МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Физико-технический факультет

Кафедра оптоэлектроники

Допустить к защите
Заведующий кафедрой
д-р техн. наук, профессор
_____Н. А. Яковенко
_____2018 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

ИССЛЕДОВАНИЕ МОДЕЛЕЙ РЕАЛИЗАЦИИ ВИРТУАЛЬНЫХ ЧАСТНЫХ СЕТЕЙ РЕГИОНАЛЬНОГО ПРОВАЙДЕРА

Работу выполнил_____ Нечитайло Роман Дмитриевич

Направление подготовки 11.03.02 Инфокоммуникационные технологии и системы связи

Направленность (профиль) Оптические системы и сети связи

Научный руководитель
канд. физ.-мат. наук, доцент _____ А. С. Левченко

Нормоконтролер инженер _____ И. А. Прохорова

Краснодар
2018

РЕФЕРАТ

Выпускная квалификационная работа: 78 с., 14 рис., 4 табл., 6 источников, 4 прил.

JUNIPER MX, IP-OVER-IP ТУННЕЛЬ, GRE ТУННЕЛЬ, MPLS, POLICY AND FILTER

Объектом исследования выпускной квалификационной работы являются сети пакетной передачи данных на оборудовании провайдера сетевых услуг.

Целью работы является разработка описания лабораторных работ по практическому изучению в эмуляционной среде GNS3 модели реализации VPN в городской сети регионального провайдера на маршрутизаторах фирмы Juniper серии MX.

В результате выполнения выпускной квалификационной работы были смоделированы следующие методы реализации VPN: «проброс» VLAN’ов, организация IP-over-IP и GRE туннелей, а также с помощью протокола MPLS. Были описаны этапы настройки каждого метода и применены политики и фильтры в одном из них.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современное развитие информационных технологий и, в частности, сети Internet, приводит к необходимости защиты информации, передаваемой в рамках распределенной корпоративной сети, использующей сети открытого доступа. При использовании своих собственных физических каналов доступа эта проблема так остро не стоит, так как в эту сеть не имеет доступа никто из посторонних. Однако стоимость таких каналов высока, поэтому не каждая компания позволит себе использовать их. В связи с этим Internet является наиболее доступным. Internet является незащищенной сетью, поэтому приходиться изобретать способы защиты конфиденциальных данных, передаваемых по незащищенной сети.

VPN – это технология, которая объединяет доверенные сети, узлы и пользователей через открытые сети, которые могут быть небезопасны. Эта технология получает всё большее распространение среди не только технических специалистов, но и среди рядовых пользователей, которым также требуется защищать свою информацию (например, пользователи Internet-банков или Internet-порталов).

Специалисты в области технологии VPN используют сугубо технические понятия, такие как «туннелирование», «сервер сертификатов» и другие. Но для конечных пользователей эта терминология, как правило, ничего не объясняет. Скорее их интересует другие вопросы, вопросы безопасности и скорости работы сети.

1 Архитектура Juniper MX

1.1 Маршрутизаторы Juniper

Ещё в 1998 году Juniper Networks выпустила свой первый маршрутизатор M40. Используя специальные интегральные схемы (ASIC), M40 смог превзойти любые другие маршрутизатор. M40 также был первым маршрутизатором, имеющим истинное разделение управляющего и передающего уровней, и родилась серия M. Первоначально название модели M40 ссылается на его способность обрабатывать 40 миллионов пакетов в секунду.

Основным вариантом использования для серии M было разрешить поставщикам услуг предоставлять услуги на основе IP, в то же время поддерживая устаревшее ретрансляционное устройство и сети ATM.

Крупные корпоративные компании становятся все более похожими на поставщиков услуг и предлагают IP для департаментов и дочерних компаний.

Почти все сетевое оборудование подключается через Ethernet. Это один из самых понятных и развернутых сетевых технологий, используемых сегодня. Компании испытывают трудности требования по сокращению эксплуатационных расходов. Ethernet обеспечивает упрощение сетевых операций, администрирования и поддержку пользователей.

Серия MX была представлена в 2007 году для решения этих новых задач. Данная серия маршрутизаторов оптимизирована для доставки высокоскоростных Ethernet-сервисов уровня 2 и уровня 3.

Серия M по-прежнему относится к использованию нескольких служб, таких как MPLS и VPN, а серия X уже относится к новому поколению маршрутизаторов, которые сосредоточены на работе на интерфейсах 10G и выше; интересно также отметить, что римская цифра для номера 10 - «X».

Нелёгкая задача – создать платформу, способную решать эти новые задачи. Серия маршрутизаторов MX имеет хорошую родословную: хотя она механически отличается, она использует технологии из серий M и T для управления коммутационной сетью и маршрутизацией.

Поддерживаются методы обеспечения высокой доступности, такие как Non-Stop Routing (NSR), Non-Stop Bridging (NSB), Graceful Routing Engine Switch over (GRES), Graceful Restart (GR) и обновление программного обеспечения (ISSU), различные методы маршрутизации, такие как RIP, OSPF, IS-IS, BGP и Multicast.

Коммутация включает в себя полный набор Spanning Tree Protocols (STP), VLAN, QinQ и способность масштабировать более 4094 модовых доменов, используя виртуальные коммутаторы.

Ещё имеются такие функции, как трансляция сетевых адресов (NAT), экспорт информации о потоке IP (IPFIX), MPLS, L3VPN, L2VPN и VPLS, возможность использования протокола PPPoX, DHCP, иерархическое QoS и отслеживание IP-адресов, а также виртуализация.

1.2 Операционная система Junos

Junos – это специализированная сетевая операционная система, основанная на одной из самых стабильных и безопасных операционных систем в мире: FreeBSD. Junos спроектирован как монолитная архитектура, которая размещает все службы операционной системы в пространстве ядра.

Основные компоненты Junos написаны как фоновые процессы, которые обеспечивают полный процесс и разделение памяти.

Одним из преимуществ монолитной архитектуры ядра является выполнение его функций в режиме администратора на центральном процессоре, а приложения и фоновые процессы выполняются в пространстве пользователя. Один неудачный фоновый процесс не приведёт к сбою

операционной системы и не окажет абсолютно никакого влияния на другие фоновые процессы, которые не зависят от него. Например, если возникла проблема с фоновым процессом SNMP протокола, которая нарушила его работу, то это абсолютно никаким образом не повлияет на фоновые процессы маршрутизации, которые обрабатывают OSPF и BGP [1].

1.3 Единая операционная система Junos

Создание единой сетевой операционной системы, которая может быть использована во всех маршрутизаторах, коммутаторах и брандмауэрах позволяет упростить сетевые операции, администрирование и обслуживание.

Операторам сети необходимо только изучить Junos один раз, и они смогут получить высококлассных специалистов, способных выполнять функции настройки и администрирования многих других устройств фирмы Juniper. Дополнительным преимуществом единой операционной системы Junos является то, что нет необходимости «изобретать колесо» и иметь 10 различных реализаций BGP или OSPF.

Возможность писать эти основные протоколы один раз, а затем повторно использовать их во всех продуктах позволяет достигнуть высокого уровня стабильности, поскольку код уже проверен на реальных устройствах.

1.4 Программное обеспечение

Каждый квартал в течение почти 15 лет происходит обновление программного обеспечения операционной системы Junos. Разработка основной операционной системы - это единый её выпуск для многих устройств. Это позволяет разработчикам создавать новые функции или исправлять ошибки один раз и делиться ими сразу на нескольких платформах.

Номера выпуска находятся в основном и младшем формате. Основное число - это версия операционной системы Junos за конкретный календарный год, а младший релиз указывает, в каком квартале было выпущено программное обеспечение. Это происходит, чтобы можно было подобрать, к примеру, версии операционной системы Junos 11 и Junos 12, поскольку они напрямую связаны с выпущенным годом. Например, операционная система Junos 11 была выпущена в 2011 году.

Но это не всегда было так. Исторически релиз «.0» был зарезервирован для крупных событий, таких как выпуск программного обеспечения для новых продуктов, таких как MX240 с Junos 9.0.

Каждый выпуск Junos поддерживается в течение 18 месяцев. Последний выпуск Junos в календарный год известен как Extended End of Life (EEOL), и этот выпуск поддерживается в течение 36 месяцев.

На рисунке 1 представлена модель выпуска операционной системы Junos.

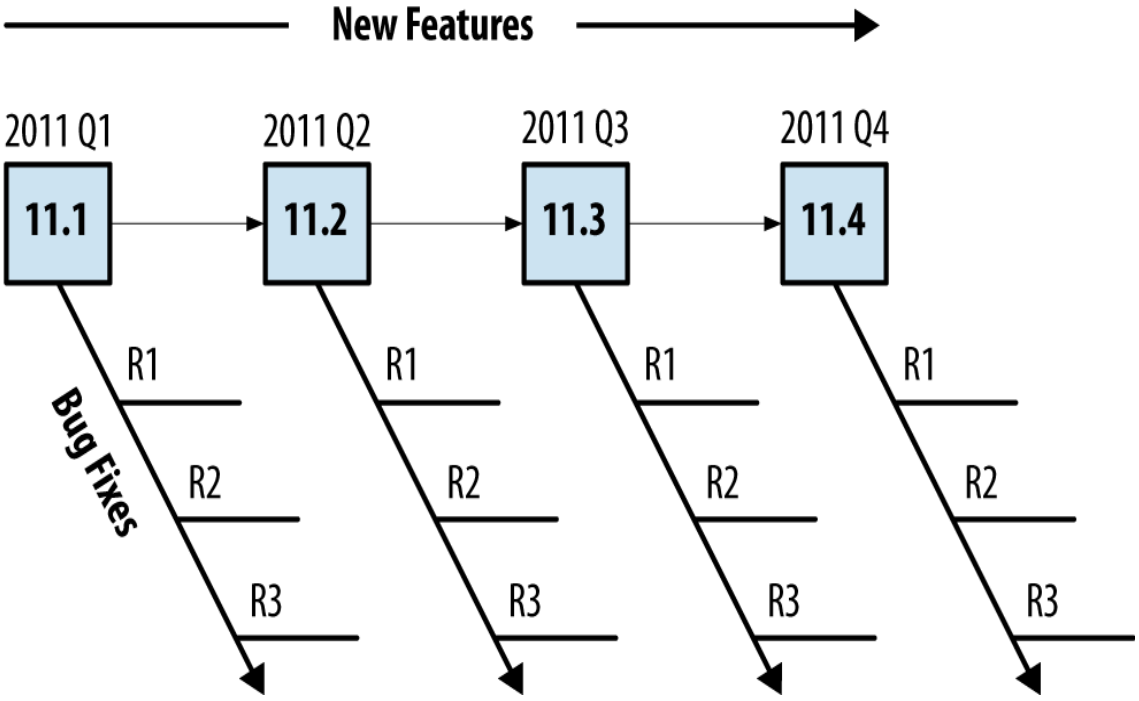

Рисунок 1 – Модель выпуска операционной системы Junos

Существует несколько различных типов Junos, которые выпущены чаще для решения вопросов, касающихся технического обслуживания, а также сервисного обслуживания. Обычно каждые шесть недель публикуются релизы технического обслуживания, чтобы исправить набор проблем, и они имеют префикс «R».

Например, Junos 11.1R2 будет вторым релизом технического обслуживания для Junos 11.1.

Версии сервисного обслуживания выпускаются по требованию специально для решения важной проблемы, которую ещё только предстоит решить с помощью технического обслуживания. Эти выпуски имеют префикс «S.». Например, Junos 11.1S2.

Общее эмпирическое правило заключается в том, что новые функции добавляются каждый незначительный релиз и исправления ошибок добавляются каждый релиз сервисного обслуживания.

Например, Junos 11.1-11.2 означает, что были введены новые функции, в то время как «Junos 11.1R1 - 11.1R2» означает, что были исправления ошибок.

Большинство производственных сетей предпочитают использовать последний выпуск Junos предыдущего календарного года, так как эти выпуски Junos - это выпуски EEOL, которые поддерживаются в течение трех лет.

Преимуществом является то, что релизы EEOL становятся более стабильными со временем. К примеру, версия Junos 11.1 прекратит исправление ошибок через 18 месяцев, в то время как в версии Junos 11.4 будут исправлены ошибки на следующие 36 месяцев. Таким образом можно сделать вывод о том, почему именно этими версиями предпочитают пользоваться различные компании. Сотрудники могут не беспокоиться о поддержке данной версии в течении трёх лет, что, естественно, способствует улучшению качества выпускаемых решений.

1.5 Модель «Три выпуска в год»

В 2012 году Junos создала новую модель выпуска для перехода от четырех выпусков в год до трёх, согласно рисунку 2. Это увеличило частоту выпуска обслуживания, чтобы чаще решать больше проблем.

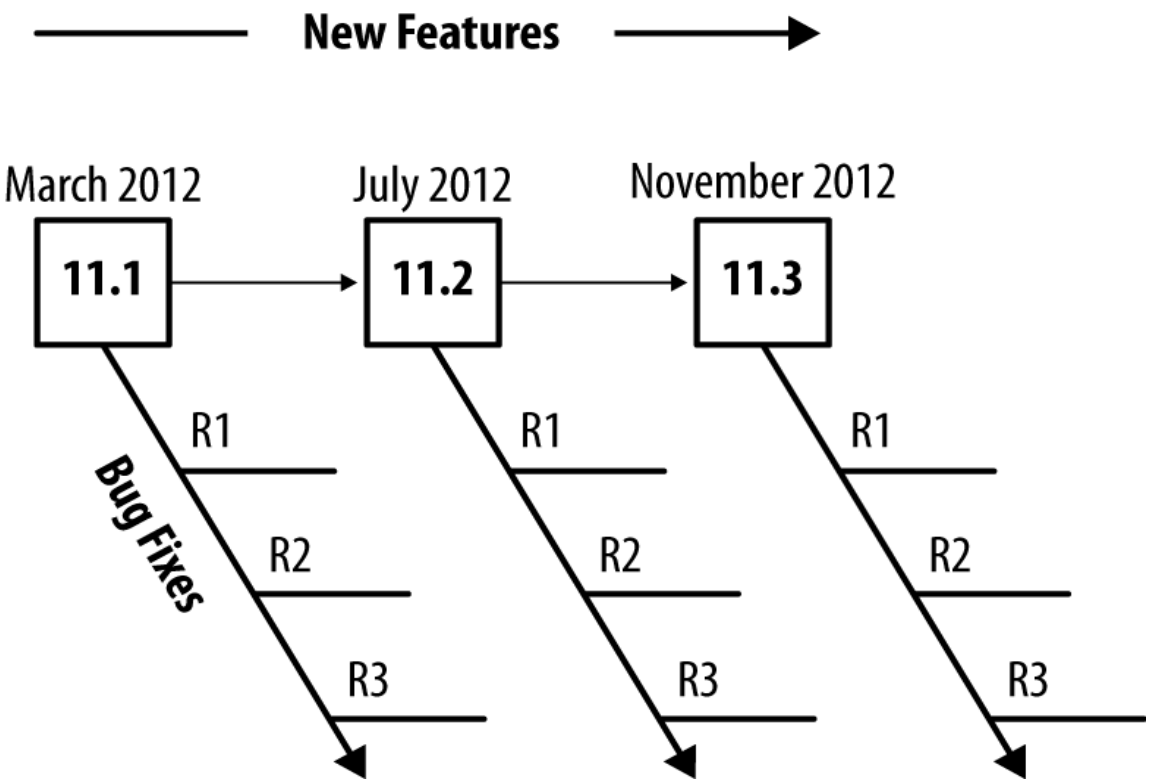

Рисунок 2 – Модель «Три выпуска в год»

Другое преимущество заключается в том, что все выпуски операционной системы Junos с 2012 года поддерживаются в течение 24 месяцев, в то время как последний выпуск операционной системы Junos за календарный год по-прежнему будет считаться EEOL версией, и он имеет сервисную поддержку в течение 36 месяцев.

Расширяя время технической поддержку и сокращая количество выпусков, сетевые операторы должны иметь возможность уменьшить частоту обновления до получения новой версии кода.

Благодаря данной модели, сетевые операторы всё чаще используют не только так называемые EEOL версии, но и промежуточные версии операционной системы Junos.

1.6 Архитектура программного обеспечения

Junos была разработана с самого начала, чтобы поддерживать разделение функции управления и функции пересылки трафика. Это справедливо для серии MX, где все функции управления выполняются механизмом маршрутизации, в то время как вся пересылка выполняется посредством механизма пересылки пакетов (PFE).

Благодаря такому уровню разделения обеспечивается взаимное невлияние одного функционала на другой. Например, функция пересылки может быть маршрутизирована на трафик с линейной скоростью, и выполняются при этом множество других различных функций, пока запущен и работает механизм маршрутизации.

Функции управления бывают разных форм и размеров. Существует распространенное заблуждение, что уровень управления обрабатывает только обновления протокола маршрутизации. Фактически, есть ещё много функций уровня управления. Некоторые примеры включают в себя обновление таблицы маршрутизации, ответы на запросы SNMP, обработку SSH или HTTP-трафика для администрирования маршрутизатора, а также изменение скорости вращения вентилятора.

Согласно рисунку 3, на верхнем уровне уровень управления полностью внедрён в механизм маршрутизации, в то время как уровень пересылки реализован с использованием небольшого специально сконфигурированного ядра, которое содержит только необходимые функции для маршрутизации и переключения трафика.

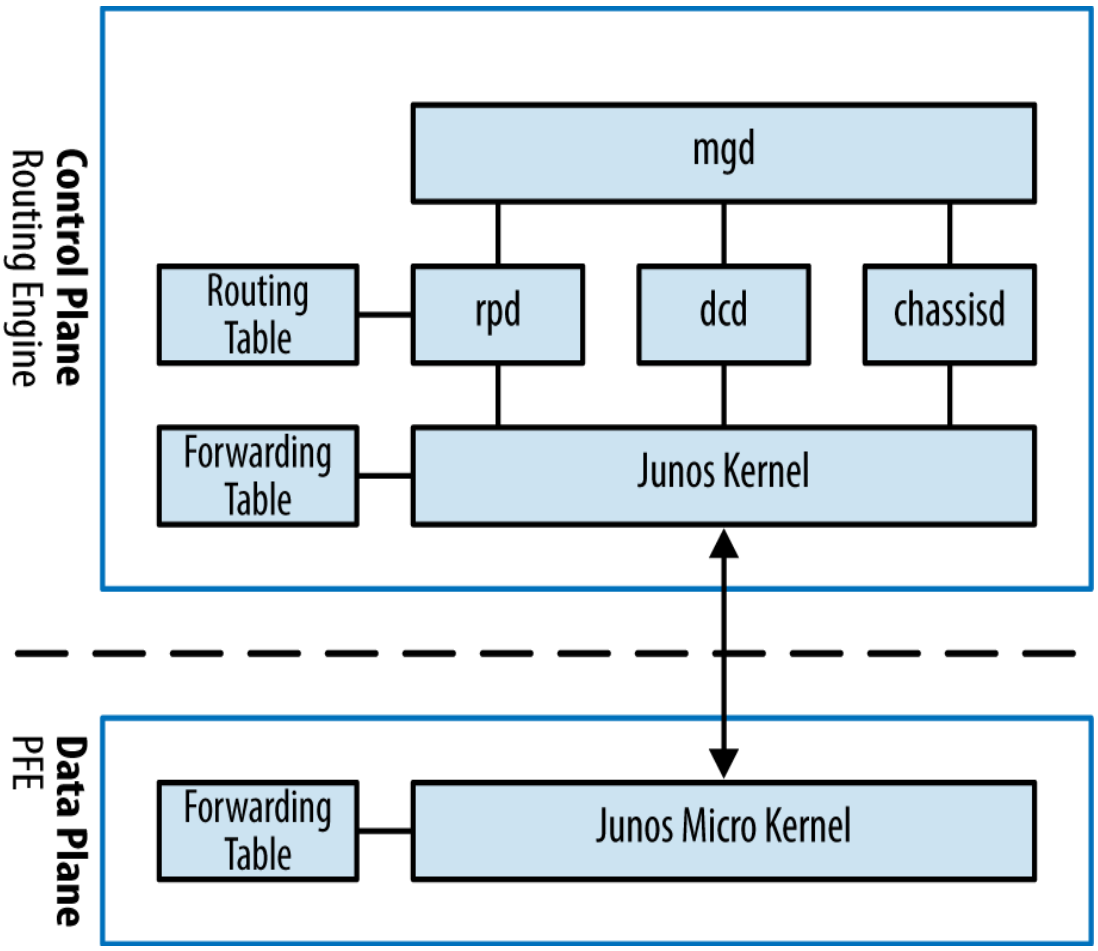

Рисунок 3 – Архитектура программного обеспечения Junos

Преимущество разделения управления и пересылки заключается в том, что любой трафик, который маршрутизируется или переключается через маршрутизатор, всегда будет обрабатываться с линейной скоростью, например, если маршрутизатор обрабатывал трафик между веб-серверами и в Интернете, вся обработка будет выполняться функционалом уровня пересылки.

1.7 Фоновые процессы

Ядро операционной системы Junos имеет четыре основных фоновых процесса; каждый из этих фоновых процессов играет критическую роль

внутри продуктов серии MX и работает вместе через межпроцессные взаимодействия (IPC) и маршрутизацию сокетов для связи с ядром операционной системы Junos и другими фоновых процессами [2].

Следующие фоновый процессы занимают центральное место и требуются для работы Junos:

1. Фоновый процесс управления (mgd)
2. Фоновый процесс протокола маршрутизации (rpd)
3. Фоновый процесс управления устройством (dcd)
4. Фоновый процесс chassis

Есть ещё много фоновых процессов для таких задач, как NTP, VRRP, DHCP и других технологий, но они играют меньшую и более конкретную роль в архитектуре программного обеспечения.

1.7.1 Фоновый процесс управления

Пользовательский интерфейс операционной системы Junos хранит всю информацию в централизованной базе данных. Это позволяет Junos обрабатывать данные интересными способами и открывает двери для расширенных функций, таких как откат конфигурации, применение групп, а также активировать и деактивировать целые части конфигурации благодаря разделению на группы.

Пользовательский интерфейс состоит из четырёх основных компонентов: базы данных конфигурации, схемы базы данных, фонового процесса управления (mgd) и интерфейса командной строки (cli).

Фоновый процесс управления (mgd) - это фоновый процесс, который содержит весь пользовательский интерфейс операционной системы Junos вместе. На высоком уровне фоновый процесс управления предоставляет механизм для обработки информации как для сетевые операторы, так и для других фоновых процессов.

Интерактивным компонентом фонового процесса управления является «Junos cli». Это приложение на основе терминального управления, что позволяет сетевому оператору использовать интерфейс в Junos. Другая положительная для операторов сторона фонового процесса управления – это интерфейс расширения языка разметки (XML) удаленного вызова процедур (RPC).

Это обеспечивает интерфейс прикладного уровня (API) через такие приложения, как Junoscript и Netconf, чтобы обеспечить развитие автоматизации приложений.

Функции Junos cli:

1. Редактирование командной строки
2. Терминальная эмуляция
3. Терминальный пейджинг
4. Отображение завершений команды и переменных
5. Мониторинг файлов журналов и интерфейсов
6. Выполнение дочерних процессов, таких как ping, traceroute и ssh

Функции фонового процесса управления:

1. Передача команд от cli соответствующему фоновому процессу
2. Поиск выполнения команд и переменных
3. Команды парсинга

Интересно отметить, что большинство рабочих команд операционной системы Junos используют XML для передачи данных. Чтобы увидеть пример этого, просто необходимо добавить отображение xml-команды на любую другую команду. Рассмотрим простую команду, такую как «show isis adjacency».

```
{master}dhanks@R1-RE0> show isis adjacency | display xml
<rpc-reply
xmlns:junos="http://xml.juniper.net/junos/11.4R1/junos">
```

```
    <isis-adjacency-information
xmlns="http://xml.juniper.net/junos/11.4R1/junosrouting"
    junos:style="brief">
    <isis-adjacency>
    <interface-name>ae0.1</interface-name>
    <system-name>R2-RE0</system-name>
    <level>2</level>
    <adjacency-state>Up</adjacency-state>
    <holdtime>22</holdtime>
    </isis-adjacency>
    </isis-adjacency-information>
    <cli>
    <banner>{master}</banner>
    </cli>
    </rpc-reply>
```

Данные были отформатированы в XML и получены из фонового процесса управления через удалённый вызов процедуры.

1.7.2 Фоновый процесс протокола маршрутизации

Фоновый процесс протокола маршрутизации обрабатывает все настроенные протоколы маршрутизации в пределах операционной системы Junos. На высоком уровне его обязанности распространяются на типовые объявления маршрутизации и обновления, поддерживая таблицу маршрутизации и устанавливая активные маршруты в пересылку в таблицу маршрутизации.

Чтобы поддерживать разделение процессов, каждый протокол маршрутизации настраивается в системе выполняется отдельная задача внутри фонового процесса протокола маршрутизации.

Другая функция фонового процесса протокола маршрутизации заключается в том, что он является посредником для обмена информацией с ядром операционной системы Junos для получения модификаций интерфейса, отправления информации о маршруте и отправлять изменения самого интерфейса.

Давайте взглянем на фоновый процесс протокола маршрутизации и посмотрим, что происходит. Задача его заключается в том, что он ведёт учёт включения и выключения учётной записи процессора. Также используется функция учёта заданий, чтобы увидеть результаты выполнения команд.

```
{master}
dhanks@R1-RE0> set task accounting on
Task accounting enabled.
```

Junos в настоящее время профилирует все фоновые процессы и задачи, чтобы получить лучшее использование процессора для маршрутизации. Нужно подождать несколько минут, чтобы он собрал некоторые данные и вывел результат.

Команда подсчета заданных задач скрыта по какой-то причине. Возможно, она добавляет дополнительную нагрузку на ядро Junos, в то время как учёт данных включён. Не рекомендуется запускать эту команду в производственной сети, если это не предусмотрено особым образом. После завершения вашей отладки необходимо выключить функцию учёта данных, когда выполнение заданного задания закончено.

```
{master}
dhanks@R1-RE0> set task accounting off
Task accounting disabled.
```

1.7.3 Фоновый процесс управления устройством

Фоновый процесс управления устройством отвечает за настройку интерфейсов на основе текущей конфигурации и доступного сетевого оборудования.

Одна из особенностей операционной системы Junos – возможность настроить «несуществующее» оборудование, поскольку предполагается, что аппаратное обеспечение может быть добавлено
на более поздний срок. Тогда при его подключении можно будет сразу же приступить к работе.

Примером может служить функционал, который можно настроить, например установить интерфейсы ge-1/0 / 0.0 семейного inet-адреса 192.168.1.1/24 и поднять интерфейс. Если предположить, что сейчас нет оборудования, эта конфигурация ничего не сделает. Как только необходимое оборудование будет подключено, поскольку аппаратное обеспечение уже было установлено, первый порт будет настроен немедленно с IP-адресом 192.168.1.1/24.

1.7.4 Фоновый процесс chassis

Фоновый процесс chassis поддерживает все процессы шасси, индикации неполадок и окружающих процессов.

На высоком уровне это включает в себя мониторинг работоспособности оборудования, управление базой данных инвентаризации оборудования в режиме реального времени и координация с аварийным фоновым процессом (alarmd) и фоновым процессом оборудования (craftd) для управления аварийными сигналами и светодиодами.

На рисунке 4 изображён интерфейс оборудования Juniper MX960.

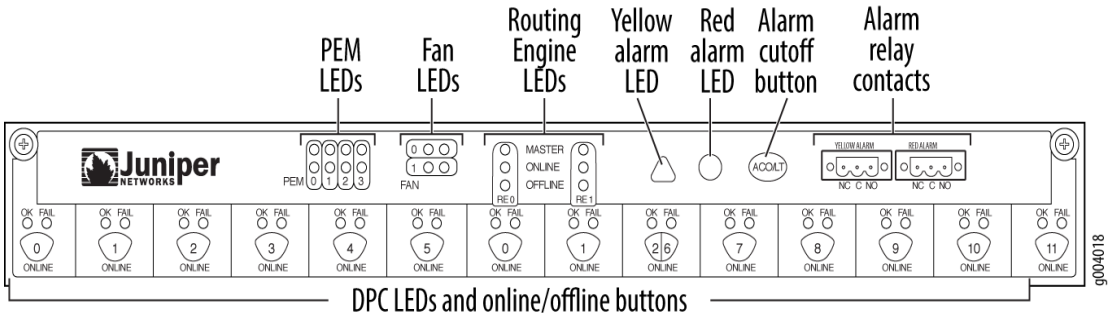

Рисунок 4 – Интерфейс оборудования Juniper MX960

Всё должно казаться само собой разумеющимся, за исключением данного интерфейса, который является видом панели устройства.

Интерфейс представляет собой набор кнопок и светодиодных индикаторов для отображения текущего состояния оборудования и аварийных сигналов. Также благодаря этому интерфейсу можно получить визуальную и программную информацию о работоспособности устройства.

Введём специальную команду «`show chassis craft-interface`», и на мониторе отобразится следующая информация:

```
dhanks@R1-RE0> show chassis craft-interface
Front Panel System LEDs:
Routing Engine 0 1
--------------------------
OK * *
Fail . .
Master * .
Front Panel Alarm Indicators:
-----------------------------
Red LED .
Yellow LED .
Major relay .
Minor relay .
Front Panel FPC LEDs:
```

```
FPC 0 1 2
------------------
Red . . .
Green . * *
CB LEDs:
CB 0 1
-------------
Amber . .
Green * *
PS LEDs:
PS 0 1 2 3
--------------------
Red . . . .
Green * . . .
Fan Tray LEDs:
FT 0
----------
Red .
Green *
```

Одна главная весьма ответственная функция фонового процесса chassis – это контроль окружающей среды и охлаждения.

Необходимо постоянно контролировать напряжение всех компонентов устройства и отправлять предупреждения, если напряжение пересекает любые пороговые значения. То же самое верно для охлаждения.

Фоновый процесс chassis постоянно контролирует температуру на всех компонентах и чипах, а также скорости вращения вентиляторов. Если произойдёт что-то необычное, данный фоновый процесс создаст заданные предупреждения. В экстремальных температурных условиях также возможно автоматическое выключение компонентов устройства, чтобы избежать повреждений.

1.8 Сокеты маршрутизации

Сокеты маршрутизации – это механизм операционной системы UNIX для управления таблицей маршрутизации операционной системы Junos.

Ядро принимает этот же механизм и расширяет его, чтобы включить дополнительную информацию для поддержки дополнительных атрибутов для создания сетевой операционной системы операторского класса.

На высоком уровне есть два участника при использовании роутинговых сокетов: «производитель» и «потребители», согласно рисунку 5. Фоновый процесс протокола маршрутизации отвечает за обработку обновлений маршрутизации и, таким образом, является производителем. Другие фоновые процессы считаются потребителями, поскольку они используют информацию процесса, полученную от сокетов маршрутизации.

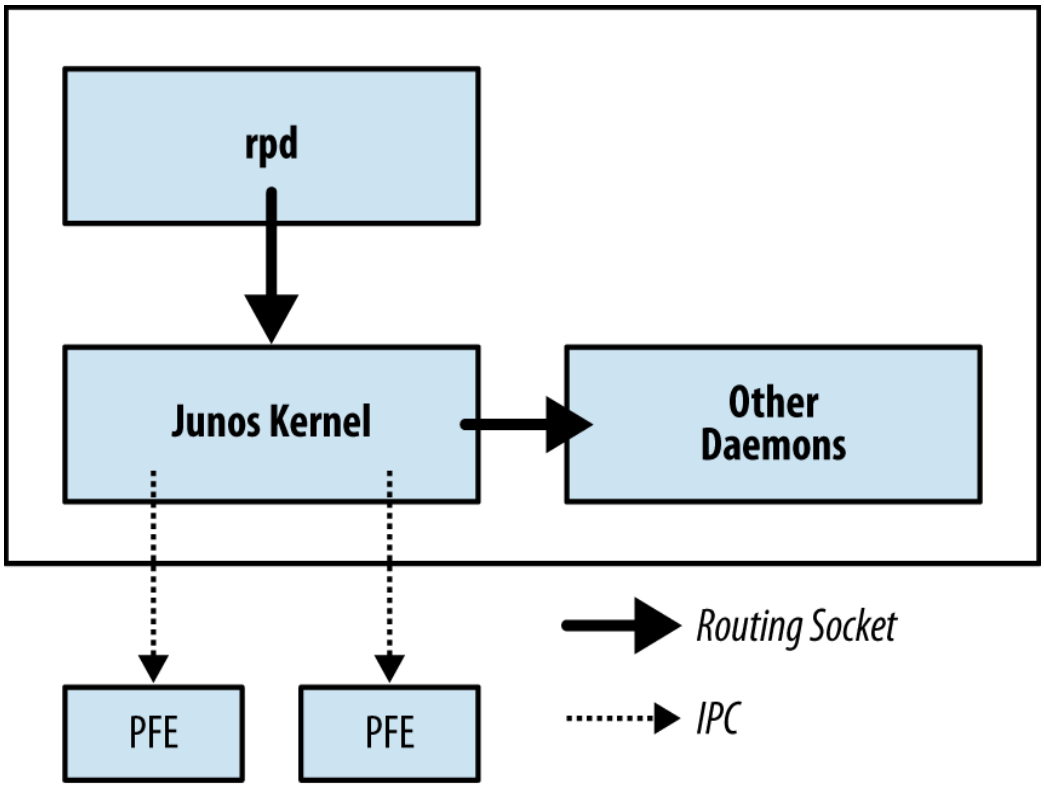

Рисунок 5 – Сокеты маршрутизации

Таким образом, именно благодаря фоновому процессу протокола маршрутизации другие фоновые процессы способны реализовать свой потенциал.

Протестируем текущие сокеты маршрутизации и посмотрим, что произойдёт, когда будет настроен интерфейс ge-1/0 / 0.0 с IP-адресом 192.168.1.1/24. Использование команды «rtsockmon» позволит увидеть, что команды подаются в ядро операционной системы Junos благодаря фоновым процессам.

```
{master}
dhanks@R1-RE0> start shell
dhanks@R1-RE0% rtsockmon –st
[15:38:21] dcd P iflogical add ge-1/0/0.0 flags=0x8000
[15:38:21] dcd P ifdev change ge-1/0/0 mtu=1514
dflags=0x3
[15:38:21] dcd P iffamily add inet mtu=1500
flags=0x8000000200000000
[15:38:21] dcd P nexthop add inet 192.168.1.255 nh=bcst
[15:38:21] dcd P nexthop add inet 192.168.1.0 nh=recv
[15:38:21] dcd P route add inet 192.168.1.255
[15:38:21] dcd P route add inet 192.168.1.0
[15:38:21] dcd P route add inet 192.168.1.1
[15:38:21] dcd P nexthop add inet 192.168.1.1 nh=locl
[15:38:21] dcd P ifaddr add inet local=192.168.1.1
[15:38:21] dcd P route add inet 192.168.1.1 tid=0
[15:38:21] dcd P nexthop add inet nh=rslv flags=0x0
[15:38:21] dcd P route add inet 192.168.1.0 tid=0
[15:38:21] dcd P nexthop change inet nh=rslv
[15:38:21] dcd P ifaddr add inet local=192.168.1.1
dest=192.168.1.0
[15:38:21] rpd P ifdest change ge-1/0/0.0, af 2, up, pfx
192.168.1.0/24
```

Команда «rtsockmon» является командой оболочки операционной системы Junos, которая дает пользователю возможность увидеть сообщения, передаваемые фоновым процессом протокола маршрутизации.

Пункты маршрутизации разбиты на четыре основных компонента: отправитель, тип, операция и аргументы. Поле отправителя используется для определения того, какой фоновый процесс записывает в таблицу маршрутизации. Поле тип идентифицирует фоновый процесс, который может изменяться. Поле операции показывает, что на самом деле было выполнено. Существует три основных операции: добавление, изменение и удаление. Последнее поле нужно для аргументов, переданных ядру операционной системы Junos. Это набор пар ключей и значений, которые также могут изменяться.

В предыдущем примере можно увидеть, как происходит взаимодействие с таблицей маршрутизации для настройки интерфейса ge-1/0 / 0.0 и назначение IPv4-адрес.

Фоновый процесс управления устройством создает новый логический интерфейс (IFL), затем изменяет интерфейсное устройство (IFD), чтобы установить соответствующий максимальный объём данных (MTU). Фоновый процесс управления устройством добавляет новое семейство интерфейсов (IFF) для поддержки IPv4. Далее он устанавливает ненужные, широковещательные и другие атрибуты, необходимые для системы и добавляет адрес интерфейса (IFA) 192.168.1.1. Фоновый процесс протокола маршрутизации, наконец, добавляет маршрут для 192.168.1.1 и выводит его на монитор.

1.9 Семейство оборудования Juniper MX

Оборудование фирмы Juniper серии MX поставляется в разных формах и конфигурациях.

На рисунке 6 показано оборудование (слева направо): MX80, MX240, MX480, MX960 и MX2020. Модели MX240 и выше имеют высокоуровневую архитектуру, в котором размещаются все компоненты. Модели MX80 и ниже считаются средними и принимают только интерфейс модули.

Рисунок 6 – Семейство оборудования Juniper MX

Ёмкости отложенного вызова процедур (DPC) и модульного концентратора портов (MPC) основаны на текущих аппаратно-технических характеристиках (4x10GE для DPC и 32x10GE для MPC) и могут быть изменены в будущем по мере выпуска нового оборудования.

Например, маршрутизатор Juniper MX960 может принимать до 12 линейных карт с использованием 4x10GE для DPC. Таким образом, пропускная способность достигает 960 гигабит в секунду.

Используя тот же расчёт для пропускной способности MPC (32x10GE для MPC), она может достигать 7,640 терабит в секунду.

Поскольку платформа MX обновляется, эти расчёты будут меняться с выходом нового оборудования и программного обеспечения.

2 Основы построения и работы сети пакетной передачи данных провайдера

В настоящее время проектирование внутригородских сетей пакетной передачи данных провайдера производится на основе трехуровневой модели построения сети компании. Такая модель строения сети провайдера называется иерархической и представляет собой определенное логическое разделение сети на три уровня, согласно рисунку 7: уровень доступа (access), уровень распределения (distribution), уровень ядра (core) [3].

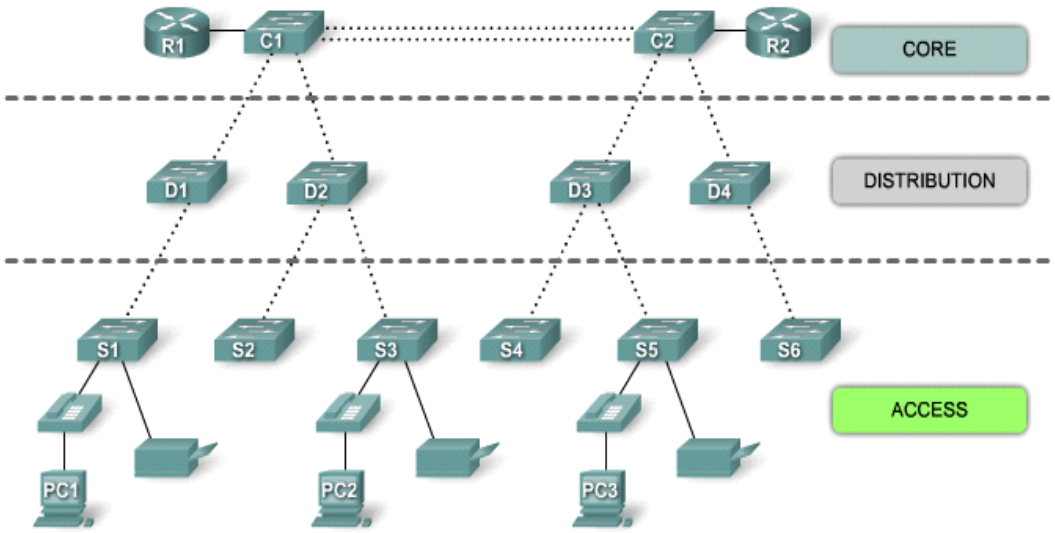

Рисунок 7– Деление сети провайдера на уровни

Вследствие использования провайдерами топологии построения типа «звезда», в иерархической модели можно обозначить ряд достаточно больших преимуществ. Сети, построенные на основе данной топологии, имеют хороший потенциал к масштабированию, так как при добавлении дополнительных модулей оборудования сеть не нуждается в изменении архитектуры.

Улучшенная безопасность достигается из-за возможности функционала так называемых политик, настраиваемых на уровне распределения. Данные политики регулируют доступность определённых узлов сети. В виду того, что иерархические сети являются модульными и достаточно легко масштабируются, их обслувивание также достаточно простое.

Каждый уровень иерархической модели реализует свои определенные функции. Однако уровни являются логическими и не всегда могут быть согласованы с физическими единицами оборудования.

Для более подробного исследования архитектуры и схемы работы сети пакетной передачи данных провайдера рассмотрим спроектированную схему части такой сети.

В данной смоделированной схеме сети провайдера пакетной передачи данных использовано оборудования различных видов, а также различных компаний-производителей. Основу уровня ядра иерархической сети составляют маршрутизаторы фирмы Juniper модели MX480.

Функциональная особенность данного уровня нуждается в оборудовании с высокой производительностью и большой пропускной способностью, чтобы максимально эффективно работали используемые технологии и протоколы.

Работа протоколов на данном уровне обусловлена функциями этого уровня, в числе которых быстрая коммутации большого объема трафика и обработка приходящих запросов с уровня доступа и уровня распределения. Работа с командной строкой в JUNOS разделяется на два режима: эксплуатационный режим и режим конфигурирования.

Эксплуатационный режим (Operational mode) – это режим для просмотра информации посредством команды «show», отслеживания ошибок через команды «monitor», «ping», «test», «traceroute». Данный режим в командной строке обозначается символом «>».

```
root@% cli
root> tracerout [ip-адрес]
```

Режим конфигурирования (Configuration mode) – это режим, в котором происходит непосредственно настройка устройства. Любое подключение и отключение протоколов происходит именно в этом режиме. В консоли он обозначается символом «#». В данный контекст можно попасть с помощью введения команд «`configure`» или «`edit`» в экплуатационном режиме.

```
root> configure
[edit]
root#
```

Структура команд эксплуатационного режима маршрутизатора представлена на рисунке 8.

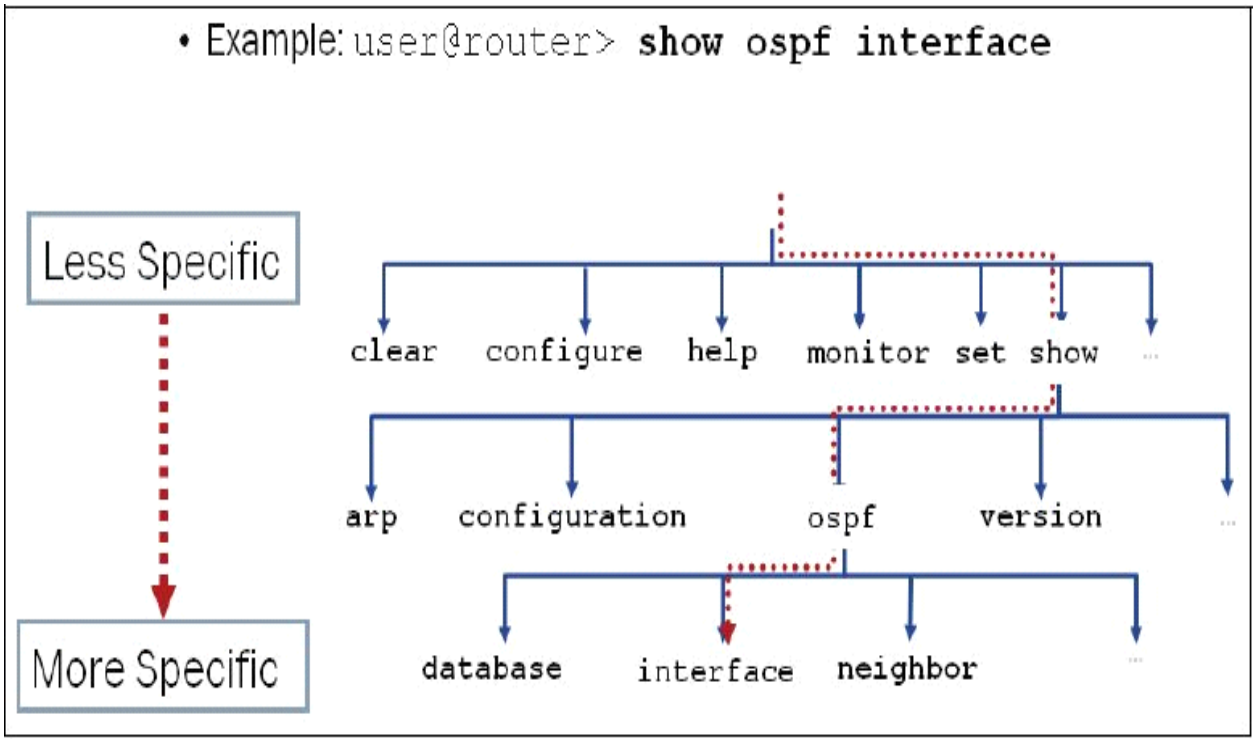

Рисунок 8 – Структура команд эксплуатационного режима

В режиме же конфигурирования работа устройства осуществляется засчёт уже прописанной конфигурации, которая сохранена в памяти оборудования. Такая конфигурация называется действующей. Однако, существует и другая конфигурация – «кандидат», используемая для настройки устройства в настоящее время. В данной конфигурации происходит изменение настроек и подключение протоколов.

Любая настройка протоколов в данной конфигурации происходит с помощью команды «`edit`», согласно рисунку 9.

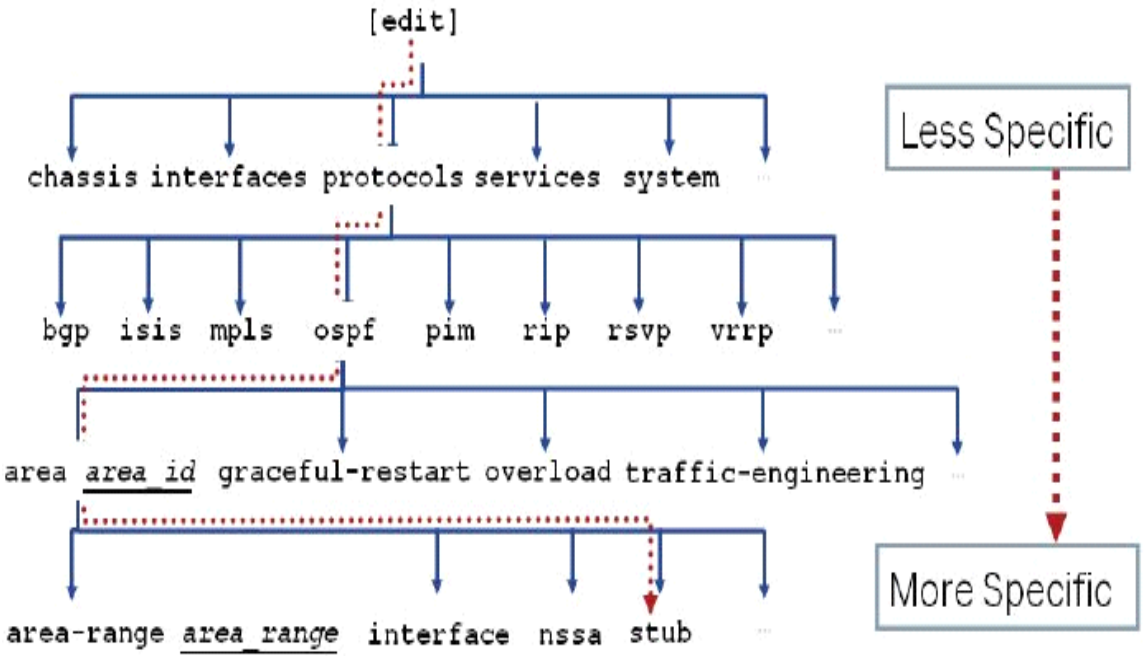

Рисунок 9 – Структура команды «edit»

Данная команда обозначает переход с одного уровня настройки протокола на другой. Таким образом существует несколько вариантов перехода на определенный уровень.

Изначальная настройка маршрутизаторов уровня ядра включает в себя обычные для любого другого оборудования настройки логина и пароля для обеспечения безопасности пользования, а также для возможности применения конфигурации протоколов и технологий. Также это дает

возможность подключения удаленно. Для настройки этих параметров нужно войти в конфигурационный режим.

```
login: root
Router1% cli
Router1> edit
```

Далее с помощью команды «`set system host-name`» прописывается логин, а с помощью команды «`set system root-auth pl`» пользователь задаёт пароль.

```
RouterMX-1# set system host-name <<name>>
RouterMX-1# set system root-auth pl
New password:
Retype new password:
```

В дальнейшем пароль и логин шифруются для обеспечения лучшей безопасности. Таким образом, при выводе настроенной конфигурации маршрутизатора для просмотра в поле пароля и логина будут отображаться различные символы, которые не имеют ничего общего с логином и паролем пользователя:

```
RouterMX-1#set system root-authentication encrypted-password "$1$E/cUvZpS$xkUCM8vUyXsqNpgISUAlp1"
```

3 Метод «проброса» VLAN’ов

Чтобы создать виртуальные сети на локальном уровне, используется технология VLAN, которая позволяет сконфигурировать несколько виртуальных широковещательных сетей в пределах одного физического домена. При этом эти сети не сообщаются между собой. Разграничение доступа и трафика происходит на уровне коммутатора, который имеет возможность выделять на канальном уровне одного или нескольких пользователей в группу по признакам, которые задаёт сетевой администратор.

Для объединения сетей и организации между ними VPN, офисы клиента разделяют на отдельные VLAN’ы. То есть существует группа компьютеров, подключенных к сети, логически объединённые в домен рассылки широковещательных сообщений по заданным параметрам. Например, такие группы компьютеров могут быть выделены в зависимости от структуры предприятия.

Сети VLAN дают несколько преимуществ: значительно более эффективное использование пропускной способности (в сравнении с традиционными локальными сетями), повышенная степень защиты передающейся информации, а также упрощенной схеме администрирования.

Повышенная пропускная способность достигается засчёт разбиения всей сети на широковещательные домены. Трафик внутри такой структуры передается только между её членами, а не всем компьютерам в физической сети. Получается, что широковещательный трафик, который генерируется серверами, ограничен предопределенным доменом, то есть не транслируется всем станциям в этой сети. Так достигается оптимальное распределение пропускной способности сети между выделенными группами компьютеров. Компьютеры из разных VLAN’ов не видят друг друга.

Для ускорения передачи информации VLAN’ы отключают от IP маршрутизации и пробрасывают их сквозь маршрутизаторы, то есть именно маршрутизаторы осуществляют коммутацию каналов. При этом сохраняются резервирование и уникальность подключения. Благодаря этому повышается защищенность их соединения [4].

Построение VLAN для технологии Ethernet основано на стандарте 802.11Q, согласно которому в заголовке кадра Ethernet устанавливается номер подсети, обрабатываемый коммутаторами и/или сетевыми картами. Один и тот же порт коммутатора (сетевую карту) можно ассоциировать с несколькими номерами виртуальных подсетей для организации доступа к общему сетевому ресурсу (серверу). При организации VLAN не предполагается использование какой-либо защиты для кадра. Независимость виртуальных подсетей друг от друга построена на отправке нужного кадра коммутатором или пропуске прочих кадров сетевой картой. Отсюда можно сделать вывод, что такой подход оправдан только тогда, когда информация физически не выходит за пределы одной организации и когда есть возможность гарантировать защиту от несанкционированного перехвата.

Почтовое сообщение содержит конверт с необходимой для доставки информацией, заголовка с полезными для автоматизированной обработки адресатом данными и собственно сообщения. Конверт и заголовок имеют формализованные поля, тело - набор строк из не более, чем 1000 (рекомендуется до 78) ASCII- знаков. Для передачи символов в национальных кодировках (например, знаков кириллицы), двоичных файлов (например, с аудио, или видео информацией) используется соглашение MIME, предусматривающее дополнительные поля заголовка.

На данный момент многие современные организации и предприятия практически не используют такую весьма полезную, а часто и необходимую, возможность, как организация виртуальной локальной сети

(VLAN) в рамках цельной инфраструктуры, которая предоставляется большинством современных коммутаторов. Связано это со многими факторами, поэтому стоит рассмотреть данную технологию с позиции возможности ее использования в таких целях.

В качестве исследуемой сети рассматривается сеть на рисунке 10. Настройки будут приведены в приложении А.

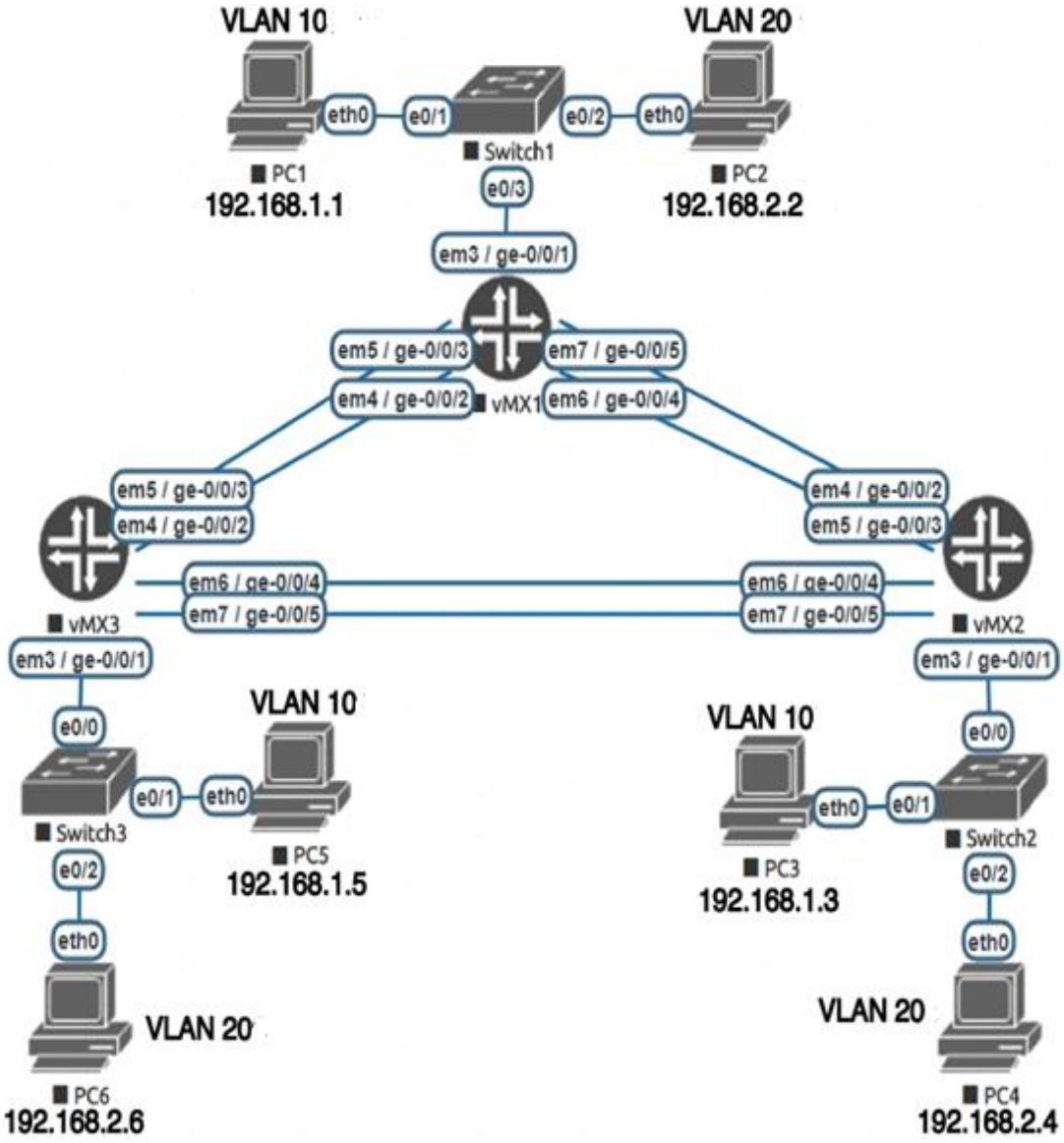

Рисунок 10 – Схема моделируемой частной сети

В этой сети находятся два клиента с тремя офисами. Первый клиент находится в сети 192.168.1.0 для которой выделен VLAN 10, второй клиент в сети 192.168.2.0 с VLAN’ом 20. В результате необходимо организовать проброс VLAN’ов из одних сетей в другие через маршрутизаторы. На роутерах Juniper серии MX это осуществляется с помощью технологии VLAN bridge. Для увеличения пропускной способности несколько физических интерфейсов будут объединены в один логический. Таким образом представленная на рисунке схема должна функционировать следующим образом: PC1, PC3, PC5 принадлежат первому клиенту, получившему VLAN 10, PC2, PC4, PC6 относятся ко второму клиенту, и отделены во VLAN 20. Оба клиента могут обмениваться информацией только между своими офисами, доступа друг к другу они не имеют. Так как данная сеть замкнута, то необходимо использовать технологию на подобии STP, которая будет убирать топологические петли, чтобы избежать broadcast шторма, уменьшающего полезный трафик, или вовсе парализующего систему.

4 Организация IP-over-IP и GRE туннелей

4.1 Организация IP-over-IP туннеля

IP-туннели предназначены для соединения между собой двух разных сетей, не имеющих прямого маршрута друг к другу. Пользователи IP-over-IP туннеля должны иметь внешние IP-адреса (или находится в одной сети), а также между ними не должно быть никакой трансляции адресов NAT. Это необходимые условия для установки туннеля.

В IP-over-IP туннелях каждый IP-пакет, в том числе и адресная информация источника и назначения IP-сети, инкапсулируется в формат пакета, родной для транзитной сети.

На границах между сетями источника и транзитной сети, а также транзитной сетью и сетью назначения, используются шлюзы, которые устанавливают концы IP-туннеля через транзитную сеть. Таким образом, концы IP-туннеля становятся связанными IP-маршрутизаторами, которые устанавливают стандартный IP-маршрут между сетями источника и получателя. У пакетов, которые пришли в эти конечные точки, удаляется их транзитный заголовок кадра, и, таким образом, превращается в исходный формат IP и входит в IP-стек конечной точки туннеля. Кроме того, инкапсуляция любого другого протокола, используемого при транспортировке удаляется.

В простом виде, никаких механизмов безопасности для этих туннелей не предусмотрено, шифрование данных для защиты информации, передаваемой через публичные сети доступа, обеспечивается с помощью различных протоколов шифрования данных, например, IPsec или Transport Layer Security передаваемых по межсетевым протоколам, обеспечивая функциональность VPN. Принцип работы IP туннелей заключается в инкапсуляции любого протокола сетевого уровня внутри виртуального двухточечного канала. Для формирования соединения необходима промежуточной транспортной сети. Наиболее часто используемые протоколы, поддерживающие IP

туннелирование, это GRE и IP- over -IP(IPIP).

Они представляются в системе в виде интерфейсов gr-* и ip-*, и через них можно прокидывать маршрутизацию точно также, как и через любые другие интерфейсы. Плюс ко всему для этих интерфейсов можно настроить уровень доступа security level — private, protected или public.

Главным минусом IP туннелей является то, что между двумя точками туннеля обязательно должна быть прямая связь. То есть, либо они обе находятся в одной серой сети, либо у обоих серверов есть публичные адреса, отсутствует NAT адресация и firewall между ними разрешает пропускать любой трафик. IP over IP (IPIP) один из самых простых в настройке туннелей (инкапсулирует только unicast IPv4-трафик). Его можно настроить как на UNIX/Linux-системе, так и на различных маршрутизаторах. GRE (Generic Routing Encapsulation) туннель является одним из популярных разновидностей VPN. Работает он по принципу добавления к исходному пакету данных своего заголовка, за которым следует новый IP заголовок с новыми IP адресами. Туннели GRE совместимы с аппаратными шлюзами безопасности.

В качестве исследуемой сети рассматривается сеть на рисунке 11. Настройки будут приведены в приложении Б.

В этой сети находятся два клиента с двумя офисами. Первый клиент находится в сети 192.168.3.0 для которой выделен VLAN 100, второй клиент в сети 192.168.4.0 с VLAN’ом 200, сети клиентов не имеют доступа друг к другу. В офисах клиентов передача пакетов информации происходит с помощью IP маршрутизации. Далее трафик через маршрутизатор инкапсулируется в IP туннель, транспортируемый промежуточным маршрутизатором vMX3, выступающим в роли интернета, связывающим между собой две сети клиентов. На маршрутизаторах vMX1 vMX2, от сетей в которых расположены офисы, настроена статическая маршрутизация к интерфейсам туннеля. Маршрутизация трафика между концами туннеля происходит с помощью транспортной сети интернет. Соединение создается через интерфейс ip-*/*/* роутера vMX1, на которых указывается сеть-

источник трафика и сеть-получатель, находящаяся на другом конце. Аналогично туннель создается и на роутере vMX2. Компьютеры PC1,PC3 принадлежат VLAN’у 100, PC2,PC4 во VLAN’е 200 и выступают в качестве конечных точек клиентских сетей.

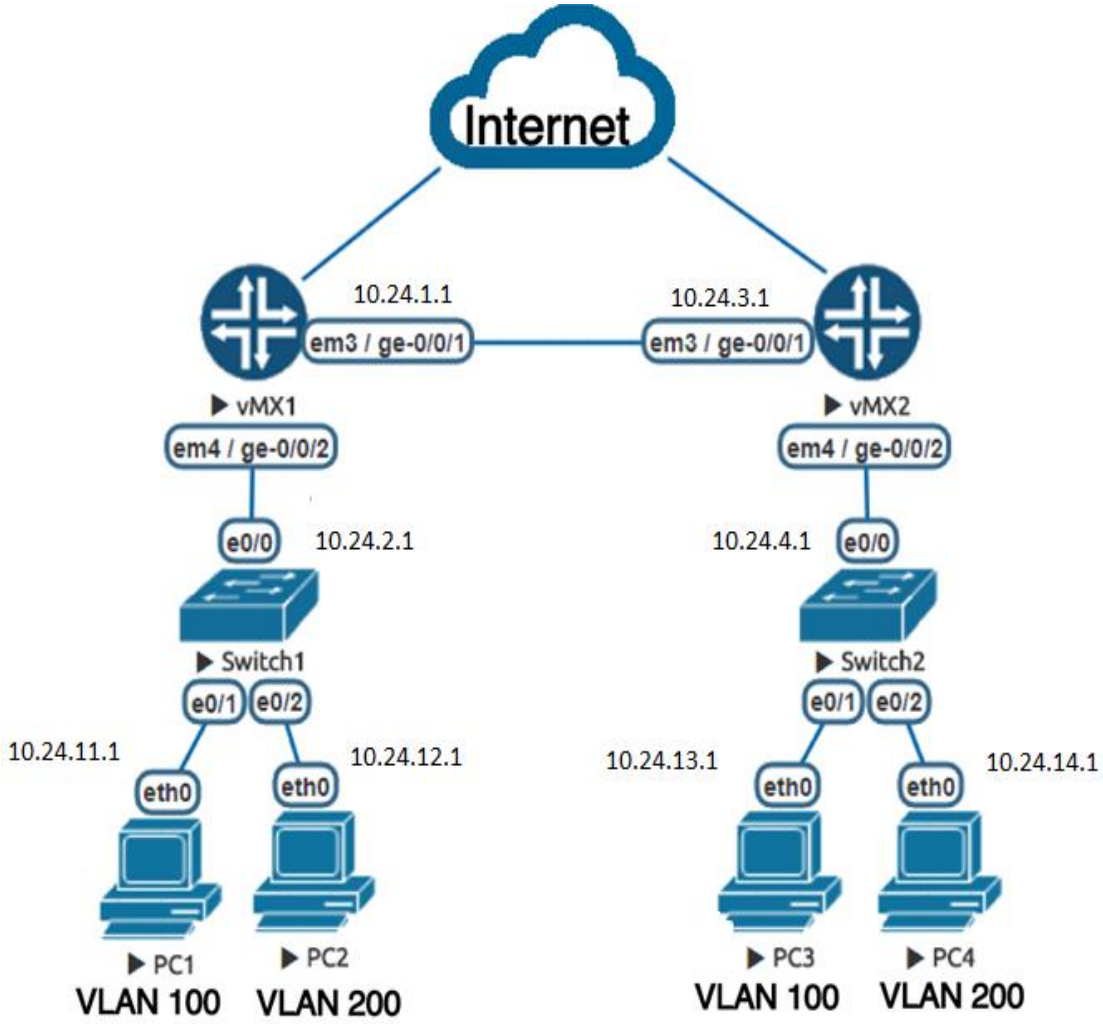

Рисунок 11 – Схема работы IP-over-IP

4.2 Организация GRE туннеля

Протокол туннелирования GRE, как и IP-over-IP, передаёт широковещательный трафик, что позволяет пропускать через такой туннель протоколы шифрования, например такие как IPSec и IPsec-Tools Racoon. В отличии от IP-over-IP туннеля, инкапсулирующего только unicast IPv4-трафик, GRE туннель работает с IPv4 и с IPv6 unicast и multicast трафиком.

Так же GRE туннель может «пробросить» особые протоколы, такие как NetBios, IPX, AppleTalk. Главный недостаток туннеля GRE в том, что он добавляет два дополнительных заголовка к нашему исходному пакету IP. Таким образом, если клиент обеспокоен пропускной способностью, из-за данной особенности могут возникнуть проблемы.

Однако, если поменять туннельный режим на другой, то можно будет уменьшить передаваемые данные, поскольку IP-over-IP туннель добавляет один заголовок вместо двух по сравнению с GRE туннелем. Так что это делает его более эффективным. Другим недостатком является то, что туннели GRE не являются очень масштабируемым решением.

Для конфигурирования GRE туннеля взята аналогичная топология (рисунок 11) с сетями двух клиентов, которым принадлежат два офиса. Настройки будут приведены в приложении В. Первый клиент находится в сети 192.168.3.0, для которой выделен VLAN 100, второй клиент находится в сети 192.168.4.0 с VLAN’ом 200. При этом сети клиентов не имеют доступа друг к другу.

GRE туннель создаётся через интерфейс gr-*/*/* роутера vMX1, на котором указывается сеть-источник трафика и сеть-получатель, находящаяся на другом конце сети. Аналогичный GRE туннель создаётся и на роутере vMX2. Компьютеры PC1, PC3 принадлежат VLAN’у 100, PC2, PC4 во VLAN’е 200 и выступают в качестве конечных точек клиентских сетей.

5 Исследование метода организации VPN на базе MPLS

MPLS (мультипротокольная коммутация по меткам) — механизм передачи данных, который эмулирует различные свойства сетей с коммутацией каналов поверх сетей с коммутацией пакетов данных на основе меток. Ключевой плюс технологии - крайне высокая скорость обработки. Схема работы достаточно проста - для каждого потока пакетов, например, протокола IPv4, создаётся собственная числовая метка, которая добавляется между заголовками L2 и L3, и дальнейшая обработка пакетов идёт не путём достаточно сложной маршрутизации, а быстрой коммутации.

Была рассмотрена организация сети с применением MPLS L2, так как сервисы L2 VPN обычно используются для построения корпоративных сетей в рамках одного города (или города и ближайших окрестностей) [5]. Для таких сервисов характерны большие скорости каналов при меньшей (по сравнению с L3 VPN) стоимости соединения. Достоинствами L2 VPN являются также поддержка кадров увеличенного размера (jumbo frame), относительная простота и дешевизна оборудования клиента, устанавливаемого на границе с провайдером (L2). Рост популярности сервисов L2 VPN во многом связан с потребностями отказоустойчивых территориально распределенных центров обработки данных (ЦОД): для «путешествий» виртуальных машин требуется прямое подключение между узлами на уровне L2. Такие сервисы, по сути, позволяют растянуть домен L2. Это хорошо отлаженные решения, но часто требующие сложной настройки. В частности, при подключении ЦОД к сети сервис-провайдера в нескольких точках — а это крайне желательно для повышения отказоустойчивости — требуется задействовать дополнительные механизмы, чтобы обеспечить оптимальную загрузку соединений и исключить возникновение «петель коммутации» .

Для построения любого L2VPN существуют два концептуально разных подхода. Подход «точка-точка» изображён на рисунке 12, применим к любым типам протоколов канального уровня и в принципе, в полной мере

исчерпывает все сценарии применения L2VPN. Он поддерживает все мыслимые и немыслимые протоколы, соединяет два узла друг с другом. Сеть провайдера будет как один виртуальный кабель — то, что вошло в него на одном конце, обязательно выйдет на другом без изменений. В его основе лежит концепция псевдопровода. Общее название услуги: VPWS — Virtual Private Wire Service.

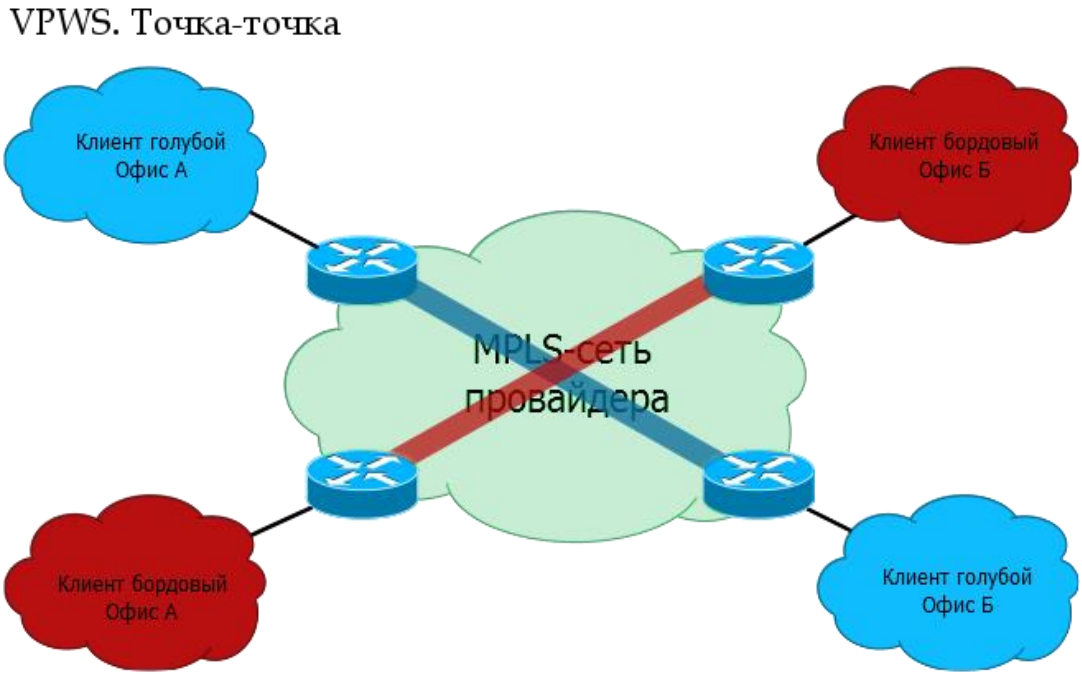

Рисунок 12 – Схема работы VPWS

Подход точка-многоточка изображён на рисунке 13. Этот режим только для Ethernet, поскольку только в нём фактически такая необходимость есть. В этом случае у клиента может быть три-пять-десять-сто точек подключения/филиалов, и все они должны передавать данные друг другу, причём, как одному конкретному филиалу, так и всем сразу.
Название технологии: VPLS — Virtual Private LAN Service.

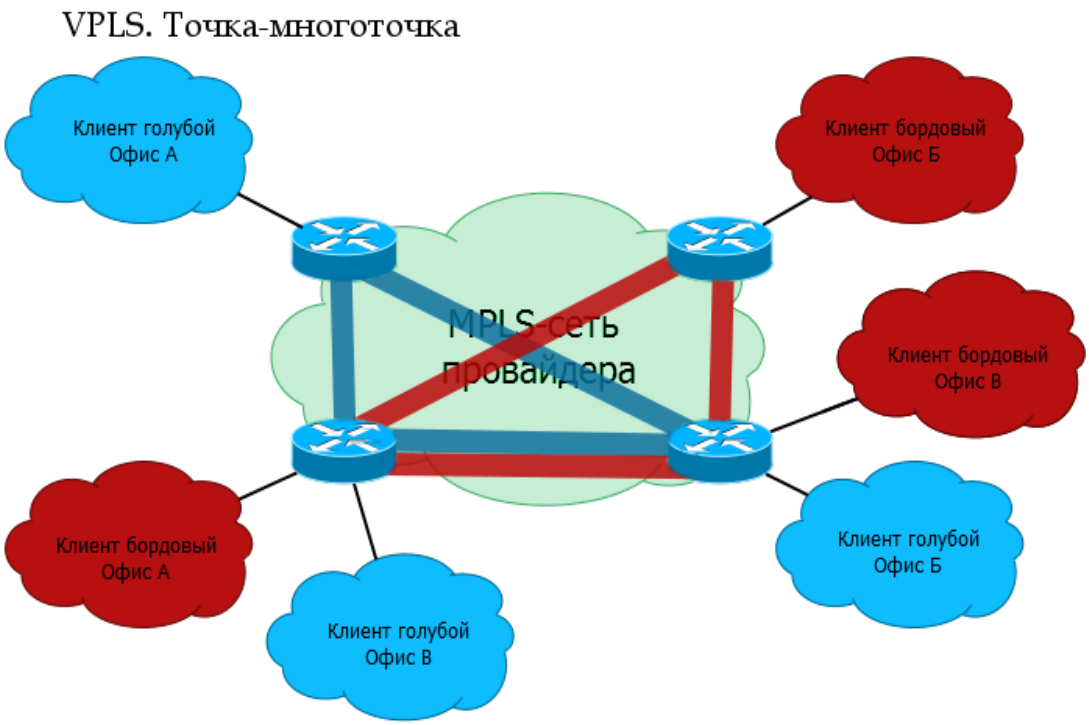

Рисунок 13 – Схема работы VPLS

В качестве исследуемой локальной MPLS L2 сети рассматривается сеть на рисунке 14. Настройки будут приведены в приложении Г. Для реализации маршрутизации через MPLS на маршрутизаторах vMX1, vMX3, принадлежащих клиенту, необходимо отсутствие специальной конфигурации. Маршрутизатор vMX2 настраивается в качестве промежуточного коммутатора, передающего трафик от граничных роутеров сети провайдера. Коммутаторы поставщика требуют, чтобы MPLS и LDP были настроены на интерфейсах, которые будут принимать и передавать MPLS пакеты от офисов клиента. LDP отвечает за распространение сервисных меток, протокол ищет соседей среди непосредственно подключенных маршрутизаторов. Как только с обеих сторон появятся AC-интерфейсы с одинаковым параметром VC-ID в сетях двух сетях одной виртуальной частной сети, LDP поможет им сообщить друг другу метки [6].

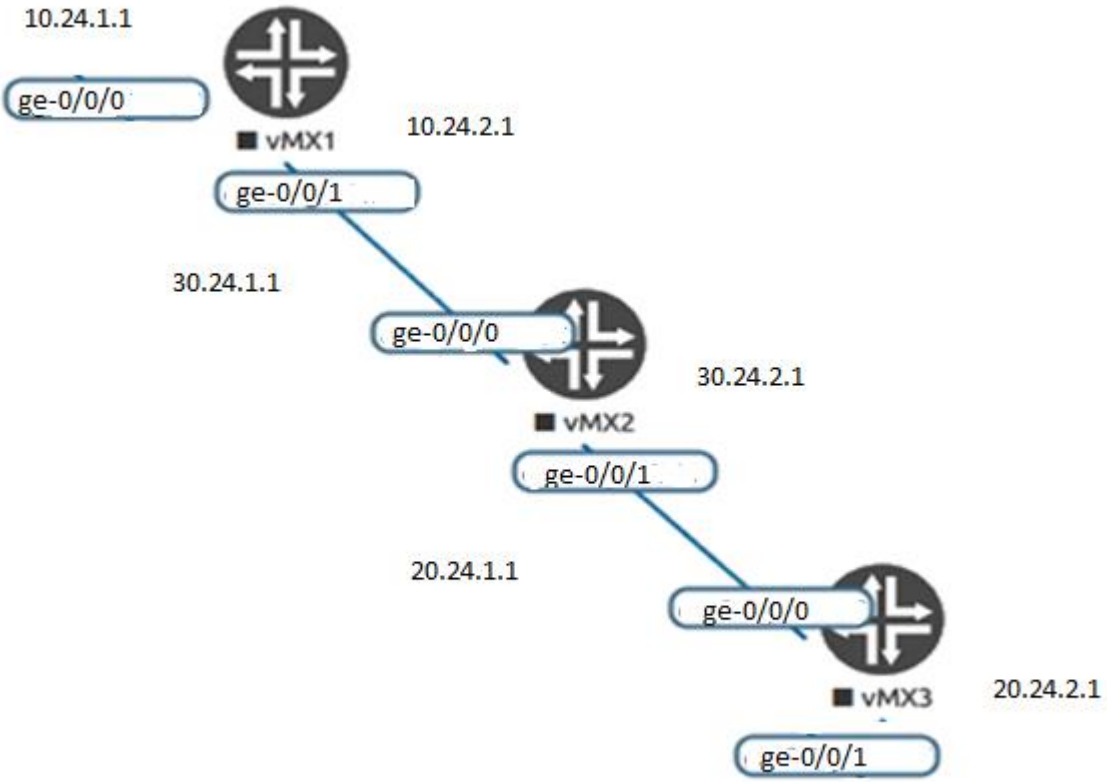

Рисунок 14 – Схема работы MPLS L2

6 Политики и фильтры

Фильтры и политики широко применяются в процессе управления трафиком в сетях, служат различным целям, и подходят для разных ситуаций.

Фильтр – это механизм, который применяется для отсеивания пакетов на входе в маршрутизатор и который выполняет определённые действия, необходимые оператору. Чаще всего это принять либо отбросить данные, в зависимости от выбранных свойств, также для применения механизмов называемых filter based routing (fbr) посредством которого представлен механизм policy based routing в операционной системе Junos [7].

Политики используются для управления маршрутами на сети, изменения стандартного поведения протоколов динамической маршрутизации.

Простейший пример фильтра, чтобы можно было представить, как выглядит его структура.

```
lab@mxA-1# show firewall filter test
interface-specific;
term 1 {
from {
protocol udp;
}
then {
count test;
discard;
}
}
term 2 {
then accept; }
```

В приведенном примере показан фильтр, который считает и отбрасывает все пакеты, полученные по протоколу UDP, но принимает все другие пакеты. Все фильтры делятся на части, называемые term. Каждый term имеет свое название и содержит свой набор действий, применимых к пакету. Сначала, после параметра «from», выставляется условие, в нашем случае это protocol udp. Потом указывается действие, которое необходимо выполнить при выполнении заданного условия, т.е. посчитать и отбросить. В фильтре так же содержится второй term, который вступает в действие, в случае невыполнения условий, заданных в первом term.

Все фильтры и политики состоят из частей, называемых «term». Каждый term содержит условия совпадения и действия, которые надо выполнить в результате совпадения условий. Все term’ы выполняются строго по порядку, до выполнения одного из действий. Как только у нас сработал один из них, фильтр или политика прекращают свою работу. Для того, чтобы выполнить несколько term подряд, необходимо добавить специальное действие, которое называется «next term».

```
nagru# show firewall filter test
term 1 {
from {
source-address {
192.168.1.230/32;
}
protocol udp;
}
then {
count test;
next term;
}
}
term 2 {
```

```
then accept;
}
```

В данном случае у нас будут работать оба term даже в случае выполнения условия в первом term.

Если более детально рассмотреть команду «from», то можно получить список условий, соответствие которым ставить фильтр.

```
nagru# set firewall filter test term 1 from ?
Possible completions:
> address Match IP source or destination address
+ apply-groups Groups from which to inherit configuration data
+ apply-groups-except Don't inherit configuration data from these groups
> destination-address Match IP destination address
+ destination-port Match TCP/UDP destination port
+ destination-port-except Do not match TCP/UDP destination port
> destination-prefix-list Match IP destination prefixes in named list
+ dscp Match Differentiated Services (DiffServ) code point
+ dscp-except Do not match Differentiated Services (DiffServ) code point
+ esp-spi Match IPSec ESP SPI value
+ esp-spi-except Do not match IPSec ESP SPI value
first-fragment Match if packet is the first fragment
+ forwarding-class Match forwarding class
+ forwarding-class-except Do not match forwarding class
fragment-flags Match fragment flags (in symbolic or hex formats) — (Ingress only)
```

+ fragment-offset Match fragment offset
+ fragment-offset-except Do not match fragment offset
+ icmp-code Match ICMP message code
+ icmp-code-except Do not match ICMP message code
+ icmp-type Match ICMP message type
+ icmp-type-except Do not match ICMP message type
> interface Match interface name
+ interface-group Match interface group
+ interface-group-except Do not match interface group
> interface-set Match interface in set
+ ip-options Match IP options
+ ip-options-except Do not match IP options
is-fragment Match if packet is a fragment
+ packet-length Match packet length
+ packet-length-except Do not match packet length
+ port Match TCP/UDP source or destination port
+ port-except Do not match TCP/UDP source or destination port
+ precedence Match IP precedence value
+ precedence-except Do not match IP precedence value
> prefix-list Match IP source or destination prefixes in named list
+ protocol Match IP protocol type
+ protocol-except Do not match IP protocol type
service-filter-hit Match if service-filter-hit is set
> source-address Match IP source address
+ source-port Match TCP/UDP source port
+ source-port-except Do not match TCP/UDP source port
> source-prefix-list Match IP source prefixes in named list
tcp-established Match packet of an established TCP connection
tcp-flags Match TCP flags (in symbolic or hex formats)

```
tcp-initial Match initial packet of a TCP connection
+ ttl Match IP ttl type
+ ttl-except Do not match IP ttl type
[edit]
```

Можно сделать отбор по широкому спектру параметров ip пакета, на которые можно настроить фильтр для выполнения определенных действий. Таким образом можно очень гибко настроить фильтрацию трафика на порту устройства, задав необходимые критерии отбора. Останавливаться на каждом подробно нет смысла, каждый инженер сам знает какие критерии ему нужны.

Возможные варианты действий над выбранными пакетами:

```
nagru# set firewall filter test term 1 then ?
Possible completions:
accept Accept the packet
+ apply-groups Groups from which to inherit configuration data
+ apply-groups-except Don't inherit configuration data from these groups
count Count the packet in the named counter
> discard Discard the packet
forwarding-class Classify packet to forwarding class
log Log the packet
loss-priority Packet's loss priority
next Continue to next term in a filter
packet-mode Bypass flow mode for the packet
policer Name of policer to use to rate-limit traffic
port-mirror Port-mirror the packet
> reject Reject the packet
> routing-instance Packets are directed to specified routing instance
sample Sample the packet
```

```
    service-accounting Count the packets for service
accounting
    service-filter-hit Marked when packet processing by the
current type of chained filters is done, the packet is
directed to the next type of filters
    syslog System log (syslog) information about the packet
    topology Packets are directed to specified topology
    virtual-channel Set the output interface virtual channel
    [edit]
```

Как видно из примера, возможных действий меньше чем возможных критериев отбора. Остановимся на основных из них:

1. discard — отброс пакета, без отсылки ICMP уведомления.
2. reject — отброс пакета с отсылкой ICMP уведомления.
3. count filename — посчитать пакет с записью в специально создаваемый файл.
4. log — залогировать пакет в файл логирования.
5. policer policer name — данная функция используется для ограничения полосы пропускаемого трафика на интерфейсе. Может применяться как на входящий трафик так и на исходящий. Для этого необходимо создать специальный параметр policer:

```
    nagru# show firewall policer test
    if-exceeding {
    bandwidth-limit 1024000000;
    burst-size-limit 10k;
    }
    then discard;
    [edit]
```

В данном случае мы поставили ограничение на 1 Гбит/с. Данный синтаксис показывает пример настройки, ограничивающей полосу пропускания.

Теперь рассмотрим механизмы filter based routing (fbr). Приведем пример настройки функционала fbr.

```
nagru# show firewall filter fbr
term 1 {
from {
address {
10.10.10.1/30;
}
}
then {
routing-instance fbr;
}
}
term 2 {
then accept;
}
[edit]

nagru# show routing-instances fbr
instance-type forwarding;
routing-options {
static {
route 0.0.0.0/0 next-hop 10.10.15.1;
} }
[edit]
nagru# show routing-options
interface-routes {
rib-group fbr;
```

```
}
rib-groups {
fbr {
import-rib [ inet.0 fbr.inet.0 ];
} }
```

В данной конфигурации мы изменили маршрутизацию пакетов, приходящих с источника с IP-адресом 10.10.10.1.

По умолчанию все пакеты, приходящие на маршрутизатор от источника 10.10.10.1, проходят через общие правила маршрутизации, настроенные заранее.

Изменим движение трафика от данного источника, направив его через оборудование с адресом: 10.10.15.1. Для этого необходимо создать специальный фильтр, который выполняет действие:

```
}
then {
routing-instance fbr;
}
```

При выполнении условия, наш трафик помещается в специальный vrf, который используется для форвардинга пакетов.

Далее в данном vrf создается статический маршрут по умолчанию, используемый для маршрутизации пакетов к другому получателю, т.е. 10.10.15.1.

Теперь самое интересное. В нашем vrf отсутствует маршрут до 10.10.15.1, и соответственно маршрутизатор не знает куда направлять пакеты. Для того что бы исправить ситуацию, нужно сделать последнюю настройку с помощью функционала rib-groups.

Каждое устройство под управлением Junos OS ведет несколько таблиц маршрутизации, таких как например inet.0 inet.3 vrf.inet.0. В разных таблицах

содержатся различная маршрутная информация. Например, в inet.0 содержится основная (глобальная) информация о маршрутах на устройстве.

С помощью функционала rib-groups возможно осуществить перенос маршрутов из одной таблицы маршрутизации в другую.

В нашем примере мы взяли все маршруты локально назначенные на наших интерфейсах, и поместили их в таблицу маршрутизации.

```
nagru# show routing-options
interface-routes {
rib-group fbr;
}
rib-groups {
fbr {
import-rib [ inet.0 fbr.inet.0 ];
}}
```

Теперь наш маршрут по умолчанию сможет направить пакеты по заданному пути.

Общая структура политик выглядит следующим образом:

```
nagru# show policy-options
policy-statement test {
term 1 {
from community 65400:2000;
then as-path-prepend 65400;
}
term 2 {
from {
prefix-list list1;
}
```

```
then {
community add 65400:3000;
}}}
[edit]
nagru#
```

Как видно из примера, политика как и фильтр состоит из частей, называемых «term». Каждый term содержит условия совпадения и действия, выполняемые в результате совпадения условий.

Политики являются очень гибким механизмом для осуществления маршрутизации пакетов, и применяются уже не на интерфейс, а в самых различных местах в конфигурации оборудования Juniper Network. Например, целиком к таблице «forwarding table», или к какому-либо протоколу, на входе или выходе из vrf.

В приведенном выше примере политика состоит из двух term, в случае выполнения условия в основном term, перехода к следующему не будет осуществлено, так же обстоит дело и в цепочке политик.

Если в каком-либо разделе конфигурации присутствует цепочка политик, состоящая из двух или трех штук, к примеру, то при выполнении одной из политик перехода к следующей не будет осуществлено если не указать специальное действие:

```
policy-statement test {
term 1 {
then next policy;
} }
```

Тогда при выполнении данной политики, произойдет переход к следующей политике в цепочке.

Рассмотрим варианты условий в политике:

```
nagru# set policy-options policy-statement test from ?
Possible completions:
aggregate-contributor  Match more specifics of an aggregate
+ apply-groups Groups from which to inherit configuration data
+ apply-groups-except Don't inherit configuration data from these groups
area OSPF area identifier
+ as-path Name of AS path regular expression (BGP only)
+ as-path-group Name of AS path group (BGP only)
color Color (preference) value
color2 Color (preference) value 2
+ community BGP community
> community-count Number of BGP communities
+ condition Condition to match on
> external External route
family
instance Routing protocol instance
+ interface Interface name or address
level IS-IS level
local-preference  Local preference associated with a route
metric Metric value
metric2 Metric value 2
metric3 Metric value 3
metric4 Metric value 4
> multicast-scope Multicast scope to match
+ neighbor Neighboring router
+ next-hop Next-hop router
next-hop-type Next-hop type
origin BGP origin attribute
+ policy Name of policy to evaluate
```

```
preference Preference value
preference2 Preference value 2
> prefix-list List of prefix-lists of routes to match
> prefix-list-filter List of prefix-list-filters to match
+ protocol Protocol from which route was learned
rib Routing table
> route-filter List of routes to match
route-type Route type
> source-address-filter List of source addresses to match
state Route state
+ tag Tag string
tag2 Tag string 2
[edit]
nagru# set policy-options policy-statement test from
```

Выбор условий для работы политики довольно большой, даёт очень гибкий инструмент для операций над маршрутами. Описывать подробно каждый атрибут не имеет смысла, все зависит от целей политики и от уровня знаний каждого специалиста. Лучше рассмотреть какие действия можно совершить после отбора.

Также любой провайдер услуг, может создать список своих личных community, которые при назначении на маршрут будут выполнять определенные действия.

Описание простой политики, которая очень часто применяется в маршрутизации: экспорт либо импорт маршрутов между протоколами. Например, необходимо отдать статический маршрут в BGP или IGP или передать маршруты между BGP и IGP.

Для этого в условии соответствия выбирается необходимый протокол:

```
set policy-options policy-statement test from protocol ?
Possible completions:
[Open a set of values]
access Access server routes
access-internal Internal routes to directly connected clients
aggregate Aggregate routes
bgp BGP
direct Directly connected routes
dvmrp Distance Vector Multicast Routing Protocol
esis End System-to-Intermediate System
isis Intermediate System-to-Intermediate System
l2circuit Layer 2 circuits
l2vpn Layer 2 MPLS virtual private networks
ldp Label Distribution Protocol
local Local system addresses
msdp Multicast Source Discovery Protocol
ospf Open Shortest Path First
ospf2 Open Shortest Path First Version 2
ospf3 Open Shortest Path First Version 3
pim Protocol Independent Multicast
rip Routing Information Protocol
ripng Routing Information Protocol next generation
rsvp Resource Reservation Protocol
rtarget Local route target VPN membership
static Statically defined addresses
[edit]
```

Выбор протоколов довольно большой. Нужно указать, откуда будет импортироваться маршрут. Далее следует просто выбрать действие и присвоить эту политику на импорт или на экспорт к протоколу, в который следует произвести импорт маршрутов, или экспорт.

```
show policy-options
policy-statement test {
from protocol static;
then {
accept;
}
[edit]
```

ЗАКЛЮЧЕНИЕ

Основные результаты выпускной квалификационной работы состоят в следующем:

Проведено моделирование в эмуляционной среде GNS3 соединения части сети провайдера по технологии «проброс» VLAN`ов на маршрутизаторах серии MX фирмы Juniper с использованием интерфейса irb.

В эмуляционной среде GNS3 настроен пример соединения двух сайтов при помощи IP-over-IP и GRE туннелей. В данном случае сети клиентов работают в едином адресном пространстве.

В эмуляционной среде GNS3 была смоделирована часть сети провайдера, в которой был реализован протокол MPLS, настроенный между несколькими маршрутизаторами.

В смоделированных сетях были применены политики и фильтры по ограничению скорости передачи данных и блокировке нескольких протоколов, а также по IP-адресам.

Выявлены преимущества и недостатки данных методов, выбраны наиболее подходящие области применения. В результате было выявлено, что для городской сети наиболее подходящим методом является метод организации VPN на базе протокола MPLS. По сравнению с методом «проброса» VLAN`ов и с методами GRE и IP-over-IP туннелирования, именно данная технология позволяет настраивать приоритет проходящего через маршрутизатор трафика. Во многих городских сетях на данный момент преобладает технология «проброса» VLAN`ов, так как её могут настраивать инженеры второго уровня доступа, а для настройки VPN на базе протокола MPLS L2 обычно осуществляется инженерами удалённо из головного терминала по стране. Поэтому при всём преимуществе технологии из-за ограничения политик безопасности доступа персонала к возможностям оборудования применение VPN на базе протокола MPLS L2 осуществляется только в том случае, если решить задачу «пробросом» VLAN`ов не удаётся, а

это обычно, случается, когда сайты клиента находятся территориально в разных автономных сетях провайдера или разных регионах страны.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Дуглас Ричард Хэнкс. Juniper MX series/ Д. Р. Хэнкс, Х. Рейнольдс. – O'Reilly Media, 2012.

2 Джонатан Луней. Automating JunOS administration. Д. Луней, Стейси Смит. – O'Reilly Media ISBN: 978- 1- 4493- 1663-1, 2016.

3 Питер Соуфвик. Juniper networks warrior. П. Соуфвик. – O'Reilly Media ISBN: 978- 1- 4493- 1663-1, 2012.

4 Сэм Хелеби. Принципы маршрутизации в интернет. 2-ое изд/ Д. Мк-Ферсон, С. Хелеби. – Москва Вильяме, Cisco Press. 2000.

5 Левченко А. С. Оптические цифровые телекоммуникационные системы: основные работы распределенных сетей на базе протоколов BGP и MPLS. Лабораторный практикум А. С. Левченко, Е. А. Лаврентьева, Ю. А. Тихонова, В. В. Слюсаревский, Н. А. Яковенко – Краснодар: КубГУ, 2013.

6 Гольдштейн А.Б. Механизм эффективного туннелирования в сети MPLS// Вестник связи. 2004. №2. – С. 48-54.

7 Juniper Networks. ACX Series Universal Access Router Configuration Guide, Inc. 1133 Innovation Way Sunnyvale, California 9 4089 USA 408- 745-2000, Copyright 2015.

ПРИЛОЖЕНИЕ А

Исходные настройки будут выполнены согласно таблице 1.

Таблица 1 – Исходные настройки

Устройство	Интерфейс	IP-адрес	Маска	VLAN
vMX1	ae0	-	-	10,20
	ae1	-	-	10,20
	ge-0/0/1	-	-	10,20
vMX2	ae0	-	-	10,20
	ae1	-	-	10,20
	ge-0/0/1	-	-	10,20
vMX3	ae0	-	-	10,20
	ae1	-	-	10,20
	ge-0/0/1	-	-	10,20
Switch1	e0/1	-	-	10
	e0/2	-	-	20
	e0/3	-	-	trunk
Switch2	e0/1	-	-	10
	e0/2	-	-	20
	e0/0	-	-	trunk
Switch3	e0/1	-	-	10
	e0/2	-	-	20
	e0/0	-	-	trunk
PC1	eth0	192.168.1.1	255.255.255.0	10
PC2	eth0	192.168.2.2	255.255.255.0	20
PC3	eth0	192.168.1.3	255.255.255.0	10
PC4	eth0	192.168.2.4	255.255.255.0	20
PC5	eth0	192.168.1.5	255.255.255.0	10
PC6	eth0	192.168.2.6	255.255.255.0	20

Для того чтобы настроить VLAN, сначала нужно будет подготовить каждый маршрутизатор, настроив агрегированные интерфейсы, а также стандарты их работы.

Поэтапно рассмотрим настройки на примере маршрутизатора vMX1.

Войдём в режим конфигурации:

```
root@vMX1> configure
```

Затем переходим в раздел «chassis»:

```
[edit]
root@vMX1# edit chassis
```

После этого нужно задать количество агрегированных интерфейсов:

```
[edit                                                chassis]
root@vMX1# set aggregated-devices ethernet device-
count 3
```

Далее физические интерфейсы добавляются в созданные ранее агрегированные каналы. Данные агрегированные каналы работают по стандарту IEEE 802.3ad:

```
[edit                                                chassis]
root@vMX1# exit
[edit]
root@vMX1# edit interfaces
[edit                                             interfaces]
root@vMX1# set ge-0/0/2 gigether-options 802.3ad ae0
[edit                                             interfaces]
root@vMX1# set ge-0/0/3 gigether-options 802.3ad ae0
[edit                                             interfaces]
root@vMX1# set ge-0/0/4 gigether-options 802.3ad ae1
[edit                                             interfaces]
root@vMX1# set ge-0/0/5 gigether-options 802.3ad ae1
```

На интерфейсе ge-0/0/1, который направлен в сторону клиентов, настраиваем VLAN10 и VLAN20:

```
[edit interfaces]
root@vMX1# set interface ge-0/0/1 unit 10
encapsulation vlan-bridge vlan-id 10
[edit interfaces]
```

```
root@vMX1# set interface ge-0/0/1 unit 20 encapsulation vlan-bridge vlan-id 20
```

Далее включаем VLAN-мосты и irb на интерфейсах ae0 и ae1:

```
[edit interfaces]
root@vMX1# set interfaces ae0 unit 10 encapsulation vlan-bridge vlan-id 10
root@vMX1# set interfaces ae0 unit 10 family inet address 192.168.1.1/24
[edit interfaces]
root@vMX1# set interfaces ae0 unit 20 encapsulation vlan-bridge vlan-id 20
root@vMX1# set interfaces ae0 unit 20 family inet address 192.168.2.2/24
[edit interfaces]
root@vMX1# set interfaces ae1 unit 10 encapsulation vlan-bridge vlan-id 10
root@vMX1# set interfaces ae1 unit 10 family inet address 192.168.1.1/24
[edit interfaces]
root@vMX1# set interfaces ae1 unit 20 encapsulation vlan-bridge vlan-id 20
root@vMX1# set interfaces ae1 unit 20 family inet address 192.168.2.2/24
[edit interfaces]
```

В той же самой последовательности настраиваем маршрутизаторы vMX2 и vMX3.

Маршрутизатор vMX2.

Войдём в режим конфигурации:

```
root@vMX2> configure
```

Затем переходим в раздел «chassis»:

```
[edit]
root@vMX2# edit chassis
```

После этого нужно задать количество агрегированных интерфейсов:

```
[edit chassis]
root@ vMX2# set aggregated-devices ethernet device-count 3
```

Далее физические интерфейсы добавляются в созданные ранее агрегированные каналы. Данные агрегированные каналы работают по стандарту IEEE 802.3ad:

```
[edit chassis]
root@vMX2# exit
[edit]
root@vMX2# edit interfaces
[edit interfaces]
root@vMX2# set ge-0/0/2 gigether-options 802.3ad ae0
[edit interfaces]
root@vMX2# set ge-0/0/3 gigether-options 802.3ad ae0
[edit interfaces]
root@vMX2# set ge-0/0/4 gigether-options 802.3ad ae1
[edit interfaces]
root@vMX2# set ge-0/0/5 gigether-options 802.3ad ae1
```

На интерфейсе ge-0/0/1, который направлен в сторону клиентов, настраиваем VLAN10 и VLAN20:

```
[edit interfaces]
root@vMX2# set interface ge-0/0/1 unit 10 encapsulation vlan-bridge vlan-id 10
```

```
[edit interfaces]
root@vMX2# set interface ge-0/0/1 unit 20 encapsulation vlan-bridge vlan-id 20
```

Далее включаем VLAN-мосты и irb на интерфейсах ae0 и ae1:

```
[edit interfaces]
root@vMX2# set interfaces ae1 unit 10 encapsulation vlan-bridge vlan-id 10
root@vMX2# set interfaces ae1 unit 10 family inet address 192.168.1.3/24
[edit interfaces]
root@vMX2# set interfaces ae1 unit 20 encapsulation vlan-bridge vlan-id 20
root@vMX2# set interfaces ae1 unit 20 family inet address 192.168.2.4/24
[edit interfaces]
root@vMX2# set interfaces ae2 unit 10 encapsulation vlan-bridge vlan-id 10
root@vMX2# set interfaces ae2 unit 10 family inet address 192.168.1.3/24
[edit interfaces]
root@vMX2# set interfaces ae2 unit 20 encapsulation vlan-bridge vlan-id 20
root@vMX2# set interfaces ae2 unit 20 family inet address 192.168.2.4/24
```

Маршрутизатор vMX3.

Войдём в режим конфигурации:

```
root@vMX3> configure
```

Затем переходим в раздел «chassis»:

```
[edit]
root@vMX3# edit chassis
```

После этого нужно задать количество агрегированных интерфейсов:

```
[edit chassis]
root@vMX3# set aggregated-devices ethernet device-count 3
```

Далее физические интерфейсы добавляются в созданные ранее агрегированные каналы. Данные агрегированные каналы работают по стандарту IEEE 802.3ad:

```
[edit chassis]
root@vMX3# exit
[edit]
root@vMX3# edit interfaces
[edit interfaces]
root@vMX3# set ge-0/0/2 gigether-options 802.3ad ae0
[edit interfaces]
root@vMX3# set ge-0/0/3 gigether-options 802.3ad ae0
[edit interfaces]
root@vMX3# set ge-0/0/4 gigether-options 802.3ad ae2
[edit interfaces]
root@vMX3# set ge-0/0/5 gigether-options 802.3ad ae2
```

На интерфейсе ge-0/0/1, который направлен в сторону клиентов, настраиваем VLAN10 и VLAN20:

```
[edit interfaces]
root@vMX3# set interface ge-0/0/1 unit 10 encapsulation vlan-bridge vlan-id 10
[edit interfaces]
```

```
root@vMX3# set interface ge-0/0/1 unit 20
encapsulation vlan-bridge vlan-id 20
```

Включаем VLAN-мосты и добавляем irb на интерфейсах ae0 и ae1 :

```
[edit interfaces]
root@vMX3# set interfaces ae1 unit 10 encapsulation
vlan-bridge vlan-id 10
root@vMX3# set interfaces ae1 unit 10 family inet
address 192.168.1.5/24
[edit interfaces]
root@vMX3# set interfaces ae1 unit 20 encapsulation
vlan-bridge vlan-id 20
root@vMX3# set interfaces ae1 unit 20 family inet
address 192.168.2.6/24
[edit interfaces]
root@vMX3# set interfaces ae2 unit 10 encapsulation
vlan-bridge vlan-id 10
root@vMX3# set interfaces ae2 unit 10 family inet
address 192.168.1.5/24
[edit interfaces]
root@vMX3# set interfaces ae2 unit 20 encapsulation
vlan-bridge vlan-id 20
root@vMX3# set interfaces ae2 unit 20 family inet
address 192.168.2.6/24
```

После того как были созданы VLAN-мосты, необходимо присвоить интерфейсы каждому VLAN’у.

Маршрутизатор vMX1:

```
[edit]
root@vMX1#set bridge-domain vlan10 domain-type
bridge interface interface ge-0/0/1.10
[edit]
```

```
    root@vMX1#set  bridge-domain  vlan10  domain-type
bridge interface interface ae0.10
    [edit]
    root@vMX1#set  bridge-domain  vlan10  domain-type
bridge interface interface ae1.10
    [edit]
    root@vMX1#set  bridge-domain  vlan20  domain-type
bridge interface interface ge-0/0/1.20
    [edit]
    root@vMX1#set  bridge-domain  vlan20  domain-type
bridge interface interface ae0.20
    [edit]
    root@vMX1#set  bridge-domain  vlan20  domain-type
bridge interface interface ae1.20
```

Маршрутизатор vMX2:

```
    [edit]
    root@vMX1#set  bridge-domain  vlan10  domain-type
bridge interface interface ge-0/0/1.10
    [edit]
    root@vMX1#set  bridge-domain  vlan10  domain-type
bridge interface interface ae1.10
    [edit]
    root@vMX1#set  bridge-domain  vlan10  domain-type
bridge interface interface ae2.10
    [edit]
    root@vMX1#set  bridge-domain  vlan20  domain-type
bridge interface interface ge-0/0/1.20
    [edit]
    root@vMX1#set  bridge-domain  vlan20  domain-type
bridge interface interface ae1.20
```

```
[edit]
root@vMX1#set bridge-domain vlan20 domain-type
bridge interface interface ae2.20
```

Маршрутизатор vMX3:

```
[edit]
root@vMX1#set bridge-domain vlan10 domain-type
bridge interface interface ge-0/0/1.10
[edit]
root@vMX1#set bridge-domain vlan10 domain-type
bridge interface interface ae0.10
[edit]
root@vMX1#set bridge-domain vlan10 domain-type
bridge interface interface ae2.10
[edit]
root@vMX1#set bridge-domain vlan20 domain-type
bridge interface interface ge-0/0/1.20
[edit]
root@vMX1#set bridge-domain vlan20 domain-type
bridge interface interface ae0.20
[edit]
root@vMX1#set bridge-domain vlan20 domain-type
bridge interface interface ae2.20
```

Для того чтобы устранить топологические петли, которые приводят к повторным передачам «пакетов» и снижают пропускную способность, нужно настроить Spanning Tree Protocol (STP) протокол.

Так как в сети есть несколько одинаково сконфигурированных VLAN-групп, идеально подойдёт Multiple Spanning Tree Protocol (MSTP) протокол, работающий с несколькими сетями одновременно.

Настройка MSTP на маршрутизаторе vMX1 :

```
[edit protocols mstp]
```

```
root@vMX1#set configuration-name mstp-for-R1-2-3
root@vMX1#set revision-level 3
root@vMX1#set bridge-priority 0
root@vMX1#set interface ae0
root@vMX1#set interface ae1
root@vMX1#set mstp 1 vlan10
root@vMX1#set mstp 2 vlan20
```

Настройка MSTP на маршрутизаторе vMX2 :

```
[edit protocols mstp]
root@vMX2#set configuration-name mstp-for-R1-2-3
root@vMX2#set revision-level 3
root@vMX2#set bridge-priority 0
root@vMX2#set interface ae0
root@vMX2#set interface ae1
root@vMX2#set mstp 1 vlan10
root@vMX2#set mstp 2 vlan20 bridge-priority 4096
```

Настройка MSTP на маршрутизаторе vMX3:

```
[edit protocols mstp]
root@vMX3#set configuration-name mstp-for-R1-2-3
root@vMX3#set revision-level 3
root@vMX3#set bridge-priority 0
root@vMX3#set interface ae1
root@vMX3#set interface ae2
root@vMX3#set mstp 1 vlan10 bridge-priority 4096
root@vMX3#set mstp 2 vlan20
```

Добавим фильтр на ограничение скорости передачи данных до 1 гигабита в секунду для маршрутизатора vMX3:

```
root@vMX3# show firewall policer test
root@vMX3# if-exceeding bandwidth-limit 1024000000
burst-size-limit 10k then discard
```

```
[edit]
```

Добавим фильтр для маршрутизатора vMX1 на ограничение получения данных по протоколу UDP:

```
root@vMX1# show firewall filter test
root@vMX1# interface-specific from protocol udp then discard
[edit]
```

ПРИЛОЖЕНИЕ Б

Исходные настройки будут выполнены согласно таблице 2.

Таблица 2 – Исходные настройки

Устройство	Интерфейс	IP-адрес	Маска	VLAN
vMX1	ge-0/0/1	10.24.1.1	255.255.255.0	-
	ge-0/0/2	10.24.2.1	255.255.255.0	-
	ip-0/0/0	1.1.1.1	255.255.255.252	-
vMX2	ge-0/0/1	10.24.3.1	255.255.255.0	-
	ge-0/0/2	10.24.4.1	255.255.255.0	-
	ip-0/0/0	1.1.1.1	255.255.255.252	-
PC1	eth0	10.24.11.1	255.255.255.0	-
PC2	eth0	10.24.12.1	255.255.255.0	-
PC3	eth0	10.24.13.1	255.255.255.0	-
PC4	eth0	10.24.14.1	255.255.255.0	-

Настройка интерфейсов маршрутизатора vMX1:

```
root@vMX1# set interface ge-0/0/1 unit 0 family inet address 10.24.1.1/24
root@vMX1# set interface ip-0/0/0 unit 0 family inet address 1.1.1.1/30
```

После этого необходимо указать источник передаваемых данных и их получателя:

```
root@vMX1# set interface ip-0/0/0 unit 0 tunnel source 10.24.1.1 destination 10.24.2.1
root@vMX1# set interface ge-0/0/2 unit 0 family inet address 10.24.1.1/24
```

Настройка интерфейсов маршрутизатора vMX2:

```
root@vMX2# set interface ge-0/0/1 unit 0 family inet
```

```
address 10.24.10.1/24
    root@vMX2# set interface ip-0/0/0 unit 0 family inet
address 1.1.1.2/30
    root@vMX2# set interface ip-0/0/0 unit 0 tunnel
source 10.24.2.1 destination 10.24.1.1
    root@vMX2# set interface ge-0/0/2 unit 0 family inet
address 10.24.2.1/24
```

Далее настраиваем статический маршрут от сетей клиента к ip-интерфейсу для каждого маршрутизатора:

```
    root@vMX1# set routing-options static route
10.24.3.0/24 next-hop ip.0
    root@vMX2# set routing-options routing-options
static route 10.24.4.0/24 next-hop ip.0
```

ПРИЛОЖЕНИЕ В

Исходные настройки будут выполнены согласно таблице 3.

Таблица 3 – Исходные настройки

Устройство	Интерфейс	IP-адрес	Маска	VLAN
vMX1	ge-0/0/1	10.24.1.1	255.255.255.0	-
	ge-0/0/2	10.24.2.1	255.255.255.0	-
	gr-0/0/0	1.1.1.1	255.255.255.252	-
vMX2	ge-0/0/1	10.24.3.1	255.255.255.0	-
	ge-0/0/2	10.24.4.1	255.255.255.0	-
	gr-0/0/0	1.1.1.1	255.255.255.252	-
PC1	eth0	10.24.11.1	255.255.255.0	-
PC2	eth0	10.24.12.1	255.255.255.0	-
PC3	eth0	10.24.13.1	255.255.255.0	-
PC4	eth0	10.24.14.1	255.255.255.0	-

Настройка интерфейсов маршрутизатора vMX1 :

```
root@vMX1# set interface ge-0/0/1 unit 0 family inet address 10.24.1.1/24
root@vMX1# set interface gr-0/0/0 unit 0 family inet address 1.1.1.1/30
```

После этого необходимо указать источник передаваемых данных и их получателя:

```
root@vMX1# set interface gr-0/0/0 unit 0 tunnel source 10.24.1.1 destination 10.24.2.1
root@vMX1# set interface ge-0/0/2 unit 0 family inet address 10.24.1.1/24
```

Настройка интерфейсов маршрутизатора vMX2 :

```
root@vMX2# set interface ge-0/0/1 unit 0 family inet
```

address 10.24.10.1/24

root@vMX2# set interface gr-0/0/0 unit 0 family inet address 1.1.1.2/30

root@vMX2# set interface gr-0/0/0 unit 0 tunnel source 10.24.2.1 destination 10.24.1.1

root@vMX2# set interface ge-0/0/2 unit 0 family inet address 10.24.2.1/24

Далее настраиваем статический маршрут от сетей клиента к gr-интерфейсу :

root@vMX1# set routing-options static route 10.24.3.0/24 next-hop gr.0

root@vMX2# set routing-options routing-options static route 10.24.4.0/24 next-hop gr.0

ПРИЛОЖЕНИЕ Г

Исходные настройки будут выполнены согласно таблице 4.

Таблица 4 – Исходные настройки

Устройство	Интерфейс	IP-адрес	Маска
vMX1	ge-0/0/0	10.24.1.1	255.255.255.0
	ge-0/0/1	10.24.2.1	255.255.255.0
	lo0	1.1.1.1	255.255.255.255
vMX2	ge-0/0/0	30.24.1.1	255.255.255.0
	ge-0/0/1	30.24.2.1	255.255.255.0
	lo0	1.1.1.1	255.255.255.252
vMX3	ge-0/0/0	20.24.1.1	255.255.255.0
	ge-0/0/1	20.24.2.1	255.255.255.0
	lo0	1.1.1.1	255.255.255.255

Настройка маршрутизатора vMX1 происходит следующим образом:

Создание UNIT и добавление в него VLAN’a

```
root@vMX1# set interfaces ge-0/0/0 vlan-tagging
root@vMX1# set interfaces ge-0/0/0 encapsulation vlan-ccc
root@vMX1# set interfaces ge-0/0/0 unit 222 encapsulation vlan-ccc
root@vMX1# set interfaces ge-0/0/0 unit 222 vlan-id 222
```

Для того чтобы интерфейс ge-0/0/1 принимал и обрабатывал IP-пакеты, укажем на данном интерфейсе «family inet» вместе с IP-адресом:

```
root@vMX1# set interfaces ge-0/0/1 unit 0 family inet address 10.0.0.1/30
```

Для «восприятия» и обработки этим же интерфейсом MPLS-пакетов, настроим на нём «family mpls»:

```
root@vMX1# set interfaces ge-0/0/1 unit 0 family mpls
root@vMX1# set interfaces lo0 unit 0 family inet address 1.1.1.1/32
```

Подключим протоколы MPLS и LDP ко всем интерфейсам маршрутизатора:

```
root@vMX1# set protocols mpls interface all
root@vMX1# set protocols ldp interface all
```

Включим OSPF для маршрутизации данных, исходящих от vMX3:

```
root@vMX1# set protocols ospf traffic-engineering
root@vMX1# set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
root@vMX1# set protocols ospf area 0.0.0.0 interface lo0.0 passive
```

Сообщим маршрутизатору о его соседе, маршрутизаторе vMX2, подключенному через интерфейс ge-0/0/0

```
root@vMX1# set protocols l2circuit neighbor 30.24.1.1 interface ge-0/0/0.222 virtual-circuit-id
root@vMX1# set protocols l2circuit neighbor 30.24.1.1 interface ge-0/0/0.222 no-control-word
```

Команды настройки маршрутизатора vMX2:

Маршрутизатор vMX2 настраивается в качестве промежуточного коммутатора:

```
root@vMX2# set interfaces ge-0/0/0 unit 0 family inet address 10.0.0.0/30
root@vMX2# set interfaces ge-0/0/0 unit 0 family mpls
root@vMX2# set interfaces ge-0/0/1 unit 0 family inet address 20.0.0.0/30
```

```
root@vMX2# set interfaces ge-0/0/1 unit 0 family mpls
root@vMX2# set interfaces lo0 unit 0 family inet
address 1.1.1.1/32
```

Далее включаем протоколы LDP и MPLS на всех интерфейсах маршрутизатора:

```
root@vMX2# set protocols ldp interface all
root@vMX2# set protocols mpls interface all
```

Затем необходимо включить OSPF для маршрутизации данных, приходящих с обеих сторон сети:

```
root@vMX2# set protocols ospf traffic-engineering
root@vMX2# set protocols ospf area 0.0.0.0 interface
ge-0/0/0.0
root@vMX2# set protocols ospf area 0.0.0.0 interface
ge-0/0/1.0
root@vMX2# set protocols ospf area 0.0.0.0 interface
lo0.0 passive
```

Команды настройки маршрутизатора vMX3.

```
root@vMX3# set interfaces ge-0/0/0 vlan-tagging
root@vMX3# set interfaces ge-0/0/0 encapsulation
vlan-ccc
root@vMX3# set interfaces ge-0/0/0 unit 222
encapsulation vlan-ccc
root@vMX3# set interfaces ge-0/0/0 unit 222 vlan-id
222
root@vMX3# set interfaces ge-0/0/1 unit 0 family inet
address 10.0.0.1/30
root@vMX3# set interfaces ge-0/0/1 unit 0 family mpls
root@vMX3# set interfaces lo0 unit 0 family inet
address 1.1.1.1/32 primary
```

```
root@vMX3# set interfaces lo0 unit 0 family inet
address 127.0.0.1/32
root@vMX3# set protocols mpls interface all
root@vMX3# set protocols ldp interface all
```

Включение OSPF для маршрутизации данных, исходящих от vMX1:

```
root@vMX3# set protocols ospf traffic-engineering
root@vMX3# set protocols ospf area 0.0.0.0 interface
ge-0/0/1.0
root@vMX3# set protocols ospf area 0.0.0.0 interface
lo0.0 passive
```

Сообщим маршрутизатору о его соседе, маршрутизаторе vMX2, подключенному через интерфейс ge-0/0/1:

```
root@vMX3# set protocols l2circuit neighbor
30.24.2.1 interface ge-0/0/1.222 virtual-circuit-id
root@vMX3# set protocols l2circuit neighbor
30.24.2.1 interface ge-0/0/1.222 no-control-word
```