Министерство науки высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«УНИВЕРСИТЕТ ИТМО»
Факультет «инфокоммуникационных технологий»
Направление подготовки «11.04.02 Инфокоммуникационные технологии и системы связи»

УНИВЕРСИТЕТ ИТМО

Реферат

На тему: «Open vSwitch»
По дисциплине: Основы технологии программно-конфигурируемых сетей

Выполнила:
Студентка гр. К41101
Трегубова Анастасия
Руслановна

Проверил:
Шкребец Александр
Евгеньевич

г. Санкт-Петербург
2021 г.

ВВЕДЕНИЕ

Сегодня виртуализация серверов и рабочих станций помогает оптимизировать множество процессов, но она также привносит некоторые специфические проблемы, требующие нового подхода. Одной из таких непростых задач является контроль трафика ВМ, его приоритизация и мониторинг.

Так как на одном узле может работать множество ВМ и трафик между виртуальными интерфейсами может не выходить за его рамки, применение классических, аппаратных коммутаторов и фаерволов в данной среде становится неэффективным. По этой причине возникла необходимость в реализации виртуального коммутатора, работающего на каждом из физических узлов и умеющего отслеживать и контролировать трафик ВМ.

Какое-то время в средах виртуализации на базе KVM и Xen для решения данной задачи использовался встроенный в ядро Linux коммутатор второго уровня — Linux Bridge. Он и сейчас используется по умолчанию в некоторых дистрибутивах как наиболее простое решение. Linux-мост позволяет создавать виртуальные интерфейсы для ВМ, а также коммутировать их между собой и внешним миром через физические интерфейсы узла. Также возможны некоторые методы фильтрации трафика на канальном уровне. Но даже этот функционал едва ли применим в крупных инфраструктурах из-за еще одной особенности виртуализации — высокой мобильности объектов.

Linux-мост не подходит для динамичных сред с большим количеством узлов и миграцией ВМ по причине отсутствия централизованности и взаимодействия между узлами. Связанность состояний и правил, таких как правила маршрутизации, ACL, QoS, мониторинга, закрепленные за ВМ при перемещении на другой физический узел — это основной список требований к коммутатору в виртуальной среде. Здесь нам на помощь и приходит распределённый, управляемый, многоуровневый виртуальный коммутатор, каковым и является Open vSwitch, основанный на протоколе OpenFlow.

OPENFLOW SWITCH

OpenFlow Switch состоит из одной или нескольких flow-таблиц, групповой таблицы и OpenFlow канала к удаленному контроллеру. Switch обменивается сообщениями с контроллером при помощи протокола OpenFlow (рисунок 1).

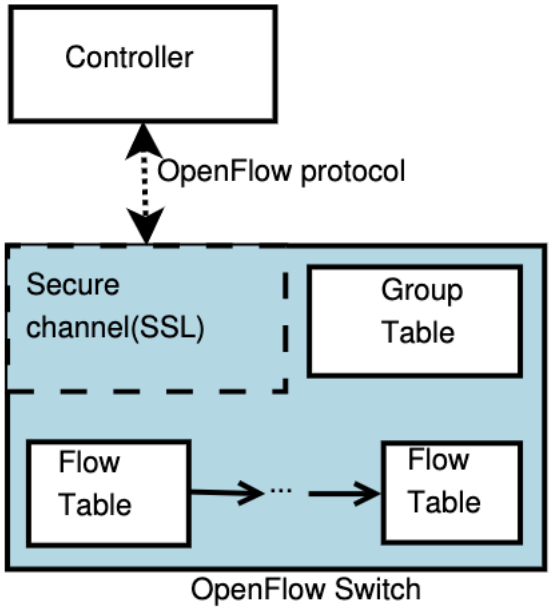

Рисунок 1 – Структура OpenFlow Switch

Используя данный протокол, контроллер может добавлять, обновлять и удалять flow-записи (flow-entries) во flow-таблицах. Каждая flow-таблица содержит набор flow-записей; каждая запись определяется полями сравнения, счетчиками и набором инструкций, которые применяются к пакету с совпавшими полями.

Сравнение начинается с первой flow-таблицы и может продолжаться в других таблицах. Поля сравнений flow-записей сравниваются с заголовком пакета в порядке приоритета. Если найдена совпадающая flow-запись, к пакету применяются инструкции, ассоциированные с данной записью. Если не найдено ни одной записи, результат зависит от конфигурации коммутатора (пакет может быть сброшен либо передан контроллеру по OpenFlow каналу для анализа и принятия решения).

Инструкции в соответствующих записях содержат либо действия, которые применяются к пакету и определяют непосредственно сами правила пересылки, либо определяют дальнейшую обработку пакета, в последующих flow-таблицах (конвейерная обработка). Конвейерная обработка позволяет передавать анализ пакета из одной таблицы в другую, дополнительно пересылая специальные метаданные (например, числовые переменные, связанные с предыдущей обработкой). Такой подход позволяет создавать “ветвление” в обработке трафика. Обработка заканчивается, когда соответствующие инструкции не содержат команд по пересылки в следующую таблицу; в этот момент выполняется модификация пакета, выполнение “накопленных” действий.

ПОРТ OPENFLOW

Порт – сетевой интерфейс, для передачи пакетов между OpenFlow обработкой и остальной частью сети. OpenFlow коммутатор должен поддерживать три типа OpenFlow портов: физический порт, логический порт, зарезервированный порт.

- Физический порт – порт, который соответствует аппаратному сетевому интерфейсу.
- Логический порт – порт, который не обязательно соответствуют физическому интерфейсу. Логический порт представляет собой более высокую абстракцию и может быть определен без использования OpenFlow (например: туннели, loopback интерфейсы). Единственное различие между физическим портом и логическим – пакет пересылаемый через данные порты в заголовке может иметь дополнительное поле Tunnel-ID, и когда пакет приходит с логического порта он пересылается на контроллер.
- Зарезервированный порт – представляет собой специальное действие по пересылки пакета:

- all – представляет собой все порты, на которые может быть продублирован пакет (может быть использован, только как output port)
- controller – канал связи с OpenFlow контроллером (может быть использован как для входящих, так и для исходящих пакетов)
- table – действие по передачи обработки пакета в другую таблицу
- in_port – представляет собой ID порта входящего пакета

OPENFLOW ТАБЛИЦА

Flow-таблица состоит flow-записей, где каждая состоит из:

Поля сравнения	Приоритет	Счетчики	Инструкции	Временные метки

- Поля сравнения – параметры, содержащиеся в заголовке пакета (данные L2-L4), а также некоторые OpenFlow данные (номер входящего порта, метаданные обработки процесса обработки пакета);
- Приоритет – числовое поле от 0 до 65535, среди flow-записей, совпавших с заголовком пакета, выбрана будет одна с большим приоритетом;
- Счетчики – хранят статистические данные о работе записи (например, сколько пакетов было обработано в соответствии с данной записью);
- Инструкции – набор команд, которые необходимо выполнить над пакетом в процессе обработки;
- Временные метки – максимальное время до удаление соответствующей flow-записи.

Соответствующая запись определяется по полям сравнения и приоритету: среди записей, у которых совпали поля выбирается уникальная с наибольшим приоритетом.

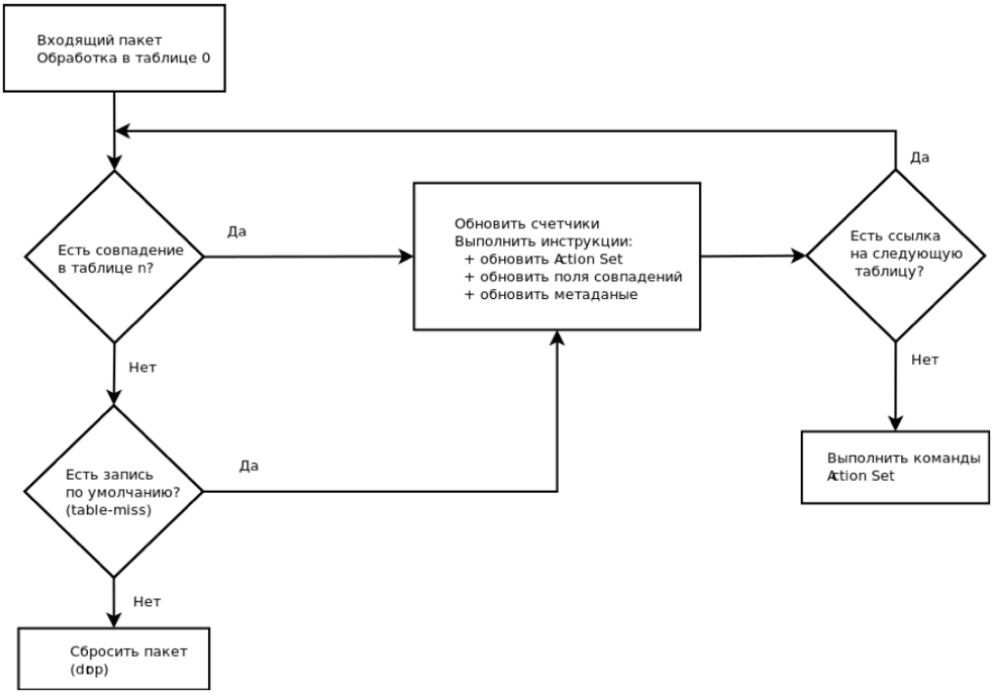

Рисунок 2 – Обработка пакетов

Анализ начинается с нулевой таблицы (см. рисунок 2), из пакета извлекаются данные заголовка. Далее проходит сравнение этих данных со всеми полями во flow-таблице (стоит отметить, что поиск идет не только по заголовку пакета, но, например, по номеру входящего порта и специальным метаданным, которые могли быть получены на предыдущих этапах обработки), если подходит несколько записей с одинаковым приоритетом, будет выбрана только одна запись, причем выбор конкретной записи не определен.

После совпадения записи инструкции добавляются в Action Set (своеобразный накопитель действий, которые будут выполнены после окончания анализа пакета) (есть возможность выставить бит OFPFF_CHECK_OVERLAP для запрета использования совпадающих правил).

OPEN VSWITCH

Open vSwitch (OVS) – программная реализация коммутатора, совместимого с протоколом OpenFlow. OVS — это проект с открытым исходным кодом, который позволяет гипервизорам виртуализировать сетевой уровень. Это позволяет обслуживать большое количество виртуальных машин, работающих на одном или нескольких физических узлах. Виртуальные машины подключаются к виртуальным портам на виртуальных мостах (внутри виртуализированного сетевого уровня).

Это очень похоже на подключение физического сервера к физическим портам сетевого коммутатора уровня 2. Эти виртуальные мосты затем позволяют виртуальным машинам связываться друг с другом на одном физическом узле. Эти мосты также подключают эти виртуальные машины к физической сети для связи за пределами узла гипервизора.

OVS базируется на некоторых уже имеющихся в ядре компонентах, например, том же Linux Bridge и штатном стеке QoS, плюс собственных разработках, реализующих дополнительный функционал. С помощью OVS можно создавать VLAN’ы, фильтровать трафик на сетевом уровне, агрегировать каналы, зеркалировать трафик, ограничивать ширину канала для конкретных ВМ. OVS значительно упрощает администрирование виртуализированных сред, так как вся сетевая конфигурация ВМ и статистика остаются связанными, даже если ВМ мигрирует с одного физического узла на другой.

Управление коммутаторами, создание правил и их перемещение на каждом из узлов осуществляется централизованно с помощью протокола OpenFlow. Поддержка NetFlow и sFlow позволяет мониторить и анализировать состояние сети по множеству аспектов, учитывать трафик ВМ. Также реализована сетевая база данных состояний (OVSDB), которая поддерживает удаленные триггеры. Поэтому часть программного обеспечения оркестровки может наблюдать различные аспекты сети и сообщать об их изменениях,

например, отслеживать миграцию ВМ. Кроме всего, OVS может работать с логическими тегами также, как и его коммерческие конкуренты. Поддержка логического контекста в сети достигается посредством добавления тегов или управления ими в сетевых пакетах. Это может использоваться, например, чтобы однозначно определить ВМ. Так же имеется реализация GRE-туннелирования, способная обрабатывать тысячи одновременных GRE-туннелей. Это, например, может использоваться, для объединения частных сетей ВМ в различных центрах обработки данных.

СТРУКТУРА OPEN VSWITCH

Open vSwitch условно можно разделить на две части – user-space и kernel-space.

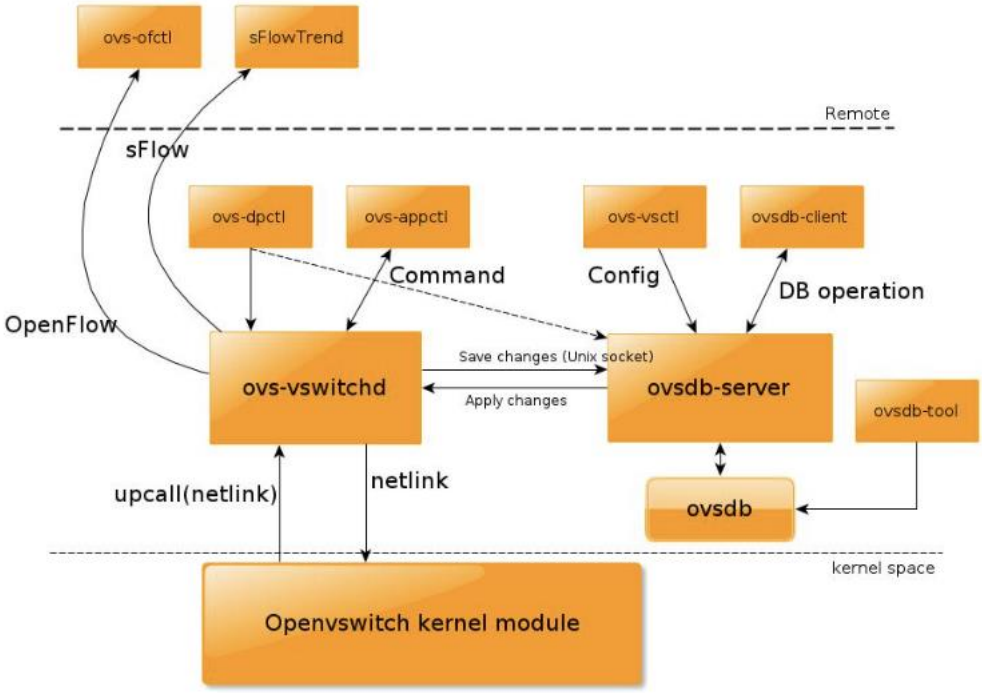

Рисунок 3 – Структура Open vSwitch

User-space состоит из нескольких наиболее важных компонент: это демоны, которые реализуют свитч с таблицей потоков, ядро Open vSwitch, и набор утилит, которые позволяют конфигурировать свитч, его базу данных (ovsdb) и напрямую посылать сообщения в ядро.

Open vswitch service запускает три демона:

- ovs-vswitchd – сохраняет и отвечает за изменение конфигурации свитча, а также сохраняет состояние в базу данных (ovsdb);
- ovsdb-server – манипулирует базой данных, конфигурацией и набором потоков;
- ovs-brcompatd – создает совместимость Open vSwitch с обычными Linux мостами (которые создаются командой brctl).

Демон ovs-vswitchd – единственная компонента, связанная с kernel-space через протокол netlink, также меняет конфигурацию Open vSwitch и сохраняет ее в базу данных (ovsdb), которая управляется при помощи демона ovsdb-server (коммуникация ovs-vswitchd и ovsdb-server осуществляется при помощи сокетов).

- ovs-dpctl – позволяет напрямую обращаться к kernel-space, без сторонних обращений к базе данных.
- ovsdb-client – утилита для конфигурирования базы данных ovsdb.
- ovs-ofctl – команда для работы с протоколом OpenFlow, модификацией flow таблиц, настройки соединения с удаленным контроллером.

Стоит отметить, что Open vSwitch является гибридным коммутатором – помимо поддержки OpenFlow, коммутатор поддерживает специализированные команды (QoS, tunneling и.т.д), но наибольший интерес он представляет как проект поддерживающий протокол OpenFlow.

ЗЕРКАЛИРОВАНИЕ ТРАФИКА

Для использования системы обнаружения вторжения необходимо использовать зеркалирование трафика на порт для анализа. Для настройки зеркалирования необходимо создать порт-зеркало и добавить его в соответствующий маршрутизатор:

```
[host]# ovs-vsctl create mirror name=mirror select_all=1 output_port=port_id
```

Для того, чтобы узнать ID порта, который необходим:

```
[host]# ovs-vsctl list port <IDS target port>
```

Добавление зеркала:

```
[host]# ovs-vsctl add bridge extern0 mirrors=mirror_id
```

НАСТРОЙКА GRE ТУННЕЛИРОВАНИЯ

Open vSwitch – позволяет использовать GRE туннелирование между хостами, как способ инкапсуляции трафика и создание оверлейных сетей. На следующей схеме показано, как настраивать GRE туннель между двумя виртуальными машинами:

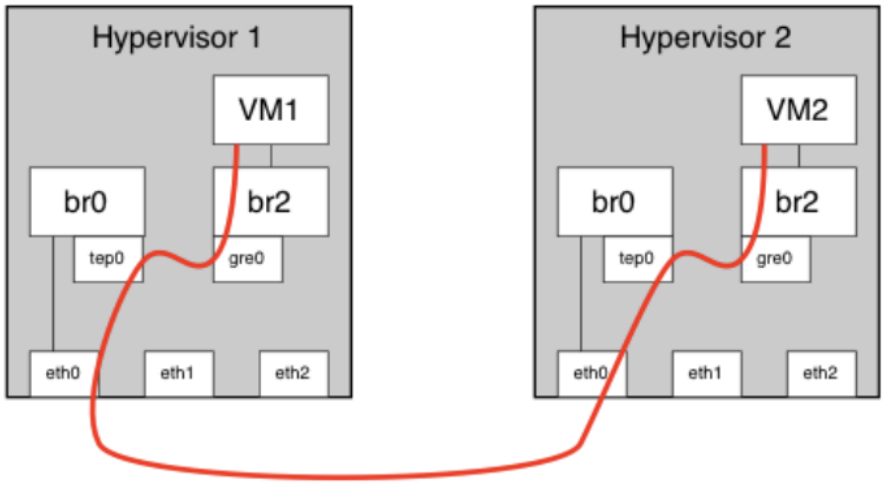

Рисунок 4 – GRE туннель

Процесс настройки состоит из трех частей:

- Создать изолированный мост для виртуальной машины
- Создать конечные точки GRE туннеля на каждом из гипервизоров.
- Добавить GRE интерфейс и инициализировать туннель.

Создание изолированного моста Open vSwitch (его можно назвать изолированным, т.к. он не содержит реальных физических интерфейсов):

```
[hypervisor1]# ovs-vsctl add-br br2
```

Смысл использования “изолированного моста” и отдельного GRE интерфейса в том, чтобы отделить системный трафик гипервизора от трафика GRE, что в свою очередь отделяет управление сетью для виртуальной машины, от особенностей сети хоста, на котором запущен гипервизор. Для этого создается внутренний интерфейс и ему присваивается IP адрес:

```
[hypervisor1]# ovs-vsctl add-port br0 tep0 -- set Interface tep0 type=internal
[hypervisor1]# ifconfig tep0 192.168.200.20 netmask 255.255.255.0
```

На втором гипервизоре прописываются те же команды, выставляется другой IP адрес интерфейса tep0. После этого в маршрутизатор добавляется порт GRE туннеля:

```
[hypervisor1]# ovs-vsctl add-port br2 gre0 -- set interface gre0 type=gre options:remote_ip=<GRE tunnel enpoint on other hypervisor>
```

После данных команд на каждом из гипервизоров, между маршрутизаторами инициализируется GRE туннель.

ЗАКЛЮЧЕНИЕ

Рассмотренный в данной работе программный компонент Open vSwitch является практически главным инструментом для организации сетей различной сложности. Знание деталей его работы и умение его конфигурировать являются весьма востребованными среди сетевых администраторов. Open vSwitch зарекомендовал себя как надежный, стабильный продукт, использующийся для создания программно-конфигурируемых сетей в крупных компаниях.