основе формата YAML, который содержит необходимое описание виртуальной функции.[10]

3.5 Проприетарные решения

Автору не известны проприетарные решения, реализующие концепцию NFV. Наиболее популярные продукты Microsoft Azure и VMware vSphere работают в рамках модели IaaS. Указанные платформы можно использовать только в качестве менеджера инфраструктуры в рамках архитектуры ETSI NFV MANO.

4 Исследование и построение решения задачи

4.1 Анализ результатов обзора

Результаты обзора, приведенные в таблице 4.1, показали, что ни одно из существующих решений полностью не удовлетворяет установленным требованиям.

Платформа	ETSI NFV MANO	Независимость от платформы виртуализации ресурсов	Одновременная работа с несколькими VIM	Мониторинг состояния VNS	автоматическое срабатывание обработчиков scaling, healing
OPNFV	+	+	-	?	?
Cloudify	+	+	?	+	-
Openstack Tacker	+	-	-	?	?
OpenBaton	+	+	+	+	-

Таблица 4.1: Сравнение существующих NFV решений.

4.2 Требования к решению

В результате анализа существующих NFV платформ сформируем требования к разрабатываемому решению:

1. по запросу осуществлять подписку и отписку пользователей от виртуальных сетевых сервисов в рамках модели Software as a Service (SaaS);
2. при возникновении неисправности принимать меры по восстановлению корректной работы сервиса в автоматическом режиме (healing);
3. обеспечивать масштабируемость инфраструктуры сервиса в автоматическом режиме (scaling);
4. решение должно быть независимым от платформы виртуализации ресурсов;
5. решение должно быть согласовано с высокоуровневой архитектурой ETSI NFV MANO;
6. поддерживать работу с несколькими плафтормами виртуализации ресурсов одновременно;

4.3 План построения решения задачи

Прежде чем перейти непосредственно к решению задачи необходимо понять, какие модули архитектуры ETSI NFV MANO относятся к установленным требованиям.

Оркестратор сетевых сервисов согласно ETSI NFV MANO занимается:

- подпиской и отпиской пользователей от сервиса;

- масштабируемостью инфраструктуры сервиса (scaling) в автоматическом режиме;
- восстановлением корректной работы сервиса при возникновении неисправности (healing) в автоматическом режиме.

Поддержка нескольких модулей VIM также решается на уровне оркестратора. Независимость от платформы виртуализации решается на уровне менеджера инфраструктуры (этот модуль должен использовать плагин для работы с конкретной платформой виртуализации ресурсов).

Таким образом, для решения поставленной задачи достаточно:

1. реализовать оркестратор сетевых сервисов (NFVO) согласно архитектуры ETSI NFV MANO;
2. внедрить оркестратор в платформу, удовлетворяющую которая:
    - обеспечивает виртуализацию ресурсов;
    - обладает средствами мониторинга для слежения за виртуальными машинами и связующими их каналами.

5 Описание практической части

В текущей главе приведено описание реализации модуля VNF-O платформы C2 Platform. В результате реализации такого модуля платформа будет полностью удовлетворять требованиям, описанным в разделе 4.2. Проект разрабатывается отечественной организацией Центр Прикладных Исследований Компьютерных Сетей (ЦПИКС).

5.1 Облачная платформа C2 Platform

Рассматриваемая облачная платформа ориентирована на предоставление услуг по модели IaaS. Основной задачей проекта является управление несколькими платформами виртуализации ресурсов (в частности Openstack). Взаимодействие между внутренними модулями ощуществляется через библиотеку RabbitMQ (RMQ). [13]

Проект C2 Platform состоит из нескольких модулей:

- GUI-client;
- GUI-server;
- модуль виртуализации инфраструктуры (VIM);
- модуль мониторинга Monitoring (Mon);

- менеджер функций (VNF-M, в разработке);
- менеджер сервисов (VNF-O, в разработке);

Далее более подробно будут рассмотрены уже разработанные модули.

5.1.1 Графический интерфейс

Оба модуля (GUI-client и GUI-server) отвечают за:

- отображение актуальной информации о текущей загрузке физических серверов;
- отображение размещенных тенантов (тенант – это сети виртуальных машин);
- авторизацию пользователей;
- предоставление управления виртуальными сетевыми функциями и виртуальными сетевыми сервисами (в разработке);
- отображение актуального состояния сетевых функций и сервисов (в разработке).

Модуль GUI-server будет использовать интерфейс модуля VNF-O для отображения и управления виртуальными сетевыми функциями и сервисами. Заметим, что модуль VNF-O взаимодействуют напрямую только с GUI-server, поэтому в дальнейшем под GUI будем подразумевать модуль GUI-server.

5.1.2 Модуль виртуализации ресурсов

Модуль VIM состоит из двух основных частей: плагин для существующей платформы виртуализации ресурсов (используется Openstack) и независимая

часть для обработки сообщений от других модулей проекта C2 Platform (Mon, VNF-M, GUI-server и т.д.). Плагин можно менять в зависимости от используемой платформы виртуализации ресурсов (Openstack, VMWare и т.д.). Задача второй части заключается в управлении тенантами (сеть с виртуальными машинами) по запросу от других модулей проекта.

Основные интерфейсы для других модулей: размещение и удаление тенанта, предоставление консоли к конкретной виртуальной машине, предоставление информации о размещенных сетях и виртуальных машинах, информация о занятых ресурсах и т.д..

5.1.3 Модуль мониторинга

Модуль мониторинга Mon так же состоит из двух частей: плагин для существующей программы мониторинга (используется Zabbix [14]) и независимая часть для обработки сообщений от других модулей проекта C2 Platform (GUI-server, VNF-M, и т.д.). Плагин можно менять в зависимости от используемой программы мониторинга. Независимая часть обеспечивает слежение за виртуальными машинами, соединениями между ними.

Основные интерфейсы для других модулей: установка и удаление мониторов за виртуальными машинами, оповещение подписанных модулей о событиях, предоставление статистики по работе виртуальных машин.

5.1.4 Модуль управления виртуальными сетевыми функциями

Модуль управления виртуальными сетевыми функциями VNF-M состоит из двух основных частей: анализатор описания виртуальных сетевых функций и часть для взаимодействия с другими модулями проекта C2 Platform

(VNF-O, VNF-M). Модуль умеет регистрировать виртуальные сетевые функции, внося соответствующую информацию в базу данных. Модуль анализирует описание фукнции и занимается конфигурацией ее инфраструктуры.

Основной интерфейс, предоставляемый модулю VNF-O: управление, конфигурирование и поддержка виртуальных сетевых функций.

5.2 Описание реализованного модуля

Модуль VNF-O занимается управлением виртуальных сетевых сервисов. Он состоит из следующих основных классов:

- VNFOrchestrator – класс, инициализирующий все остальные части программы (обработчики сообщений, базу данных, логгирование и т.д.).
- менеджеры сообщений, взаимодействующие с другими модулями платформы:
  - GUIManager – класс, обрабатывающий запросы от модуля GUI;
  - MonitoringManager – класс для отправки запросов на добавление и удаление мониторов за виртуальными машинами и для обработки событий, связанных с функционированием виртуальных машин;
  - VIMManager – класс, запрашивающий управление инфраструктурой сети виртуальных машин;
  - VNFManager – класс, запрашивающий управление виртуальными сетевыми функциями и обрабатывающий запросы на изменение инфраструктуры конкретной виртуальной функции;
- DataBase – класс, отвечающий за взаимодействие с базой данных (БД). Реализует соединение с БД и предоставление сессии для взаимодействия. В качестве БД используется MySQL;

- Messenger – класс, обобщающий взаимодействие между модулями платформы. Занимается отправкой и приемом сообщений используя адаптеры для библиотеки RabbitMQ:
  - RabbitMQConsumer – адаптер для приема сообщений через библиотеку RabbitMQ;
  - RabbitMQPublisher – адаптер для отправки сообщений через библиотеку RabbitMQ;

Как было сказано выше, в рамках данной работы разрабатывался модуль, соответствующий уровню оркестратора сетевых сервисов в архитектуре ETSI NFV MANO. На рисунке 5.1 можно увидеть диаграмму классов разработанного модуля.

Рассмотрим схему взаимодействия классов разработанного модуля на примере подписке клиента на виртуальный сетевой сервис. Подписка клиента осуществляется через серию шагов:

1. GUIManager получает запрос на подписку виртуальный сетевой сервис и начинает его обработку (subscribe_handler);
2. GUIManager с помощью VNFManager запрашивает подписку на каждую виртуальную сетевую функцию, содержащуюся в сервисе (vnf_subscribe_request);
3. в результате предыдущего запроса модуль управления виртуальными сетевыми функциями должен запросить разместить инфраструктуру для каждой функции. Для этого он обращается к разработанному модулю с соответствующим запросом, который обрабатывает VNFManager (update_deploy_handler);

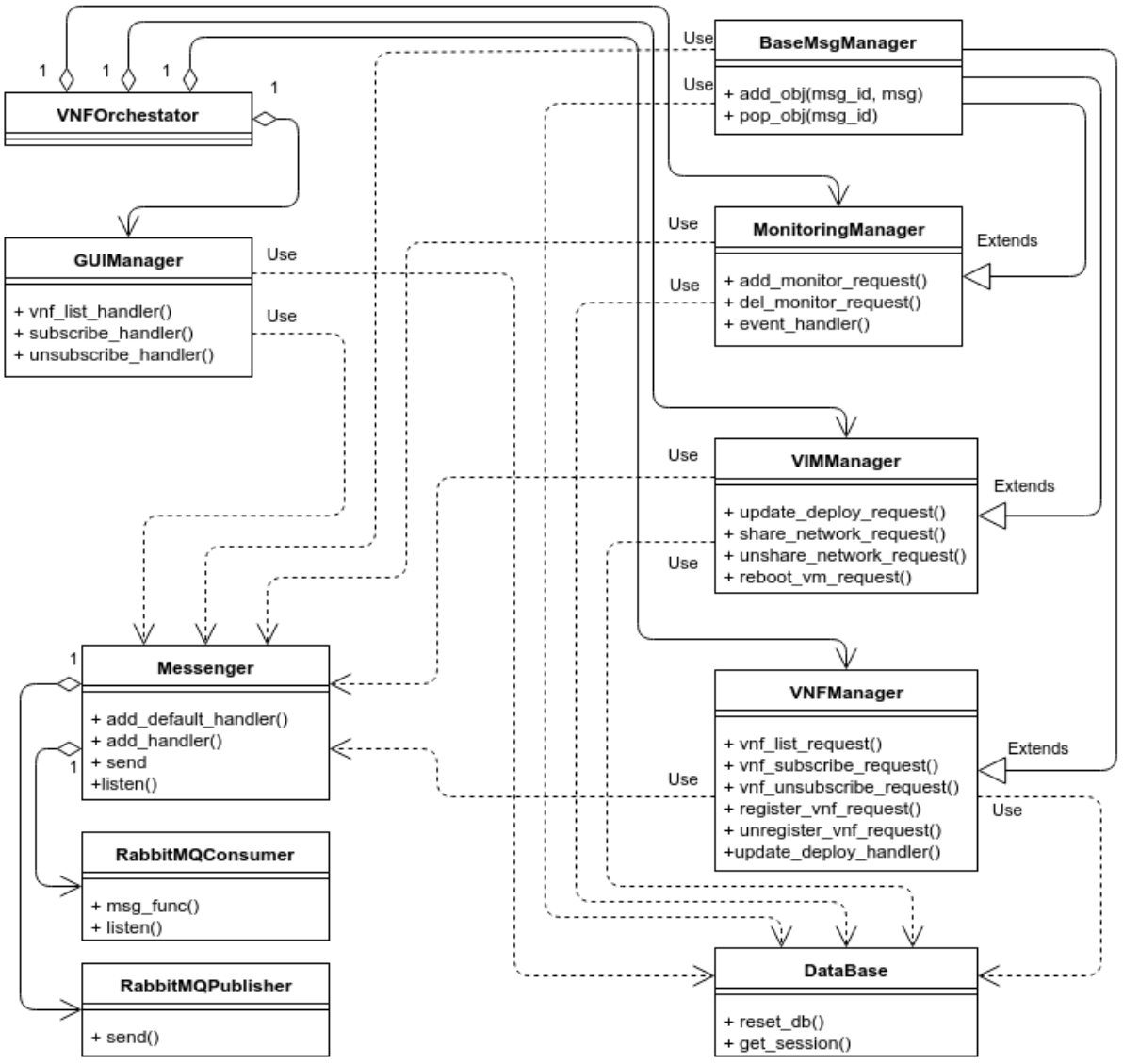

Рис. 5.1: Диаграмма классов модуля реализованного модуля

4. VNFManager выполняет запрос на обновлении инфраструктуры, используя VIMManager (update_deploy_request);

5. после успешного выполнения предыдущего запроса VNFManager передает информацию о размещенной инфраструктуре в модуль управления виртуальными сетевыми функциями;

6. модуль управления функциями сообщает об успешно произведенной настройке всех виртуальных функций, входящих в состав сервиса.

7. GUIManager с помощью MonitoringManager устанавливает мониторы за размещенными виртуальными машинами (add_monitor).

8. GUIManager с помощью VIMManager соединяет виртуальную машину клиента с инфраструктурой виртуального сетевого сервиса (share_network);

9. после успешной подписки клиента на сервис GUIManager сообщает модулю GUI об успешной подписке клиента.

6 Экспериментальные исследования

В данной главе проведем описание экспериментов и полученных в результате их проведения результатов. Целью эксперимента является проверка выполнения требований 2 и 3 из раздела 4.2: автоматическое срабатывание обработчиков на события, связанные со стабильной работой сервиса. Проведем эксперименты healing и scaling, в которых покажем автоматическое восстановление работы сервиса и автоматическое расширение инфраструктуры сервиса.

6.1 Эксперимент healing

6.1.1 Описание эксперимента

Во время использование клиентом виртуального сетевого сервиса происходит отказ в инфраструктуре функции. Оркестратор должен восстановить работу сервиса.

6.1.2 Входные данные

- зарегистрирована виртуальная функция *proxy* в менеджере функций;

- зарегистрирован виртуальной сетевой сервис прокси, состоящий из одной функции *proxy*;
- размещена виртуальная машина клиента в облаке (напомним, что плафторму C2 Platform позволяет размещать виртуальные машины клиентов в своем облаке);
- виртуальная машина клиента подключена к сервису прокси.

6.1.3 Ожидаемая реакция

В результате недоступности одной из виртуальных машин, модуль мониторинга генерирует событие, соответствующее отказу соединения между виртуальной машиной клиента и экземпляром сервиса, и отправляет его оркестратору сетевых сервисов. Оркестратор должен перезапустить ту виртуальную машину, на которой работает программное обеспечение виртуальной сетевой функции. Запрос на перезапуск виртуальной машины отправляется в менеджер инфраструктуры. После получение ответа о успешном перезапуске оркестратор уведомляет клиента о том, что сервис снова доступен.

6.1.4 Результаты эксперимента

Порядок действий, которые произвел оркестратор:

1. получил сообщение от Mon и уведомил клиента о сбое в работе сервиса и о начале ее восстановления;
2. отправил запрос в VIM на разъединение виртуальных машин клиента и функции;
3. отправил запрос в VIM на перезагрузку виртуальной машины с программным обеспечением функции *proxy*;

4. отправил запрос в VIM на соединение виртуальных машин клиента и функции;

5. уведомил клиента о доступности функции.

6.2 Эксперимент scaling

6.2.1 Описание эксперимента

В результате высокой нагрузки инфраструктура сервиса оказывается перегружена. Поэтому клиент испытывает проблемы при работе с сервисом (задержки). Оркестратор должен увеличить объем ресурсов, выделяемых для работы сервиса, чтобы исправить ситуацию.

6.2.2 Входные данные

- оркестратор работает с двумя менеджерами инфраструктуры, каждый из которых управляет ресурсами подконтрольного ему центра обработки данных (ЦОД).
- зарегистрирована виртуальная функция *proxy* в менеджере функций;
- зарегистрирован виртуальной сетевой сервис прокси, состоящий из одной функции *proxy*;
- размещена виртуальная машина клиента в облаке;
- экземпляр функции, использующийся клиентом, размещен на первом ЦОД;
- виртуальная машина клиента подключена к сервису прокси.

6.2.3 Ожидаемая реакция

При обнаружении перегрузки одной из виртуальных машин сервиса, модуль мониторинга генерирует соответствующее событие и отправляет его менеджеру функций. Получив событие о некорректной работе виртуальной машины из функции *proxy*, менеджер считывает описание функции. Согласно описанию менеджер делает запрос в оркестратор на расширение инфраструктуры сервисного тенанта. Оркестратор оценивает запрос менеджера на инфраструктуру и оставшийся запас ресурсов в первом ЦОД. Ресурсов в первом ЦОД оказывается недостаточно, но во втором ЦОД их хватает для удовлетворения запроса. Поэтому оркестратор решает переместить тенант с функцией с первого ЦОД на второй. После получения ответа об успешном размещение функции оркестратор уведомляет клиента о том, что сервис снова доступен.

6.2.4 Результаты эксперимента

Порядок действий, которые произвел оркестратор:

1. получил запрос от VNF-M на расширение инфраструктуры виртуальной функции *proxy*;
2. уведомил клиента о сбое в работе сервиса и о начале ее восстановления;
3. отправил запрос в VIM на разъединение виртуальных машин клиента и функции;
4. удалил старую инфраструктуру функции (одна виртуальная машина);
5. разместил новую инфраструктуру на втором ЦОД (две связанные между собой виртуальные машины);

6. отправил данные о рамещенной инфраструктуре в VNF-M для настройки;

7. получил ответ от VNF-M об успешной инициализации инфраструктуры и подключил клиента к функции с помощью запроса в VIM;

8. уведомил клиента о доступности сервиса.

6.3 Выводы из экспериментального исследования

Экспериментальное исследование показало, что модуль оркестратор успешно справился со своими задачами в обоих экспериментах:

- обеспечил корректное восстановление работы сервиса в автоматическом режиме;

- уведомил клиента о недоступности, а затем о результатах восстановления сервиса;

Заключение

Обзор существующих платформ виртуализации сервисов показал, что ни одно из существующих решений не удовлетворяет поставленной задаче. В рамках данной работы был разработан модуль для платформы C2 Platform, позволяющий автоматизировать процессы восстановления и расширения виртуальных сетевых сервисов. Решение позволяет осуществлять управление подключением пользователей к сервисам.

Литература

1 Network Functions Virtualisation (NFV); Management and Orchestration. URL: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf (дата обращения 01.04.2016)

2 ETSI NFV Management and Orchestration (MANO) простым языком. URL: https://sdnblog.ru/etsi-nfv-mano-beginners-tutorial/ (дата обращения 01.04.2016)

3 Network Functions Virtualisation (NFV); use cases. URL: http://www.etsi.org/technologies-clusters/technologies/nfv (дата обращения 01.04.2016)

4 NFV для корпоративных сервисов - № 12, 2014. URL: http://www.osp.ru/lan/2014/12/13044225 (дата обращения 01.04.2016)

5 NFV виртуализация сетевых функций. URL: http://sci-article.ru/stat.php?i=1455156066 (дата обращения 01.04.2016)

6 Open Platform for NFV (OPNFV). URL: https://www.opnfv.org (дата обращения 01.04.2016)

7 Чем занимается сообщество OPNFV? URL: https://sdnblog.ru/who-is-opnfv/ (01.04.2016)

8 Cloudify Overwiew. URL: http://getcloudify.org/guide/3.1/overview-architecture.html (дата обращения 01.04.2016)

9 Tacker. URL: https://wiki.opfirewallenstack.org/wiki/Tacker (дата обращения 01.04.2016)

10 OpenBaton. URL: http://openbaton.github.io/ (дата обращения 01.04.2016)

11 Модель SaaS в мире и в России. URL: http://www.bytemag.ru/articles/detail.php?ID=12825 (дата обращения 01.04.2016)

12 Google App Engine. URL: https://appengine.google.com/ (дата обращения 01.04.2016)

13 RabbitMQ - Messaging that just works. URL: http://www.rabbitmq.com/ (дата обращения 01.04.2016)

14 Homepage of Zabbix. URL: http://www.zabbix.com/ (дата обращения 01.04.2016)