Тема:
Основные сервисы на Linux .
Роутер на Linux.

План занятия:

1.

План занятия:

Роутер на Linux - это компьютер с операционной системой Linux, который выполняет функции маршрутизации пакетов в сети. Обычно роутер на Linux используется для следующих целей:

1.Маршрутизация пакетов:

2.Роутер на Linux может маршрутизировать пакеты между различными сетями, определяя наилучший путь для доставки данных на основе информации в маршрутной таблице.

3.Переключение сетевого трафика:

4.Он может также выполнять функции коммутатора, пересылая кадры между устройствами в локальной сети.

5.Фильтрация трафика:

6.Роутер на Linux может выполнять фильтрацию пакетов на основе определенных правил, что позволяет контролировать доступ к сети и обеспечивать безопасность.

7.Выполнение сетевых служб:

8.В дополнение к маршрутизации и фильтрации, роутер на Linux может также выполнять другие сетевые службы, такие как DHCP, DNS, NAT, VPN и т. д.

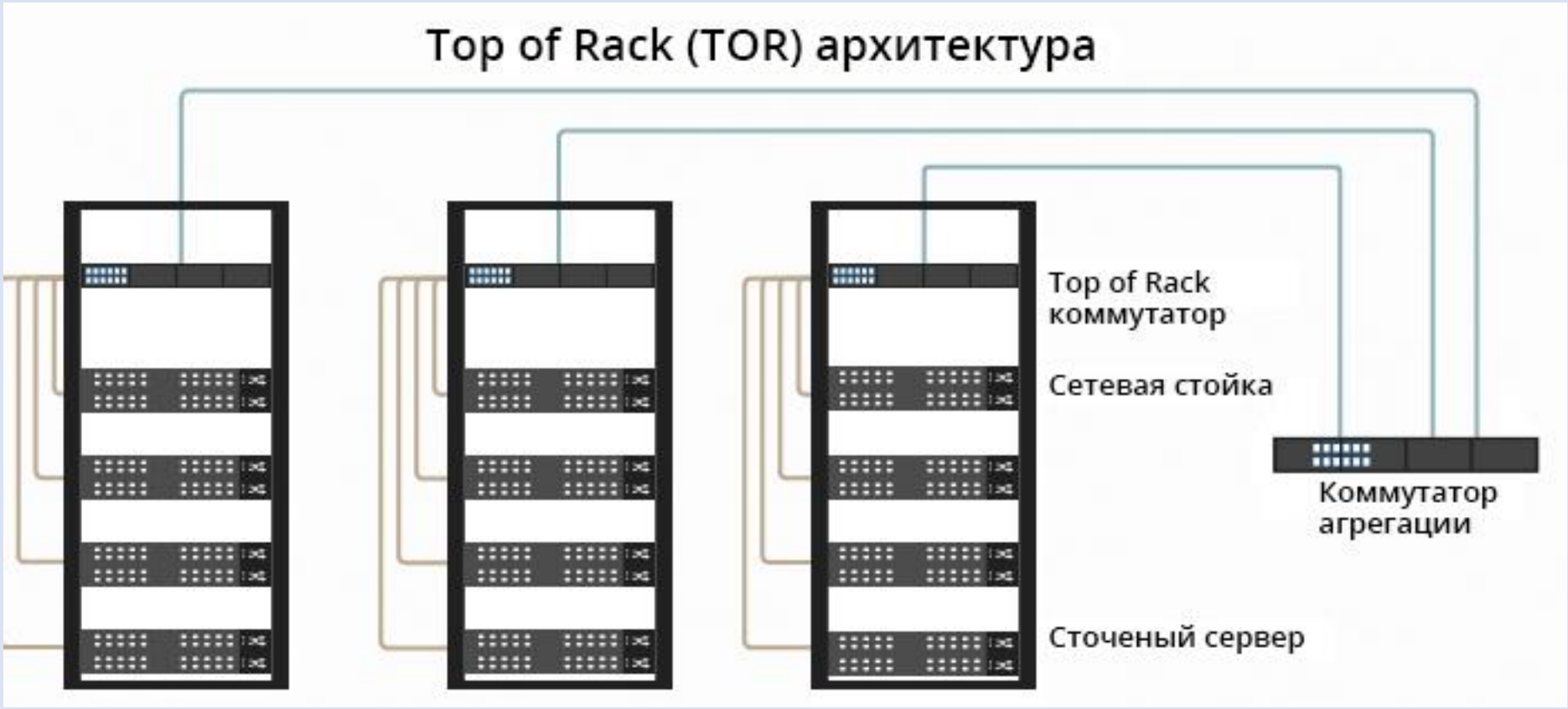
Роутер на Linux является гибким и настраиваемым инструментом, который может быть настроен для выполнения различных функций сетевого управления и обеспечения связности. Он может быть полезен как для домашнего использования, так и для организаций, которые нуждаются в настраиваемом и гибком решении для управления своей сетью.

Top of Rack

Top of Rack (ToR) — это концепция и архитектурное решение, применяемое в компьютерных сетях и центрах обработки данных (ЦОД). ToR относится к размещению коммутаторов (сетевых переключателей) уровня доступа внутри стойки (rack) серверов.

В традиционной архитектуре ЦОД коммутаторы размещались в отдельных помещениях или в отдельных стойках, а сетевые кабели для подключения серверов пролегали по всему помещению. Однако с ростом количества серверов и требований к пропускной способности сети, такая архитектура стала неэффективной и приводила к сложностям в управлении и масштабировании сети.

ToR-архитектура решает эту проблему, перемещая коммутаторы ближе к серверам, внутрь стоек с серверами. Каждая стойка в ЦОД оборудуется своим собственным коммутатором, который обеспечивает подключение всех серверов внутри стойки. Это позволяет сократить длину сетевых кабелей и упростить управление сетью.


Top of Rack (TOR) архитектура
Top of Rack
коммутатор
Сетевая стойка
Коммутатор
агрегации
Сточеный сервер

Top of Rack

Преимущества архитектуры ToR включают:

1.Уменьшение влияния задержек в сети: Коммутаторы ToR находятся очень близко к серверам, что снижает задержки и улучшает производительность сети.

2.Упрощенное управление и масштабирование: Размещение коммутаторов ToR внутри стоек облегчает управление сетью и добавление новых серверов. Масштабирование сети происходит за счет добавления дополнительных стоек с коммутаторами.

3.Уменьшение длины сетевых кабелей: Размещение коммутаторов внутри стоек позволяет сократить длину сетевых кабелей, что уменьшает затраты на кабельную инфраструктуру и упрощает управление проводами.

4.Гибкость и высокая пропускная способность: Каждая стойка имеет свой собственный коммутатор, что позволяет достичь высокой пропускной способности и обеспечивает гибкость в настройке сети.

Top of Rack

Недостатки архитектуры ToR включают:

Затраты на оборудование: При использовании ToR каждая стойка серверов оборудуется своим собственным коммутатором.

Увеличение потребления энергии и теплопроизводства: Увеличенное количество коммутаторов в ToR архитектуре может привести к увеличению потребления электроэнергии и теплопроизводства в ЦОД.

Ограничения пропускной способности: В некоторых сценариях ToR архитектура может иметь ограничения пропускной способности, особенно при передаче трафика между стойками серверов.

Ограниченные возможности масштабирования: ToR архитектура обычно предназначена для локального масштабирования внутри ЦОД и может столкнуться с ограничениями при масштабировании на уровне целых дата-центров или распределенных сетей.

Управление: Увеличение количества коммутаторов в ToR архитектуре может сделать управление и конфигурирование сети более сложным и трудоемким процессом, особенно при использовании традиционных методов управления сетью.

Top of Rack

Если один из серверов в стойке начнет флудить широковещательные пакеты, это может иметь негативное воздействие на всю сеть, включая другие серверы и устройства в ЦОДе. Вот несколько потенциальных проблем:

1. **Потеря пропускной способности**: Флуд широковещательных пакетов может заблокировать значительную часть пропускной способности сети, что приведет к уменьшению производительности и задержек для других устройств и серверов в сети.

2. **Перегрузка коммутаторов**: Коммутаторы в стойке и других частях сети могут быть перегружены обработкой большого объема широковещательного трафика, что может привести к их отказу или снижению производительности.

3. **Отказ в обслуживании**: Если широковещательный трафик становится слишком интенсивным, некоторые устройства могут перестать отвечать на запросы или потерять связь с сетью, что приведет к недоступности сервисов и данных.

4. **Повышенное энергопотребление**: Повышенная активность сети, вызванная флудом широковещательных пакетов, может привести к увеличению энергопотребления в ЦОДе из-за увеличенной нагрузки на сетевое оборудование.

End of row network

"End of row network" (EoR) - это архитектурный подход в сетевом дизайне, который используется в центрах обработки данных (ЦОД) для организации сети. В этой архитектуре сетевое оборудование, такое как коммутаторы, устанавливается на концах (концах) каждого ряда стоек серверов в ЦОДе.

Основные характеристики архитектуры "End of Row":

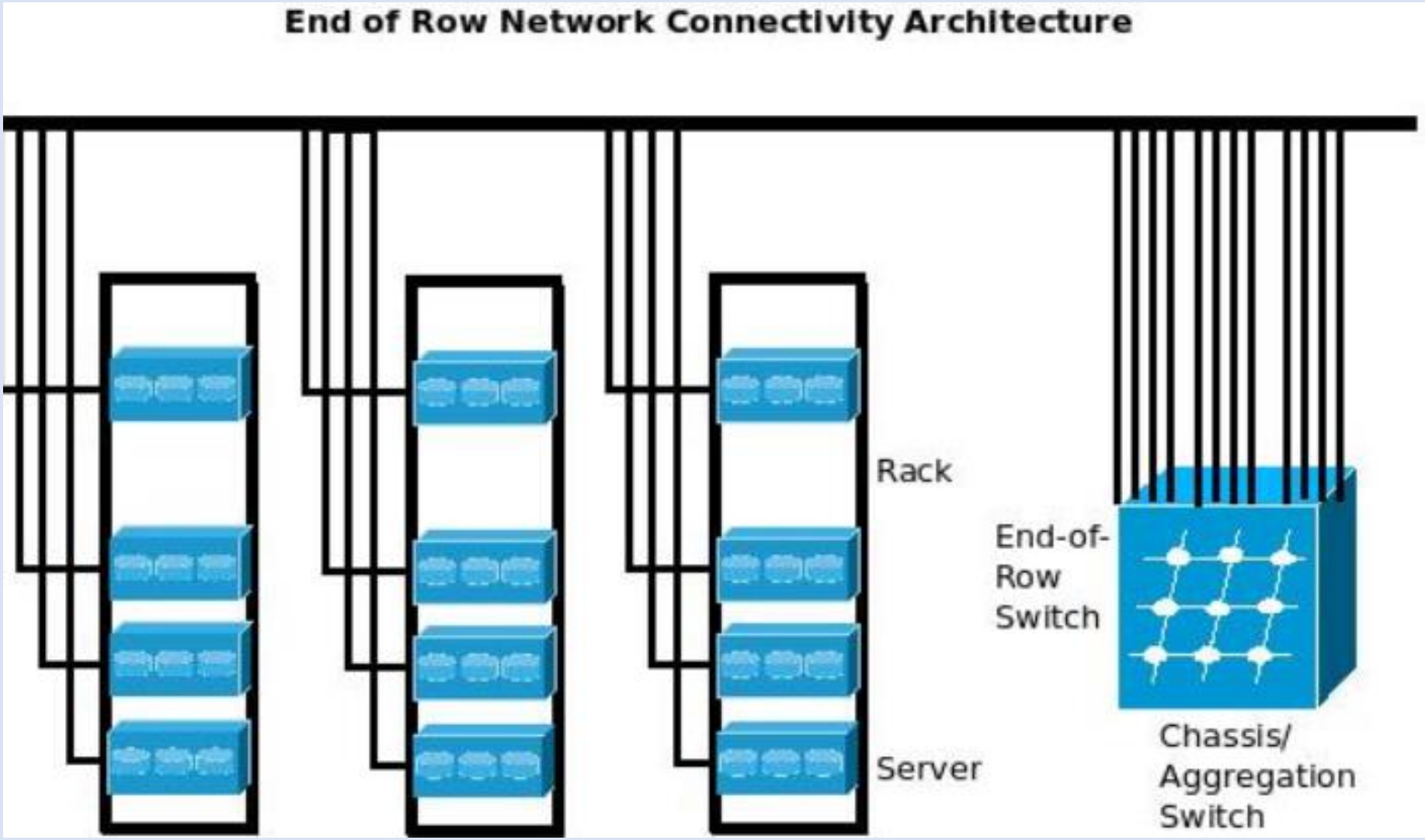
1.Простота управления и масштабирования: Каждый ряд стоек серверов оборудуется своим собственным коммутатором, что упрощает управление и масштабирование сети. Это позволяет уменьшить количество необходимых уровней сетевой абстракции и упростить конфигурацию сети.

2.Улучшенная пропускная способность и производительность: Передача трафика между серверами в одном ряду происходит без необходимости прохода через центральный коммутатор, что позволяет уменьшить задержки и повысить пропускную способность.

3.Гибкость и адаптивность: Архитектура End of Row позволяет быстро адаптироваться к изменениям в сетевой инфраструктуре, добавляя или удаляя ряды стоек серверов и связанные с ними коммутаторы по мере необходимости.

4.Легкость обслуживания и обнаружения проблем: Каждый ряд стоек серверов имеет свой собственный коммутатор, что упрощает обслуживание и обнаружение проблем с сетью в определенной области ЦОДа.

5.Повышенная надежность: Архитектура End of Row обеспечивает дополнительный уровень отказоустойчивости, так как каждый ряд стоек серверов может обслуживаться своим коммутатором, минимизируя влияние отказов на другие части сети.


End of Row Network Connectivity Architecture
Rack
Server
End-of-
Row
Switch
Chassis/
Aggregation
Switch

End of row network

Недостатки:

1.Сложность масштабирования: Хотя архитектура EoR обеспечивает гибкость в масштабировании, добавление новых рядов стоек серверов может потребовать дополнительного оборудования и изменений в сетевой конфигурации. Это может усложнить процесс масштабирования и требовать дополнительных инвестиций.

2.Потенциальные точки отказа: Каждый коммутатор, установленный на конце ряда стоек серверов, представляет собой потенциальную точку отказа. Если один из коммутаторов выходит из строя, это может привести к недоступности всего ряда стоек серверов, что повлияет на доступность сервисов.

3.Ограниченные варианты резервирования: Архитектура EoR ограничивает варианты резервирования и отказоустойчивости. Поскольку каждый ряд стоек серверов обслуживается своим собственным коммутатором, создание полностью резервированных или отказоустойчивых конфигураций может быть сложным и дорогостоящим процессом.

4.Увеличение нагрузки на коммутаторы: Коммутаторы, установленные на концах рядов стоек серверов, могут быть подвержены увеличенной нагрузке из-за проходящего через них трафика между стойками. Это может привести к перегрузке коммутаторов и снижению производительности сети.

5.Ограничения в управлении трафиком: Архитектура EoR ограничивает возможности управления трафиком и маршрутизации данных. В некоторых случаях это может ограничить гибкость и эффективность сетевого управления в сравнении с другими архитектурными решениями.

6. Один бродкаст домен.

End of row network

Недостатки:

1.Сложность масштабирования: Хотя архитектура EoR обеспечивает гибкость в масштабировании, добавление новых рядов стоек серверов может потребовать дополнительного оборудования и изменений в сетевой конфигурации. Это может усложнить процесс масштабирования и требовать дополнительных инвестиций.

2.Потенциальные точки отказа: Каждый коммутатор, установленный на конце ряда стоек серверов, представляет собой потенциальную точку отказа. Если один из коммутаторов выходит из строя, это может привести к недоступности всего ряда стоек серверов, что повлияет на доступность сервисов.

3.Ограниченные варианты резервирования: Архитектура EoR ограничивает варианты резервирования и отказоустойчивости. Поскольку каждый ряд стоек серверов обслуживается своим собственным коммутатором, создание полностью резервированных или отказоустойчивых конфигураций может быть сложным и дорогостоящим процессом.

4.Увеличение нагрузки на коммутаторы: Коммутаторы, установленные на концах рядов стоек серверов, могут быть подвержены увеличенной нагрузке из-за проходящего через них трафика между стойками. Это может привести к перегрузке коммутаторов и снижению производительности сети.

5.Ограничения в управлении трафиком: Архитектура EoR ограничивает возможности управления трафиком и маршрутизации данных. В некоторых случаях это может ограничить гибкость и эффективность сетевого управления в сравнении с другими архитектурными решениями.

6. Один бродкаст домен.

Host-based router

Архитектура "роутер на уровне хоста" (Host-based router) - это концепция сетевой архитектуры, при которой маршрутизация и пересылка пакетов выполняются непосредственно на хосте, а не на отдельном сетевом устройстве, таком как маршрутизатор.

В традиционной сетевой архитектуре маршрутизация выполняется на маршрутизаторах или коммутаторах, которые принимают решения о том, как направить пакеты между различными сетевыми сегментами. Однако с архитектурой "роутер на уровне хоста" функции маршрутизации вместо этого выполняются на самих хостах.

При использовании архитектуры "роутер на уровне хоста" хост может быть настроен для выполнения следующих функций:

1.Маршрутизация: Хост может принимать решения о том, как направить пакеты между различными сетевыми интерфейсами на основе информации о сетевых протоколах, маршрутах и таблицах маршрутизации.

2.Фильтрация трафика: Хост может фильтровать пакеты на основе различных критериев, таких как IP-адрес и порт назначения или источника. Это позволяет применять политики безопасности, контролировать доступ и управлять сетевым трафиком.

3.Трансляция сетевых адресов (NAT): Хост может выполнять функцию Network Address Translation (NAT), переписывая IP-адреса и портов в заголовках пакетов для обеспечения связи между сетевыми сегментами с различными IP-адресами.

Host-based router

Архитектура "роутер на уровне хоста" находит применение в различных сценариях, таких как виртуализация, контейнеризация и сети в облаке. В этих сценариях хосты могут выполнять функции маршрутизации и пересылки пакетов, обеспечивая гибкость, масштабируемость и изоляцию сетей.

Однако стоит отметить, что архитектура "роутер на уровне хоста" имеет свои ограничения. Она может быть ограничена производительностью хоста и не всегда является оптимальным решением для больших сетей с высокой пропускной способностью. Кроме того, эта архитектура требует правильной настройки и обслуживания на каждом хосте, что может быть сложно при большом количестве хостов в сети.

Host-based router

В архитектуре "роутер на уровне хоста" бродкаст домен ограничен на уровне самого хоста. Бродкаст домен - это область сети, в пределах которой широковещательные пакеты (broadcast) доставляются ко всем устройствам.

В случае архитектуры "роутер на уровне хоста", каждый хост, выполняющий функции роутера, имеет свой собственный бродкаст домен. Это означает, что широковещательные пакеты, отправленные с одного хоста, не будут автоматически доставлены ко всем остальным хостам в сети.

Таким образом, бродкаст домен в архитектуре "роутер на уровне хоста" ограничен до пределов каждого отдельного хоста. Это позволяет более гибко управлять широковещательным трафиком и предотвращать его распространение по всей сети.

Однако стоит отметить, что ограничение бродкаст домена на уровне хоста также означает, что широковещательные пакеты не могут быть доставлены ко всем устройствам в сети без дополнительных механизмов или настройки.

Free Range Routing

Free Range Routing (FRR) - это свободное программное обеспечение с открытым исходным кодом для маршрутизации с открытой архитектурой, предназначенное для Linux и Unix-подобных операционных систем. Оно представляет собой мощное средство для настройки и управления различными протоколами маршрутизации, такими как BGP (Border Gateway Protocol), OSPF (Open Shortest Path First), IS-IS (Intermediate System to Intermediate System), RIP (Routing Information Protocol) и другими.

Основные характеристики и возможности FRR:

1. **Поддержка различных протоколов маршрутизации**: FRR поддерживает большое количество протоколов маршрутизации, что делает его универсальным инструментом для настройки маршрутизации в различных сетевых средах.

2. **Открытая архитектура**: FRR предоставляет открытую архитектуру и API для интеграции с другими сетевыми приложениями и инструментами, что обеспечивает высокую гибкость и расширяемость.

3. **Гибкие возможности настройки**: FRR предоставляет широкие возможности настройки и управления протоколами маршрутизации, включая настройку различных параметров протоколов, фильтрацию маршрутов, настройку политик маршрутизации и многое другое.

4. **Высокая производительность и надежность**: FRR обладает высокой производительностью и надежностью благодаря оптимизированной реализации протоколов маршрутизации и механизмам обнаружения и восстановления отказов.

5. **Поддержка IPv4 и IPv6**: FRR полностью поддерживает как IPv4, так и IPv6, что позволяет использовать его для настройки маршрутизации в сетях с обоими протоколами.

Free Range Routing

1. Для настройки маршрутизации с помощью FRR на Linux, как правило, требуется разрешить транзитные пакеты в ядре. Это необходимо, чтобы ядро могло принимать и пересылать пакеты между интерфейсами.

Чтобы разрешить транзитные пакеты в ядре Linux, вам нужно установить определенные параметры с помощью команды sysctl

```
sudo sysctl net.ipv4.ip_forward=1
sudo sysctl net.ipv6.conf.all.forwarding=1
```

или изменить конфигурационный файл /etc/sysctl.conf:

```
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
```

применить изменения

```
sudo sysctl -p
```

Free Range Routing

2. Установка FRR:

```
sudo apt-get update
sudo apt-get install frr
```

Free Range Routing

3. Настройка конфигурационных файлов. Отредактируйте файлы конфигурации, такие как frr.conf и daemons, для определения параметров маршрутизации, активации протоколов маршрутизации (например, OSPF, BGP) и указания интерфейсов, на которых маршрутизатор будет работать.

Free Range Routing

4. Запустите службу FRR с помощью команды, подходящей для вашей системы.

```
sudo systemctl start frr
sudo systemctl enable frr
```

Free Range Routing

5. Используйте утилиту командной строки vtysh для взаимодействия с FRR и проверки текущего состояния маршрутизации.

```
sudo vtysh
```

Внутри vtysh вы можете выполнить различные команды, такие как show ip route или show ospf neighbors, чтобы проверить текущую маршрутизацию и соседей OSPF.

Домашнее задание:

1. Изучить дополнительные материалы.