NETWORK NAMESPACES

- Что такое Network Namespaces
- Зачем нужно?
- Как использовать?
- Пример использования

MIRANTIS

Зачем это нужно?

VRF теперь не только у CISCO!

- Несколько таблиц маршрутизации
- Несколько экземпляров iptables
- Разные настройки sysctl
- В том числе работа с дублированными адресами

MIRANTIS

Требования к системе

- Начну с неприятного — в **CentOS НЕ РАБОТАЕТ** без пересборки ядра и **iproute2**
- Зато работает в **Ubuntu 12.04**
- В Gentoo тоже :)

Простой способ проверки - посмотреть наличие директории **/proc/self/ns/**

MIRANTIS

Если не работает в моем дистрибутиве а очень надо?

- Хорошо подумать. А может не надо?
- Пересобрать **iproute2**.
- Пересобрать ядро. Возможно обновить ядро
- Для CentOS 6.4 есть репозиторий с iproute2 и ядром с OVS и NETNS

MIRANTIS

Так что все таки включать то?

- Activate the network namespaces **CONFIG_NET_NS=y**
- Add the virtual ethernet pair device **CONFIG_VETH=y**
- Add the mac-vlan device **CONFIG_MACVLAN=y**

MIRANTIS

Итак, попробуем!

- Создаем name space: **ip netns add R0**

MIRANTIS

Добавляем интерфейс

- Обратить внимание — тип интерфейса «**Virtual Ethernet**»

- **ip link add name ve0a type veth peer name ve0b**

MIRANTIS

Проверим:

ip link show

20: **ve0b:** <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN qlen 1000

link/ether **2e:0c:69:1a:43:d1** brd ff:ff:ff:ff:ff:ff

21: **ve0a:** <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN qlen 1000

link/ether **b2:21:f3:95:e3:f6** brd ff:ff:ff:ff:ff:ff

MIRANTIS

Перенесем один из интерфейсов в namespace

- **ip link set dev ve0b netns R0**

- Эта команда переносит интерфейс ve0b

  из initial network namespace в namespace R0

MIRANTIS

Проверим что получилось

- **ip link show**

  21: ve0a: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN qlen 1000

  link/ether b2:21:f3:95:e3:f6 brd ff:ff:ff:ff:ff:ff

- **Интерфейс пропал из initial namespace**
- **ip netns exec R0 ip link show**

  18: lo: <LOOPBACK> mtu 16436 qdisc noop state DOWN

  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

  20: ve0b: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN qlen 1000

  link/ether 2e:0c:69:1a:43:d1 brd ff:ff:ff:ff:ff:ff

- **Зато появился в namespace R0**

MIRANTIS

С линком разобрались — теперь IP адреса

- ip netns exec R0 ip addr add 172.19.0.1/24 dev ve0b
- ip addr add 172.19.0.2/24 dev ve0a
- ip link set up dev ve0a
- ip link set up dev ve1a
- ip netns exec R0 ip route add 0.0.0.0/0 via 172.19.0.2

MIRANTIS

Ну вот и все — должно работать

- **ip netns exec R0 mtr -n 8.8.8.8**

MIRANTIS

Не понятно?

- Не понятно, зачем это нужно в жизни?
- Да и вообще ерунда это все
- Куда применить, Quantum мне не нужен!

MIRANTIS

Зачем?

- Как будет вести себя наше приложение когда между сервером и клиентом будет **100** (сто!) хопов?
- Да, реальный компьютер у нас всего **1** (один)
- Памяти тоже немного.
- Варианты? (про LXC не вспоминаем пока)

MIRANTIS

Итак, магия

- Шаг первый - создаем 100 network namespaces
- Скриптом конечно же!

MIRANTIS

Что дальше? Маршрутизация!

- У нас есть 100 роутеров
- Да, они включены в цепочку, один за одним.
- Но на уровне IP они знают только о соседях
- Решение?

MIRANTIS

Как настроить маршрутизацию в 100 экземплярах?

Вообще-то вариантов несколько,

но я выбрал **BGP**

- Достаточно просто (в нашем простом случае)
- Наглядно
- Анонсы распространяются достаточно медленно — можно посмотреть состояние на разных стадиях

MIRANTIS

Маленькое отступление

Вообще-то **bgpd** нельзя запускать во многих экземплярах. Но мне очень хотелось. Потому:

- **CHROOT**
- Отдельная **FS** для каждого экземпляра процесса (mount -o bind)
- Отдельный Network NameSpace

MIRANTIS

Конфиг BGP

- **Номера AS — от 100 до 1099**
- **2 сессии — к соседям**
- **Дефолт (0.0.0.0/0) анонсируется только нулевым namespace (ROUTER_0)**

MIRANTIS

Конфиг (часть 2)

```
router bgp 101
    bgp router-id 172.31.1.1
    network 172.31.1.0/24
    network 172.31.2.0/24
    neighbor 172.31.1.2 remote-as 100
    neighbor 172.31.2.1 remote-as 102
    neighbor 172.31.1.2 next-hop-self
    neighbor 172.31.2.1 next-hop-self
```

MIRANTIS

Пробуем взлететь?

- Собственно все экземпляры BGPd запускаем скриптом, окружения созданы заранее
- **ps -auxfw | grep bgpd | grep -v grep | wc -l**

  101

MIRANTIS

Ну а теперь в консоль

- Проверить как распространяются маршруты
- Проверить как ведут себя процессы

MIRANTIS

Сначала маршрутизация

- Подключится к BGPd:
- **ip netns exec ROUTER_50 telnet 127.0.0.1 2605**
- **2605** — Управляющий порт BGPd
- НЕ путать с портом **179!**

MIRANTIS

Как там соседи?

- Проверим — работают ли наши BGP сессии.

```
sh ip bgp summary
Peers 2, using 9120 bytes of memory
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
172.31.50.2     4  1049     578     580        0    0    0 08:46:17        52
172.31.51.1     4  1051     577     581        0    0    0 08:46:15        51
```

MIRANTIS

И полная таблица

- sh ip bgp
- Нет, скриншота не будет — очень много там :)

MIRANTIS

Ну и напоследок — наше виртуальное приложение

- В качестве приложения для теста используем обычный traceroute
- Как поведет себя traceroute -n 8.8.8.8 ?
- Работать будет?

MIRANTIS

TTL по умолчанию - 30

- Не работает потому что traceroute отправляет пакеты с ttl=**30**
- А что б работало — добавить ключ **-m**

MIRANTIS

И все равно не работает!

- Потому что линукс не может слать пакеты с TTL больше 64
- Точнее может — но требуется настройка параметров ядра
- Но это уже совсем другая история
- А тем кто ОЧЕНЬ хочет знать. **net.ipv4.ip_default_ttl**

MIRANTIS

Спасибо!