УДК 004.738

В. А. Антоненко, канд. физ.-мат. наук, мл. науч. сотр., e-mail: anvial@lvk.cs.msu.su,
Р. Л. Смелянский, член-корр. РАН, проф., e-mail: smel@cs.msu.su,
А. А. Ермилов, ст. программист-разработчик, e-mail: aermilov@arccn.ru,
А. Р. Романов, студент, e-mail: xanter@lvk.cs.msu.su,
Н. М. Пинаева, студент, e-mail: pinaeva@lvk.cs.msu.su,
А. В. Плакунов, программист-разработчик, e-mail: artacc@lvk.cs.msu.su,
Московский государственный университет имени М. В. Ломоносова

Платформа управления виртуальными сетевыми функциями C2[1]

Рассматривается проблема построения универсальной облачной платформы. Универсальной в том смысле, что она равно хорошо обеспечивает виртуализацию и управление как сервисов для телеком-операторов, так и сервисов для корпоративных приложений. Показано, что ключевое значение для решения этой проблемы имеет система управления жизненным циклом сервисов. Представлена архитектура облачной платформы C2, в которой реализованы предлагаемые решения. Архитектура этой платформы следует стандарту ETSI NFV MANO. Эффективность полученного решения продемонстрирована на экспериментах, в которых были получены оценки накладных расходов на масштабирование и поддержание доступности виртуальных сервисов.

Ключевые слова: *SDN, NFV, облачная инфраструктура*

Введение

Современные телекоммуникационные сети содержат большое количество проприетарного и, как правило, узкоспециализированного оборудования, созданного для реализации конкретной функциональности. Внедрение новой услуги требует установки нового комплекта оборудования, поддерживающего необходимую функциональность, и его настройки. Такой подход имеет высокие капитальные и операционные затраты, а также не позволяет динамически масштабировать, автоматически настраивать и поддерживать работоспособность сервиса. Использование технологии виртуализации сетевых функций (Network Functions Virtualization, NFV) [1] позволяет решить вышеперечисленные проблемы. Стоит отметить, что технология виртуализации сетевых функций пришла именно из области телекоммуникации.

Виртуализация сетевых функций — это технология, позволяющая за счет виртуализации физических ресурсов (вычислительных, сетевых и хранилищ данных) программно реализовать необходимую функциональность на типовом оборудовании. Тем самым достигается независимость логики сервиса от оборудования, на котором он выполняется. Инженерия "виртуальной сетевой функции" (Virtual Network Function, VNF) зависит от целей, для которых строится инфраструктура виртуализации сетевых функций, от того, кто и для чего строит эту инфраструктуру. Можно условно выделить две основные группы потребителей такой инфраструктуры: телеком-операторы и операторы корпоративных центров обработки данных (далее ЦОД).

Цель виртуализации сетевых функций для телеком-операторов заключается в представлении программными методами сетевых сервисов для анализа, управления и инжиниринга сетевого трафика. Для телеком-операторов виртуальная сетевая функция — это сущность, реализующая функциональность специализированных программно-аппаратных сетевых устройств (так называемых appliance) для коммутирования, маршрутизации, фильтрации, балансировки и тому подобной обработки трафика. Здесь объектом обработки является не запрос/данные пользователя, а трафик, им вызванный. Примерами сервисов телеком-операторов (телко-сервисов)

[1] Данная работа поддержана грантом РФФИ № 16-07-01261, 2016.

могут быть IP телефония, видеоконференцсвязь, EPC, билинг, DPI (Deep Package Inspection), трафик инжиниринг и мониторинг, и т. п.

Вторая группа — это операторы корпоративных ЦОД. В этом случае объектом обработки виртуальных функций являются данные, возникающие в процессе обработки запроса пользователя. Здесь виртуальная сетевая функция — это, например, программа для управления базами данных, web-сервер и любая другая функциональность, нацеленная на обработку запросов пользователей — корпоративных клиентов.

Для корпоративных клиентов важна доступность облачных сервисов. Непрерывность доступа обычно не столь критична. Для телетрафика важна не только доступность, но и непрерывность доступа. Потеря трафика не допустима! Например, если произойдет ошибка на виртуальной машине, где запущен сервер раздачи видео, то виртуальная машина просто будет перезапущена, а для пользователя это лишь небольшое неудобство по причине недоступности видео в течение нескольких секунд. Для телеком-операторов, если абонент звонит в службу спасения с сообщением о пожаре в своей квартире, то не допустимо сбросить его звонок и заставить перезвонить.

Гарантированная производительность — еще одно важное свойство телко-сервисов. Она может быть обеспечена только при наличии предсказуемого количества ресурсов — вычислительных ядер, памяти, сетевых ресурсов, при которых виртуальные машины будут успешно запущены и взаимодействовать между собой с нужной скоростью. Поэтому телеком-операторы накладывают ограничения на время реакции сервиса, что влечет определенные ограничения на размещение как самих сервисов в сети, так и выделяемых под них ресурсов.

Корпоративные клиенты приложений в корпоративных ЦОД не столь требовательны к задержкам прохождения пакетов, хотя и здесь могут быть исключения, например, видеосервисы, приложения безопасности внутри корпоративного ЦОД, такие как межсетевой экран веб-приложений, система предотвращения или обнаружения вторжений и т. д.

Важно понимать, что объекты обработки для этих двух разных видов сервисов (телко и корпоративных) разные: для телко-сервиса объектом обработки является трафик, поданный на вход телко-сервиса; для корпоративных сервисов — запрос пользователя, обработка его данных. Также важно осознавать, что инфраструктура для реализации как телко, так и корпоративных сервисов может быть распределенной, с весьма сложной топологией.

В настоящее время отсутствует унифицированное решение, которое бы равно эффективно поддерживало как телко-сервисы, так и корпоративные сервисы в виде виртуальных сетевых функций. В статье будет показано, что:

- причина этого заключается в различии требований к доступности и производительности телко-сервисов и корпоративных сервисов;
- решение этой проблемы заключается в организации и настройках системы обеспечения жизненного цикла виртуальной сетевой функции, что позволит использовать ресурсы одного ЦОД для обеих категорий пользователей.

В статье представлена архитектура облачной платформы C2, которая следует эталонной модели ETSI NFV MANO [1] и которая является решением сформулированной выше проблемы. Дизайн и архитектура этой платформы полностью соответствуют требованиям виртуальных сетевых функций, реализующих телко-сервисы. Платформа поддерживает полный жизненный цикл виртуальной сетевой функции: создание, настройка, мониторинг, удаление. Как будет показано ниже, предложенная эталонная модель полностью соответствует требованиям поддержки корпоративных сервисов.

Введем несколько определений, следуя [1]. Виртуальная сетевая функция (далее "функция") — программная реализация сетевой функции, которая может быть установлена и запущена в инфраструктуре виртуализации сетевых функций. Виртуальный сервис (сервис) — определенная композиция функций, заданных функциональной и поведенческой спецификацией [1]. Под инфраструктурой виртуальных сетевых функций (Network Function Virtualization Infrastructure, NFVI) понимается программно-аппаратное окружение (физические серверы с установленным программным обеспечением (ПО) виртуализации, объединенные в сеть), в котором размещаются и функционируют функции.

Как видно, эти определения не охватывают случай корпоративного сервиса. Поэтому введем понятие виртуальной прикладной функции (Virtualized Application Function — VAF) как программы с четко определенными внешними интерфейсами и логикой функционирования, предназначенной для обработки запросов, поступающих на ее выходы. Особенность и главное отличие VAF от VNF заключаются в том, что VAF может быть точкой предоставления VNS и подписываться на сервис, состоящий из цепочки VNF.

Везде далее, если не оговорено противное, термин "виртуальная функция" мы будем использовать как для VNF, так и для VAF. Жизненный цикл виртуальной функции охватывает все этапы ее существования — от создания до удаления (рис. 1).

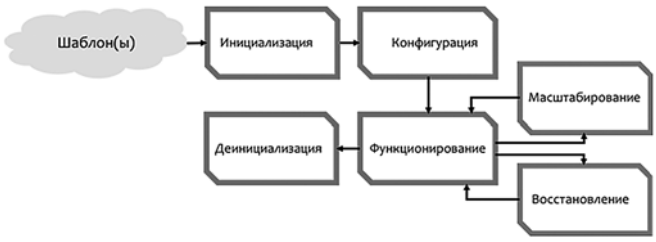

Рис. 1. Стадии жизненного цикла виртуальной функции

Определим экземпляр виртуальной функции как набор виртуальных машин, развернутых из образа операционной системы, с прикладным программным обеспечением, реализующим сервисы виртуальной функции, и сетей, возникающих при инициализации виртуальной функции. Состояние экземпляра функции определяется значениями параметров из заранее определенного списка, за которыми ведется мониторинг, параметры могут отображать различные аспекты работы виртуальных машин или сетей, входящих в состав экземпляра.

Дальнейшая структура статьи такова: в разделе 1 представлен обзор схожих работ в области виртуализации сетевых функций для корпоративного ЦОД и телеком-операторов. В разделе 2 описаны основные модули платформы C2 и взаимосвязи между ними, а также процессы регистрации и поддержки жизненного цикла функции, описание функций на языке TOSCA в виде TOSCA-шаблона. В разделе 3 показано соответствие платформы C2 модели ETSI NFV MANO, выполнены оценки накладных расходов на организацию, масштабирование и восстановление сервиса, показана зависимость пропускной способности каналов и задержек трафика от числа экземпляров функции на его пути и проведены эксперименты с пропускной способностью цепочек функций, включая функции от сторонних поставщиков.

1. Схожие работы

В данном разделе рассматриваются основные решения в области "NFV management and organization (MANO)" платформ на сегодняшний день. Сразу отметим, что в основном в этих проектах не сформулировано, на поддержку какого вида функций (VAF или VNF) ориентирована та или иная платформа. Из статей [5—21] можно сделать вывод, что эти платформы предназначены для применения в ЦОД, а вот конкретно для какого вида сервисов, авторы не уточняют. Тем не менее, как мы уже сказали выше, здесь есть существенные различия и особенности.

Рассмотрим основные проекты реализации MANO с точки зрения следующих критериев: соответствие стандарту ETSI и полнота поддержки жизненного цикла функции.

Концептуально модель MANO состоит из трех уровней: оркестрация сервисов, управление функциями и управление инфраструктурой. Однако не все проекты, которые реализуют MANO, используют данное разделение на уровни с открытыми API. Следовательно, они не являются открытыми системами и могут быть интегрированы, использованы только как единое целое.

Например, в одной из первых реализаций MANO (проект OPNFV [5]) понятие сервиса эквивалентно понятию функции. В проекте Cloudify [6] понятия сервиса и функции, хотя и отличаются, но пользователь не может определять сервис как суперпозицию функций с определенным именем. В проекте Cloudify приходится управлять непосредственно экземпляром каждой функции, даже если они логически находятся в составе единой сервисной цепочки. Следовательно, средства управления и оркестрации этой платформы позволяют работать только с функциями, но с сервисом как суперпозицией функций. Это противоречит эталонной модели MANO.

Проект OPNFV [5] в основном разрабатывает интерфейс для менеджера виртуальной инфраструктуры, подходящий для использования будущими платформами виртуализации сетевых функций. Проект Cloudify [6] в основном сфокусирован на функциональности системы оркестрации и управления жизненным циклом (оркестратор платформы, менеджер функций) на базе различных менеджеров виртуальной инфраструктуры: OpenStack [7], ESXi [8] и Azure [9].

Проект с открытым исходным кодом OpenStack Tacker [10] — это еще одна реализация менеджера функций и оркестратора платформы. В отличие от Cloudify этот проект реализует MANO-платформу в виде расширения облачной платформы OpenStack. Это означает, что он не допускает поддержки других менеджеров виртуальной инфраструктуры. На момент написания статьи не доведены до конца такие возможности Tacker, как: интеграция с пользовательским интерфейсом платформы OpenStack и автоматическое управление восстановлением и масштабированием сетевых функций. Отдельно следует отметить необходимость выделять внешний адрес каждому экземпляру функции, что сокращает область применения системы и накладывает дополнительные требования к инфраструктуре сетевой функции.

Важным является то, как задавать сервис, т. е. как сообщать платформе о его организации и требованиях к виртуальной инфраструктуре. Проект Cloudify для этой цели использует язык спецификаций TOSCA. В Tacker для этих целей разработан собственный язык спецификации на

базе языка YAML, с возможностью поддержки языка TOSCA в будущем.

Для любой реализации MANO важным аспектом является совместимость со средствами мониторинга инфраструктуры ЦОД, например, проекты Zabbix [11] и NAGIOS [12].

Один из наиболее перспективных и гибких вариантов решения проблемы мониторинга сетевых ресурсов ЦОД — это использование программно-конфигурируемого сетевого (ПКС) контроллера [13], который не только может управлять трафик-инжинирингом в сети ЦОД, но и выполнять мониторинг физических и виртуальных сетевых компонентов в режиме реального времени. На момент написания статьи первые шаги в направлении интеграции ПКС контроллера и MANO-платформы сделали только проекты OPNFV, где реализован интерфейс для интеграции с контроллером OpenDayLight [14], и проект OpenContrail [15], где для этих целей используется собственный OpenContrail контроллер.

CORD (Central Office Re-architected as a Datacenter, [16]) — единственная MANO-платформа, ориентированная как на корпоративные ЦОД, так и ЦОД телеком-операторов. В проекте предлагается установить в центральном офисе провайдера платформу виртуализации, реализующую модель издателя-подписчика для сервисов трех типов: сервисы уровня управления (например, контент-ориентированная сеть или виртуальная сеть по запросу), сервиса уровня данных (например, родительский контроль или NAT) и общие облачные сервисы (например, сеть доставки контента или интернет-вещей). Однако проект не объединяет эти три типа сервисов в единой платформе, а предоставляет три типа решений: для провайдеров интернета (R-CORD), для мобильных операторов (M-CORD) и для корпоративных клиентов (E-CORD).

На основании сказанного можно сделать вывод, что ни один из рассмотренных проектов не нацелен на управление виртуальными сервисами как в корпоративных ЦОД, так и в ЦОД телеком-операторов. Платформа виртуализации для телеком-операторов должна уделять особенное внимание качеству сервиса при размещении функций, а также производительности и доступности при поддержке жизненного цикла функции. Платформа для операторов корпоративных ЦОД в главной мере должна озаботиться вопросом стабильности работы функции, тогда как качество сервиса, производительность и доступность отходят на второй план.

Также следует отметить, что публичных демонстраций работы каждой из систем (за исключением проекта Cloudify), с примерами масштабирования и восстановления сетевой функции и с оценкой накладных расходов на построения цепочек сетевых функций отсутствуют.

2. Архитектура облачной платформы C2

Как было сказано во введении, основное отличие телеком NFV MANO-платформ от корпоративных — это то, до какой степени и какие требования к работе сетевых функций учитываются при их оркестрации. Поэтому при разработке архитектуры облачной платформы C2 учитывались требования, предъявляемые как к сервисам телеком, так и к сервисам операторов корпоративных центров обработки данных.

Экземпляром виртуальной функции назовем набор виртуальных машин, соединенных изолированными виртуальными сетями. Под масштабированием экземпляра виртуальной функции будем иметь в виду горизонтальное масштабирование. Другими словами, будем наращивать число экземпляров виртуальной функции в случае нехватки производительности действующих экземпляров. Под восстановлением экземпляра виртуальной функции будем понимать процесс детектирования некорректного функционирования экземпляра и выполнение определенных действий (конфигурация, перезагрузка, выполнения дополнительных команд и скриптов) для приведения экземпляра в нормальное состояние.

Облачная платформа C2 представляет собой расширение проекта Self-Organized Cloud (SOC) [17] и научных проектов, посвященных задачам размещения ресурсов в ЦОД [18, 19]. В проекте SOC функциональность менеджера виртуальной инфраструктуры практически полностью была реализована на базе платформы OpenStack. Это позволяет собирать информацию о доступных MANO-платформе физических ресурсах (ядрах ЦП, оперативной памяти, дисковых подсистемах) и создавать пользователю виртуальные сети — тенанты. Под тенантом будем понимать совокупность из виртуальной сети, виртуальных хранилищ данных и виртуальных машин. Платформа C2 реализует функциональность MANO поверх платформы SOC типа "инфраструктура как услуга" и спроектирована как классическая трехуровневая система MANO, задача реализации которой ложится в основном на MANO-оркестратор платформы, менеджер функций и менеджер виртуальной инфраструктуры.

Структура облачной платформы C2 на уровне основных подсистем представлена на рис. 2.

Ключевым модулем является NFV оркестратор (C2-Orc). Его основная цель — это поддержка жизненного цикла всех экземпляров виртуальных функций. C2-Orc знает, из каких функций состо-

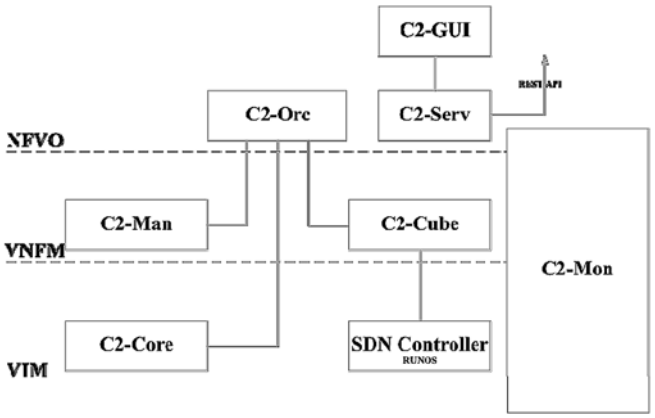

Рис. 2. Соответствие модулей облачной платформы С2 стандарту ETSI NFV MANO

ит каждый сервис, однако не имеет представления о детальном описании состава экземпляра функции (характеристики и число виртуальных машин, виртуальных сетей и т. д.). C2-Orc наблюдает только за состоянием экземпляра виртуальной функции в целом (например, "нормальное" состояние, или "перегружен", или "недоступен").

В случае, когда C2-Orc необходимо инициировать создание нового экземпляра виртуальной функции, он отправляет запрос в Virtual Function (VF) менеджер (C2-Man). Предполагается, что C2-Man заранее принял Topology and Orchestration Specification for Cloud Applications (TOSCA) [3] описание необходимой виртуальной функции. Под TOSCA-описанием понимается шаблон виртуальной функции, который описывает топологию виртуальных машин, состав программного обеспечения, а также политики оркестрации виртуальной функции. TOSCA — расширяемый формат и позволяет описать любое облачное приложение. Спецификация TOSCA описывает шаблон виртуальной функции в терминах "узлов" (подсеть, сеть, сервер или программный компонент) и "отношений" (связи между узлами).

Комбинация спецификаций на TOSCA и образа виртуальной машины (подготовленной специально для виртуальной функции) будем называть C2-Template. Стоит отметить, что часто необходимо ссылаться в TOSCA на внешние по отношению к текущему шаблону скрипты (например, для сложной конфигурации экземпляра виртуальной функции), такие скрипты также должны входить в состав C2-Template.

C2-Man предоставляет C2-Orc детальную информацию о шаблоне экземпляра виртуальной функции. Тот, в свою очередь, перенаправляет данные в менеджер инфраструктуры (C2-Core). Модуль C2-Core работает с системой виртуализации инфраструктуры, используя Python API. На данный момент OpenStack — единственная среда виртуализации инфраструктуры, поддерживаемая облачной платформой С2. Однако уровневая организация платформы С2 с открытым API каждого уровня позволяет утверждать, что она может быть интегрирована с разными средами виртуализации инфраструктуры.

Следует отметить, что функциональности среды виртуализации инфраструктуры недостаточно для оркестрации жизненного цикла виртуальной функции. Например, в модуле Neutron платформы OpenStack отсутствует функциональность "зеркалирования" трафика, нет L2/L3 балансировки трафика. Для этих целей был разработан и реализован модуль C2-Cube, который с помощью ПКС контроллера RUNOS [13] расширяет функциональность выше указанного модуля с точки зрения работы с виртуальными сетями.

Для оперативного мониторинга каждого экземпляра виртуальной функции C2-Orc получает (в режиме реального времени) информацию от модуля C2-Mon. Этот модуль использует систему мониторинга Zabbix. На данный момент — это единственная система мониторинга, поддерживаемая облачной платформой С2, однако в планах команды проекта в будущем добавить поддержки системы мониторинга NAGIOS.

Облачная платформа С2 имеет полнофункциональный графический интерфейс, реализуемый в модуле C2-GUI. Данный интерфейс позволяет конфигурировать, размещать и мониторить экземпляры виртуальных функций, а также подписывать пользователей на сервисы. Графический интерфейс — это клиент-серверное приложение, серверная часть которого — модуль C2-Serv. C2-Serv представляет два типа API: websocket API для связи с C2-GUI и REST API для упрощения интеграции с программной инфраструктурой телком или корпоративного оператора.

3. Экспериментальное исследование

Целями экспериментального исследования являлись: оценка накладных расходов на разворачивание, масштабирование и восстановление функции; оценка зависимости задержки прохождения пакетов через цепочку функций от длины этой цепочки; проведение нагрузочного тестирования на стойке из восьми серверов*; демонстрация реализации C2 MANO на функциях от сторонних поставщиков, интересных как клиентам телеком-операторов, так и операторам корпоративных ЦОД.

* Intel Xeon CPU E5-2640 v2 @ 2.00 GHz, 64 GB RAM, 6.4 TB HDD.

Исследование затрат времени на разворачивание функции. Эксперимент разделялся на три этапа:

1. Создание экземпляров функции.
2. Создание цепочки функций:

— с чистого образа операционной системы (ОС) с установкой нужного ПО и последующей настройкой функции;

— с заранее подготовленного образа с установленным ПО, где требуется только настройка функции.

3. Масштабирование сервиса.

Во всех тестах оценены затраты времени на развертывание конкретной сетевой функции от момента инициализации подписки до момента маршрутизации пользовательского трафика в уже настроенный экземпляр сетевой функции. В табл. 1 представлено время, необходимое на разворачивание инфраструктуры функции.

Исследование затрат времени на поддержку жизненного цикла функции. Данный эксперимент демонстрирует затраты времени на отработку политики масштабирования и восстановление функции. Результаты эксперимента представлены в табл. 2.

Данные эксперимента показывают, что наихудшее время восстановления нормального состояния функции составляет 2 мин 19 с. Строка "Триггер" определяет параметры триггера: триггер восстановления срабатывал после 60 с недоступности по ICMP, триггер масштабирования срабатывал после того, как загрузка центрального процессора (ЦП) машины превышала 60 % в течение 60 с. В таблице показано время, потребовавшееся платформе C2, чтобы восстановить нормальное состояние функции.

Исследование влияния длины цепочки функций на задержку прохождения пакетов. В данном эксперименте для тестирования задержки используется "fping":

fping-e — cNUM — H STEP URL,

где

- NUM — число ICMP запросов. Использовались значения 15, 50 и 100.
- STEP — значение поля Time-to-live (TTL) пакета. Это значение равнялось 70.
- URL — адрес назначения пакета. Назначением был домен "ubuntu.com".

На рис. 3 (см. третью сторону обложки) представлена зависимость задержки прохождения пакета от длины цепочки функций. Длина цепочки указана на горизонтальной оси, средняя задержка — на вертикальной оси.

Результаты эксперимента показывают линейную зависимость между числом функций в цепочке и задержкой прохождения трафика через цепочку. В данном случае каждый экземпляр функции увеличивал задержку менее чем на 1 мс, однако в общем случае задержка может зависеть от реализации самой функции. Стоит отметить, что эти накладные расходы могут быть уменьшены путем оптимизации ядра операционной системы в области работы с сетевым стеком протоколов, например, путем использования Intel DPDK [20].

Таким образом, платформа C2 может создавать достаточно длинные цепочки функций с минимальным влиянием на качество сервиса для пользователя. Это соответствует требованию корпоративных сервисов, а именно, поддерживать большое число простых и разнородных функций.

Таблица 1

Результаты измерения времени разворачивания функции

Сервис	Squid	Snort	Squid-Snort	Snort-Squid
Чистая ОС	37 с	29 с	53 с	52 с
Подготовленная ОС	35 с	28 с	51 с	51 с

Примечание: с — секунды; чистая ОС — инсталляция приложения на виртуальную машину с чистым образом ОС; подготовленная ОС — образ с уже инсталлированным приложением.

Таблица 2

Результаты измерения времени оркестрации функции

Сервис	Squid Чистая ОС	Snort Чистая ОС	Squid Подготов. ОС	Snort Подготов. ОС
Масштабирование (новый экз.)	139 с	Масштабирование недопустимо	135 с	Масштабирование недопустимо
Восстановление (перезагрузка)	73 с	70 с	70 с	70 с
Восстановление (переустановка)	129 с	121 с	126 с	119 с
Триггер (загрузка ЦП)	60 % в течение 60 с	—	60 % в течение 60 с	—
Триггер (ICMP таймаут)	60	60 с	60 с	60 с

Примечание: с — секунды; чистая ОС — инсталляция приложения на виртуальную машину с чистым образом ОС; подготовленная ОС — образ с уже инсталлированным приложением.

Сравнение с другими проектами. В данном эксперименте сравниваются результаты, показанные платформой C2 в трех предыдущих экспериментах с результатами проектов Cloudify и OPNFV. На рис. 4 (см. третью сторону обложки) представлено сравнение времени ответа и времени разворачивания функции между проектами.

В данном эксперименте для вычисления времени ответа также использовалась программа "fping". Эксперимент со временем разворачивания функции для проектов OPNFV и Cloudify проводился следующим образом:

- инсталляция проекта в соответствии с официальной документацией;
- добавление образа ОС для запуска функции в хранилище образов проекта;
- создание TOSCA шаблона в соответствии с документацией и запуск виртуальной машины из образа;
- ручная настройка экземпляра функции.

Результаты эксперимента показывают незначительные отличия во всех значениях, кроме времени размещения и настройки функции. Причина этого отличия заключается в отсутствии возможности автоматической настройки функции в проекте OPNFV и бесплатной версии проекта Cloudify.

Также в проекте OPNFV нет средств мониторинга за состоянием функции, поэтому в случае выхода функции из нормального состояния все действия по восстановлению администратору приходится выполнять вручную.

В проекте Cloudify отсутствует определение цепочки функций и возможность мониторинга за целой цепочкой и за сложными параметрами, такими как список процессов или код ошибки в журнале событий.

Нагрузочное тестирование. Платформа C2 позволяет предоставить функциональность CPE (Customer Premises Equipment) внутри платформы виртуализации сетевых функций. Виртуализация CPE — это способ доставки сетевых сервисов, таких как маршрутизация, межсетевой экран, виртуальные изолированные сети, на корпоративное предприятие в виде программного обеспечения, а не специализированного оборудования. Виртуализация CPE позволяет платформе C2 упростить и ускорить доставку сервисов, удаленную конфигурацию и управление устройствами, а также позволяет пользователям по требованию заказывать новые сервисы и конфигурировать существующие.

В эксперименте проводилось нагрузочное тестирование цепочек из двух функций: межсетевой экран и NAT, в тестовом окружении, состоящем из девяти физических серверов*.

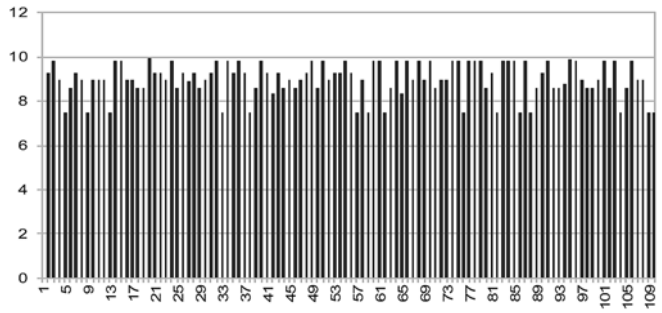

Рис. 5. Разделение пропускной способности внешнего канала между пользователями CPE. По оси *X* — номер клиента, по оси *Y* — средняя загрузка в Мбит/с

Один из серверов был источником трафика, создающего нужное число прямых подключений к платформе C2 по VXLAN-туннелям. Туннель в данном случае эмулирует подключение CPE-устройства и объединяет локальную сеть, находящуюся за CPE-устройством, и виртуальную сеть в платформе в единый сегмент L2. В локальной сети располагалось виртуальное устройство-клиент, генерирующее трафик. Устройство получало необходимые настройки сети (IP-адрес, маску подсети, шлюз, DNS-сервер) от платформы C2. После этого трафик от устройства направлялся через соответствующую цепочку функций.

Сценарий тестирования был следующим:

- создание 111 цепочек функций для пользователей. Такое число пользователей обусловлено количеством ресурсов на всех семи серверах, запускающих виртуальные машины;
- с помощью специального агента наблюдение за тем, как цепочки функций используют канал в 1 Гбит между инфраструктурой платформы и внешней сетью.

Предполагалось, что все цепочки функций поделят пропускную способность канала поровну. Полученные результаты представлены на рис. 5.

Средняя пропускная способность цепочки оказалась равной 8,99 Мбит/с, тогда как суммарная пропускная способность — 989,92 Мбит/с, т. е. использование внешнего канала оказалось близким к оптимальному.

Заключение

В данной работе представлено описание архитектуры платформы C2. Проанализировано различие требований к платформе виртуализации сетевых функций, предъявляемых клиентами телекомоператоров и операторов корпоративных ЦОД.

Был проведен обзор схожих решений, который показал, что ни одна из существующих на момент написания статьи платформ виртуализации сетевых функций не удовлетворяет тре-

* Intel Xeon CPU E5-2640 v2 @ 2.00 GHz, 64 GB RAM, 6.4 TB HDD.

бованиям, определяемым двумя основными вариантами использования, — корпоративных ЦОД и телеком-оператора. Стоит отметить, что в открытых источниках не представлены данные о характеристиках работы рассматриваемых решений (за исключением системы Cloudify), где анализировались бы накладные расходы при масштабировании и восстановлении функций, и проводился анализ накладных расходов на построение цепочек функций.

Эффективность платформы C2 была продемонстрирована в экспериментах, связанных с числом функции в сервисной цепочке. В ходе экспериментального исследования были продемонстрированы возможности облачной платформы C2 и их соответствие стандарту ETSI NFV MANO. Проведена оценка накладных расходов на развертывание сетевого сервиса, оценка времени масштабирования и восстановления сетевого сервиса, оценка ресурсоемкости экземпляра сетевого сервиса и зависимость задержек прохождения пакетов от числа экземпляров сетевого сервиса на физической инфраструктуре.

Результаты экспериментального исследования показывают, что с использованием платформы C2 ресурсы ЦОД эффективно могут быть использованы как в интересах клиентов телеком-операторов, так и операторов корпоративного ЦОД.

Список литературы

1. **ETSI** Network Functions Virtualisation // Introductory White Paper 22 October 2012.
2. **Intel**. Different NFV/SDN Solutions for Telecoms and Enterprise Cloud. 2014.
3. **OASIS**. Topology and Orchestration Specification for Cloud Applications (TOSCA) TC. URL: http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html
4. **Ain't** Markup Language (YAML). Version 1.2. URL: http://www.yaml.org/spec/1.2/spec.html
5. **Linux** Foundation, OPNFV — An open platform to accelerate NFV. October 2014.
6. **Cloudify.** White Paper. URL: http://getcloudify.org
7. **OpenStack.** OpenStack Compute Admin Manual, November 2011.
8. **Waldspurger C.** Memory Resource Management in VMware ESX Server. Operating Systems Design and Implementation (OSDI), 2002.
9. **Microsoft.** Azure. White Paper. URL: https://azure.microsoft.com
10. **OpenStack.** OpenStack Tacker White Paper. URL: https://wiki.openstack.org/wiki/Tacker
11. **Zabbix** Manual. URL: http://www.zabbix.com/downloads/ZABBIX%20Manual%20v1.6.pdf
12. **Nagios** overview. URL: https://www.nagios.org/about/overview
13. **Shalimov A., Nizovtsev S., Morkovnik D., Smeliansky R.** The runos openflow controller // Proceedings of the 4th European Workshop on Software Defined Networks. IEEE Bilbao, Spain, 2015.
14. **OpenDaylight** technical overview. 2013
15. **Overview** of the Contrail system, components and usage // White Paper. January 2014. URL: http://contrail-project.eu/documents/18553/341152/WhitePaper.pdf
16. **Central** Office Rearchitected as a Datacenter (CORD) // White Paper. URL: http://opencord.org/wp-content/uploads/2016/03/CORD-Whitepaper.pdf
17. **Kostenko V., Plakunov A., Nikolaev A., Tabolin V., Smeliansky R., Shakhova M.** Selforganizing cloud platform // In Proceedings of the 2014 international science and technology conference "Modern Networking Technologies (MoNeTec)", Moscow, Russia, 2014.
18. **Kostenko V., Plakunov A.** Ant algorithms for scheduling computations in data processing centers // Moscow University Computational Mathematics and Cybernetics. 2017. Vol. 41, N. 1. P. 44—50.
19. **Vdovin A. V., Zotov I. A., Kostenko V. A., Plakunov A. V., and Smelyansky R. L.** Comparing various approaches to resource allocating in data centers // Journal of Computer and Systems Sciences International, 2014. Vol. 53, N. 5. P. 689—701.
20. **Intel** Corporation, Intel Data Plane Development Kit (Intel DPDK) Programmer's Guide, August 2013.

V. A. Antonenko, Researcher, e-mail: anvial@lvk.cs.msu.su,
R. L. Smeliansky, Professor, e-mail: smel@cs.msu.su, **A. A. Ermilov,** Developer, e-mail: aermilov@arccn.ru,
A. R. Romanov, Student, e-mail: xanter@lvk.cs.msu.su, **N. M. Pinaeva,** Student, e-mail: pinaeva@lvk.cs.msu.su,
A. V. Plakunov, Developer, e-mail: artacc@lvk.cs.msu.su,
Lomonosov Moscow State University

C2: Cloud Platform with NFV Life-cycle Management

The paper presents C2 Platform that covers both the telecommunication and enterprise cloud cases. The C2 architecture incorporates as the network service orchestration as service life-cycle management. C2 implementation follows ETSI NFV MANO standard and uses TOSCA to describe network functions. The paper provides the experiment's methodology and experimental measurements of the platform efficiency, function scaling and healing overheads.

Keywords: *SDN, NVF, Telco, Cloud*

References

1. **ETSI** Network Functions Virtualisation, *Introductory White Paper*, 22 October, 2012.
2. **Intel.** *Different NFV/SDN Solutions for Telecoms and Enterprise Cloud*, 2014.
3. **OASIS.** *Topology and Orchestration Specification for Cloud Applications (TOSCA) TC*, available at: http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html
4. **Ain't** Markup Language (YAML) Version 1.2, available at: http://www.yaml.org/spec/1.2/spec.html
5. **Linux** Foundation, OPNFV — An open platform to accelerate NFV. October 2014.
6. **Cloudify.** White Paper, available at: http://getcloudify.org
7. **OpenStack**, *OpenStack Compute Admin Manual*, November 2011.
8. **Waldspurger C.** Memory Resource Management in VMware ESX Server, *Operating Systems Design and Implementation (OSDI)*, 2002.
9. **Microsoft.** *Azure White Paper*, available at: https://azure.microsoft.com
10. **OpenStack.** *OpenStack Tacker White Paper*, available at: https://wiki.openstack.org/wiki/Tacker
11. **Zabbix** Manual, available at: http://www.zabbix.com/downloads/ZABBIX%20Manual%20v1.6.pdf
12. **Nagios** overview, available at: https://www.nagios.org/about/overview
13. **Shalimov A., Nizovtsev S., Morkovnik D., Smeliansky R.** The runos openflow controller, *Proceedings of the 4th European Workshop on Software Defined Networks. IEEE Bilbao*, Spain, 2015.
14. **Open Day** *light technical overview.* 2013
15. **Overview** of the Contrail system, components and usage. *White Paper.* January 2014, available at: http://contrail-project.eu/documents/18553/341152/WhitePaper.pdf
16. **Central** Office Rearchitected as a Datacenter (CORD) // *White Paper*, available at: http://opencord.org/wp-content/uploads/2016/03/CORD-Whitepaper.pdf
17. **Kostenko V., Plakunov A., Nikolaev A., Tabolin V., Smeliansky R., Shakhova M.** Selforganizing cloud platform, *In Proceedings of the 2014 international science and technology conference "Modern Networking Technologies (MoNeTec)"*, Moscow, Russia, 2014.
18. **Kostenko V., Plakunov A.** Ant algorithms for scheduling computations in data processing centers, Mo*scow University Computational Mathematics and Cybernetics*, 2017, vol. 41, no. 1, pp. 44—50.
19. **Vdovin A. V., Zotov I. A., Kostenko V. A., Plakunov A. V., and Smelyansky R. L.** Comparing various approaches to resource allocating in data centers, *Journal of Computer and Systems Sciences International*, 2014, vol. 53, no. 5, pp. 689—701.
20. **Intel** *Corporation*, Intel Data Plane Development Kit (Intel DPDK) Programmer's Guide, August 2013.

УДК 004.047

С. В. Мальцева, д-р техн. наук, проф., e-mail: smaltseva@hse.ru,
Д. В. Думский, канд. физ.-мат. наук, доц., e-mail: ddumsky@hse.ru,
М. М. Комаров, канд. техн. наук, PhD, доц., e-mail: mkomarov@hse.ru,
Национальный исследовательский университет "Высшая школа экономики", Москва

Исследование рынка новых доменных имен

Представлен анализ и прогнозирование изменений рынка доменных имен, связанных с масштабной программой ICANN по внедрению новых доменов верхнего уровня. Приведена статистика регистраций новых доменов во всем мире и в России в частности на конец 2016 г. Предложены критерии, характеризующие степень реального использования имен в новых доменных зонах и осведомленности пользователей об информационных ресурсах, размещаемых в них. Выявлены некоторые отрицательные и положительные стороны внедрения новых доменов.

Ключевые слова: *доменные имена, gTLD, ICANN*

Введение

Развитие экономики в современном мире тесно связано с развитием телекоммуникаций и связи. Большие объемы данных, присутствующих в экономике, требуют своевременного обмена и обработки и являются двигателем экономического развития не только в нашей стране, но и в целом в мире. Тенденции, связанные с внедрением в последние годы цифровой экономики и с развитием промышленного Интернета и фабрик будущего, предъявляют новые требования к информационным технологиям, в том числе и к существующей базе доступных для регистрации информационных ресурсов доменных имен. Промышленный Интернет тесно связан с таким новым понятием, как Интернет вещей, предполагающий наличие большого числа датчиков для получения информации о состоянии окружающей среды и работающего оборудования, а также постоянный обмен данными между машинами, датчиками и управляющими системами. Корпорация ICANN, управляющая адресным пространством сети Интернет, в октябре 2013 г. утвердила первый список доменов верхнего уровня согласно новому порядку внедрения новых до-