УДК 004.7

ПРОГРАММНО-КОНФИГУРИРУЕМЫЕ СЕТИ НА ОСНОВЕ ПРОТОКОЛА OPENFLOW

В. А. Лихачев

Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых

Поступила в редакцию 09.01.2014 г.

Аннотация. Попытки внедрения новых технологий в существующих сетях – это всегда проблема, потому что выбрав одного производителя оборудования, не всегда можно с легкостью внедрить новое оборудование от другого производителя.

За счет развития программно-конфигурируемых сетей (ПКС) можно упростить модернизацию сети и ее контролируемость благодаря централизованному управлению. Однако ПКС тоже имеют свои минусы, что и рассмотрено в данной статье.

Ключевые слова: программно-конфигурируемые сети, виртуализация сети, DDoS, балансировка нагрузки, OpenFlow.

Annotation. Attempts to introduce new technologies in modern networks are always problematic, because when we choose one vendor it is not always possible to rollout new hardware from another vendor with ease.

Due to the development of software-defined networks (SDN) it is possible to ease network's modernization and controllability thanks to centralized management. But SDN has it's minuses too, what is discussed in this article.

Keywords: software-defined networks, network's virtualization, DDoS, load balancing, OpenFlow.

ВВЕДЕНИЕ

Стремительное развитие технологий привело к необходимости обработки больших объемов данных. В последнее время сильно изменились концепции обработки и все чаще сеть становится ограничивающим фактором развития вычислительной инфраструктуры. Причина заключается в том, что сети являются статичными, в отличие от серверов, которые обязаны этим технологии виртуализации. Для оптимизации загрузки серверов виртуальные машины в крупных центрах обработки данных часто мигрируют, что меняет точки привязки трафика. Существующие способы адресации, логического деления сетей и способы конфигурирования сетевого оборудования в таких сетях становятся неэффективными. Имеющиеся в наличии сетевых инженеров инструменты ориентированы на работу с каждым сетевым устройством в отдельности. Если необходимо, к примеру, внести одно небольшое изменение на пути от одной виртуальной машины к другой, то максимум, чего можно добиться, автоматизации разворачивания требуемых настроек на оборудовании одного модельного ряда конкретного производителя.[1]

Для решения этих проблем была разработана концепция *SDN (Software Defined Networks)*. Архитектура SDN зародилась в Стэндфордском университете, когда понадобилось создать сеть для выполнения некоторых экспериментов, а строить отдельную сеть было дорого.

ПРОТОКОЛ OpenFlow

Концепция *SDN* не нова и, еще до ее применения в Стэнфорде, у крупных производителей уже были свои реализации, но полно-

стью привязанные к их технологиям. В связи с этим был разработан протокол *OpenFlow*, который является тем связующим звеном, которое призвано оградить разработчиков сетевых приложений от особенностей сетевого оборудования конкретного производителя. Часто этот способ сравнивают с созданием сетевой операционной системы, призванной предоставить программные абстракции для работы с сетевой инфраструктурой, как классические операционные системы берут на себя сокрытие особенностей аппаратной части компьютера.

Протокол *OpenFlow*, как и следует из названия, использует концепции потоков для описания трафика. В любом коммутаторе, поддерживающем этот протокол, имеется таблица потоков, разбиваемая на 3 части: поля соответствия (англ. *match fields*), счетчики (англ. *counters*) и инструкции (англ. *instructions*). Поля соответствия позволяют определить для каждого проходящего через коммутатор пакета, к какому потоку он относится. Тут следует учесть, что *OpenFlow* стирает грань между 2, 3 и 4 уровнем модели OSI и пакет можно выбрать на основании одновременно полей 2 уровня (mac-адрес, номер vlan), 3 уровня (ip-адрес) и 4 уровня (номер порта в tcp или udp дейтаграмме). Таким образом, разделение сетевых устройств на коммутаторы и маршрутизаторы в *SDN* исчезает.

Реализация просмотра таблицы потоков остается на усмотрение производителя, то есть можно как просматривать таблицу последовательно для каждого проходящего через коммутатор пакета, так и использовать специальную коммутационную матрицу, которая позволяет сравнивать заголовки пакета сразу со всеми записями в таблице потоков.

Поле счетчиков позволяет реализовать мониторинг и получение статистики по любым необходимым показателям, что в классических сетях является более сложно реализуемым. Например, без использования прокси-сервера можно определить, с какого ip-адреса было больше всего запросов к web-серверам. Для этого достаточно для каждого из требуемых ip-адресов в таблице потоков создать запись с этим адресом, типом протокола – TCP, и портом назначения – 80. Поле инструкций при этом для всех таких записей будет одинаковым. В нем может быть реализована как простая коммутация (направить пакет в выходной порт), так и аппаратный файрвол (отбросить пакет, подходящий под эти условия), так и модификация заголовков пакета (например, уменьшение *TTL (Time To Live)* для имитации работы маршрутизатора на основе коммутатора). Способов использования *OpenFlow* существует множество и выше были описаны лишь самые простые из них.

ЦЕНТРАЛИЗОВАННЫЙ КОНТРОЛЬ

Помимо *OpenFlow*-коммутаторов в *SDN* используется выделенный контроллер, который реализует такие задачи, как построение маршрутов и принятие решений об обработке того или иного вида трафика. Каждый коммутатор устанавливает с контроллером свой управляющий зашифрованный канал, по которому получает инструкции по обработке трафика и запрашивает у контроллера, что делать с трафиком, который не подходит ни под одно правило в таблице потоков.

Если в классических сетях задачи построения маршрута и реализации маршрута реализуются на одном устройстве, то в *SDN* построением занимается контроллер, а реализацией – коммутаторы. Поскольку им не надо выполнять сложных вычислений, они могут быть упрощены. Однако контроллер должен быть достаточно надежным, так как он является единой точкой отказа в такой сети. Выход из строя контроллера, в зависимости от возможностей коммутаторов, может как парализовать сеть, так и быть практически незамеченным, так как правила обработки трафика на коммутаторах уже есть и «консультироваться» с контроллером нет необходимости.

Поскольку это является самым слабым звеном *SDN*, существует возможность поставить в сети 2 контроллера, один из которых будет основным, а второй – запасным. Пока первый контроллер работает, второй только синхронизируется с ним, чтобы иметь

актуальное представление сети. После того, как связь с первым контроллером пропадает, OpenFlow-коммутаторы после истечения таймаута обращения к первому контроллеру будут устанавливать связь со вторым контроллером.

Сами контроллеры работают на обычных серверах (а в рамках тестов это может быть и любой компьютер вплоть до виртуальной машины) и ожидают подключений от клиентов-коммутаторов. Существуют несколько открытых и несколько закрытых реализаций контроллера. Одной из самых известных открытых реализаций является контроллер *NOX*, который предоставляет *API (Application Programming Interface)* для программирования сети. Это является еще одной особенностью SDN. Поскольку коммутаторы ничего сами не решают, контроллер должен выполнять вычисления за них. К примеру, реализация протокола маршрутизации *OSPF (Open Shortest Path First)* выполняется на контроллере и если сетевой инженер захочет внести какие-то изменения в алгоритм (например, чтобы алгоритм выбирал маршруты на основании загруженности каналов), то для этого необходимо будет провести цикл разработки и тестирования программы. Это относят как к минусам, так и к плюсам *SDN*. С одной стороны, написание такой программы может быть сопряжено со значительной сложностью и затратой значительного времени на тестирование и все равно остается шанс, что ошибка в программе может нарушить работу сети, а с другой стороны в классических сетях реализация произвольных алгоритмов просто невозможна ввиду того, что сетевое оборудование ограничено теми возможностями, которое в него заложил производитель.

Здесь следует учитывать тот факт, что *SDN* – технология для крупных, постоянно меняющихся сетей, таких, как сеть в крупном центре обработки данных (ЦОД) или кластерах. Этот факт подтверждает то, что членами консорциума Open Networking Foundation, развивающего SDN, являются такие компании, как Google, Facebook, Yahoo, HP, IBM и другие, работа которых полностью завязана на обработку больших массивов данных, что невозможно без гибкой сетевой инфраструктуры.

ВОЗМОЖНЫЕ ПРОБЛЕМЫ В SDN

В рамках моего исследования рассматривается использование OpenFlow для балансировки нагрузки и недостатки протокола в определенных условиях (как, например, при DDoS-атаке).

Рассмотрим пример, каким образом *OpenFlow* позволяет организовать «легкую» балансировку нагрузки на 2 уровне модели OSI. Но для начала заметим, что классическим способом решения этой проблемы является один сервер, который либо случайным образом, либо циклично выбирает один из нескольких серверов для обработки входного запроса. Сервер-балансировщик выполняет перенаправление запросов на сервер, причем обработка запроса длится больше, чем перенаправление, в противном случае балансировщик является лишней сущностью, увеличивающей время ожидания ответа клиентов. Затем ответ на запрос возвращается балансировщику и он выдает результат клиенту. Между запросом и ответом для одного клиента при этом могут прийти тысячи запросов от других клиентов, каждый из которых будет выполняться одним из N серверов, находящихся за балансировщиком. При этом последний может учитывать текущую загрузку серверов и направлять запрос наименее загруженному, но обычно этого не делают, так как происходит значительное усложнение балансировщика нагрузки. Однако в OpenFlow-сетях благодаря централизованному сбору статистики можно по крайней мере определить, к какому из серверов было направлено меньше всего запросов.

В SDN балансировку можно организовать другим, более эффективным способом [2]. Всем серверам назначается один IP-адрес. Когда приходит запрос, контроллер решает, какому из серверов направить этот запрос и по какому пути он должен пойти. После того, как поток (*flow*) был создан и во все коммутаторы добавили соответствующее правило, все пакеты для этого и последующих запросов пойдут к нужному серверу на максимально возможной скорости. Для определения пути контроллер учитывает текущую загру-

женность сети и задержки на разных путях. Для этого периодически считывается информация о текущей загруженности сети во всех узлах и затем на контроллере рассчитываются оптимальные пути для прохождения различных видов трафика.

Рассмотрим пример, в котором сеть состоит из 4 коммутаторов[3]: балансировщика нагрузки S и 3 фильтрующих коммутаторов F1, F2 и F3. К S подключены 2 класса трафика: на порты 1–2 приходит трафик от недоверенных пользователей, на порты 3–6 – от доверенных. Сеть рассматривает пакеты типов A и B от недоверенных хостов как потенциальные вторжения, и, возможно, отвергает их на основе политик. В то же время все пакеты с портов 3–6 должны проходить через сеть без изменений. Изначально предполагаем, что S отправляет пакеты с портов 1 и 2 на F1, с портов 3 и 4 – F2, с портов 5 и 6 – F3. F1 анализирует и, возможно, отбрасывает пакеты типов A и B, в то время как F2 и F3 пропускают весь трафик. Предположим, что нагрузка изменилась и нужно больше ресурсов для мониторинга трафика от недоверенных хостов. Тогда сеть может быть перенастроена так, что трафик с портов 1 и 2 распределяется между F1 и F2, а трафик с доверенных хостов с портов 3–6 идет на F3. Поскольку эти изменения невозможно произвести атомарно, все коммутаторы должны быть реконфигурированы по очереди. Однако это может привести к нарушению политик. Например, если правила для F2 изменяются так, что он отбрасывает трафик A и B, то они пересекаются с трафиком, отправленным доверенными хостами. Если сначала изменяются правила на S так, что он распределяет A и B трафик между F1 и F2, то пакеты от недоверенных хостов окажутся разрешенными. Однако существует правильная последовательность обновления правил:

– Обновить правила S так, что трафик с портов 1–2 идет на F1, с портов 3–6 – на F3.

– Дождаться, пока все оставшиеся в пути пакеты не будут обработаны F2.

– Обновить F2 так, что он анализирует и отбрасывает трафик A и B

– Обновить S так, что трафик с порта 1 идет на F1, с порта 2 – на F2, с портов 3–6 – на F3.

Как видно из этого примера, балансировка нагрузки не всегда позволяет реализовать то, что задумано, из-за задержек между выдачей команд на изменение правил и фактической установкой этих правил.

При правильно сформированной распределенной атаке можно перегрузить контроллер запросами, тем самым осуществив *DoS (Denial of Service)* контроллера, что при определенных условиях скажется на работоспособности сети. Тут следует учитывать множество условий, каждое из которых по отдельности не является критическим, но вместе они дают определенные проблемы. Например, если при балансировке нагрузки коммутатор-балансировщик настроен таким образом, что на каждый приходящий TCP-пакет с установленным флагом SYN будет отправлять запрос контроллеру, что делать с пакетами этого потока, то даже абсолютно легальные запросы могут устроить отказ в обслуживании (англ. *denial of service*) контроллера. Однако, если реализовывать разворачивание правил на коммутаторе совершенно другим способом, то этого не будет. Например, вместо создания потока для каждого нового tcp-пакета с флагом SYN, осуществлять проверку по совершенно другим полям, а именно, создавать поток на основе сети, к которой принадлежит IP-адрес клиента, осуществляющего запрос, или на основе адреса назначения, тем самым реализуя классическую систему маршрутизации. Таким же образом можно организовать балансировку на основе географического расположения клиента. Поскольку существуют базы, по которым можно определить, часто с точностью до города, к которому принадлежит IP-адрес, то контроллер позволит создать поток, который будет направлять запросы из определенной области на один и тот же географически самый близкий сервер. Нельзя сказать, что этот подход является чем-то новым, но его реализация классически все равно выполняется на сервере-балансировщике, а при работе этого подхода в географически распределенной OpenFlow-сети можно значительно разгрузить балансировщик, так как после того, как очередной поток для определенного региона создан на соответствующем

коммутаторе, он будет там существовать до тех пор, пока его используют (то есть пока из этого региона приходят запросы). Конечно, есть и другие способы реализации этой концепции: например, на основе DNS.

ЗАКЛЮЧЕНИЕ

Преимущества, который дает концепция *SDN* – очевидны. Это централизованное управление, мониторинг и сбор статистики в мультивендорной среде, независимость от технологий конкретного производителя, упрощение модернизации и обслуживания сети. Однако все вышеперечисленное не повод отказываться от разрабатываемых годами подходов к управлению сетями, поэтому переход к SDN будет реализовываться постепенно. К сожалению, на текущий момент OpenFlow находится в состоянии постоянной разработки, а материальные затраты, необходимые для перехода на него останавливают различные компании от перехода к SDN. Лишь через несколько лет можно будет увидеть, как пойдет внедрение этой технологии в жизнь. В настоящее время SDN в основном интересно исследователям в области сетей, хотя, к примеру, Google уже использует OpenFlow в своих датацентрах, но подробной информации об этом нет в свободном доступе.

СПИСОК ЛИТЕРАТУРЫ

1. *Барсков А.* SDN: кому и зачем это надо? [Электронный ресурс]. Журнал сетевых решений/LAN, № 12, 2012. URL: http://www.osp.ru/lan/2012/12/13033012/

2. *Flajslik M., Handigol N., Seetharaman S., McKeown N.* Plug-n-Serve: Load-Balancing Web Traffic using OpenFlow. [Электронный ресурс]. ACM Sigcomm conference, 2009. URL: http://conferences.sigcomm.org/sigcomm/2009/demos/sigcomm-pd-2009-final26.pdf

3. *Reitblatt M., Foster N., Rexford J., Walker D.* Consistent Updates for Software-Defined Networks: Change You Can Believe In! [Электронный ресурс]. Computer Science Department at Princeton University. URL: http://www.cs.princeton.edu/~dpw/papers/change-hotnets11.pdf

Лихачев В.А. – Ведущий специалист-эксперт, Владимирстат. E-mail: lihachev@rester.tk

Likhachev V.A. – Leading specialist-expert, Vladimirstat. E-mail: lihachev@rester.tk