**Министерство образования и науки Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет инфокоммуникационных технологий**

**Реферат по дисциплине: Основы технологии программно-конфигурируемых сетей**

**«Программно-конфигурируемые cети на основе протокола OpenFlow»**

**Выполнила студентка гр. K41114: Дмитриева К.А.**

**Проверил: Шкребец А.Е.**

Санкт – Петербург 2020

Современные сети являются статичными, в отличие от серверов, которые обязаны этим технологии виртуализации. Для оптимизации загрузки серверов виртуальные машины в крупных ЦОД (Центр обработки данных) часто мигрируют, что меняет точки привязки трафика. Существующие способы адресации, деление сетей и конфигурирование сетевого оборудования в таких сетях становится неэффективными. Для решения этих проблем была разработана концепция SDN (Software-Defined Network).



Рисунок 1 - Передача пакетов в традиционной сети и в сети SDN.

**Протокол OpenFlow**

Протокол OpenFlow использует концепции потоков для описания трафика. Коммутатор OpenFlow состоит из одной или более таблиц потока и таблицы группы, с помощью которых выполнятся поиск и передача пакетов. OpenFlow формирует канал к внешнему управляющему серверу – контроллеру Коммутатор связывается с контроллером и контроллер управляет коммутатором через OpenFlow-протокол.

При помощи протокола OpenFlow контроллер может добавить, обновить и удалить записи потока в таблицах потока, как мгновенно в ответ на запрос для пакета, так и заранее. Каждый поток в таблице коммутатора содержит множество записей потоков, каждая запись потока состоит из полей соответствия, счетчиков и инструкций, применяемых к пакетам. Поля соответствия позволяют определить, к какому потоку относится проходящий через него пакет. Протокол стирает грань между 2, 3 и 4 уровнем модели OSI, и пакет можно выбрать на основании одновременно mac-адреса, ip-адреса и номер порта в tcp- или udp-датаграмме. Деление на коммутаторы и маршрутизаторы исчезает.

Поле счетчиков позволяет реализовать съем статистики по любым показателям, что в классических сетях является сложно реализуемым. Например, можно определить, с какого ip-адреса было больше всего запросов к web-серверам. С этой целью для каждого из требуемых ip-адресов в таблице потоков создается 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) и модификация заголовков пакета (например, уменьшение значения TTL) и т.д.



Рисунок 2 - Структура управления OpenFlow.

При получении пакета коммутатор OpenFlow выполняет последовательность действий, в соответствии с приведенной схемой (рис. 3). Из пакета извлекаются служебные данные, включающие адрес назначения. Поиск соответствия начинается с первой таблицы и может продолжаться в каждой из дополнительных таблиц. Если необходимая запись найдена, выполняются инструкции, связанные с конкретной записью потока. Если никакого соответствия не найдено в таблице потоков, то в зависимости от конфигурации коммутатора пакет может быть либо направлен к контроллеру по каналу OpenFlow, либо пропущен, либо перенаправлен к следующей таблице потока.



Рисунок 3 – Схема прохождения пакета через коммутатор OpenFlow.

Инструкции, связанные с каждой записью потока, или содержат действия, или изменяют конвейерную обработку. Действия, включенные в инструкции, описывают пересылку и модификацию пакетов, а также обрабатывают группы таблиц. Инструкции конвейерной обработки позволяют пакетам быть отправленными к последующим таблицам для дальнейшей обработки, и позволяют информации быть переданной между таблицами в форме метаданных. Обработка таблиц останавливается в том случае, когда система команд, связанная с соответствующей записью потока, не указывает следующую таблицу; в этой точке пакет обычно изменяется и передается.

**Централизованный контроль**

В SDN используется выделенный контроллер, который реализует такие задачи, как построение маршрутов и принятие решений об обработке трафика. Каждый коммутатор устанавливает с контроллером свой управляющий зашифрованный канал, по которому получает инструкции по обработке трафика. Вследствие этого коммутаторы могут быть упрощены.

Поскольку это является самым слабым звеном SDN, можно поставить в сети 2 контроллера. Пока 1-й работает, 2-й только синхронизируется с ним, чтобы иметь актуальное представление сети. После того, как связь с первым контроллером пропадает, OpenFlow-коммутаторы, после таймаута обращения к 1-му контроллеру, будут устанавливать связь со 2-м.



Рисунок 4. – Централизованный контроль.

**Детектирование потерь пакетов**

Часто возникает задача определения, на каком участке сети наблюдаются потери пакетов. Причин может быть несколько: повреждение кабелей, перегрузка каналов, перегрузка оборудования. Для решения этой задачи «научим» контроллер отслеживать потери пакетов. Для этого создавать потоки надо не только на основе адреса назначения (классическая коммутация), но и адреса источника. Рассмотрим линейную сеть из трех коммутаторов и трех хостов, показанную на рисунке 5. Если использовать только адрес назначения, то количество пакетов, предназначенных h3, на s3 может быть равно 100, в то время как количество таких пакетов на s1 и s2 может быть равно 40 и 60 соответственно. Если учитывать и адрес источника, то такой проблемы не возникнет, но количество потоков будет расти экспоненциально.



Рисунок 5. – Тестовая топология.

Также необходимо периодически считывать показания по всем потокам и сравнивать количество байт или пакетов, прошедших по связанным потокам. Чтобы определить, что поток X на s1 соответствует потоку Y на s2, проверяется значение поля cookie в потоке. Это значение устанавливается контроллером при создании потока. Оно же возвращается коммутатором контроллеру при запросе информации о потоках. Под соответствием подразумевается, что потоки соответствуют одной паре «адрес источника – адрес назначения». Контроллер должен гарантировать, что для каждого потока будет использовано уникальное значение cookie.

Контроллер может получать статистику по всем потокам всех коммутаторов в сети. На основе этих данных он определяет, какими путями, какой трафик и в каком объеме прошел. Вследствие природы SDN, невозможно сказать о состоянии сети в каждый конкретный момент времени, так как между запросом статистики и ее получением проходит некоторый промежуток времени, и полученные данные в загруженной сети всегда будут отличаться от реального состояния. Для учета этого факта ставится определенный допуск на отклонение показаний для связанных потоков в рамках некоторого пути в сети. Степень допуска уменьшается по мере того, как по этому пути проходят все новые пакеты.

Чтобы избавиться от ограничения, по которому надо создавать поток для каждой пары адресов источник-приемник, нужно определить, к какому коммутатору приемник (хост) подключен напрямую. В таком случае, для определения потерь пакетов достаточно будет сравнить количество пакетов на этом коммутаторе и суммарно на всех остальных, предназначенных для одного и того же хоста.

 В разработанном алгоритме реализуется привязка mac-адреса назначения и cookie. При этом учитывается, какой коммутатор увидел определенный mac-адрес первым. В таблице 1 показано сравнение требований к памяти у 2-х методов. 1-й метод хранит статистику для каждой пары источник-приемник, а 2-й метод – только для приемника. Однако 2-й метод имеет недостаток: невозможно определить, на каком именно участке наблюдаются потери пакетов.

Таблица 1 – Сравнение двух способов хранения информации о потоках.

|  |  |  |
| --- | --- | --- |
| Количество хостов | Метод 1 | Метод 2 |
| 2 | 2 | 2 |
| 3 | 3 | 3 |
| 4 | 9 | 4 |
| N | N\*(N-1) | N |

Для проведения экспериментов использовалась топология, показанная на рисунке 5. Полезная нагрузка моделировалась посредством flood-ping, когда запросы посылаются без ожидания ответа. Трафик запускался с узла h1 на узел h3. В первые 5 секунд при таком виде трафика отклонения на коммутаторах достигали значения в 5%. В дальнейшем, еще через 5 секунд, эти значения уменьшались до 1%. По причине, описанной выше, необходимо использовать некоторое пороговое значение (например, 1000 пакетов на поток), чтобы можно было полагаться на получаемые значения отклонений.

Для моделирования проблем в сети периодически выключался сетевой интерфейс на коммутаторе h3. Результаты тестов показывают, что описанный подход удовлетворителен только на начальных порах работы алгоритма, а именно – когда количество прошедших через поток пакетов не достигло больших значений. В дальнейшем потери пакетов будут вносить малые отклонения. Именно поэтому нужна следующая модификация – периодический сброс собранных статистических данных. Самый простой способ – просто начинать считать заново, обнулив все счетчики. Но такой подход является неудовлетворительным с точки зрения необходимости дождаться порогового значения (в это время не будут отслеживаться потери).

Поэтому мы используем подход, основанный на хранении предыдущего и текущего состояния сети (до сброса и после). Когда пройдет очередной интервал в N секунд, запускается новый анализ отклонений. При этом значения счетчиков на всех коммутаторах сбрасываются. Новый анализ начинает работать по описанному выше алгоритму, но в течение некоторого интервала (K секунд или L пакетов) не участвует в принятии решений. Вместо этого, статистика считается в ранее запущенном анализе. Когда новый анализ достигнет порогового значения, предыдущий останавливается. При этом на сеть не создается дополнительной нагрузки, так как собранные данные можно использовать многократно.

Таблица 2 – Зависимость способности алгоритма детектировать потери пакетов от длительности работы.

|  |  |
| --- | --- |
| Длительность работы, секунд | Детектирование потерь в последнюю секунду |
|  | 10% | 50% | 100% |
| 10 | Да | Да | Да |
| 30 | Нет | Да | Да |
| 60 | Нет | Нет | Да |
| 120 | нет | Нет | Нет |

Зависимость способности детектирования потерь пакетов от длительности работы алгоритма приведена в таблице 2. Считается, что в секунду через поток проходит 1000 пакетов. В таблице указано, способен ли алгоритм определить определенное количество потерь пакетов в последнюю секунду работы. Такой способ подсчета связан с тем, что обычно проблемы возникают в неожиданные моменты времени и не являются распределенными по всему временному интервалу, а привязаны к конкретным моментам.

Из данной таблицы следует, что длительность работы алгоритма должна быть как можно меньше. Также иногда на компьютерах могут возникать перегрузки, вследствие чего некоторый трафик может отбрасываться в связи с тем, что компьютер не успевает его обрабатывать. Описанный выше подход при использовании только адресов не позволит определить, действительно ли проблема заключается в том, что компьютер не справляется с нагрузкой, или же, к примеру, кабель, идущий от коммутатора к сетевой карте компьютера, поврежден.

Данный алгоритм позволяет эффективно определять потери пакетов в сети, не создавая дополнительной нагрузки на коммутаторы. Но следует учитывать, что если пакет не дошел до получателя, то это не всегда означает проблемы с линиями связи или перегрузкой оборудования. Проблема может заключаться и в неправильных настройках таблицы маршрутизации, и в блокировании определенных типов пакетов. Однако OpenFLOW позволяет гибко задавать правила, по которым пакет относится к тому или иному потоку. Таким образом, можно определить, какой тип трафика (какие протоколы, порты) блокируется.

**К преимуществам данной технологии можно отнести:**

1. *Производительность.* Разделение тракта данных и тракта управления позволяет централизованно управлять маршрутизацией всей сети, а контроллерам направить все свои ресурсы на ускорение перемещения трафика.

2*. Стоимость*. За счет виртуализации управления сетью технология OpenFlow снижает расходы на построение и сопровождение сетей.

3. *Функциональность*. Благодаря открытому и единому интерфейсу прикладных программ (API) администраторы могут создавать различное ПО управления. Ранее данная задача была труднореализуемой, поскольку у маршрутизаторов и коммутаторов, предлагаемых крупными поставщиками, не было единого API.

4. *Администрирование*. За счет использования централизованного сервера повышается удобство администрирования. Так же предполагается возможность создания виртуальных сетевых топологий — построение виртуальных локальных или глобальных сетей без физического изменения сети.

**К опасениям относительно данной технологии можно отнести:**

1. *Масштабируемость.* Сосредоточение всех функций сетевого управления на одном сервере вызывает опасения. В обычных сетях все коммутаторы участвуют в управлении, и с ростом сети ее масштабируемость обеспечивается за счет новых коммутаторов.

2. *Безопасность*. Поскольку в OpenFlow вся функциональность управления реализуется на одном сервере, взломать такую сеть будет проще.

3. *Совместимость*. Имеются опасения, что добавление к сети OpenFlow проприетарных (собственных) расширений, может породить реализации, несовместимые с другими. Таким образом, на настоящий момент OpenFlow скорее является интересным научным проектом, чем готовой к внедрению технологией. Пока большинству пользователей нужны коммутаторы и маршрутизаторы, которые работают без трудоемкой настройки. Если технологии OpenFlow удастся достигнуть такого уровня функциональности, который позволит упростить архитектуру сети и управление ею, то в этом случае за данной технологией будет большое будущее.