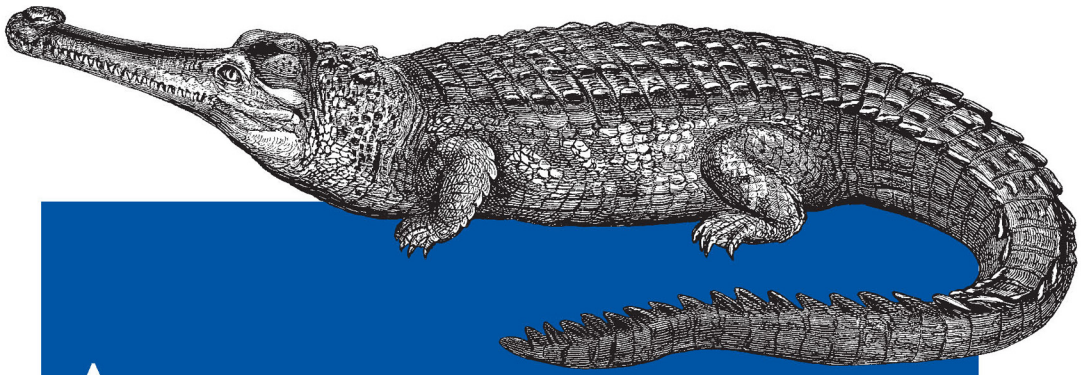
O'REILLY®



Автоматизация программируемых сетей

Джейсон Эделман
Скотт С. Лоу
Мэтт Осуолт

ДМК ПРЕСС
ИЗДАТЕЛЬСТВО

УДК 004.5
ББК 32.812
Э30

Эделман Дж., Лоу С. С., Осуолт М.
Э30 Автоматизация программируемых сетей / пер. с анг. А. В. Снастина. – М.: ДМК Пресс, 2019. – 616 с.: ил.

ISBN 978-5-97060-699-5

Постоянное появление новых протоколов, технологий, моделей доставки и ужесточение требований к интеллектуальности и гибкости бизнес-процессов сделали сетевую автоматизацию чрезвычайно важной. Это практическое руководство наглядно демонстрирует сетевым инженерам, как использовать широкий спектр технологий и инструментальных средств, в том числе Linux, Python, JSON и XML, для автоматизации систем с помощью написания программного кода.

Книга поможет вам упростить выполнение задач, связанных с конфигурированием, управлением и эксплуатацией сетевого оборудования, топологий, сервисов и поддержкой сетевых соединений. Внимательно изучая ее, вы получите основные практические навыки и освоите инструментальные средства, необходимые для сложного перехода к автоматизации сети.

УДК 004.5
ББК 32.812

ISBN 978-1-491-93125-7 (анг.)
ISBN 978-5-97060-699-5 (рус.)

Содержание

Глава 1

Тенденции в современной промышленной эксплуатации сетей

Вам мало знаком термин «программно определяемая сеть» (Software Defined Networking, SDN)? Или вы захвачены массовым увлечением технологией SDN в последние несколько лет? В какую бы категорию вы ни попали, не стоит беспокоиться. Эта книга проведет вас по всем основным темам и поможет начать изучение сетевой программируемости и автоматизации с момента появления SDN. В данной главе представлен обзор тенденций в современной промышленной эксплуатации сетей, при этом особое внимание уделено SDN, ее значимости и воздействию этой технологии на современный мир сетей. Начнем мы с исторического обзора внедрения SDN в основной пул технологий и окончательного формирования направлений сетевой программируемости и автоматизации.

Возникновение технологии программно определяемой сети

Если бы нужно было назвать только одного человека, который изменил всю сетевую индустрию, это был бы Мартин Касадо (Martin Casado), в настоящее время являющийся главным партнером (General Partner) и вкладчиком-инвестором (Venture Capitalist) в венчурном фонде Andreessen Horowitz. Ранее Касадо был действительным членом совета VMware, первым вице-президентом

и генеральным директором подразделения Networking and Security Business в компании VMware. Он оказал огромное воздействие на всю сетевую индустрию не только непосредственным участием в разработках (включая OpenFlow и Nicira), но и прояснением общей ситуации в отношении обязанностей, возлагаемых на крупные сети, а также острой необходимости внесения изменений в функционирование, адаптируемость и управляемость сетей. Рассмотрим эти факты более подробно.

OpenFlow

Как бы то ни было, OpenFlow послужил в качестве первого основного протокола в процессе внедрения и продвижения программно определяемой (или конфигурируемой) сети (Software Defined Network, SDN). Мартин Касадо разрабатывал протокол OpenFlow в процессе получения кандидатской степени (PhD) в Стэнфордском университете (Stanford University) под руководством Ника МакКеона (Nick McKeown). OpenFlow представляет собой протокол, который всего лишь позволяет отделить уровень управления сетевыми устройствами от уровня представления данных (см. рис. 1.1). Проще говоря, уровень управления можно интерпретировать как разум, мозг (brains) сетевого устройства, а уровень представления данных – как аппаратуру (hardware) или интегральную схему специального назначения (application-specific integrated circuits, ASIC), которая действительно выполняет перенаправление (forwarding) пакетов.

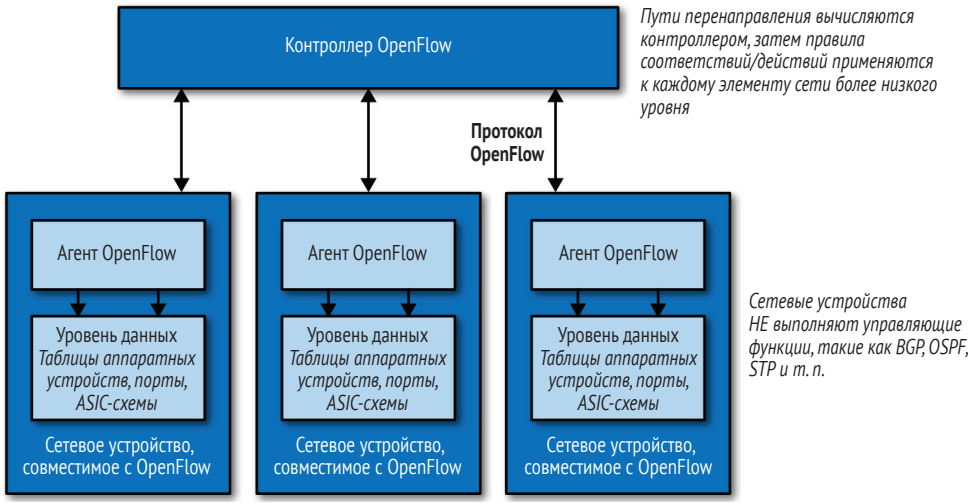


Рис. 1.1 ❖ Разделение уровня управления и уровня данных с помощью протокола OpenFlow

Работа OpenFlow в гибридном режиме

На рис. 1.1 показаны элементы сети, не имеющие уровня управления. Эта схема представляет «чистый» вариант развертывания на основе только протокола OpenFlow. Многие устройства также поддерживают работу OpenFlow в гибридном режиме, то есть протокол OpenFlow может быть развернут на заданном порту, в виртуальной локальной сети (VLAN) или даже в обычном канале перенаправления пакетов таким образом, что если в таблице OpenFlow не найдено соответствие, то используются существующие таблицы перенаправления (MAC-адреса, таблицы маршрутизации и т. п.). Такой режим работы в большей степени похож на маршрутизацию на основе правил (Policy Based Routing, PBR).

Все сказанное выше означает, что OpenFlow представляет собой протокол низкого уровня, используемый для прямого взаимодействия с таблицами аппаратных устройств (например, с информационной базой данных переадресации (Forwarding Information Base, FIB)), которые указывают сетевому устройству, как перенаправлять трафик (например, «трафик на целевой адрес 192.168.0.100 должен исходить из порта 48»).

OpenFlow – это протокол низкого уровня, который управляет таблицами потоков (данных), то есть напрямую воздействует на перенаправление (переадресацию) пакетов. OpenFlow не предназначен для взаимодействия с атрибутами уровня управления, такими как процедура аутентификации или параметры протокола SNMP.

Поскольку таблицы OpenFlow не ограничиваются поддержкой только целевых адресов, в отличие от обычных протоколов маршрутизации, они обеспечивают бóльшую детализацию (соответствие полей в пактах) при определении пути перенаправления. Но этот подход отличается от детализации, предлагаемой маршрутизацией на основе правил (PBR). Подобно OpenFlow в далеком будущем, PBR-маршрутизация позволяет сетевым администраторам перенаправлять трафик на основе «не совсем обычных» атрибутов, таких как адрес источника пакетов. Но при этом для поставщиков сетевого оборудования потребовалось некоторое время, чтобы обеспечить соответствующую производительность при PBR-маршрутизации трафика, поэтому окончательный результат оставался в весьма сильной зависимости от поставщиков. Появление протокола OpenFlow означало, что теперь можно достичь того же уровня детализации в принятии решений о переадресации, но при этом полностью устраняется зависимость от производителей и поставщиков оборудования. Появилась возможность расширить функциональные возможности сетевой инфраструктуры без ожидания следующей версии аппаратных устройств от производителей.

История программируемых сетей

OpenFlow не был самым первым протоколом или технологией, используемой для отделения функций управления и принятия решений от сетевых устройств. Его появлению предшествовала долгая история создания технологий и исследований, тем не менее

именно OpenFlow стал технологией, начавшей «революцию программно определяемых сетей». Предшественниками OpenFlow являются такие технологии, как Forwarding and Control Element Separation (ForCES), Active Networks, Routing Control Platform (RCP) и Path Computation Element (PCE). Чтобы более подробно ознакомиться с историей развития программно определяемых сетей и соответствующих технологий, прочтите статью «The Road to SDN: An Intellectual History of Programmable Networks» (https://www.cs.princeton.edu/courses/archive/fall13/cos597E/papers/sdnhistory.pdf), авторы: Джен Рексфорд (Jen Rexford), Ник Фимстер (Nick Feamster) и Элен Зегура (Ellen Zegura).

Почему именно OpenFlow?

Несмотря на важность понимания того, что представляет собой протокол OpenFlow, еще более важно понять основания и причины, по которым начались исследования и разработка первоначальных технических характеристик OpenFlow и последующее появление программно определяемых сетей.

Мартин Касадо работал на национальное правительство во время своего обучения в Стэнфордском университете. В процессе этой работы возникла необходимость организации ответных действий при атаках, угрожающих безопасности IT-систем (прежде всего IT-систем правительства США). Касадо быстро понял, что можно создать программу, которая управляет компьютерами и серверами именно так, как ему нужно. Реальные примеры использования этой программы никогда не публиковались, но это был тот тип управления конечными точками сети, который позволял предпринимать ответные действия, анализировать и в перспективе перепрограммировать хост или группу хостов в тех случаях, когда это необходимо.

В то время в реальных сетях было почти невозможно выполнить задуманное исключительно программным способом. Каждое сетевое устройство было «закрытым» (например, блокировалась установка любого стороннего программного обеспечения) и предлагало только интерфейс командной строки (command-line interface, CLI). Хотя интерфейс командной строки был и остается широко распространенным и даже предпочитаемым инструментом сетевых администраторов, Касадо ясно понял, что этот инструмент не способен обеспечить гибкость, требуемую для управления, эксплуатации и защиты сети.

В действительности за последние 20 лет способ управления сетями не изменился, за исключением добавления функциональных возможностей с помощью новых команд в интерфейс командной строки. Самым крупным изменением был переход с Telnet на SSH, послуживший основой для шутки, часто используемой SDN-компанией Big Switch Networks на своих демонстрационных слайдах, один из которых можно видеть на рис. 1.2.

Но если говорить серьезно, то технология управления сетями значительно отстала от других технологий, поэтому Касадо в конечном итоге занялся этой проблемой, чтобы изменить положение дел в течение нескольких следующих лет. Скудность средств управления часто лучше осознается при изучении других технологий. Другие технологии почти всегда предлагают более современ-

ные способы управления большим количеством устройств и для управления конфигурацией, и для сбора и анализа данных – например, менеджеры гипервизоров, контроллеры беспроводных сетей, телефонные системы IP PBX, PowerShell, инструменты DevOps – список можно продолжать бесконечно. Некоторые из перечисленных средств являются коммерческим программным обеспечением, то есть собственностью производителя, но прочие инструменты можно свободно применять для управления, эксплуатации и обеспечения гибкости сетей независимо от конкретной платформы.

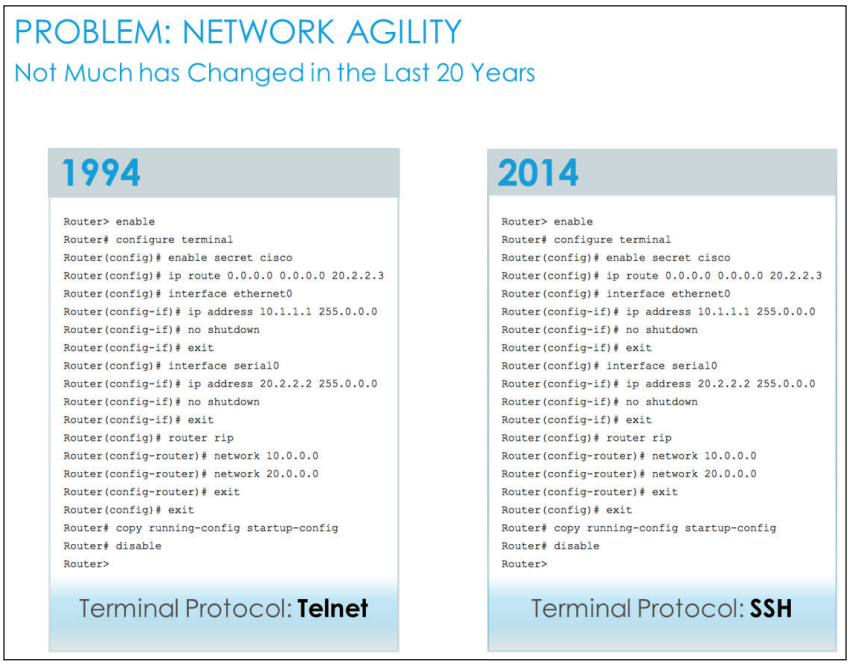


Рис. 1.2 ❖ Что изменилось? Telnet заменили на SSH (источник: Big Switch Networks)

Если вновь вернуться к ситуации, когда Касадо работал на правительство, то возникает вопрос: существовала ли возможность перенаправления трафика на уровне прикладного программного обеспечения? Существовали ли в то время прикладные программные интерфейсы (API) для сетевых устройств? Существовал ли единый пункт обмена информацией для всей сети? На все эти вопросы в большинстве случаев следовал отрицательный ответ. Каким образом появилась возможность программирования сети для динамического управления перенаправлением пакетов, стратегиями и конфигурациями с такой же простой реализацией, как написание программы и выполнение ее на оконечной хост-машине?

Первоначальная спецификация OpenFlow стала результатом исследований Мартина Касадо, который искал пути решения перечисленных выше проблем. Когда ажиотаж вокруг OpenFlow прекратился, поскольку в итоге внимание сосредоточилось в большей степени на вариантах применения и практических решениях, нежели на протоколах низкого уровня, первая работа Касадо стала катализатором для переосмысления всего процесса создания, управления и эксплуатации сетей. Благодарим тебя, Мартин.

Кроме того, это значит, что если бы не было Мартина Касадо, то, возможно, эта книга никогда не была бы написана. Хотя кто знает…

Что такое программно определяемая сеть

В предыдущем разделе был кратко описан протокол OpenFlow, но что такое программно определяемая (или конфигурируемая) сеть (Software Defined Networking, SDN)? Это одно и то же или нет, а может быть, это совершенно разные вещи? Откровенно говоря, SDN – это практически то же самое, чем была облачная среда (Cloud) около десяти лет назад, до тех пор, пока мы не узнали о различных типах облака: инфраструктура как сервис (Infrastructure as a Service, IaaS), платформа как сервис (Platform as a Service, PaaS) и программное обеспечение как сервис (Software as a Service, SaaS).

Есть множество примеров и проектных решений, способствующих пониманию того, чем была облачная среда и чем она стала сейчас, но даже до того, как появились вышеперечисленные термины, могли возникать споры о том, что именно подразумевается при упоминании «облачной среды». Аналогичная ситуация и с термином «программно определяемая сеть». Существуют общедоступные определения, утверждающие, что сетевая среда как «прозрачный ящик» – это SDN или что наличие прикладного программного интерфейса (API) для сетевого устройства – это SDN. Но действительно ли такие характеристики определяют термин SDN? Разумеется, нет.

Вместо попыток определения SDN мы рассмотрим следующие технологии и направления, которые весьма часто ассоциируют с SDN и упоминают при обсуждении SDN:

- OpenFlow;
- виртуализация сетевых функций (Network Functions Virtualization, NFV);
- виртуальная коммутация;
- виртуализация сети;
- прикладные программные интерфейсы (API) устройств;
- автоматизация сети;
- аппаратная коммутация;
- сетевые фабрики центров данных;
- SD-WAN (программно определяемая глобальная сеть);
- специализированные сетевые контроллеры.

Мы преднамеренно не даем определение программно определяемой сети (SDN) в этой книге. Несмотря на то что SDN упоминается в текущей главе, главное внимание здесь сосредоточено на основных направлениях и тенденциях, которые часто классифицируются как SDN. Это сделано для того, чтобы читатель узнал как можно больше о каждом из этих направлений.

Из всех перечисленных направлений основное внимание в книге уделено автоматизации сети, прикладным программным интерфейсам (API) и объемлющим (внешним) технологиям, которые чрезвычайно важны для понимания того, каким образом все эти компоненты объединяются в сетевых устройствах, предоставляющих программируемые интерфейсы в совокупности с современными инструментальными средствами автоматизации и управления.

OpenFlow

Протокол OpenFlow уже был представлен читателю, но есть еще несколько важных фактов, которые следует знать.

Одним из самых главных преимуществ использования протокола, подобного OpenFlow, между контроллером и сетевыми устройствами была реальная независимость провайдера (организатора сети) от программного обеспечения контроллера, иногда обозначаемого как сетевая операционная система (network operating system, NOS), и от виртуальных и физических сетевых устройств более низкого уровня. Тем не менее в действительности сетевые провайдеры, использующие OpenFlow в своих решениях (например, Big Switch Networks, HP, NEC), разработали расширения протокола OpenFlow в соответствии с развитием стандартов и из-за необходимости предоставления специфических дополнительных весьма полезных функциональных возможностей, которые базовая серийная версия OpenFlow предложить не может. Но можно предположить, что в конечном итоге все эти расширения будут включены в будущие версии, реализующие стандарт OpenFlow.

При использовании OpenFlow вам предоставляется больший уровень детализации при управлении маршрутами трафика в сети, но бóльшая власть влечет за собой и бóльшую ответственность. Хорошо, если у вас есть группа разработчиков. Например, корпорация Google на основе протокола OpenFlow развернула глобальную сеть B4, которая увеличила производительность почти на 100%. Для большинства других организаций использование OpenFlow или другого подобного протокола будет менее важным, чем комплексные всеобъемлющие решения, ориентированные на поддержку конкретного типа бизнеса.

Несмотря на то что этот раздел называется OpenFlow, с точки зрения архитектуры он посвящен разделению уровня управления и уровня данных. OpenFlow – это основной протокол, используемый для выполнения данной функции.

Виртуализация сетевых функций

Виртуализация сетевых функций (Network Functions Virtualization, NFV) представляет собой не слишком сложную концепцию сетевой архитектуры. По су-

ществу, она означает развертывание функций, которые ранее обычно выполнялись аппаратурой, в виде программного обеспечения. Самыми известными примерами реализации этой концепции являются виртуальные машины, работающие как маршрутизаторы, сетевые экраны (брандмауэры), балансировщики нагрузки, IDS/IPS, VPN, защитные экраны прикладного уровня (технология WAF) и выполняющие многие другие функции/сервисы.

Появление NFV позволило ослабить казавшуюся незыблемой базовую позицию аппаратуры, стоимость которой могла достигать десятков или даже сотен тысяч долларов с сотнями и тысячами строк команд, и перейти к конфигурированию сети с помощью N компонентов программного обеспечения, то есть с помощью виртуальных устройств. Эти небольшие устройства стали намного более управляемыми, если рассматривать их как отдельные независимые сетевые устройства.

При описанном выше подходе используются виртуальные устройства как форм-фактор для реальных устройств, поддерживающих концепцию виртуализации сетевых функций. Это всего лишь один из примеров. Развертывание сетевых функций как программного обеспечения может происходить во многих различных формах, включая встраивание в гипервизор, в виде контейнера или как приложение, работающее на x86-сервере.

Нередко встречаются случаи развертывания оборудования, потребность в котором с большой вероятностью возникнет в течение будущих трех–пяти лет, так как постепенные обновления слишком сложны и даже являются более дорогостоящими. Таким образом, эксплуатируемую основную стоимость составляет не только оборудование, оно лишь используется для сценариев типа «что, если» для вариантов роста и расширения. Развертывание решений, основанных на программном обеспечении, то есть NFV-решений, предлагает более эффективный способ масштабирования и сведения к минимуму проблемной области критических сбоев всей сети или конкретного приложения при использовании модели «оплата по мере роста» (pay-as-you-grow model). Например, вместо приобретения одного крупного аппаратного межсетевого экрана Cisco ASA вы можете постепенно развертывать виртуальные устройства Cisco ASAv и платить по мере роста. Точно с такой же легкостью можно масштабировать балансировщики нагрузки с помощью новейших технологий, предлагаемых компаниями, подобными Avi Networks.

Если концепция NFV способна предложить столь весомые преимущества, то почему не наблюдается изобилия реально внедренных в эксплуатацию решений и программных продуктов этой категории? По нескольким различным причинам. Во-первых, необходимо полное переосмысление архитектуры сети. При установке единственного монолитного сетевого экрана (как пример) все проходят через этот экран, то есть все приложения и все пользователи, а если не все, то определенная группа, известная системному администратору. В современной модели NFV, где возможно развертывание многочисленных виртуальных сетевых экранов, создается сетевой экран для каждого приложения или

пользователя в противоположность одному общему экрану, «ответственному за все». Такой подход существенно сужает проблемную область критических сбоев, ограничивая ее одним сетевым экраном или любым другим сервисным устройством, а при необходимости внесения изменений или развертывания нового приложения никоим образом не затрагиваются другие сетевые экраны для прочих приложений (или пользователей).

С другой стороны, в более привычном всем мире монолитных устройств чрезвычайно важно наличие единой панели управления стратегией защиты, обычно в форме интерфейса командной строки (CLI) или графического пользовательского интерфейса (GUI). Возможно, такой подход значительно расширяет потенциальную проблемную область критических сбоев, но предлагает администраторам четко определенную стратегию управления, поскольку управление требуется для единственного устройства. При наличии группы или отдела поддержки таких устройств вполне естественным становится выбор монолитно-ориентированного подхода. Сегодня это действительность, но есть надежда, что через некоторое время с появлением более усовершенствованных инструментов, оказывающих помощь в эксплуатации и управлении решениями, основанными на программном обеспечении, в промышленном масштабе мы увидим более широкое практическое применение этого типа технологии. В мире современных автоматизированных сетевых операций и автоматизированного управления сетями все меньшее значение имеет выбор сетевой архитектуры с точки зрения продуктивности выполнения функций, поскольку имеется возможность намного более эффективного управления как одним устройством, так и большим количеством устройств.

Наряду с вопросами управления другим фактором, влияющим на ситуацию, является то, что многие производители уделяют недостаточно внимания продажам своих версий виртуальных устройств. Мы не утверждаем, что у производителей отсутствуют версии с виртуализацией, но обычно эти версии не являются предпочтительными у многих известных производителей сетевого оборудования. Если поставщик начал свою деятельность по продаже сетевого оборудования в течение нескольких последних лет, то можно заметить его резкий переход к модели, основанной на программном обеспечении, с учетом перспективы продаж и получения прибыли.

Как и во многих технологичных отраслях, одним из главных преимуществ концепции NFV также является ее гибкость и адаптируемость. Исключение аппаратных устройств уменьшает время внедрения новых сервисов, поскольку не требуется время на установку и монтирование оборудования и кабелей, на настройку и подключение к существующей сетевой среде. При использовании программной методики время подключения зависит только от времени развертывания новой виртуальной машины в сетевой среде, а присущие такому подходу преимущества можно получать многократно при клонировании и резервировании виртуальных устройств для будущего тестирования, например в рабочих средах восстановления после катастроф (disaster recovery, DR).

Наконец, при развертывании NFV исчезает необходимость маршрутизации трафика через специализированное физическое устройство с целью создания требуемого сервиса.

Виртуальная коммутация

На сегодняшний день самыми известными на рынке средствами виртуальной коммутации являются VMware standard switch (VSS), VMware distributed switch (VDS), Cisco Nexus 1000V, Cisco Application Virtual Switch (AVS) и виртуальный коммутатор с открытыми исходными кодами Open vSwitch (OVS).

Эти коммутаторы весьма часто упоминаются в спорах о сущности SDN, но в действительности они представляют собой программные коммутаторы, которые размещены в ядре гипервизора, предоставляя возможность установления соединений в локальной сети между виртуальными машинами (а в последнее время и между контейнерами). Программные коммутаторы предоставляют такие функции, как распознавание MAC-адреса, и такие возможности, как объединение каналов связи, SPAN и sFlow, аналогичные функциям их физических конкурентов, выполняемым в течение многих лет. Несмотря на то что эти виртуальные коммутаторы часто применяются в более крупных решениях SDN и виртуализации сети, они представляют собой лишь коммутаторы, реализованные в форме программного обеспечения. Хотя сами по себе виртуальные коммутаторы не являются законченным решением, они чрезвычайно важны, так как способствуют прогрессу всей отрасли. Виртуальные коммутаторы создали новый уровень доступа или новое направление в деятельности центров данных. На границах сети уже не применяется физический стоечный (top-of-rack, TOR) коммутатор в аппаратном исполнении с ограниченной гибкостью (с точки зрения развития возможностей/функциональности). С тех пор, как границы сетей стали «программными», благодаря использованию виртуальных коммутаторов появилась возможность более быстрого создания новых сетевых функций, реализуемых в программном обеспечении, следовательно, существенно упростилось распространение стратегий управления по сети. Например, стратегия управления защитой может быть развернута на порту виртуального коммутатора, ближайшего к реальному конечному узлу сети. Коммутатор может быть виртуальной машиной или контейнером, предназначенным для дальнейшего расширения стратегии защиты для всей сети.

Виртуализация сети

Решения, классифицируемые как виртуализация сети, уже стали синонимами SDN-решений. В рамках тематики этого раздела виртуализация сети обозначает исключительно программные решения, основанные на оверлейном протоколе. Самыми известными решениями из этой категории являются VMware NSX, Nuage Virtual Service Platform (VSP) и Juniper Contrail.

Главная характеристика этих решений – оверлейный протокол, такой как Virtual eXtensible LAN (VxLAN), используемый для создания соединений между виртуальными коммутаторами на основе гипервизора. Этот тип соединений

и методика туннелирования (tunneling) обеспечивает совместимость на уровне канала передачи (Layer 2) между виртуальными машинами, существующими на различных независимых физических хостах физической сети, при этом подразумевается, что физическая сеть может быть сетью уровня 2 (Layer 2), уровня 3 (Layer 3) или комбинацией обоих уровней. В результате создается отделенная от физической сети виртуальная сеть, которая предоставляет свободу выбора и гибкость.

Следует отметить, что термин оверлейная сеть (overlay network) часто используется вместе с термином андерлейная (нижележащая) сеть (underlay network). Поэтому следует уточнить, что андерлейная означает нижележащая физическая сеть, созданная с помощью физических кабелей. Оверлейная сеть формируется с использованием решения виртуализации сети, которое динамически создает туннели (tunnels) между виртуальными коммутаторами в пределах центра данных. И снова отметим, что это следует рассматривать в контексте решения виртуализации сети на основе программного обеспечения. Также отметим, что многие решения на аппаратной основе в настоящее время развертываются с помощью VxLAN в качестве оверлейного протокола для создания туннелей уровня 2 (Layer 2, канальный) между стоечными (top-of-rack) устройствами в пределах центра данных уровня 3 (Layer 3, сетевой).

Несмотря на то что оверлейный протокол является лишь элементом реализации решения виртуализации сети, эти решения представляют собой гораздо большее, нежели простой набор виртуальных коммутаторов, объединяемых оверлеями. Обычно это полные, всеобъемлющие решения, предлагающие функции защиты, балансировки нагрузки и обратную совместимость при взаимодействии с физической сетью, имеющей единственный пункт управления (например, контроллер). Часто такие решения также обеспечивают объединение с сервисами лучших компаний в сфере сетевых услуг уровней 4–7 (Layer 4–7), предлагая на выбор ряд технологий, которые можно развернуть на платформах виртуализации сети.

Кроме того, гибкость (адаптируемость) достигается благодаря наличию центральной управляющей платформы, которая используется для динамического конфигурирования каждого виртуального коммутатора и необходимых сервисных компонентов. Напомним, что застой в развитии эксплуатационных характеристик сетей произошел из-за вездесущего преобладания интерфейса командной строки (CLI), предпочитаемого всеми производителями в реальном рыночном мире. При виртуализации сети нет необходимости вручную конфигурировать все виртуальные коммутаторы, так как любое решение упрощает этот процесс, предоставляя централизованный интерфейс (GUI, CLI), а также прикладной программный интерфейс (API), с помощью которого все изменения можно реализовать программно.

Прикладные программные интерфейсы сетевых устройств

В течение нескольких последних лет производители начали понимать, что одного лишь стандартного интерфейса командной строки уже недостаточно

и его использование ограничивает функциональность. Если вы когда-либо работали с любым языком программирования или скриптовым языком, то, вероятнее всего, сможете понять эту проблему. В любом случае, мы более подробно рассмотрим эту тему в главе 7.

Главным проблемным пунктом является то, что скрипты для старых и современных сетевых устройств, использующих только интерфейс командной строки, не возвращают структурированные данные. То есть предполагается, что данные передаются из устройства в скрипт в простейшем текстовом формате (например, вывод команды `show version`), затем требуется написание отдельного скрипта для синтаксического разбора этого текста, чтобы извлечь необходимые атрибуты, такие как время непрерывной работы или версию операционной системы. Даже при незначительных изменениях вывода команд `show` скрипты могут стать неработоспособными из-за неверных правил синтаксического разбора. Хотя такой подход применялся и продолжает применяться всеми администраторами, с технической точки зрения автоматизация стала возможной уже давно, но только сейчас производители постепенно переходят на сетевые устройства, управляемые прикладными программными интерфейсами (API).

Предоставление API исключает необходимость синтаксического разбора неструктурированного текста, так как сетевое устройство возвращает структурированные данные, существенно сокращая время, затрачиваемое на написание скрипта. Вместо парсинга «сырого» текста в поисках времени непрерывной работы или любого другого атрибута возвращается объект, предоставляющий именно ту информацию, которая необходима в данном конкретном случае. При этом не только сокращается время написания скрипта и снижается «входной» уровень требований для сетевых инженеров (и прочих непрограммистов), но также предоставляется более четко определенный интерфейс, с помощью которого профессиональные разработчики программного обеспечения могут быстро разрабатывать и тестировать код почти так же, как они используют API для несетевых устройств. «Тестирование кода» может означать проверку (испытания) новых топологий, сертификацию новых сетевых функциональных возможностей, валидацию конкретных сетевых конфигураций и многое другое. Все эти операции сегодня выполняются вручную, на них затрачивается очень много времени, при этом весьма высока вероятность возникновения ошибок.

Один из первых широко распространенных API-интерфейсов в сетевой сфере представила компания Arista Networks. Ее интерфейс называется eAPI и представляет собой прикладной программный интерфейс на основе протокола HTTP, который использует данные в JSON-кодировке. Разумеется, API на основе протокола HTTP и использование формата JSON будут подробно рассматриваться начиная с главы 5. После Arista компания Cisco представила свои API-интерфейсы, такие как Nexus NX-API и NETCONF/RESTCONF, на конкретных платформах, и появились производители и поставщики, такие как

Juniper, получившие в свое полное распоряжение расширяемый и адаптируемый интерфейс NETCONF, но это не привлекло особого внимания. Следует отметить, что в те времена почти каждый производитель/поставщик предлагал собственный тип API.

Более подробно эта тема будет рассматриваться в главе 7.

Автоматизация сети

По мере развития прикладных программных интерфейсов в сетевой отрасли продолжают появляться все более интересные варианты извлечения преимуществ из их использования. В ближайшем будущем автоматизация сети является главным потенциальным средством получения преимуществ программируемых интерфейсов при использовании современных сетевых устройств, предоставляющих API.

В более широком смысле автоматизация сети – это не только автоматизация процедур конфигурирования сетевых устройств. Разумеется, это самое распространенное представление об автоматизации сети, но использование API и программируемых интерфейсов обеспечивает гораздо бóльшие потенциальные возможности автоматизации, нежели простое манипулирование параметрами конфигурации.

Переход к использованию API делает простым и ясным доступ ко всем данным, скрытым в сетевых устройствах. Здесь имеются в виду данные уровня информационных потоков, таблицы маршрутизации, информационные базы данных переадресации (FIB – Forwarding Information Base), статистические данные интерфейсов, таблицы MAC-адресов, таблицы виртуальных локальных сетей (VLAN), серийные номера устройств – этот список можно продолжать бесконечно. Использование современных методик автоматизации, которые, в свою очередь, применяют API, может оказать оперативную помощь в выполнении повседневных операций управления сетью, сбора данных и автоматизированной диагностики. Кроме того, поскольку современные API позволяют извлекать структурированные данные, администраторы получают возможность выводить и анализировать именно те наборы данных, которые необходимы, и даже объединять выходные данные от нескольких различных команд `show`, значительно сокращая время на отладку и устранение проблем в сети. Вместо установления соединений с N-маршрутизаторами, работающими по протоколу граничного шлюза (BGP – Border Gateway Protocol), с целью проверки правильности конфигурации или устранения возникшей проблемы можно воспользоваться методиками автоматизации для упрощения этого процесса.

Кроме того, применение методик автоматизации в конечном итоге позволяет создать более предсказуемую и единообразную сеть в целом. Это можно наблюдать при автоматизации создания файлов конфигурации, при автоматизации создания виртуальной локальной сети (VLAN) и/или при автоматизации процесса устранения проблем и неисправностей. Это упрощает понимание обобщенного процесса всеми пользователями, поддерживающими конкретную сетевую среду, в отличие от ситуации, в которой каждый сетевой адми-

нистратор должен формировать свою собственную индивидуальную практическую методику работы.

Различные типы автоматизации сети подробно рассматриваются в главе 2.

Аппаратная коммутация

Аппаратную коммутацию (bare-metal switching) тоже часто приравнивают к SDN, но в действительности это ошибочное представление. Ранее уже было отмечено, что в этой главе даются краткие описания различных технологических направлений, которые воспринимаются как SDN, следовательно, и этой теме нужно уделить немного внимания. Если вернуться в 2014 год (и даже немного раньше), то можно обнаружить, что для обозначения аппаратной коммутации в то время использовался термин «коммутация с помощью прозрачного ящика» (white-box switching) или commodity switching. С тех пор термин изменился, и не без серьезных на то оснований.

Прежде чем рассматривать замену «прозрачного ящика» (white-box) на «голую» аппаратуру (bare-metal), необходимо понять, что это означает на самом верхнем уровне с учетом существенных изменений в представлениях о том, что представляют собой сетевые устройства. В последние 20 лет сетевые устройства всегда приобретались как физические устройства, точнее, как аппаратные устройства, сетевая операционная система и инструменты/приложения, которые можно было использовать в сетевой операционной системе. Все эти компоненты приобретались у одного поставщика/производителя.

В соответствии с концепцией прозрачного ящика и «чисто» аппаратных сетевых устройств такое устройство по большей части похоже на x86-сервер (см. рис. 1.3). Это позволяет пользователю отделить друг от друга все требуемые компоненты, предоставляя возможность купить аппаратуру у одного производителя, операционную систему у другого производителя, а затем подбирать и загружать инструменты/приложения от различных поставщиков или даже воспользоваться программным обеспечением с открытыми исходными кодами.

Коммутация по методике прозрачного ящика стала весьма популярной в период повышенного интереса к OpenFlow, поскольку целью этой методики являлось превращение аппаратуры в товар массового потребления и сосредоточение «интеллекта» сети в OpenFlow-контроллере, который сейчас называют SDN-контроллером. В 2013 году корпорация Google объявила о создании собственных коммутаторов под управлением OpenFlow. Тогда это объявление стало предметом многочисленных дискуссий между профессионалами, но в действительности далеко не каждого конечного пользователя можно сравнивать с Google и не каждый пользователь будет создавать собственную программную или аппаратную платформу.

Вместе с этими событиями наблюдалось появление нескольких компаний, сосредоточивших свои усилия на предоставлении решений на основе коммутации по методике прозрачного ящика, например Big Switch Networks, Cumulus Networks и Pica8. Каждая такая компания предлагает решения исключительно

на основе программного обеспечения, но при этом остается необходимость в аппаратуре, на которой должно работать программное обеспечение, чтобы предоставить полноценное завершенное решение. Первоначальным источником этих аппаратных платформ «прозрачного ящика» были ODM-компании[1], такие как Quanta, Super Micro и Accton. Даже если вы работали в сетевой отрасли, вряд ли вам что-либо известно об этих компаниях.

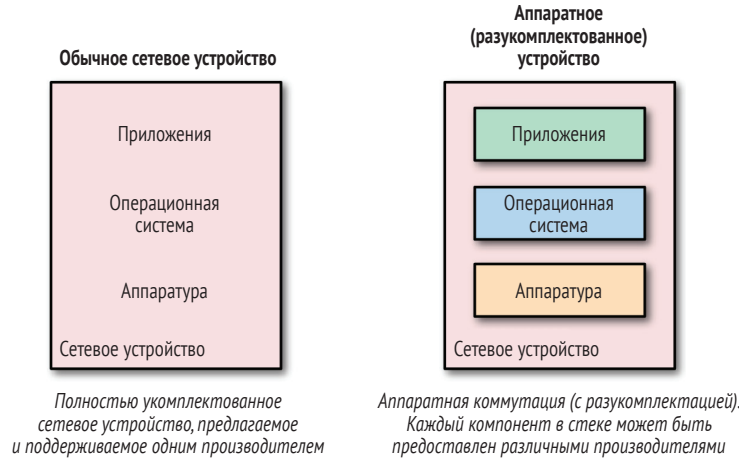


Рис. 1.3 ❖ Общая схема обычного стека коммутации и стека аппаратной коммутации

До тех пор, пока компании Cumulus и Big Switch не объявили о сотрудничестве с крупными корпорациями, в том числе с HP и Dell, не изменялось и название этого направления – с «прозрачного ящика» на «аппаратную коммутацию», но теперь крупные известные производители поддерживали операционные системы от других компаний, таких как Big Switch и Cumulus Networks, на своих аппаратных платформах.

Возможно, до сих пор остается недопонимание того факта, что аппаратная коммутация с технической точки зрения не является SDN, поскольку производители, подобные Big Switch, действуют в обоих направлениях. На возникающие в связи с этим вопросы можно дать простой ответ. Если контроллер объединен с программным решением, использующим протокол типа OpenFlow (но не обязательно OpenFlow), а обмен данными между сетевыми устройствами организован с помощью программного обеспечения, то все это в совокупности очень похоже на программно определяемую сеть (SDN). Именно так действует компания Big Switch – загружает программное обеспечение на «голую» аппаратуру (прозрачный ящик), затем инициализирует работу агента OpenFlow, обеспечивающего обмен данными с контроллером как компонентом этого решения.

[1] ODM – Original Design Manufacturers.

С другой стороны, компания Cumulus Networks предлагает специализированный дистрибутив Linux, ориентированный на работу с сетевыми коммутаторами. Этот дистрибутив, или операционная система, поддерживает реализации обычных стандартных протоколов, таких как LLDP, OSPF и BGP, без каких-либо конкретных требований к контроллеру, то есть обеспечивает сопоставимость и совместимость с сетевыми архитектурами, которые не основаны на SDN.

Приведенное выше описание должно стать свидетельством того, что Cumulus является компанией, создающей сетевые операционные системы и обеспечивающей работу своего программного обеспечения на чисто аппаратных коммутаторах, в то время как Big Switch – компания, создающая SDN на аппаратной основе, где требуется использование SDN-контроллера этой компании, но при этом Big Switch поддерживает инфраструктуру аппаратной коммутации других производителей.

Если подвести краткий итог, то аппаратную коммутацию (коммутатор как прозрачный ящик) можно приблизительно определить как разукомплектацию (разделение на отдельные компоненты) и предоставление возможности приобретения сетевой аппаратуры у одного производителя, а программного обеспечения у другого, право выбора остается за пользователем. В этом случае администраторы получают полную свободу при выборе проектных решений, архитектур и программного обеспечения без необходимости замены аппаратуры; иногда требуется заменить только основную операционную систему.

Сетевые фабрики центров данных

Возникала ли у вас ситуация, когда невозможно обеспечить простое взаимодействие между различными сетевыми устройствами, даже если все они работают по стандартным протоколам, таким как Spanning Tree или OSPF? Если эта проблема вам знакома, то вы не одиноки. Представьте себе сеть центра данных с компактным ядром и отдельными коммутаторами в каждой аппаратной стойке. А теперь подумайте, какие действия необходимо выполнить, когда наступает время обновления.

Существует много способов обновления сетей, подобных вышеупомянутой, но что, если надо обновить только стоечные (top-of-rack, TOR) коммутаторы и при исследовании текущего рынка TOR-коммутаторов был выбран новый производитель или новая платформа? Это вполне обычная ситуация, она возникает достаточно часто. Сам процесс прост – обеспечение взаимодействия между новыми коммутаторами и существующим ядром (разумеется, мы предполагаем, что в ядре имеются доступные порты) и правильное конфигурирование магистрального канала связи по стандарту 802.1Q, если это соединение уровня 2 (канальный уровень), или конфигурирование предпочитаемого протокола маршрутизации, если это соединение уровня 3 (сетевой уровень).

Здесь начинают действовать сетевые фабрики центров данных (data center network fabrics). Именно они призваны полностью изменить представление о работе сетей центров данных.

Сетевые фабрики центров данных ориентированы на изменение образа мышления сетевых операторов: от управления отдельными устройствами

поочередно к управлению всей системой как единым целым. Если вернуться к описанному в начале раздела сценарию, то здесь вряд ли появится возможность замены на TOR-коммутатор от другого производителя, так как коммутатор является всего лишь отдельным компонентом сети центра данных. Но когда сеть развертывается и управляется как единая система, необходимо воспринимать ее как единую систему. Это означает, что процесс обновления должен представлять собой переход от системы к системе, от фабрики к фабрике. В таком контексте фабрики могут полностью заменяться во время процедур обновления, но не допускается замена отдельных компонентов внутри фабрики, по крайней мере, в большинстве случаев. Такой подход возможен, если конкретный производитель/поставщик предоставляет подробный план (маршрут) перехода или обновления и только при использовании аппаратной коммутации (то есть заменяется только аппаратура). Примерами сетевых фабрик центров данных являются Cisco Application Centric Infrastructure (ACI), Big Switch Big Cloud Fabric (BCF) или фабрика и гиперконвергентная сеть (hyperconverged network) компании Plexxi (https://www.plexxi.com/).

В дополнение к интерпретации сети как единой системы сетевые фабрики центров данных также обладают следующими общими характеристиками:

- единый интерфейс для управления и конфигурирования всей фабрики, включая управление стратегиями;
- распределенные по всей фабрике шлюзы, определяемые по умолчанию;
- возможности определения многих различных маршрутов;
- использование одной из форм SDN-контроллера для управления системой.

SD-WAN

В последние два года одним из самых интенсивно разрабатываемых направлений SDN стала методика программно определяемой глобальной сети (Software Defined Wide Area Networking, SD-WAN). За последние годы выросло количество компаний, всерьез занявшихся решением проблем, связанных с глобальными сетями (WAN), в их числе Viptela (совсем недавно поглощенная корпорацией Cisco), CloudGenix, VeloCloud, Cisco IWAN, Glue Networks и Silverpeak.

В сфере глобальных сетей не наблюдалось значительных технологических прорывов со времени перехода с протокола ретрансляции кадров Frame Relay на механизм коммутации MPLS. С появлением широкополосных каналов связи и интернета затраты становятся сопоставимыми с затратами на аналогичные частные линии связи, со временем развитие продолжается с применением VPN-туннелей, связывающих сайты, в итоге все это закладывается в основу следующей важной вехи – глобальной сети (WAN).

Часто применяемые проектные решения для удаленных офисов обычно включают частную линию связи (с универсальным многопротокольным механизмом передачи данных – multiprotocol label switching, MPLS) и/или открытое интернет-соединение. При наличии обоих вариантов интернет обычно используется только как резервный вариант, главным образом для входящего («гостевого») трафика или для данных общего пользования, передаваемых