УДК 004.714

Шилина Дарья Александровна
студентка
кафедра комплексной безопасности критически важных объектов,
Российский государственный университет им. И.М. Губкина,
Россия, г. Москва
e-mail: schilina.dash@yandex.ru

Подрезов Павел Александрович
студент,
кафедра комплексной безопасности критически важных объектов,
Российский государственный университет им. И.М. Губкина,
Россия, г. Москва

НАСТРОЙКА СЕТЕВОГО МОСТА НА БАЗЕ KVM

Аннотация: *Конфигурирование мостов — важная задачей настройки сети виртуальных машин. Благодаря мостам происходит подключение к физической сети и возможность изолироваться нескольким виртуальным машинам от хост-системы. Данная статья создана с целью, дать общее понимание и представление по настройке сетевых мостов на базе KVM с использованием различных инструментов и технологий. Мы рассмотрим три основных инструмента для настройки сетевых мостов в KVM, а именно iproute2, openvswitch, systemd-networkd. Для проверки реализации воспользуемся простой топологией, представленной на рисунке 1. В качестве хоста будет выступать ALT Linux версии 10.4, а на гостевых машинах будет стоять Rosa Plasma 5.12.5.1.*

Ключевые слова: сетевой мост, конфигурация, iproute2, openvswitch, systemd-networkd, KVM.

Shilina Daria Alexandrovna
student,
Department of Integrated Security of Critical Facilities
Russian State University named after I.M. Gubkin
Russia, Moscow

Podrezov Pavel Alexandrovich
student,
Department of Integrated Security of Critical Facilities,
Russian State University named after I.M. Gubkin
Russia, Moscow

SETTING UP A KVM-BASED NETWORK BRIDGE

Abstract: *Configuring bridges is an important task of configuring a network of virtual machines. Bridges enable connection to a physical network and the ability for several virtual machines to isolate themselves from the host system. This article is designed to provide a general understanding and understanding of how to set up KVM-based network bridges using various tools and technologies. We will look at three main tools for configuring network bridges in KVM, namely iproute2, openvswitch, and systemd-networkd. To verify the implementation, we will use the simple topology shown in Figure 1. ALT Linux version 10.4 will act as the host, and Rosa Plasma 5.12.5.1 will be installed on guest machines.*

Key words: network bridge, configuration, iproute2, openvswitch, systemd-networkd, KVM.

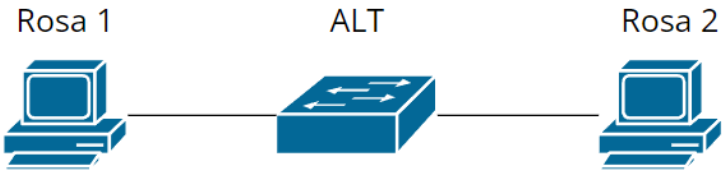

Рисунок 1. Топология (Topology)

Каждая из настроек сетевых мостов имеет свои плюсы и минусы, важно осознавать цель создания моста. Начнем более подробный обзор с **iproute2**.

Iproute2 — это набор утилит для управления параметрами сетевых устройств. Является классической для настройки и в основном используется для простых сетевых окружений. Использует всего три основных утилиты: ip, tc, ss.

Его реализация не сложная, ведь является встроенной в системах Linux. Это служит большим преимуществом при базовой настройке сети. Iproute2 позволяет настраивать физические и виртуальные интерфейсы, задавать IP-адреса, поднимать и опускать интерфейсы, а также управлять их настройками и производить настройку маршрутизации. Поддерживает QoS и VLAN. Утилита tc позволяет настраивать политику управления трафиком, а также можно создавать туннели GRE и VXLAN. Iproute2 поддерживает такие протоколы, как IPv6 и MPLS, что делает его актуальным для современных сетевых решений.

Пример настройки маршрутизации:

ip route add 10.0.0.0/8 via 192.168.1.1

Пример настройки политик:

tc qdisc add dev eth0 root handle 1: htb default 30

Кратко продемонстрируем и опишем настройку моста с помощью iproute2:

1. Создаём мост, поднимаем его и присваиваем ему ip

```
[root@host-15 ~]# ip link add name br0 type bridge
[root@host-15 ~]# ip link set enp0s3 master br0
[root@host-15 ~]# ip link set dev br0 up
[root@host-15 ~]# ip addr add 192.168.1.1/24 dev br0
[root@host-15 ~]#
```

Рисунок 2. Создание моста

2. Проверяем создание моста, выводя информация о сетевых интерфейсах на ALT:

```
[pavel@host-15 Рабочий стол]$ ip -c a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master br0
state UP group default qlen 1000
    link/ether 08:00:27:25:d7:44 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 86058sec preferred_lft 86058sec
    inet6 fe80::a00:27ff:fe25:d744/64 scope link
       valid_lft forever preferred_lft forever
3: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default qlen 1000
    link/ether 46:41:44:e7:ac:d3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.1/24 scope global br0
       valid_lft forever preferred_lft forever
    inet6 fe80::4441:44ff:fee7:acd3/64 scope link
       valid_lft forever preferred_lft forever
```

Рисунок 3. Сетевые интерфейсы

3. Добавляем на интерфейсы enp1s0 ip адреса для Rosa1 и Rosa2, а далее для проверки правильности настройки запускаем ping.

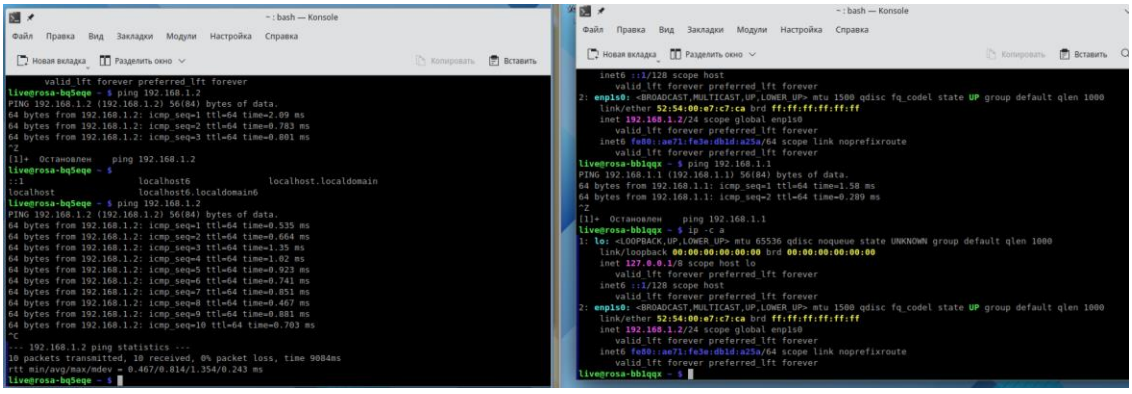

Рисунок 4. Проверка ping

Следующей рассмотрим утилиту **openvswitch**. Она уже хорошо работает в

сложных сетевых окружениях, потому что в отличие от стандартных Linux-мостов предоставляет большой функционал, включающий в себя, например, VLAN, tunneling и разные другие.

OVS, по сути, работает, как коммутатор уровня L2, так как перенаправляет трафик между портами. Его работа осуществляется как на физических серверах, так и внутри виртуальных машин. Openvswitch имеет весь основной функционал iproute2, а также обладает совместимостью с SDN, механизмом фильтрации ACL и шифрования, поддерживает мультикастинг.

Пример настройки QoS, создание очереди:

ovs-vsctl set port eth0 qos=@newq -- --id=@newq create QoS type=linux-htb other-config:max-rate=1000000

Применение QoS к порту:

ovs-vsctl set port eth0 qos=@newq

Openvswitch оптимизирован для работы в высоконагруженных окружениях. Может работать как на уровне пользовательского пространства, так и в режиме ядра, что позволяет достичь высокой производительности. Широко используется в облачных платформах, таких как OpenStack, а также в крупных корпоративных сетях. Подходит для виртуализированных сред, где необходимы сложные сценарии управления трафиком и сетевыми ресурсами.

Пример создания сетевого моста:

1. Установку при помощи команды:
   apt-get install openvswitch
2. Непосредственное создание моста при помощи таких команд как:
   Sytemctl start openvswitch
   Ovs-vsctl add-br ovs0
3. Добавление интерфейсов:
   Ovs-vsctl add-port ovs0 vnet1
   Ovs-vsctl add-port ovs0 vnet2
4. Проверка корректности настройки и запуск ping для заранее установленных ip адресов:

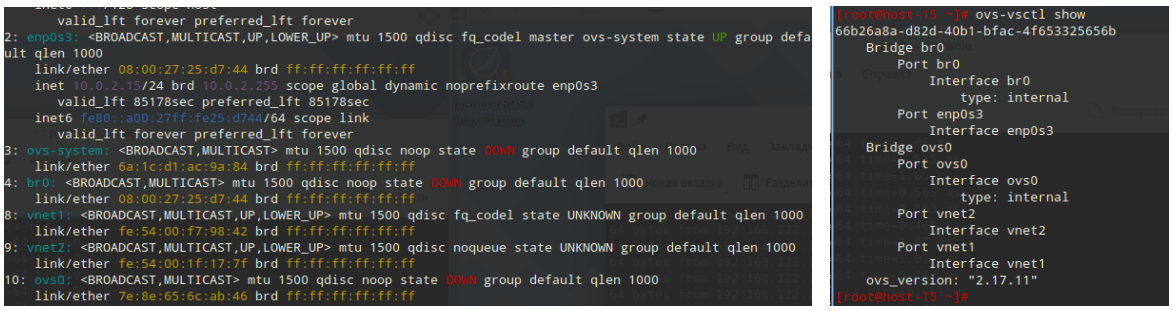

Рисунок 5. Проверка настройки

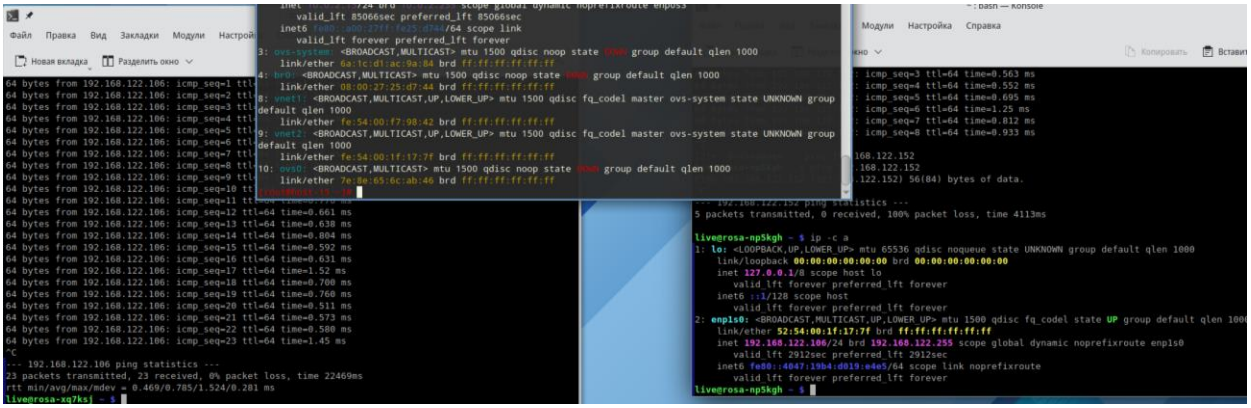

Рисунок 6. Проверка ping

Наконец, перейдем к **systemd–networkd.** Эта служба является системным демоном, предназначенным для конфигурирования сети. Этот подход является довольно современным, интегрированным с systemd.

Управление сетевыми интерфейсами происходит при помощи конфигурационных файлов, что делает его крайне удобным для администрирования. Достаточно простой формат и читаемость записей гораздо упрощают настройку, нежели работа с множеством команд в терминале. Большим преимуществом, в сравнении с другими методами настройки, можно выделить то, что для изменения какого-то конкретного файла конфигурации достаточно перезагрузить только его, а не терять время на перезагрузку всей сети.

Интеграция с другими компонентами systemd, позволяет упростить автоматизацию процессов, к примеру, можно использовать таймеры для периодической проверки сети, просто создав unit-файл со скриптом:

```
[Unit]
Description=Check Network Status
```

[Service]

Type=oneshot

ExecStart=/path/to/check_network.sh

[Install]

WantedBy=timers.target

Systemd-networkd демонстрирует наибольшую эффективность в средах, где systemd выступает в роли системы инициализации. Он широко используется на серверах контейнерного типа, а также в облачных решениях и системах, которые нуждаются в централизованном управлении сетевыми интерфейсами. Это достаточно универсальный инструмент, ведь он использует разнообразные протоколы и буквально каждый параметр сети можно адаптировать в соответствии с конкретными требованиями организации или проекта, обеспечивая полное управление конфигурацией сетевых интерфейсов.

Создаем файл конфигурации для моста по адресу: /etc/systemd/network/10-br0.netdev

И записываем туда следующие настройки:

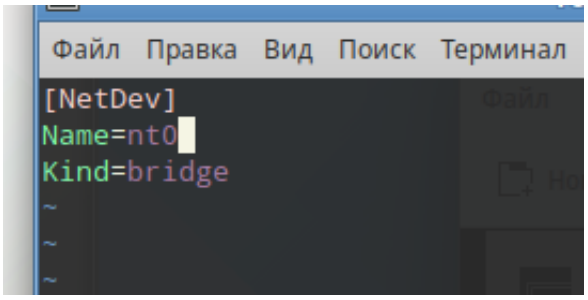

Рисунок 7. Настройки моста

Аналогично создаем файл по адресу /etc/systemd/network/10-br0.network со следующим содержимым:

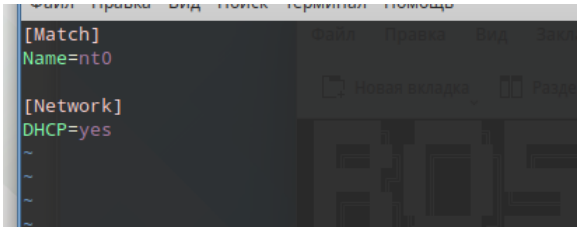

Рисунок 8. Настройки моста

Сюда же прописываем настройки для наших интерфейсов:

Рисунок 9. Настройки интерфейсов vnet0 и vnet1

После чего перезагружаем systemd-networkd командой:

Systemctl restart systemd-networkd

Проверяем появился ли наш мост nt0:

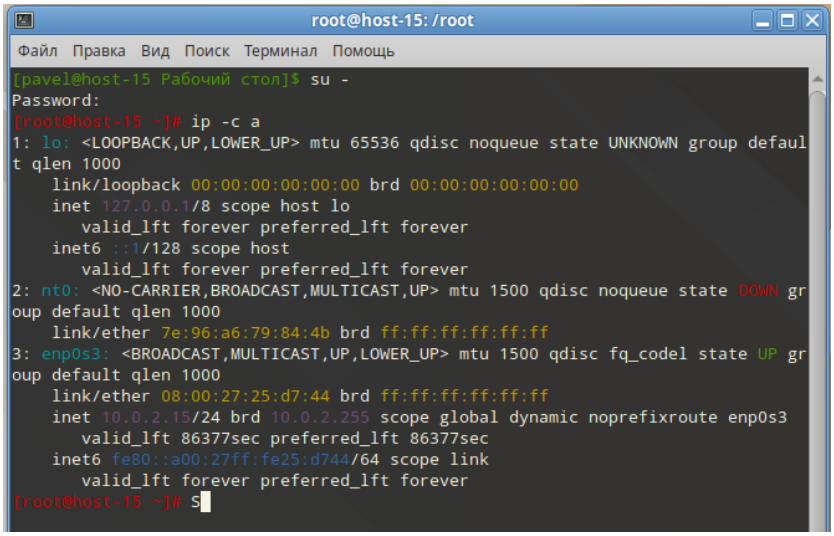

Рисунок 10. Проверка настройки

Заранее установив адреса на машины, пингуем Rosa1 и Rosa2:

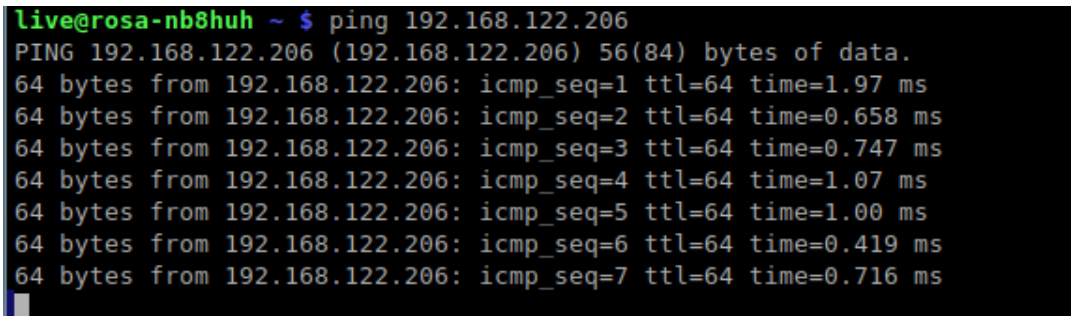

Рисунок 11. Проверка ping

А теперь разберёмся, как наши утилиты работают с KVM, и как происходит привязка адресов и интерфейсов в виртуализированной среде. Сначала виртуальная машина отправляет пакеты через виртуальный интерфейс vnet0 на мост br0, после чего эти пакеты уже пересылаются на физический

интерфейс eth0, определяя получателя по MAC-адресу.

Ip-адрес назначается в виртуальной машине внутри гостевой операционной системы и обычно является адресом из той же подсети, что и ip-адрес, назначенный br0. Таким образом, виртуальная машина получает сетевой доступ через физический интерфейс с ip-адресом, настроенным в виртуальной машине.

Физический интерфейс хоста (eth0), при использовании моста, фактически перестает иметь собственный ip-адрес, становясь частью моста. Его айпишник переходит к мостовому интерфейсу. Этот адрес используется для связи с другими машинами (физическими или виртуальными), подключенных к этому же мосту. Каждая виртуальная машина получает, свой собственный виртуальный сетевой интерфейс (vnet0) и он же подключается к созданному мосту, что позволяет виртуалкам взаимодействовать друг с другом. И, наконец, каждый виртуальный интерфейс, созданный для ВМ, получает свой уникальный MAC-адрес, который помогает мосту различать сетевые устройства, подключенные к нему.

Во всех трёх случаях, наш мост, настроенный через iproute2, openvswitch или systemd-networkd, является центральным элементом, который связывает виртуальные машины с физической сетью. Независимо от того, какой инструмент мы используем для создания моста, принцип один. Разница заключается лишь в том, как они реализуются в функциональности и инструментах управления.

Теперь мы можем привести сравнительную таблицу по трем видам настройки сетевого моста в KVM.

Таблица 1.

Три вида настройки сетевого моста в KVM

	Сложность настройки	Функционал	Преимущества	Недостатки	Использование

iproute2	Простая	Ограниченный	Простота в использовании Встроенность	Не удобен для сложных настроек	Простые сетевые окружения
openvswitch	Сложная	Расширенный	Гибкость Масштабируемость Наличие продвинутых функций	Требует дополнительных установок Сложный в настройке	Сложные сетевые окружения
systemd-networkd	Средняя	Умеренный	Современный Использует файлы конфигурации	Зависимость от systemd	Окружения, использующие systemd

Итак, после оценки каждой конфигурации становится понятно, для каких целей подходит каждых из выбранных нами подходов. Iproute2 лучше всего использовать, когда ваша задача состоит в создание базового моста, не имеющего никаких сложных требований и продвинутых функций. Openvswitch надо использовать в сложных задачах, где важную роль играют гибкость, масштабируемость и продвинутые функции. Systemd-networkd является современным решением и использует декларативную настройку, описывая параметры сети в файлах конфигурации. Благодаря этому, в данных файлах можно описывать все желаемое состояние сети, вместо ввода большого количества последовательных команд. В связи с этим, данная конфигурация является универсальной для разного рода настроек.

Список литературы:

1. BaseALT. [Электронный ресурс] // Режим доступа: URL: https://www.basealt.ru/en/download (дата обращения: 31.12.2024 г.).

2. Libvirt (Qemu+KVM+Virt-manager). [Электронный ресурс] // Режим доступа: URL: https://www.altlinux.org/Libvirt_(Qemu%2BKVM%2BVirt-manager) (дата обращения: 31.12.2024 г.).

3. Установить rosa-linux. [Электронный ресурс] // Режим доступа: URL: https://rosa.ru/rosa-linux-download-links/ (дата обращения: 21.12.2024).

4. Руководство по использованию iproute2. [Электронный ресурс] // Режим доступа: URL: https://www.altlinux.org/Etcnet (дата обращения: 31.12.2024 г.).

5. Руководство по использованию openvswitch. [Электронный ресурс] // Режим доступа: URL: https://www.altlinux.org/Etcnet/openvswitch (дата обращения: 31.12.2024 г.).

6. Руководство по использованию systemd-networkd. [Электронный ресурс] // Режим доступа: URL: https://www.altlinux.org/Systemd-networkd (дата обращения: 31.12.2024 г.).

7. Уймин А.Г. Компьютерные сети. L2-технологии: практикум. М.: Ай Пи Ар Медиа, 2024. 191 с.

8. Уймин А.Г. Периферийные устройства ЭВМ: практикум. М.: Ай Пи Ар Медиа, 2023. 429 с.