Эту книгу я посвящаю всем сетевым инженерам, начинающим свою деятельность по автоматизации сети. Я искренне надеюсь, что книга даст вам знания, необходимые для дальнейшего роста и развития. Также я хотел бы поблагодарить Скотта, Мэтта и всю рабочую группу издательства O'Reilly – понимаю, что процесс написания книги оказался более долгим, чем все мы планировали, но в итоге мы сделали это. Спасибо всем, кто сделал замысел реальностью.

– Джейсон Эделман (Jason Edelman)

Я хотел бы посвятить эту книгу Господу, давшему мне мудрость и знания, необходимые для ее написания (Исход 31:3, NIV). Также посвящаю свой труд моей жене Кристэл, без поддержки которой создание этой книги и многие другие вещи были бы невозможными.

– Скотт С. Лоу (Scott S. Lowe)

Я посвящаю эту книгу всем, кто жаждет новых знаний и всегда готов учиться с увлечением – каждое слово в книге написано для вас. Благодарю свою жену Джейми, которая поддерживала мое рабочее настроение и сохраняла нашу бодрость и жизнерадостность, даже когда жизнь становилась слегка сумасшедшей.

– Мэтт Осуолт (Matt Oswalt)

Содержание

Предисловие

Приветствуем всех читателей книги «Программно управляемые сети и их автоматизация».

Изменения в области промышленной эксплуатации сетей постоянны и значительны. Ориентация организаций и специалистов-профессионалов по сетям на восприятие идей и концепций сетевой программируемости и автоматизации сейчас приобретает гораздо большее значение, чем раньше, и стимулируется непрерывным процессом появления новых протоколов, новых технологий, новых моделей доставки, а также потребностями бизнес-процессов в плане большей интеллектуальности и гибкости для обеспечения конкурентоспособности. Но что означает сетевая программируемость и автоматизация? Начнем эту книгу с попытки найти возможности ответа на данный вопрос.

О чем эта книга

В соответствии с названием главной темой книги является сетевая программируемость (программно управляемые сети) и автоматизация. По существу, сетевая программируемость и автоматизация практически является синонимом упрощения задач по конфигурированию, управлению и эксплуатации сетевого оборудования, сетевых топологий и поддержки сетевых соединений. При этом используется множество разнообразных компонентов, в том числе операционные системы, которые в настоящее время гораздо шире применяются в сетевых средах, чем в прошлом, новые методики, например методика непрерывной интеграции (continuous integration), и привлечение инструментальных средств, прежде относящихся только к арсеналу системного администратора (например, средства управления версиями исходного кода и системы управления конфигурацией). Мы полагаем, что все перечисленные компоненты играют важную роль в базовом определении сущности сетевой программируемости и автоматизации, поэтому рассматриваем все эти темы. Цель данной книги – позволить читателям получить основы знаний о сетевой программируемости и автоматизации.

Как организована эта книга

Эту книгу не обязательно штудировать последовательно, страницу за страницей, мы разделили ее на главы таким образом, чтобы читатель с легкостью находил наиболее интересные ему темы. Возможно, правильнее всего будет начать с последовательного чтения первых трех глав, так как в них представлена общая информация, то есть фактически закладывается основа понимания

всей остальной части книги. Далее вы можете свободно переходить к темам, которые вам наиболее интересны или немедленно требуются для практической работы. Мы пытались сохранить относительную независимость глав друг от друга, но, как и в процессе применения любой технологии, это оказалось не всегда выполнимо. Везде, где это возможно, мы приводим перекрестные ссылки, чтобы помочь читателям быстро находить необходимую информацию.

Ниже приводится краткий обзор организации книги по главам:

- глава 1 «Тенденции в современной промышленной эксплуатации сетей» представляет обзор основных событий и тенденций, способствовавших появлению программно определяемых сетей (software defined networks, SDN). Как вы увидите в главе 1, программно определяемая (или конфигурируемая) сеть стала главной первопричиной постоянно возрастающего сосредоточения внимания на сетевой программируемости и автоматизации;
- глава 2 «Автоматизация сети» продолжает обсуждение программно определяемых сетей, начатое в главе 1, но здесь главное внимание уделяется автоматизации сети: истории развития, влиянию сетевой автоматизации на операционные (рабочие) модели (и влиянию операционных моделей на автоматизацию);
- глава 3 «ОС Linux» представляет общий обзор операционной системы Linux. Эта глава не является полным описанием ОС Linux, ее основная задача – быстро ознакомить специалистов по сетям с рабочей средой Linux, с основным набором команд и с сетевыми концепциями, реализованными в Linux;
- глава 4 «Изучение языка программирования Python в сетевом контексте» знакомит специалистов по сетям с языком программирования Python (http://python.org/). Язык Python часто применяется для программируемых сетей и их автоматизации, поэтому в главе рассматриваются многие основные аспекты программирования на Python: типы данных, условные выражения, циклы, работа с файлами, функции, классы и модули;
- глава 5 «Форматы и модели данных» представляет наиболее распространенные форматы данных, которые часто встречаются в проектах автоматизации сетей. Рассматриваются языки обработки данных JavaScript Object Notation (JSON), eXtensible Markup Language (XML), YAML Ain't Markup Language (YAML). Затем описываются концепции моделирования данных и приводится краткое введение в язык YANG, широко распространенный язык моделирования данных для сетевой среды;

Что означает термин «формат данных»

Если для вас это незнакомый термин, не смущайтесь. Формат данных (data format) – это всего лишь способ кодирования или инкапсуляции данных при передаче между двумя пунктами (например, при возврате данных в ответ на вызов функции прикладного программного интерфейса (API)). Форматы данных подробно рассматриваются в главе 5.

- глава 6 «Шаблоны конфигурации сети» – рассматривается использование языков описания шаблонов для создания конфигураций сетевых устройств. Основное внимание сосредоточено на языке шаблонов Jinja, так как он весьма удачно взаимодействует с языком программирования (ЯП) Python. Кроме того, рассматриваются еще два языка описания шаблонов: Mako и ERB. Mako интегрируется с ЯП Python, а ERB используется в основном совместно с ЯП Ruby;
- глава 7 «Работа с сетевыми прикладными программными интерфейсами (API)» – рассматривается роль прикладных программных интерфейсов (application programming interfaces, API) в обеспечении сетевой программируемости и автоматизации. Описываются основные термины и технологии, применяемые в API, а также применение некоторых наиболее распространенных API от конкретных производителей – API для устройств и API для контроллеров – с примерами, наглядно демонстрирующими возможности их использования для обеспечения сетевой программируемости и автоматизации;
- глава 8 «Управление исходными кодами с помощью Git» представляет систему Git (https://git-scm.com) – общеизвестное и широко используемое инструментальное средство управления исходными кодами. Подчеркивается важность управления исходными кодами, рассматриваются возможности использования такой системы с точки зрения сетевой программируемости и автоматизации. Также рассматривается работа с известными онлайновыми сервисами, например GitHub (https://github.com);
- глава 9 «Инструментальные средства автоматизации» подробно описывает использование инструментальных средств автоматизации с открытым исходным кодом, таких как Ansible (http://www.ansible.com/home), Salt (http://saltstack.com) и StackStorm (https://stackstorm.com), и способы применения этих инструментов специально для обеспечения сетевой программируемости и автоматизации;
- глава 10 «Непрерывная интеграция» описывает концепции непрерывной интеграции (continuous integration, CI) и соответствующие основные инструментальные средства и технологии. Рассматривается применение методики разработки через тестирование (test-driven development, TDD), инструментов исследования и контроля, а также комплексных сред разработки, таких как Jenkins и Gerrit, приводится пример формирования потока сетевой автоматизации, включающий все элементы методики непрерывной интеграции;
- глава 11 «Формирование культуры автоматизации сети» объясняет, почему правильно сформированная культура чрезвычайно важна и почему культура является основополагающим элементом для автоматизации сети. Рассматривается процесс формирования такой культуры;
- приложение А «Дополнительная информация о поддержке сетевой среды в ОС Linux» продолжает обсуждение, начатое в главе 3, но с более подробными описаниями работы с сетевыми интерфейсами macvlan, орга-

низации сетей с помощью виртуальных машин (VM), работы с сетевыми пространствами имен и с сетями, использующими контейнеры Linux (включая и контейнеры Docker (https://www.docker.com)). Также рассматривается использование Open vSwitch (OVS) (http://openvswitch.org);

- приложение Б «Использование NAPALM» представляет введение в использование библиотеки NAPALM (Network Automation and Programmability Abstraction Layer with Multi-vendor support), написанной на ЯП Python. Рассматривается применение NAPALM для управления конфигурацией, не зависящей от конкретных производителей, а также для извлечения данных из сетевых устройств. Кратко описываются возможности интеграции NAPALM с инструментальными средствами Ansible, Salt и StackStorm, которые подробно рассматривались в главе 9.

Для кого предназначена эта книга

Как уже было сказано ранее, главная цель этой книги – предоставить читателям основополагающие знания и набор основных практических навыков в области сетевой программируемости и автоматизации. Мы надеемся, что представители различных профессий в области информационных технологий извлекут пользу из чтения нашей книги.

Сетевые инженеры

Вполне естественно, что бóльшую часть читателей книги, посвященной сетевой программируемости и автоматизации, составляют «обычные» сетевые инженеры, то есть специалисты, достаточно хорошо разбирающиеся в сетевых протоколах, в конфигурации сетевых устройств и в вопросах функционирования и управления сетями. Мы надеемся, что книга поможет сетевым инженерам сделать свою повседневную работу более эффективной и производительной с помощью автоматизации и практического использования программируемости сетей.

Предварительные требования

Сетевым инженерам, заинтересованным в более глубоком изучении сетевой программируемости и автоматизации, не требуются какие-либо предварительные знания в области разработки ПО, программирования, автоматизации или инструментальных средств, связанных с DevOps. Единственное предварительное требование – свободное мышление и восприятие без предвзятости и желание изучать новые технологии и их влияние на деятельность специалистов по сетям и на всю сетевую индустрию в целом.

Системные администраторы

Системные администраторы, ответственные в основном за управление системами, подключенными к сетям, возможно, уже обладают некоторым предварительным уровнем знаний и практическим опытом использования инструмен-

тальных средств, рассматриваемых в книге (в особенности ОС Linux, систем управления исходным кодом и систем управления конфигурацией). Таким образом, книга может помочь им расширить знания и понимание функциональности таких инструментов, представляя их в различных контекстах (например, использование Ansible для конфигурирования сетевого коммутатора в сравнении с использованием Ansible для конфигурирования сервера, работающего под управлением дистрибутива Linux).

Предварительные требования

В этой книге не рассматриваются и не объясняются базовые сетевые протоколы и концепции. Но мы предполагаем, что многие системные администраторы, постоянно занимающиеся управлением подключенных к сетям систем, обладают достаточным уровнем знаний об основных сетевых протоколах, поэтому хорошо подготовлены к чтению. Если вы не вполне уверены в качестве своих знаний о работе в сетях, рекомендуем предварительно изучить одну из книг, подробно описывающих основные сетевые концепции и принципы функционирования. Например, неплохим вариантом является книга «Packet Guide to Core Network Protocols» издательства O'Reilly.

Разработчики программного обеспечения

Разработчики программного обеспечения также могут получить много полезной информации при чтении этой книги. Многие разработчики уже обладают опытом работы с языками программирования и средствами разработки, обсуждаемыми в книге (например, Python и Git). Как и для системных администраторов, для разработчиков может оказаться полезным рассмотрение использования средств разработки и языков программирования в применении к сетевым средам (например, описание возможностей Python для извлечения и сохранения специализированных сетевых данных).

Предварительные требования

Предполагается, что читатели обладают базовыми знаниями об основных сетевых протоколах и концепциях, поскольку все предлагаемые в книге примеры выполняются исключительно в сетевой среде. Если для разработчиков ПО сети являются новой темой, то, как и системным администраторам, мы предлагаем предварительно изучить одну из книг по основным сетевым концепциям.

Инструменты, используемые в книге

В области сетевой программируемости и автоматизации, как и в любой технической отрасли, существует множество версий и вариаций технологий и инструментальных средств. Поэтому мы придерживаемся общепризнанных стандартов при рассмотрении инструментальных средств в данной книге и выбираем самые лучшие, по нашему мнению, из этих инструментов, чтобы читатели сами оценили их полезность для своей практической деятельности. Например, при

наличии множества разнообразных дистрибутивов Linux основное внимание мы все же уделяем дистрибутивам Debian, Ubuntu (производному от Debian) и CentOS (производному от Red Hat Enterprise Linux (RHEL)). Чтобы облегчить восприятие информации для читателей, мы рассматриваем конкретную версию инструментального средства в каждой соответствующей главе.

Онлайн-ресурсы

Мы понимаем, что в одной книге невозможно представить весь материал, полностью освещающий такую обширную тему, как сетевая программируемость и автоматизация. Поэтому на протяжении всей книги мы многократно ссылаемся на дополнительные онлайн-ресурсы, которые могут оказаться весьма полезными при более глубоком изучении концепций, принципов и практических навыков, рассматриваемых в этой книге.

Условные обозначения и соглашения, принятые в книге

В книге используются следующие типографские соглашения:

Курсив – используется для смыслового выделения важных положений, новых терминов, URL-адресов и адресов электронной почты в интернете, имен команд и утилит, а также имен и расширений файлов и каталогов.

`Моноширинный шрифт` – используется для листингов программ, а также в обычном тексте для обозначения имен переменных, функций, типов, объектов, баз данных, переменных среды, операторов, ключевых слов и других программных конструкций и элементов исходного кода.

`Моноширинный полужирный шрифт` – используется для обозначения команд или фрагментов текста, которые пользователь должен ввести дословно без изменений.

`Моноширинный курсив` – используется для обозначения в исходном коде или в командах шаблонных меток-заполнителей, которые должны быть заменены соответствующими контексту реальными значениями.

Такая пиктограмма обозначает указание или примечание общего характера.

Эта пиктограмма обозначает предупреждение или особое внимание к потенциально опасным объектам.

Такая пиктограмма обозначает совет или рекомендацию.

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте www.dmkpress.com, зайдя на страницу книги, и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com на странице с описанием соответствующей книги.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг — возможно, ошибку в тексте или в коде, — мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от огорчения и поможете нам улучшить последующие версии этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и O'Reilly очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли применить санкции.

Пожалуйста, свяжитесь с нами по адресу электронной почты dmkpress@gmail.com со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, помогающую нам предоставлять вам качественные материалы.

Благодарности

Создание этой книги было бы невозможным без помощи и поддержки большой группы людей.

Во-первых, мы хотели бы выразить свою благодарность чрезвычайно активному сообществу людей, объединенных идеей сетевой автоматизации. Назвать всех по именам невозможно, потому что их слишком много, но эти люди создали проекты с открытым исходным кодом, такие как NAPALM и Netmiko, всячески способствовали вовлечению новичков в процесс обучения сетевой автоматизации и постоянно делились своими знаниями и опытом с другими. Спасибо всем участникам этого сообщества за их усилия и за участие в создании этой книги.

Сотрудничающие с нами авторы помогли сделать книгу более совершенной и всеобъемлющей, и мы не смогли бы добиться такого результата без их содействия, поэтому мы выражаем огромную благодарность за помощь. Мирча Улинич (Mircea Ulinic) написал раздел SaltStack для главы об инструментальных средствах управления сетевой конфигурацией, а Джере Джулиан (Jere Julian) предоставил материал о Puppet, который мы, к сожалению, не смогли включить в текущую версию книги. Спасибо, Мирча и Джере.

Технические рецензенты весьма серьезно относились к проверке содержимого книги, соблюдая баланс между точностью и правильностью излагаемого материала и легкостью восприятия этого материала читателем. Мы выражаем свою благодарность Патрику Огенстаду (Patrick Ogenstad), Ахилу Белу (Akhil Behl), Эрику Чоу (Eric Chou) и Сринивасу Макаму (Sreenivas Makam). Спасибо за то, что помогли!

Список людей, заслуживающих благодарности, был бы неполным без упоминания в нем сотрудников O'Reilly Media: Вирджиния Уилсон (Virginia Wilson) и Куртни Аллен (Courtney Allen) – редакторы, Дуайт Рэмси (Dwight Ramsey) – литературный редактор, Рэчел Монэхен (Rachel Monaghan) – корректор, Джуди МакКонвил (Judy McConville) – составитель предметного указателя, Колин Коул (Colleen Cole) – технический редактор, Рэнди Комер (Randy Comer) – дизайнер обложки и Ребекка Демаре (Rebecca Demarest) – художник-иллюстратор. Важность их помощи на всем пути книги от идеи до воплощения невозможно переоценить, и мы благодарим этих людей за их увлеченность работой и вклад в создание книги.

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.

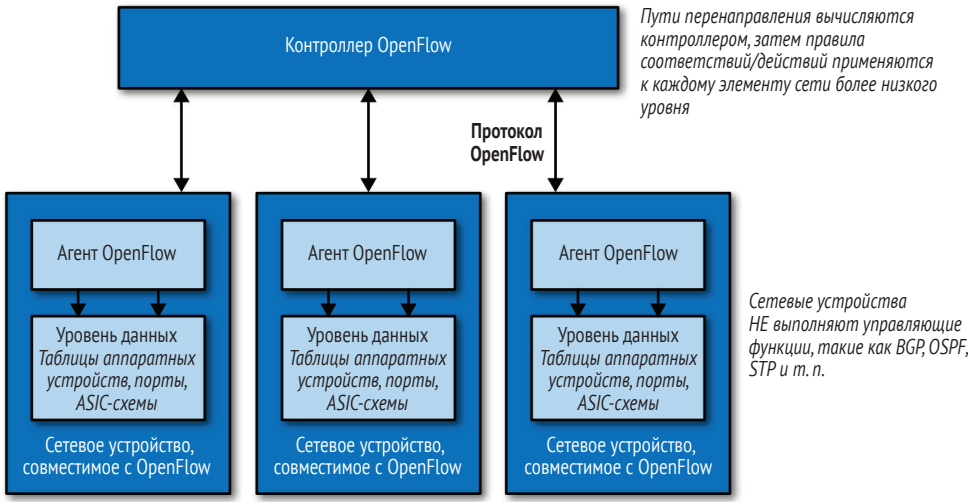

Рис. 1.1 ❖ Разделение уровня управления и уровня данных с помощью протокола OpenFlow

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.

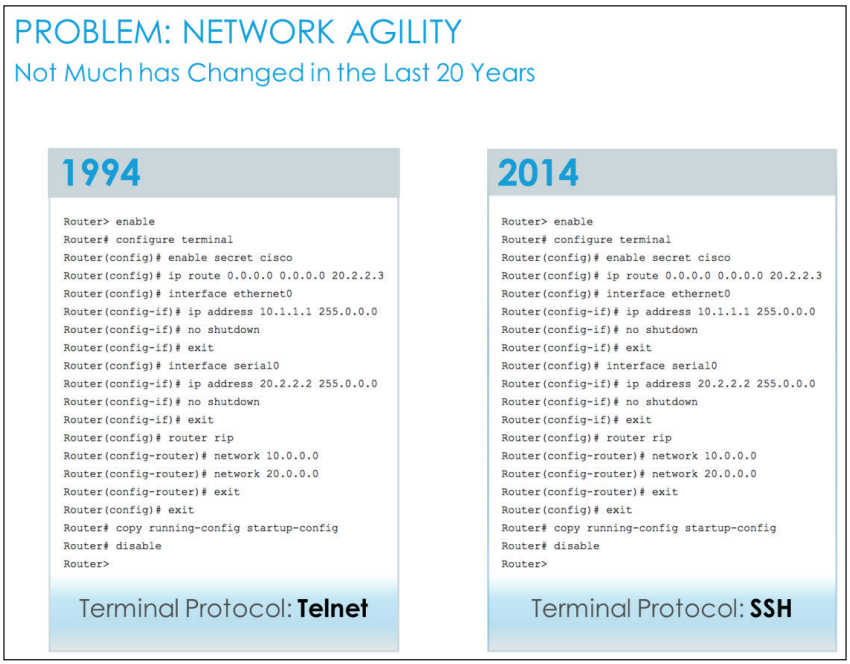

Рис. 1.2 ❖ Что изменилось? Telnet заменили на SSH (источник: Big Switch Networks)

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?