Использование Mikrotik RouterOS и Linux namespaces для моделирования реальных сетей.

Сычёв Андрей
H-Тема Сервис
Киев

Республика Беларусь
Минск
2014

Время от времени возникает необходимо
смоделировать работу сети, но н
возможности развернуть полноценн
тестовый стенд.

Некоторые особенности работы реальн
сетей – задержки и потери пакетов слож
промоделировать на реальном стенде
симуляторах типа GNS3.

Возникает задача эмуляции рабо
физического оборудован
(маршрутизаторов) и эмуляция работы сети

Для эмуляции работы маршрутизато
будем использовать RouterOS x86 вну
VirtualBox.

Для эмуляции работы сети – функционал
Linux – network namespaces.

VirtualBox

VirtualBox (Oracle VM VirtualBox) — программный продукт виртуализации для операционных систем Microsoft Windows, Linux, FreeBSD, Mac OS X, Solaris/OpenSolaris, ReactOS, DOS и других.

Будем использовать для запуска RouterOS для x86.

В виртуальной машине может быть до четырех сетевых карт.

Linux network namespaces

Network namespace - это логически отделенный от других стек сетевых протоколов в Linux - контейнерная виртуализация для сетевых интерфейсов.

Эмулируется полностью сетевой стек: сетевые интерфейсы, таблица маршрутизации, файрволл и т.д.

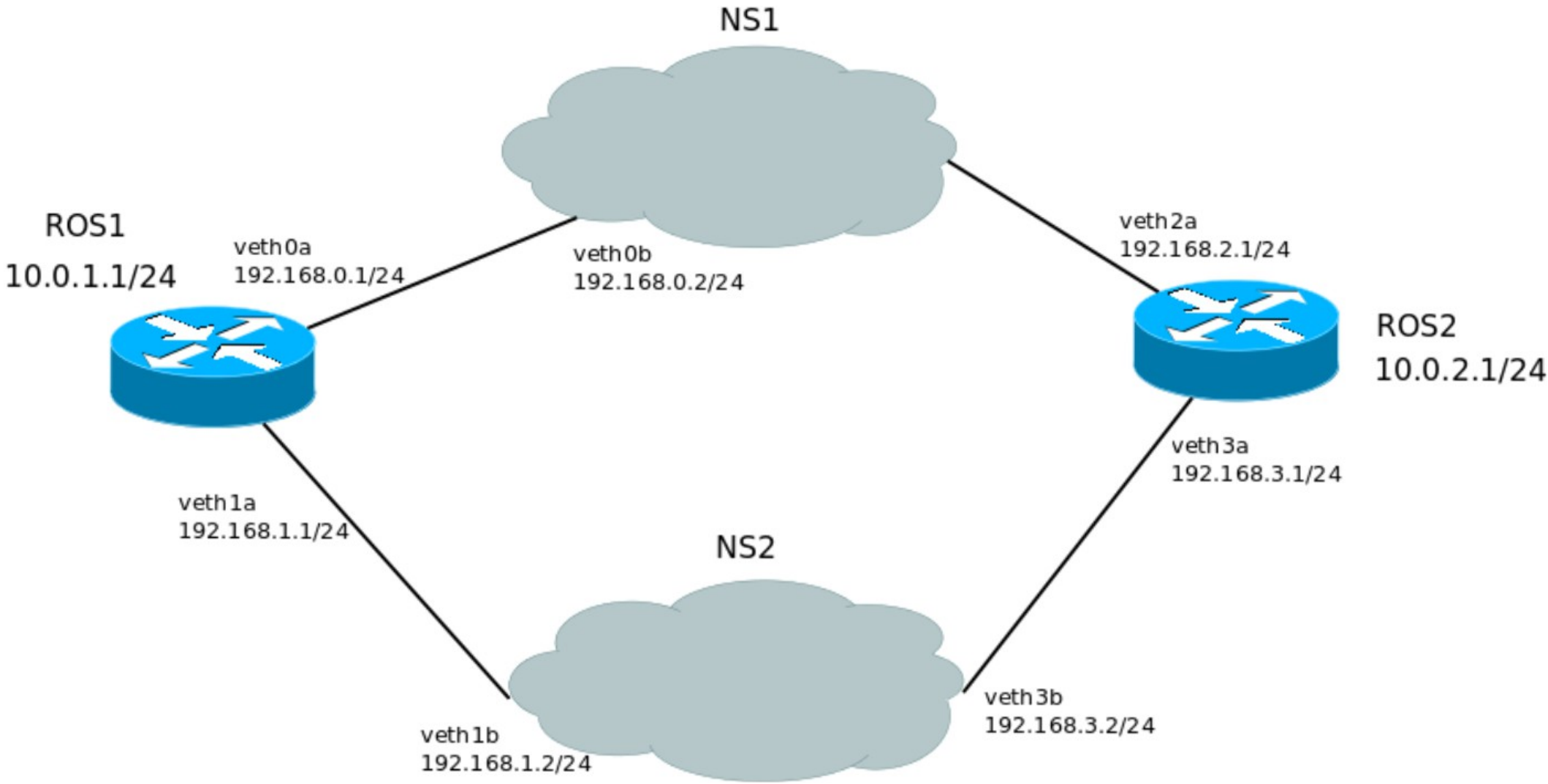
Работает на уровне ядра и для определенных процессов.

Есть по умолчанию почти во всех современных дистрибутивах Linux.

Linux network namespaces

Основные действия выполняются с помощью команды ip netns:

- ip netns list - посмотреть все network namespaces
- ip netns add NAME - создать новый network namespace
- ip netns delete NAME - удалить network namespace
- ip netns exec NAME cmd - выполнить команду внутри network namespace


NS1
ROS1
10.0.1.1/24
veth0a
192.168.0.1/24
veth0b
192.168.0.2/24
veth2a
192.168.2.1/24
ROS2
10.0.2.1/24
veth3a
192.168.3.1/24
veth1a
192.168.1.1/24
NS2
veth1b
192.168.1.2/24
veth3b
192.168.3.2/24

Создаем два namespaces и интерфейсы

```
ip netns add NS1
ip netns add NS2
```

- Разные network namespace можно объединять между собой с помощью виртуальных сетевых интерфейсов (veth). Интерфейсы создаются парой, что попало в один – выходит из другого. Интерфейсы можно добавлять в бридж в ОС Linux.

```
ip link add veth0a type veth peer name veth0b
ip link add veth1a type veth peer name veth1b
ip link add veth2a type veth peer name veth2b
ip link add veth3a type veth peer name veth3b
```

- Подключаем интерфейс в соответствующий namespaces

```
ip link set veth0b netns NS1
ip link set veth2b netns NS1
ip link set veth1b netns NS2
ip link set veth3b netns NS2
```

Смотрим что получилось

- root@it-laptop:~# ip netns add NS1
- root@it-laptop:~# ip netns add NS2
- root@it-laptop:~#
- root@it-laptop:~# ip netns list
- NS2
- NS1

- root@it-laptop:~# ip link add veth0a type veth peer name veth0b
- root@it-laptop:~# ip link add veth1a type veth peer name veth1b
- root@it-laptop:~# ip link add veth2a type veth peer name veth2b
- root@it-laptop:~# ip link add veth3a type veth peer name veth3b
- root@it-laptop:~#
- root@it-laptop:~# ip link list
- 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default
- link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
- 4: vboxnet0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
- 5: veth0b: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
- 6: veth0a: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
- 7: veth1b: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
- 8: veth1a: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
- 9: veth2b: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
- 10: veth2a: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
- 11: veth3b: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
- 12: veth3a: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN

- Внутри NS1

root@it-laptop:~# ip netns exec NS1 ip link list

1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN

4: veth0b: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state D

8: veth2b: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state D

- Внутри NS2

root@it-laptop:~# ip netns exec NS2 ip link list

1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN

6: veth1b: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state D

10: veth3b: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state I

- Разрешаем маршрутизацию для каждого NS

```
ip netns exec NS1 sysctl net.ipv4.conf.all.forwarding=1
ip netns exec NS2 sysctl net.ipv4.conf.all.forwarding=1
```

- Назначаем адреса и поднимаем интерфейсы внутри NS

```
ip netns exec NS1 ifconfig lo up
ip netns exec NS1 ifconfig veth0b 192.168.0.2/24 up
ip netns exec NS1 ifconfig veth2b 192.168.2.2/24 up
ip netns exec NS2 ifconfig lo up
ip netns exec NS2 ifconfig veth1b 192.168.1.2/24 up
ip netns exec NS2 ifconfig veth3b 192.168.3.2/24 up
```

```
root@it-laptop:~# ip netns exec NS1 ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

veth0b    Link encap:Ethernet  HWaddr e2:35:07:d7:f6:5f
          inet addr:192.168.0.2  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

veth2b    Link encap:Ethernet  HWaddr 5a:b3:c2:3c:66:e3
          inet addr:192.168.2.2  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

```
root@it-laptop:~# ip netns exec NS2 ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

veth1b    Link encap:Ethernet  HWaddr 5a:d3:58:1c:ef:21
          inet addr:192.168.1.2  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

veth3b    Link encap:Ethernet  HWaddr 3a:c2:0e:1b:98:11
          inet addr:192.168.3.2  Bcast:192.168.3.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

- Поднимаем интерфейсы в “основной” систем

```
ifconfig veth0a up
ifconfig veth1a up
ifconfig veth2a up
ifconfig veth3a up
```

Установка RouterOS в VirtualBox

Создаем две виртуальных машины ROS1 и ROS2.

Сетевые интерфейсы ROS1:

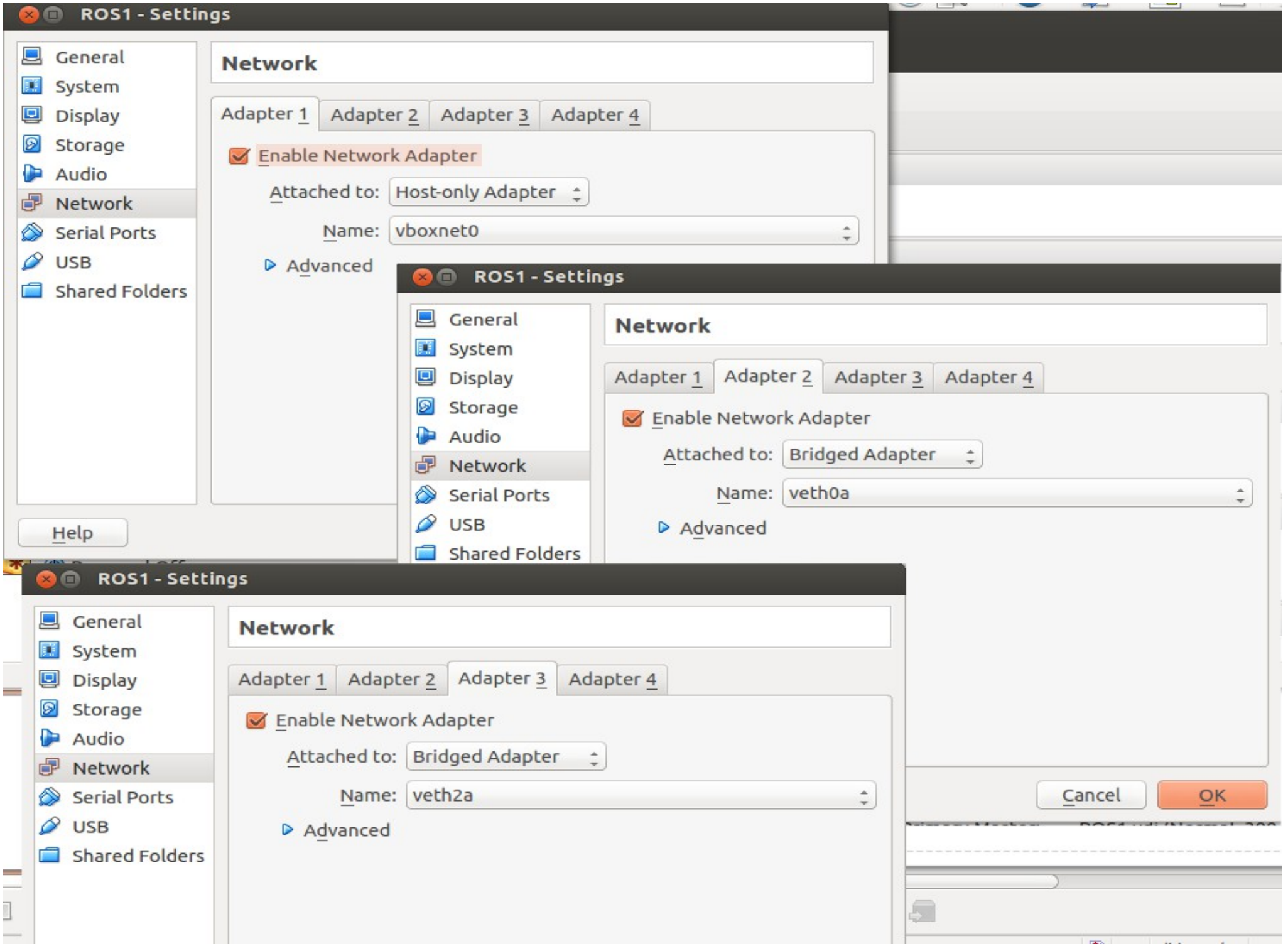
- Ether1 – Host-Only adapter -vboxnet0 – 192.168.56.101/24
- Ether2 – bridge - veth0a
- Ether3 – bridge - veth3a

Сетевые интерфейсы ROS1:

- Ether1 – Host-Only adapter -vboxnet0 – 192.168.56.101/24
- Ether2 – bridge - veth0a
- Ether3 – bridge - veth2a

Сетевые интерфейсы ROS2:

- Ether1 – Host-Only adapter -vboxnet0 – 192.168.56.102/24
- Ether2 – bridge - veth1a
- Ether3 – bridge - veth3a


ROS1 - Settings
General
System
Display
Storage
Audio
Network
Serial Ports
USB
Shared Folders
Network
Adapter 1
Adapter 2
Adapter 3
Adapter 4
Enable Network Adapter
Attached to: Host-only Adapter
Name: vboxnet0
Advanced
Help
ROS1 - Settings
General
System
Display
Storage
Audio
Network
Serial Ports
USB
Shared Folders
Network
Adapter 1
Adapter 2
Adapter 3
Adapter 4
Enable Network Adapter
Attached to: Bridged Adapter
Name: veth0a
Advanced
ROS1 - Settings
General
System
Display
Storage
Audio
Network
Serial Ports
USB
Shared Folders
Network
Adapter 1
Adapter 2
Adapter 3
Adapter 4
Enable Network Adapter
Attached to: Bridged Adapter
Name: veth2a
Advanced
Cancel
OK

Стартуем виртуальные машины и к ним подключаемся через WinBox

- [admin@ROS1] > /ip address print

  Flags: X - disabled, I - invalid, D - dynamic

#	ADDRESS	NETWORK	INTERFACE
0	192.168.56.101/24	192.168.56.0	ether1
1	192.168.0.1/24	192.168.0.0	ether2
2	192.168.2.1/24	192.168.2.0	ether3
3	10.0.1.1/24	10.0.1.0	bridge1

```
[admin@ROS2] > /ip address print
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK       INTERFAC
 0   192.168.56.102/24  192.168.56.0  ether1
 1   192.168.1.1/24     192.168.1.0   ether2
 2   192.168.3.1/24     192.168.3.0   ether3
 3   10.0.2.1/24        10.0.2.0      bridge1
```

Проверяем доступность интерфейсов в namespaces

```
[admin@ROS1] > ping 192.168.0.2
HOST                                     SIZE TTL TIME  STATUS
192.168.0.2                                56  64 0ms
192.168.0.2                                56  64 0ms
    sent=2 received=2 packet-loss=0% min-rtt=0ms avg-rtt=0ms max-rtt=0ms

[admin@ROS1] > ping 192.168.1.2
HOST                                     SIZE TTL TIME  STATUS
192.168.1.2                                56  64 0ms
192.168.1.2                                56  64 0ms
    sent=2 received=2 packet-loss=0% min-rtt=0ms avg-rtt=0ms max-rtt=0ms
```

```
[admin@ROS2] > ping 192.168.2.2
HOST                                     SIZE TTL TIME  STATUS
192.168.2.2                                56  64 0ms
192.168.2.2                                56  64 0ms
    sent=2 received=2 packet-loss=0% min-rtt=0ms avg-rtt=0ms max-rtt=0ms

[admin@ROS2] > ping 192.168.3.2
HOST                                     SIZE TTL TIME  STATUS
192.168.3.2                                56  64 0ms
192.168.3.2                                56  64 0ms
    sent=2 received=2 packet-loss=0% min-rtt=0ms avg-rtt=0ms max-rtt=0ms
```

Добавляем маршруты в namespaces

ip netns exec NS1 route add -net 10.0.1.0/24 gw 192.168.0.1

ip netns exec NS1 route add -net 10.0.2.0/24 gw 192.168.2.1

root@it-laptop:~# ip netns exec NS1 route

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
10.0.1.0	192.168.0.1	255.255.255.0	UG	0	0	0	veth0b
10.0.2.0	192.168.2.1	255.255.255.0	UG	0	0	0	veth2b
192.168.0.0	*	255.255.255.0	U	0	0	0	veth0b
192.168.2.0	*	255.255.255.0	U	0	0	0	veth2b

```
ip netns exec NS2 route add -net 10.0.1.0/24 gw 192.168.1.1
ip netns exec NS2 route add -net 10.0.2.0/24 gw 192.168.3.1

ip netns exec NS2 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.1.0        192.168.1.1     255.255.255.0   UG    0      0        0 veth1b
10.0.2.0        192.168.3.1     255.255.255.0   UG    0      0        0 veth3b
192.168.1.0     *               255.255.255.0   U     0      0        0 veth1b
192.168.3.0     *               255.255.255.0   U     0      0        0 veth3b
```

Добавляем маршруты через NS1 в ROS1 и ROS2

[admin@ROS1] > /ip route add dst-address=10.0.2.0/24 gateway=192.168.0.2

[admin@ROS1] > /ip route add dst-address=192.168.2.0/24 gateway=192.168.0.2

-

[admin@ROS2] > /ip route add dst-address=10.0.1.0/24 gateway=192.168.2.2

[admin@ROS1] > /ip route add dst-address=192.168.0.0/24 gateway=192.168.2.2

Проверяем

```
[admin@ROS1] > ping 10.0.2.1
HOST                                     SIZE TTL TIME  STATUS
10.0.2.1                                   56  63 1ms
10.0.2.1                                   56  63 1ms
10.0.2.1                                   56  63 1ms
    sent=3 received=3 packet-loss=0% min-rtt=1ms avg-rtt=1ms max-rtt=1ms

[admin@ROS1] > /tool traceroute 10.0.2.1
 # ADDRESS                                LOSS SENT    LAST     AVG    BEST   WORST
 1 192.168.0.2                            0%    2   0.7ms     0.8     0.7     0.8
 2 10.0.2.1                               0%    2   0.9ms     1.2     0.9     1.5
```

```
[admin@ROS2] > ping 10.0.1.1
HOST                                     SIZE TTL TIME  STATUS
10.0.1.1                                  56  63 0ms
10.0.1.1                                  56  63 0ms
    sent=2 received=2 packet-loss=0% min-rtt=0ms avg-rtt=0ms max-rtt=0ms

[admin@ROS2] > /tool traceroute  10.0.1.1
 # ADDRESS                          LOSS SENT    LAST     AVG    BEST   WORST
 1 192.168.2.2                      0%    3   0.5ms     0.5     0.4     0.5
 2 10.0.1.1                        0%    3   0.8ms     0.7     0.6     0.8
```

Эмуляция прохождения трафика через реальную сеть.

В ядро ОС linux входит модуль netem, который предоставляет функционал для эмуляции WAN. Текущая версия модуля имеет следующие функции:

- эмуляция задержки, с различной функцией распределения
- эмуляция потерь
- эмуляция повтора пакетов
- эмуляция перемешивания пакетов
- эмуляция искажения пакетов

Этот модуль включен по умолчанию в большинство современных дистрибутивов, основанных на ядре операционной системы Linux.

Эмуляция задержки пакетов

- Фиксированная задержка 100 мс

  ip netns exec NS1 tc qdisc add dev veth0b root netem delay 100ms

  [admin@ROS2] > ping 10.0.1.1

HOST	SIZE	TTL	TIME	STATUS
10.0.1.1	56	63	0ms	
10.0.1.1	56	63	0ms	
10.0.1.1	56	63	0ms	
10.0.1.1	56	63	101ms	
10.0.1.1	56	63	101ms	
10.0.1.1	56	63	101ms	
10.0.1.1	56	63	101ms	

sent=10 received=10 packet-loss=0% min-rtt=0ms avg-rtt=70ms max-rtt=101ms

В модуле netem существует возможность задавать неравномерное распределение задержки. Например, чтобы задать нормальную функцию распределения, нужно сделать следующие:

ip netns exec NS1 tc qdisc change dev veth0b root netem delay 100ms 20ms distribution normal

Другие таблицы распределения(normal, pareto, paretonormal) устанавливаются вместе с iproute2 в каталог /usr/lib/tc. Можно сгенерировать свою таблицу распределения, отражающую задержки канала связи, основанную на экспериментальных данных.

```
[admin@ROS2] > ping 10.0.1.1
HOST                                     SIZE TTL TIME  STATUS
10.0.1.1                                   56  63 94ms
10.0.1.1                                   56  63 68ms
10.0.1.1                                   56  63 107ms
10.0.1.1                                   56  63 76ms
10.0.1.1                                   56  63 136ms
10.0.1.1                                   56  63 146ms
10.0.1.1                                   56  63 106ms
10.0.1.1                                   56  63 108ms
10.0.1.1                                   56  63 95ms
10.0.1.1                                   56  63 98ms
10.0.1.1                                   56  63 100ms
10.0.1.1                                   56  63 105ms
10.0.1.1                                   56  63 134ms
    sent=13 received=13 packet-loss=0% min-rtt=68ms avg-rtt=105ms
   max-rtt=146ms
```

Эмуляция потери пакетов

Зададим 10% потерь в канале

ip netns exec NS1 tc qdisc change dev veth0a root netem loss 10%

[admin@ROS2] > ping 10.0.1.1

HOST	SIZE	TTL	TIME	STATUS
10.0.1.1	56	63	0ms	
10.0.1.1				timeout
10.0.1.1	56	63	0ms	
10.0.1.1	56	63	0ms	
10.0.1.1	56	63	0ms	
10.0.1.1	56	63	1ms	
10.0.1.1	56	63	1ms	
10.0.1.1	56	63	1ms	
10.0.1.1	56	63	0ms	
10.0.1.1				timeout
10.0.1.1	56	63	2ms	
10.0.1.1	56	63	1ms	
10.0.1.1	56	63	1ms	

sent=13 received=11 packet-loss=15% min-rtt=0ms avg-rtt=0ms max-rtt=2ms

Эмуляция случайных задержек и потерь

ip netns exec NS1 tc qdisc change dev veth0b root netem delay 100ms 20ms distribution normal loss 25%

```
[admin@ROS2] > ping 10.0.1.1
HOST                                     SIZE TTL TIME  STATUS
10.0.1.1                                  56  63 102ms
10.0.1.1                                         timeout
10.0.1.1                                  56  63 120ms
10.0.1.1                                  56  63 105ms
10.0.1.1                                  56  63 127ms
10.0.1.1                                  56  63 95ms
10.0.1.1                                  56  63 109ms
10.0.1.1                                  56  63 126ms
10.0.1.1                                         timeout
10.0.1.1                                         timeout
10.0.1.1                                  56  63 105ms
10.0.1.1                                  56  63 60ms
10.0.1.1                                         timeout
10.0.1.1                                  56  63 115ms
10.0.1.1                                  56  63 92ms
10.0.1.1                                  56  63 95ms
    sent=16 received=12 packet-loss=25% min-rtt=60ms avg-rtt=104ms   max-rtt=127ms
```

Источники

- http://blog.e0ne.info/post/Network-namespace-virtual-network-stack-in-linux.aspx
- http://habrahabr.ru/post/24046/
- http://blog.scottlowe.org/2013/09/04/introducing-linux-network-namespaces/
- http://lwn.net/Articles/580893/

Спасибо за внимание.

Вопросы ?