ДОКУМЕНТАЦИЯ ALT LINUX TEAM

Сетевые протоколы в Linux

Лабораторные работы

Редакция Февраль, 2026

Степан Мальчевский
ВМК МГУ им. М.В.Ломоносова Лаборатория свободного программного обеспечения
malchevskijsa@basealt.ru

Георгий Курячий
Базальт СПО Отдел разработки программного обеспечения
george@basealt.ru

Аннотация

Названия компаний и продуктов, встречающихся в руководстве, могут являться торговыми знаками соответствующих компаний.

Данное руководство соответствует текущему состоянию сведений, но какие-либо окончательные правки могли не попасть в него. В случае обнаружения ошибок и неточностей в руководство вносятся изменения.

Глава 1. Общая информация

Лабораторные работы направлены на закрепление практических навыков настройки и администрирования сети с использованием существующих сетевых технологий.

Для прохождения лабораторных работ используется разработанный образ виртуальной машины для VirtualBox на основе операционной системы ALT.

1.1. Содержание

1.1.1. Система VirtualBox VM Labs

1.1.2. Протоколы интерфейсного (канального) уровня

1.1.3. Маршрутизация в сети

1.1.4. Дополнительные возможности управления трафиком

1.2. Терминология

- Команды управления интерфейсом:
  - Просмотр доступных интерфейсов — **`ip [-d] link [show [<interface>]]`**
  - Включение/выключение интерфейса — **`ip link set <interface> {up|down}`**

- Создание/удаление виртуального интерфейса — **`ip link {add|del} dev <name> type <type>`**
- Создание/удаление связанного виртуального интерфейса — **`ip link {add|del} link <master-interface> name <name> type <type> [<parameters> <*args>]`**
- Связывание интерфейсов — **`ip link set <interface> master <master-interface>`**
- Изменение параметров интерфейса — **`ip link set dev <name> [type <type>] <parameter> <*args>`**

- Команды настройки IP-адресов:
  - Просмотр интерфейсов с описанием установленных IP-адресов — **`ip [-d] addr [show [<interface>]]`**
  - Установка/удаление IP-адреса на интерфейсе — **`ip addr {add|del} dev <interface> <IPv4>/<mask>`**
- Команды управления таблицами маршрутизации:
  - Просмотр таблиц маршрутизации — **`ip route [list [table <table-name>]]`**
  - Установка/удаление маршрута в таблицу маршрутизации — **`ip route {add|del} dev <interface> <IPv4-Net>/<mask>`**
- Команды мониторинга сети:
  - Проверка доступности абонентов в сети — **`ping [-c<N>] [-f] [-I <srcIP>] <dstIP>`**
  - Установление мониторинга — **`tcpdump [-c<N>] [-n] [-X] [-xx] -i <interface>`**
  - Отслеживание «маршрута» пакета в сети — **`traceroute [-s <srcIP>] <dstIP>`**
- Команды настройки VLAN:
  - Просмотр настроенных VLAN на интерфейсах — **`bridge vlan show`**
  - Добавление/удаление настройки VLAN на интерфейс — **`bridge vlan {add|del} vid <vlan-id> dev <interface> [pvid untagged]`**

Глава 2. Настройка системы для выполнения лабораторных

Лабораторная работа представляет из себя практико-ориентированную пошаговую инструкцию, выполняемую на одной или нескольких виртуальных машинах. В домашнее задание входит настройка некоторой топологии сети, её конфигурация и сбор этих данных в отчёт специального вида с помощью встроенных в виртуальные машины утилит.

В качестве менеджера виртуальных машин используется VirtualBox, система для выполнения лабораторных предоставляется в виде архива с виртуальной машиной (Open Virtual Appliance, **.ova**-файл), на основе которого будет создана основная система, на её основе будут создаваться копии для работы.

Важно

Для поддержки кроссплатформенности в отдельной главе будут описаны особенности, связанные с прохождением лабораторных на других ОС. Авторы гарантируют корректную работу лабораторных в рамках выполнения на операционной системе Альт.

VirtualBox поддерживает собственный набор команд для работы из терминала с помощью встроенной утилиты **VBoxManage**. Полный список команд представлен по команде:

```
$ VBoxManage --help
```

Предупреждение

VirtualBox поддерживает локализацию внутренних настроек, из-за чего переведённые файлы настроек **не работают**. **Никогда не запускайте VBoxManage** и/или **VirtualBox в русской локали**.

При обнаружении локализованных данных:

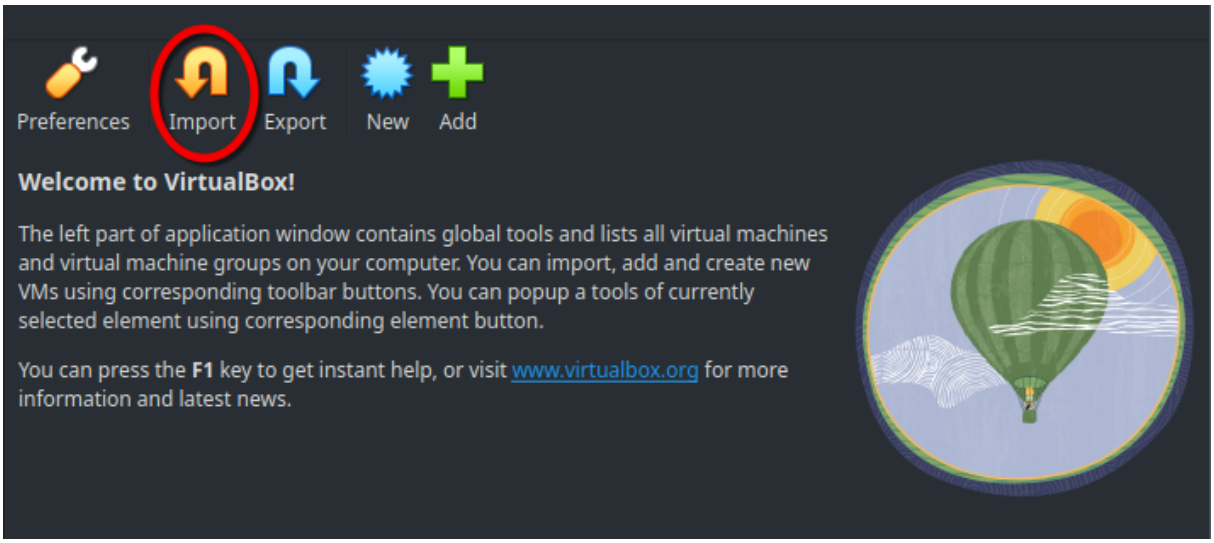
- Удалите все виртуальные машины и каталоги **~/VirtualBox VMs** и **~/.config/VirtualBox**.
- Сразу после того, как откроете терминал, выполните команду: **export LC_ALL=en_US.UTF-8**.

2.1. Импорт образа и создание виртуальных машин

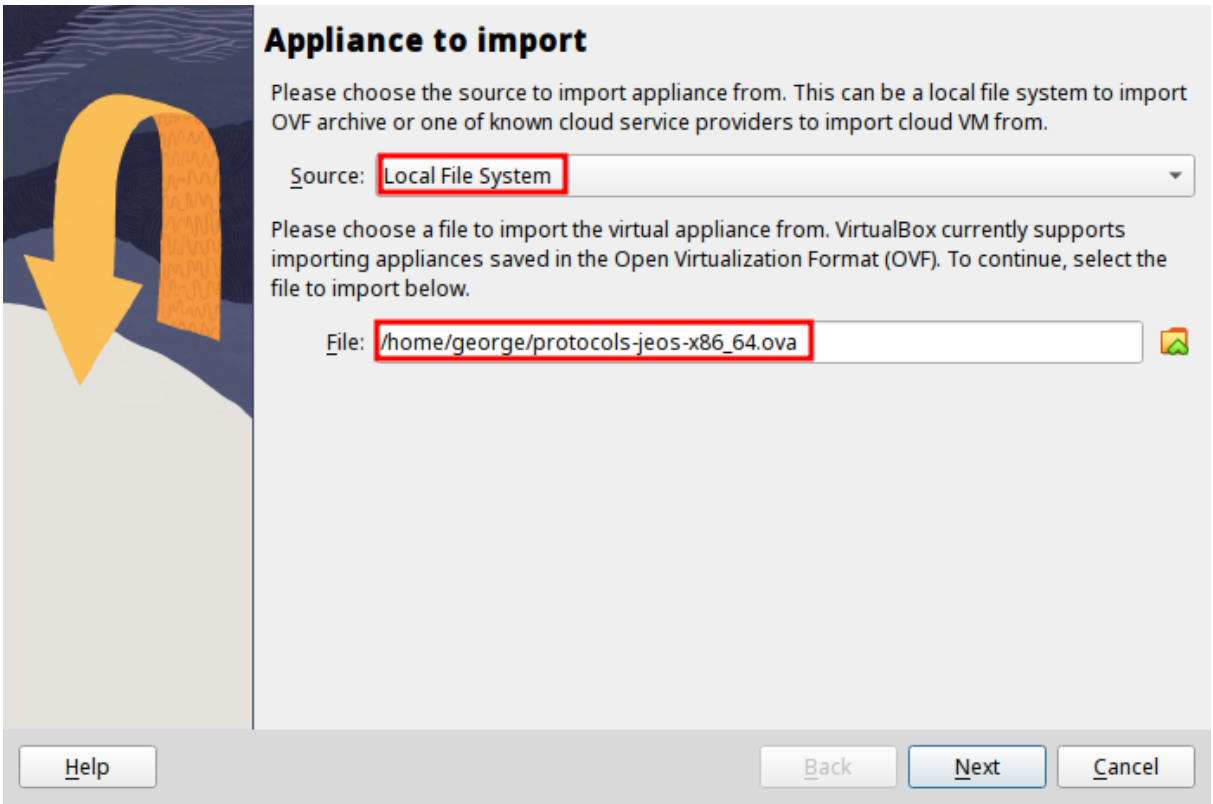
Для выполнения лабораторных необходимо единожды импортировать образ диска и после создавать клоны для работы. Для настройки можно воспользоваться интерфейсом **VirtualBox** или выполнить ряд команд **VBoxManage**.

2.1.1. Вариант 1: GUI-вариант

1. Откройте **VirtualBox**, выберите пункт «Импортировать образ».

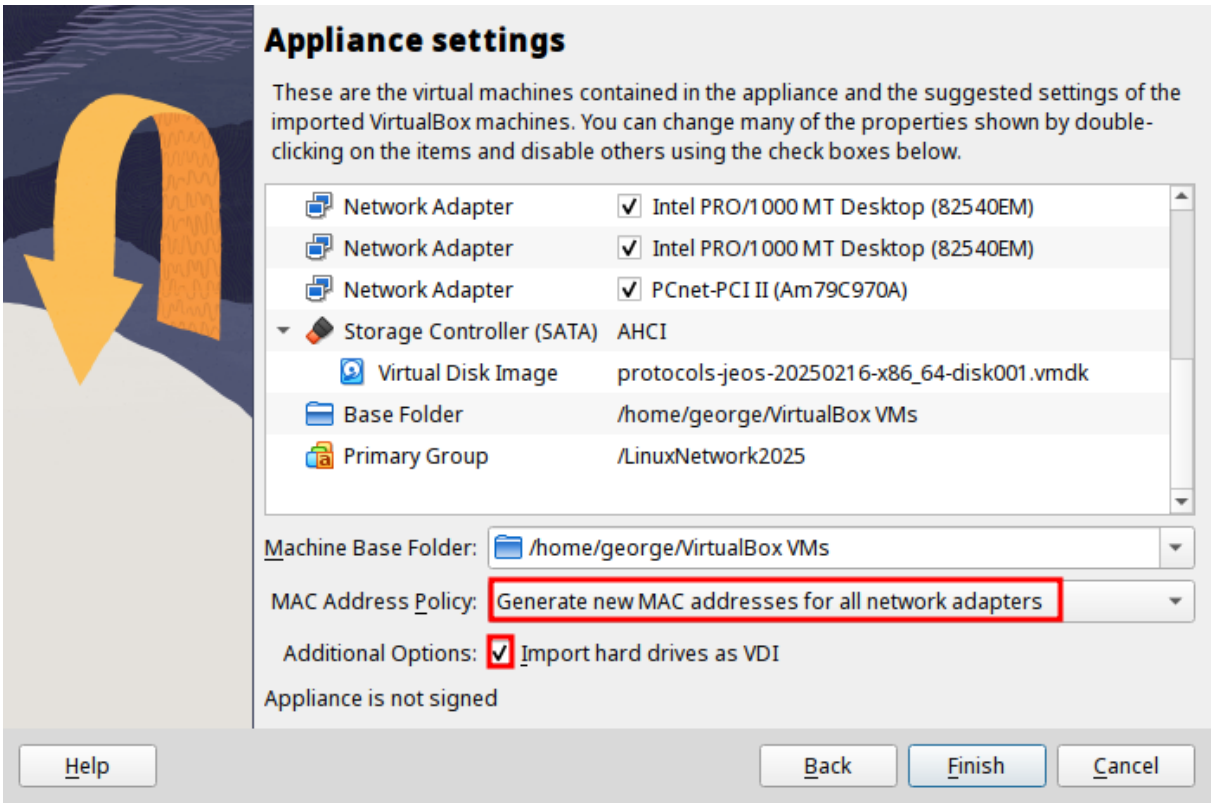


2. Выберите источником образа локальное хранилище данных и укажите путь к установленному **`.ova`**-файлу.



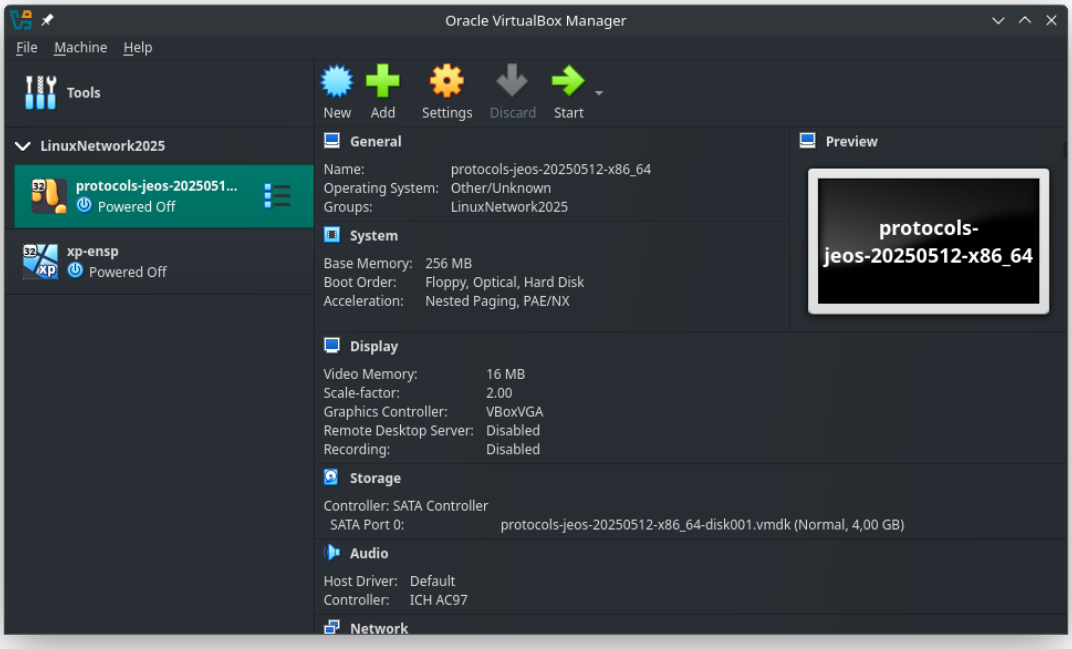
3. Перейдите к настройкам установки образа (в зависимости от версии **`VirtualBox`** настройки будут находиться на следующей странице установщика или в отдельной вкладке **`Settings`**):

   - Установите галочку в поле «Import hard drives as VDI».
   - Выберите политику MAC-адресов «Generate new MAC addresses for all network adapters».



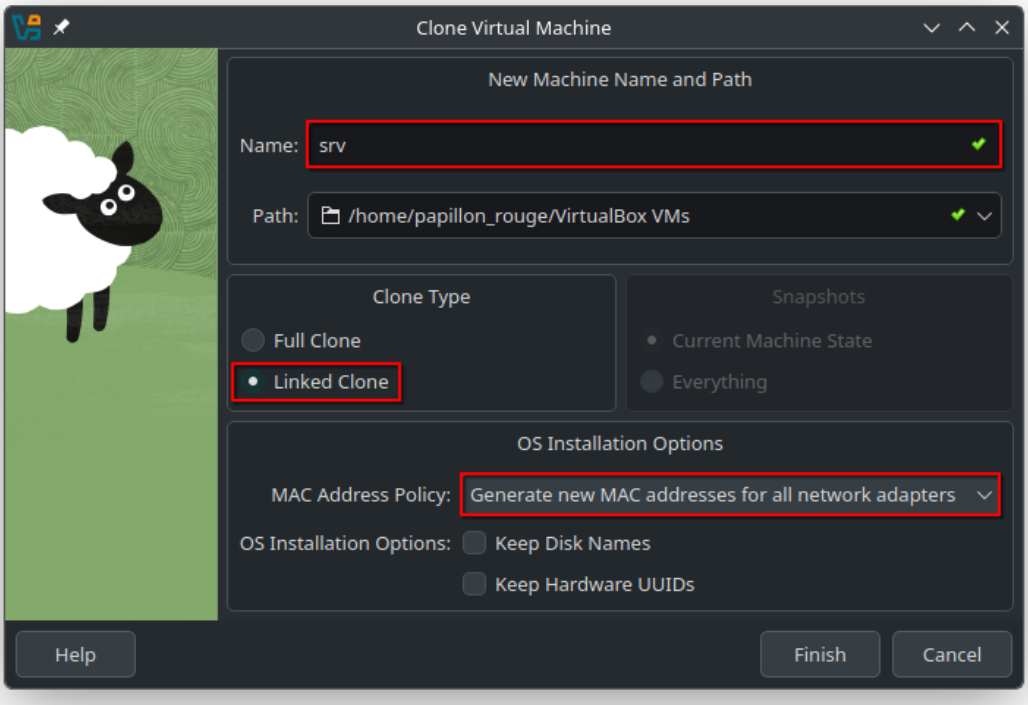
Завершите импортирование.

4. В итоге в **`VirtualBox`** должна появиться виртуальная машина с названием вида ***`protocols-jeos-<date>-x86_64`***.



5. Установленная виртуальная машина напрямую не используется для выполнения лабораторных — все действия выполняются на её *связных копиях* (клон представляет из себя не отдельную машину, а связанный блок изменений основной ВМ и клонированной).

Для клонирования машины нажмите ПКМ по основной машине и выберите во всплывающем окне поле «Clone». Окно настроек клонирования также открывается нажатием комбинации клавиш **`Ctrl+O`**.



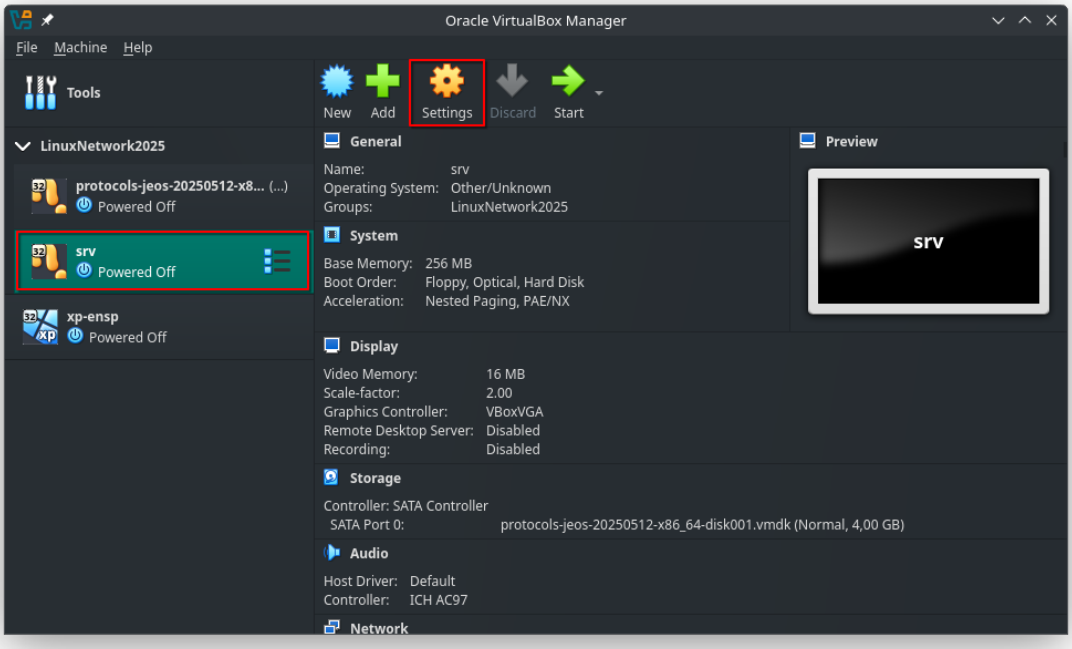
6. В настройках клонирования:

   - Укажите имя создаваемого клона.
   - Выберите тип «Связное клонирование».
   - Укажите политику MAC-адресов «Generate new MAC addresses for all network adapters».

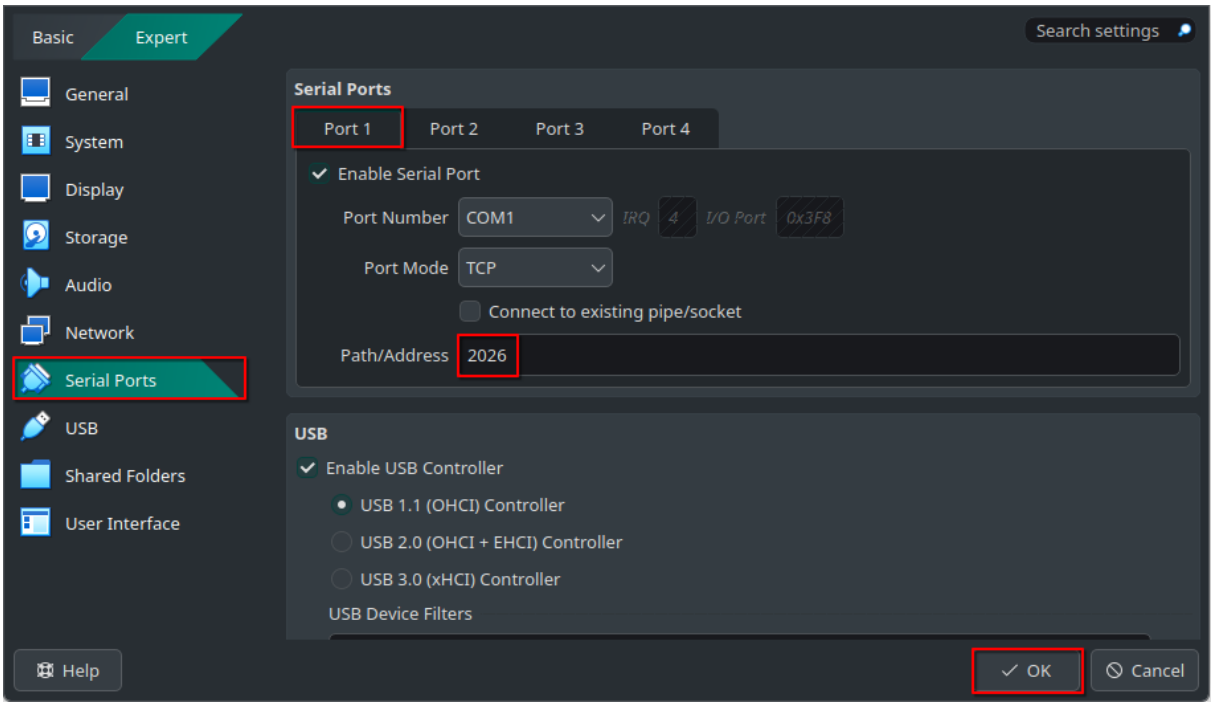
   Завершите клонирование.

7. Каждая копия наследует данные от основной машины. Для корректной работы клонов требуется перенастраивать значение COM-портов на уникальные для каждого клона.

   Выберите склонированную машину, перейдите в настройки.



В настройках выберите раздел **`Serial Ports`** и для порта **`Port 1`** укажите уникальное значение в поле **`Path/Address`**.



Поздравляем с созданием вашей первой рабочей машины!

2.1.2. Вариант 2: Настройка из командной строки

1. Импорт образа диска выполняется командой:

```
$ VBoxManage import path/to/appliance.ova
```

Пример выполнения команды:

```
[papillon_rouge@localhost ~]$ VBoxManage import Downloads/protocols-jeos-
x86_64.ova
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Interpreting /home/papillon_rouge/Downloads/protocols-jeos-x86_64.ova...
OK.
<...>
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Successfully imported the appliance.
~/papillon_rouge:
```

2. Для клонирования необходимо сохранить снапшот (состояние) основной машины и затем произвести клонирование на его основе.

   - Для создания снапшота используется команда:

     ```
     $ VBoxManage snapshot protocols-jeos-<date>-x86_64 take SNAPSHOT_NAME
     ```

     Пример выполнения команды:

     ```
     [papillon_rouge@localhost ~]$ VBoxManage snapshot protocols-
     jeos-20250216-x86_64 take srv_snapshot
     0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
     Snapshot taken. UUID: 3e6c0a9a-b9a8-44e6-92de-e96cd9db3790
     [papillon_rouge@localhost ~]$
     ```

   - Для клонирования виртуальной машины используется команда:

     ```
     $ VBoxManage clonevm protocols-jeos-<date>-x86_64 --groups=/
     LinuxNetwork2025 --name=CLONE_NAME --options=Link --
     snapshot=SNAPSHOT_NAME --register
     ```

     Пример выполнения команды:

     ```
     [papillon_rouge@localhost ~]$ VBoxManage clonevm protocols-jeos-20250216-
     x86_64 --groups=/LinuxNetwork2025 --name=srv --options=Link --
     snapshot=srv_snapshot --register
     0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
     Machine has been successfully cloned as "srv"
     ```

3. Для настройки COM-порта клона используется команда:

   ```
   $ VBoxManage modifyvm CLONE_NAME --uartmode1 tcpserver 2026
   ```

   Пример выполнения:

   ```
   [papillon_rouge@localhost ~]$ VBoxManage modifyvm srv --uartmode1 tcpserver
   2026
   ```

Итогом настройки с помощью командной строки будет состояние, аналогичное полученному при использовании графического интерфейса.

2.2. Подключение к виртуальным машинам

После создания виртуальных машин необходимо их включить и к ним подключиться. Обычный запуск виртуальных машин автоматически открывает эмулятор экрана виртуальной машины, установленный в **`VirtualBox`**. Однако из-за особенностей настройки он не позволяет производить копирование из основной системы в интерфейс виртуальной машины и, соответственно, копировать данные между виртуальными машинами с помощью буфера обмена.

Для решения этой проблемы предлагается использовать **`Shell`**-сценарий **`vbconnect`**, который автоматически производит включение и подключение к виртуальной машине через командную строку.

Для запуска необходимо:

1. Выдать сценарию права на выполнение:

   ```
   chmod +x vbconnect
   ```

2. Запустить сценарий с указанием имени запускаемой виртуальной машины:

   ```
   ./vbconnect <NAME>
   ```

Полное описание возможностей сценария доступно по *ссылке*.

Глава 3. Знакомство с системой

Цель лабораторной — познакомить изучающего с системой

Задачи лабораторной:

- Изучить логику работы последовательных портов;
- Изучить логику работы сетевых интерфейсов;
- Изучить алгоритм генерации отчёта по лабораторным;

3.1. Виртуальные машины VirtualBox

Каждая виртуальная машина по умолчанию имеет 4 *COM-порта* и 4 *сетевых адаптера* для организации подключения. При этом поддерживается до 36 псевдофизических (поскольку все интерфейсы ВМ на самом деле являются, очевидно, виртуальными; далее мы будем называть эти интерфейсы просто физическими, а виртуальными будем именовать программно реализованные внутри ВМ интерфейсы) и любое количество виртуальных интерфейсов.

Последовательные порты (COM‑порты) представляют из себя аппаратные интерфейсы, позволяющие напрямую передавать информацию между абонентами на физическом уровне стека протоколов TCP/IP. COM‑порты обозначаются в файловой системе в виде файлов специального типа, находящихся в директории **`/dev`**. Имена COM‑портов обозначаются **`ttyS0–ttyS3`** для портов 1–4 соответственно.

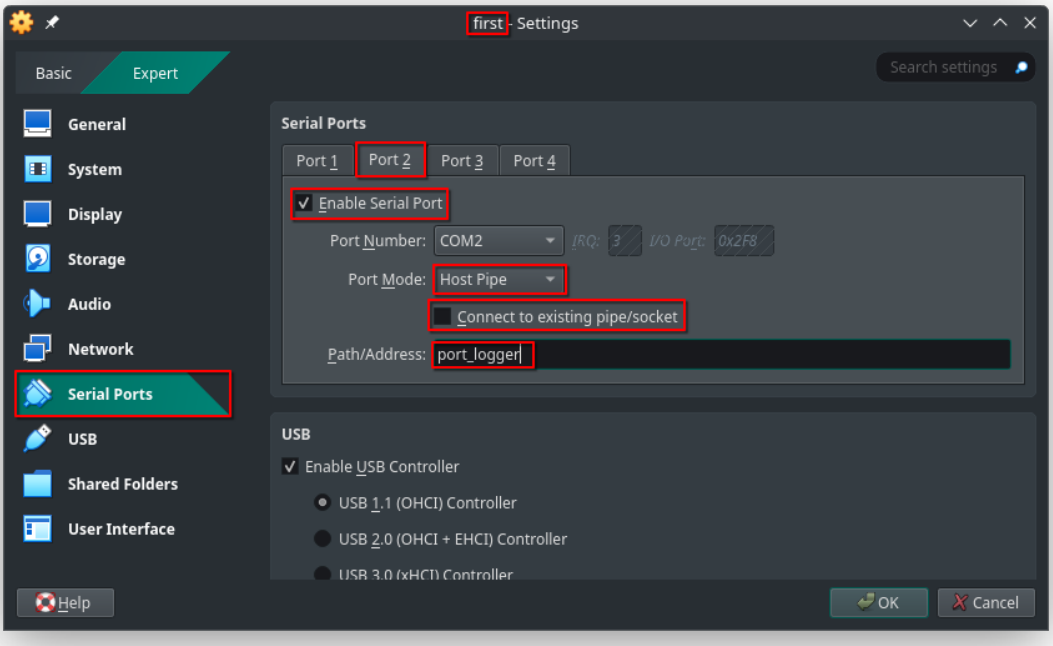
Сетевые интерфейсы представлены в виде физических Ethernet‑портов, позволяющих организовывать соединение устройств на интерфейсном и сетевом уровнях стека протоколов. Для работы с разного рода сетевыми настройками используется утилита **iproute2** (утилита предустановлена в образе виртуальной машины).

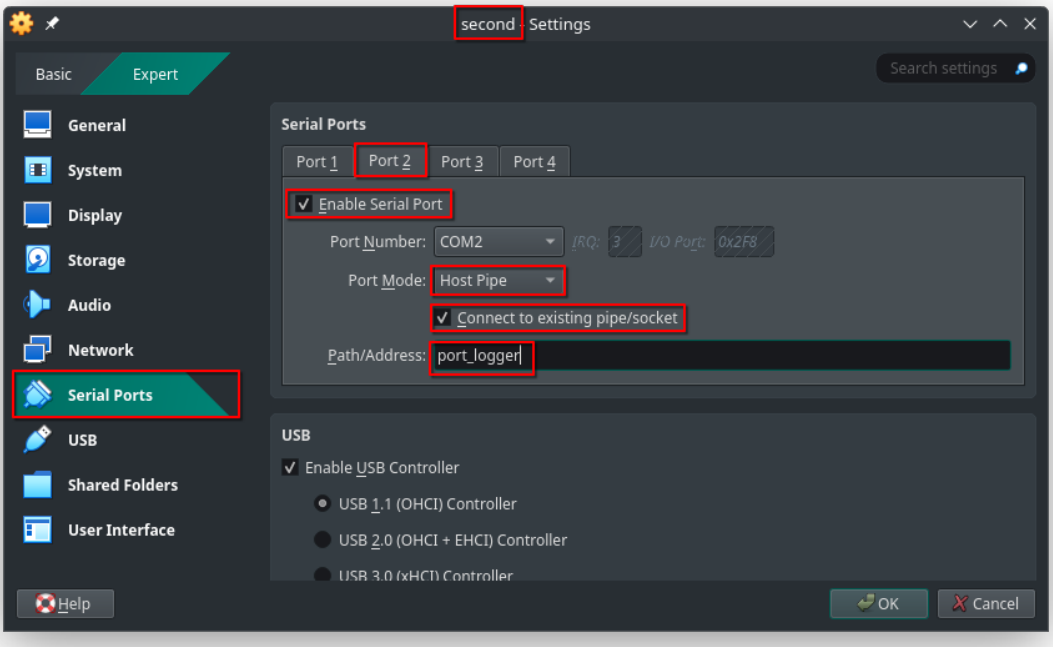
3.2. Работа с последовательным портами

Для настройки канала связи между двумя устройствами через COM‑порты используется сокетное соединение (в VirtualBox оно называется **`Host Pipe`**).

3.2.1. Создание COM-соединения

1. Создайте два клона с именами **`first`** и **`second`** согласно инструкции по клонированию.
2. В настройках виртуальных машин подключите соединение на COM‑порт 2:
   - Для одного устройства укажите настройку создания канала (убрать галочку);
   - Для второго устройства укажите только подключение к нему (оставить галочку).





3. Запустите виртуальные машины. Для входа в систему каждой машины необходимо ввести логин (**root**) и пароль (**root**).

Важно

Никогда не устанавливайте такие легко угадываемые пароли на свои персональные устройства!

3.2.2. Настройка имени пользователя

После запуска имя каждой виртуальной машины — **localhost**. Для переименования виртуальной машины используется специальная команда **sethostname** (при смене имени виртуальной машины требуется повторный вход в систему):

1. С помощью команды **sethostname** измените имена пользователей на **first** и **second** соответственно

```
localhost login: root
Password:
[root@localhost ~]# sethostname first

first login: root
Password:
Last login: Fri Sep  5 13:09:38 UTC 2025 on ttyS0
[root@first ~]#
```

```
localhost login: root
Password:
[root@localhost ~]# sethostname second
```

```
second login: root
Password:
Last login: Fri Sep  5 13:12:46 UTC 2025 on ttyS0
[root@second ~]#
```

3.2.3. Передача данных через COM-порты

Внутри виртуальной машины управление портами осуществляется с помощью команды **stty**. Без управляющих параметров она позволяет посмотреть текущие параметры портов. Для просмотра конкретного порта можно использовать перенаправление на нужный порт.

1. С помощью команды **stty -a < <dev-name>** получите информацию о параметрах COMфпорта №2 (**ttyS1**)

```
[root@first ~]# stty -a < /dev/ttyS1
speed 9600 baud; rows 0; columns 0; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>; eol2 =
<undef>;
swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V;
discard = ^O; min = 1; time = 0;
-parenb -parodd -cmspar cs8 hupcl -cstopb cread clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon
-ixoff -iuclc -ixany
-imaxbel -iutf8
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0
ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
echoctl echoke -flusho
-extproc
[root@first ~]#
```

   Для соединения устройств необходимо подготовить порты к передаче данных: включить режим **raw** для передачи просто последовательности байт, а также отключить дублирование данных на экран:

2. С помощью команды **stty raw -echo < <dev-name>** подготовьте COMфпорты для передачи

```
[root@first ~]# stty raw -echo < /dev/ttyS1
[root@first ~]#
```

```
[root@second ~]# stty raw -echo < /dev/ttyS1
[root@second ~]#
```

3. Запустите команду **cat <dev-name>** на **first** для получения данных с порта

```
[root@first ~]# cat /dev/ttyS1
```

4. С **second** передайте через порт календарь

```
[root@second ~]# cal > /dev/ttyS1
[root@second ~]#
```

   Заметьте, что календарь будет прочитан в **first**:

```
[root@first ~]# cat /dev/ttyS1
  September 2025
Su Mo Tu We Th Fr Sa
   1  2  3  4  5  6
7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

3.3. Работа с сетевыми интерфейсами

3.3.1. Настройка eth-соединения

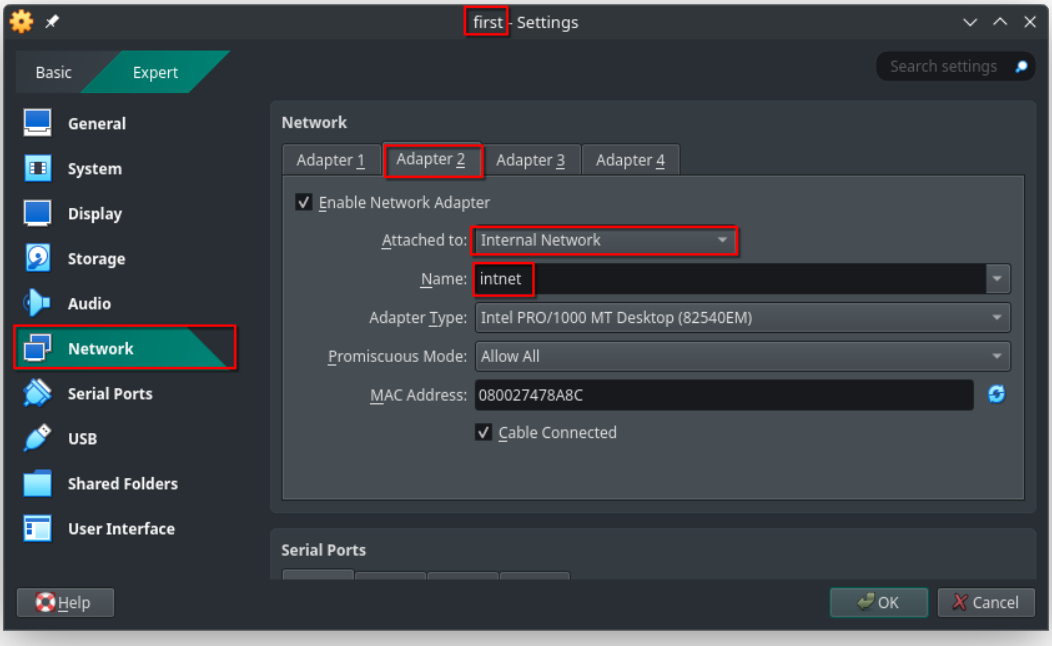
Предупреждение

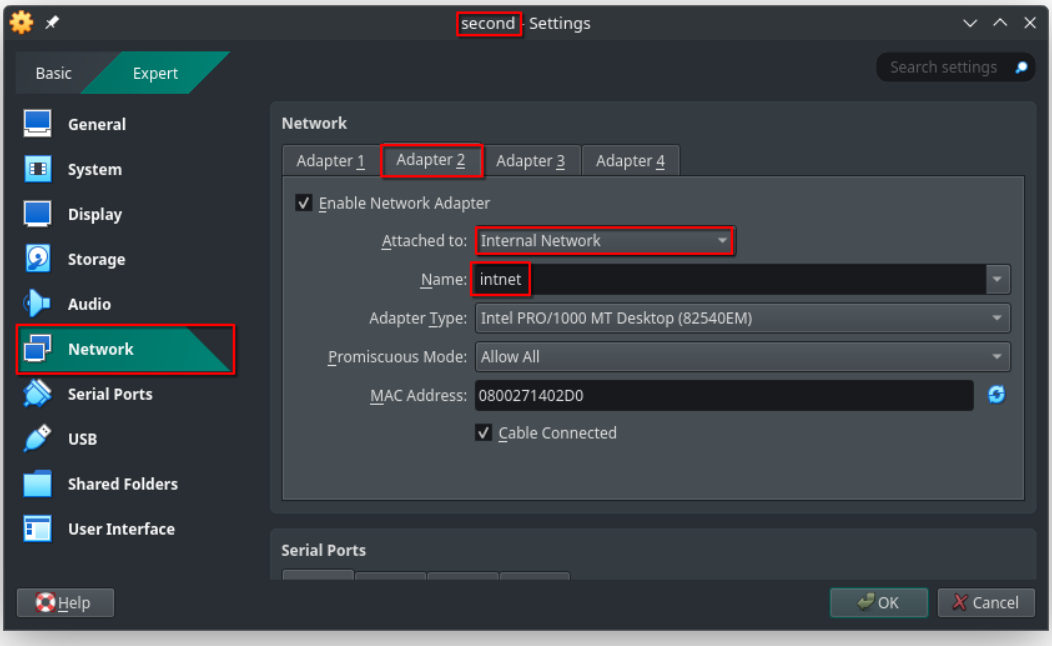
Для изменения параметров виртуальных машин их необходимо предварительно отключить (например, командой **`poweroff`** в командной строке или из графического интерфейса самого VirtualBox)

1. В настройках виртуальных машин в разделе «Network» выберите пункт «Internal Network» и укажите имя для новой сети.

Примечание

При указании у разных виртуальных машин одной и той же сети они автоматически соединяются через виртуальную ethernet-сеть и могут взаимодействовать в рамках этой сети (данное соединение аналогично соединению машин через коммутатор).





Примечание

Заметьте, что автоматически для разных виртуальных машин сгенерировались разные MAC-адреса.

2. Запустите виртуальные машины

3.3.2. Настройка сетевых интерфейсов

1. С помощью команды **`ip link`** покажите все сетевые интерфейсы на **`first`**:

```
[root@first ~]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
DEFAULT group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
   link/ether 08:00:27:77:45:6f brd ff:ff:ff:ff:ff:ff
   altname enp0s3
   altname enx08002777456f
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
   link/ether 08:00:27:47:8a:8c brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx080027478a8c
4: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
   link/ether 08:00:27:45:3d:0d brd ff:ff:ff:ff:ff:ff
   altname enp0s9
   altname enx080027453d0d
5: eth3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
```

```
    link/ether 08:00:27:7a:79:33 brd ff:ff:ff:ff:ff:ff
    altname enp0s10
    altname enx0800277a7933
[root@first ~]#
```

Система показывает 5 интерфейсов (4 физических и один логический **loopback**, для которого отправка в него вызывает приём данных из него же). Все интерфейсы на данный момент отключены.

2. С помощью **команды управления интерфейсом `:new: ip link set eth1 up`** включите интерфейс с сетью **intnet**

Примечание

Здесь и далее в лабораторных знаком **`:new:`** будет обозначаться вводимая в рамках лабораторных терминология. В будущем для команд, описываемых данной терминологией, не будет писаться сама команда, а будет использоваться упоминание термина. Список терминов и связанные с ним команды представлены на заглавной странице лабораторных

3. Покажите параметры интерфейса с помощью команды **`ip link show eth1`**:

```
[root@first ~]# ip link set eth1 up
[root@first ~]# ip link show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
mode DEFAULT group default q
len 1000
    link/ether 08:00:27:47:8a:8c brd ff:ff:ff:ff:ff:ff
    altname enp0s8
    altname enx080027478a8c
[root@first ~]#
```

Статус интерфейса стал **UP**, соединение включено.

4. С помощью **команды настройки IPv4адресов `:new: ip addr add dev <interface> <IPv4>/<mask>`** добавьте на интерфейс IPv4адрес

5. Покажите параметры интерфейса **с сетью** с помощью команды **`ip addr show eth1`**:

```
[root@first ~]# ip addr add dev eth1 10.9.0.1/24

[root@first ~]# ip addr show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 08:00:27:47:8a:8c brd ff:ff:ff:ff:ff:ff
    altname enp0s8
    altname enx080027478a8c
    inet 10.9.0.1/24 scope global eth1
       valid_lft forever preferred_lft forever
[root@first ~]#
```

6. С помощью команд управления интерфейсом и команд настройки IPv4адресов аналогично настройте **second** (При выборе IPv4адреса номер сети для всех абонентов одной сети должен быть одинаковым)

```
[root@second ~]# ip link set eth1 up
[root@second ~]# ip addr add dev eth1 10.9.0.2/24
[root@second ~]# ip addr show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
   link/ether 08:00:27:14:02:d0 brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx0800271402d0
   inet 10.9.0.2/24 scope global eth1
      valid_lft forever preferred_lft forever
[root@second ~]#
```

3.3.3. Передача данных через сетевые интерфейсы

Сгенерируем трафик для проверки соединения. Самый простой способ генерации трафика — команда **`ping -c<количество отправляемых пакетов> <dstIP>`**.

1. С помощью команды **`ping -c5 <dstIP>`** сгенерируйте ICMP-трафик между абонентами

```
[root@first ~]# ping -c5 10.9.0.2
PING 10.9.0.2 (10.9.0.2) 56(84) bytes of data.
64 bytes from 10.9.0.2: icmp_seq=1 ttl=64 time=0.948 ms
64 bytes from 10.9.0.2: icmp_seq=2 ttl=64 time=0.577 ms
64 bytes from 10.9.0.2: icmp_seq=3 ttl=64 time=0.784 ms
64 bytes from 10.9.0.2: icmp_seq=4 ttl=64 time=0.711 ms
64 bytes from 10.9.0.2: icmp_seq=5 ttl=64 time=0.751 ms

--- 10.9.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 5105ms
rtt min/avg/max/mdev = 0.577/0.749/0.948/0.109 ms
[root@first ~]#
```

   Для отслеживания трафика могут использоваться специальные утилиты, сканирующие сетевые интерфейсы и считывающие проходящий через них трафик.

2. С помощью команды **`tcpdump -i eth1 -c2`** запустите на **`second`** сканирование интерфейса

```
[root@second ~]# tcpdump -i eth1 -c2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

3. С помощью команды **`ping -c1 <ip address>`** выполните одиночный **`ping`**-запрос на **`first`**

```
[root@first ~]# ping -c1 10.9.0.2
PING 10.9.0.2 (10.9.0.2) 56(84) bytes of data.
64 bytes from 10.9.0.2: icmp_seq=1 ttl=64 time=0.570 ms

--- 10.9.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.570/0.570/0.570/0.000 ms
[root@first ~]#
```

   Убедитесь, что данные о ping-сообщении были считаны:

```
[root@second ~]# tcpdump -i eth1 -c2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
15:27:43.398843 IP 10.9.0.1 > second: ICMP echo request, id 3, seq 1, length
64
15:27:43.398872 IP second > 10.9.0.1: ICMP echo reply, id 3, seq 1, length
64
2 packets captured
2 packets received by filter
0 packets dropped by kernel
[root@second ~]#
```

3.4. Сдача самостоятельных работ

Для сдачи самостоятельных работ используется специальная утилита **report**, создающая tar-архив с записью команд в системе.

1. С помощью команды **report <#Lab> <hostname>** запустите запись отчёта на обеих машинах:

```
[root@first ~]# report 3 first
Report is started. Task: 03; host: first.
Use exit or ^D to finish.
[root@01-first ~]#
```

```
[root@second ~]# report 3 second
Report is started. Task: 03; host: second.
Use exit or ^D to finish.
[root@01-second ~]#
```

   Затем необходимо будет выполнять указанные в самостоятельной работе команды. Сейчас выполним на каждой виртуальной машине следующие команды:

   - **ip a show eth1**
   - **ping -c5 <адрес другого абонента>**

2. На каждой машине выполните описанные команды в режиме **report**. Затем завершите запись с помощью команды **exit** или сочетания клавиш **Ctrl+D**.

```
[root@01-first ~]# ip a show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
   link/ether 08:00:27:da:00:dd brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx080027da00dd
   inet 10.9.0.1/24 scope global eth1
      valid_lft forever preferred_lft forever
[root@01-first ~]# ping -c5 10.9.0.2
PING 10.9.0.2 (10.9.0.2) 56(84) bytes of data.
64 bytes from 10.9.0.2: icmp_seq=1 ttl=64 time=0.531 ms
64 bytes from 10.9.0.2: icmp_seq=2 ttl=64 time=0.635 ms
64 bytes from 10.9.0.2: icmp_seq=3 ttl=64 time=0.712 ms
64 bytes from 10.9.0.2: icmp_seq=4 ttl=64 time=0.558 ms
64 bytes from 10.9.0.2: icmp_seq=5 ttl=64 time=0.927 ms
```

```
--- 10.9.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4132ms
rtt min/avg/max/mdev = 0.531/0.672/0.927/0.142 ms
[root@01-first ~]#
<^D>exit
Report is stopped.
[root@first ~]#
```

```
[root@01-second ~]# ip a show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
   link/ether 08:00:27:e1:50:ee brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx080027e150ee
   inet 10.9.0.2/24 scope global eth1
      valid_lft forever preferred_lft forever
[root@01-second ~]# ping -c5 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
64 bytes from 10.9.0.1: icmp_seq=1 ttl=64 time=0.648 ms
64 bytes from 10.9.0.1: icmp_seq=2 ttl=64 time=0.780 ms
64 bytes from 10.9.0.1: icmp_seq=3 ttl=64 time=0.832 ms
64 bytes from 10.9.0.1: icmp_seq=4 ttl=64 time=0.569 ms
64 bytes from 10.9.0.1: icmp_seq=5 ttl=64 time=0.758 ms

--- 10.9.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 0.569/0.717/0.832/0.095 ms
[root@01-second ~]# exit
exit
Report is stopped.
[root@second ~]#
```

После записи в домашнем каталоге появится файл **report.<#Lab>.<hostname>**. С помощью **ls -l** можно убедиться в предварительной корректности записи (число после слова **root** показывает размер файла, он должен быть ненулевой), а с помощью команды **report <report-name>** можно «проиграть» запись команд.

3. С помощью команды **report <report-name>** запустите получившиеся отчёты.

```
[root@first ~]# ls -l
total 12
drwxr-xr-x 2 root root 4096 May 12 17:18 bin
-rw-r--r-- 1 root root 2621 Oct 13 12:28 report.03.first
drwx------ 2 root root 4096 May 12 17:19 tmp

[root@first ~]# report report.03.first
Replay report.03.first
[root@01-first ~]# ip a show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
   link/ether 08:00:27:da:00:dd brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx080027da00dd
   inet 10.9.0.1/24 scope global eth1
      valid_lft forever preferred_lft forever
[root@01-first ~]# ping -c5 10.9.0.2
PING 10.9.0.2 (10.9.0.2) 56(84) bytes of data.
```

```
64 bytes from 10.9.0.2: icmp_seq=1 ttl=64 time=0.531 ms
64 bytes from 10.9.0.2: icmp_seq=2 ttl=64 time=0.635 ms
64 bytes from 10.9.0.2: icmp_seq=3 ttl=64 time=0.712 ms
64 bytes from 10.9.0.2: icmp_seq=4 ttl=64 time=0.558 ms
64 bytes from 10.9.0.2: icmp_seq=5 ttl=64 time=0.927 ms

--- 10.9.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4132ms
rtt min/avg/max/mdev = 0.531/0.672/0.927/0.142 ms
[root@01-first ~]#
exit

Replay report.03.first finished
[root@first ~]#
```

```
[root@second ~]# ls -l
total 12
drwxr-xr-x 2 root root 4096 May 12 17:18 bin
-rw-r--r-- 1 root root 2498 Oct 13 12:29 report.03.second
drwx------ 2 root root 4096 May 12 17:19 tmp
[root@second ~]# report report.03.second
Replay report.03.second
[root@01-second ~]# ip a show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
   link/ether 08:00:27:e1:50:ee brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx080027e150ee
   inet 10.9.0.2/24 scope global eth1
      valid_lft forever preferred_lft forever
[root@01-second ~]# ping -c5 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
64 bytes from 10.9.0.1: icmp_seq=1 ttl=64 time=0.648 ms
64 bytes from 10.9.0.1: icmp_seq=2 ttl=64 time=0.780 ms
64 bytes from 10.9.0.1: icmp_seq=3 ttl=64 time=0.832 ms
64 bytes from 10.9.0.1: icmp_seq=4 ttl=64 time=0.569 ms
64 bytes from 10.9.0.1: icmp_seq=5 ttl=64 time=0.758 ms

--- 10.9.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 0.569/0.717/0.832/0.095 ms
[root@01-second ~]# exit
exit

Replay report.03.second finished
[root@second ~]#
```

Следующим шагом необходимо перенести полученные отчёты из виртуальных машин в основную систему.

Предупреждение

Настройка порта для передачи данных может быть проведена ещё *при стартовой настройке виртуальных машин*, но в таком случае выводы некоторых команд могут сильнее отличаться от показанных в лабораторных.

Поскольку настройка проводится над виртуальными машинами, для её проведения необходимо выключить машины, используя команду **poweroff**.

4. Выключите машины с помощью команды **poweroff** или передачей сигнала о завершении работы из VirtualBox.

Важно

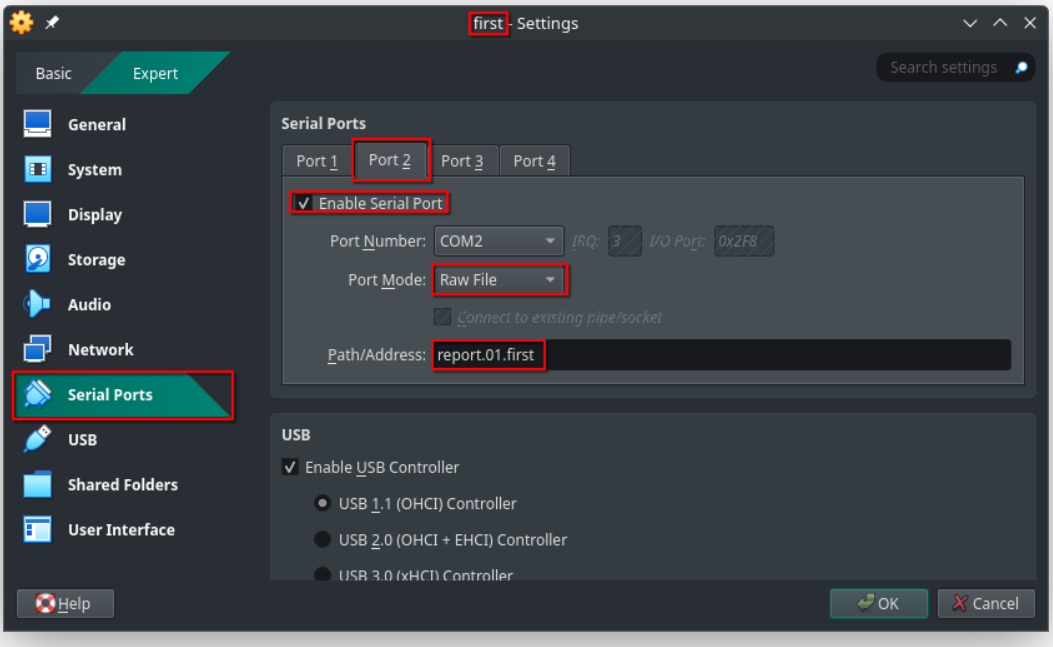
Не выключайте машины «отключением питания» (пункт **power off the machine** при выключении через VirtualBox) во избежание ошибок в сохранении данных.

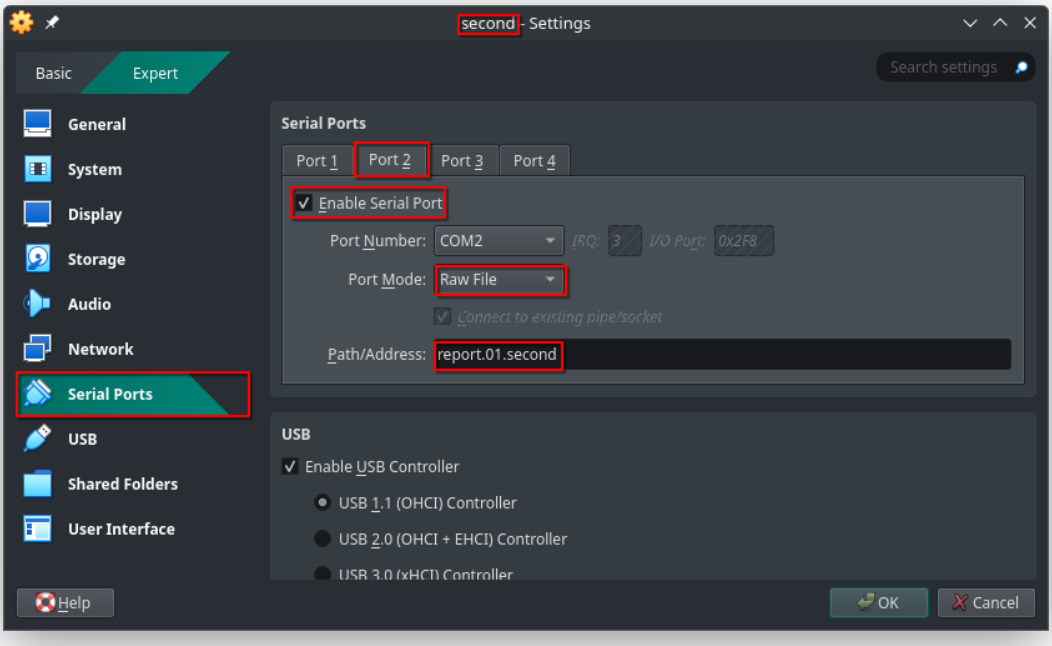
```
[root@first ~]# poweroff
----------------------------------------
[root@second ~]# poweroff
```

Для передачи используется последовательный порт (COM#порт), настраиваемый на запись в файл (**Port Mode: Raw File**). В качестве названия выводимого файла необходимо указать соответствующее имя файла **report.<#Lab>.<hostname>**.

**Имена файлов внутри ВМ и в основной системе, вообще говоря, не обязаны быть одинаковыми. Но для удобства проверки мы просим использовать те же имена.*

5. Настройте на устройствах COM#порты для передачи данных из виртуальных машин.





6. Вновь запустите виртуальные машины.

   Для передачи данных, аналогично работе с виртуальными каналами, необходимо настроить порты для передачи, после чего передать на них файлы/отчёты. По итогу команды для передачи всегда будут одинаковыми (не забывайте пользоваться автодополнением через **Tab** для записи длинных команд):

   a. **`stty raw -echo < /dev/ttyS1`**

   b. **`cat > report.<#Lab>.<hostname> > /dev/ttyS1`**

7. С помощью описанных выше команд перешлите файлы отчётов с виртуальных машин в вашу систему.

```
[root@first ~]# ls
bin  report.03.first  tmp
[root@first ~]# stty raw -echo < /dev/ttyS1
[root@first ~]# cat report.03.first > /dev/ttyS1
[root@first ~]# poweroff
```

```
[root@second ~]# ls
bin  report.03.second  tmp
[root@second ~]# stty raw -echo < /dev/ttyS1
[root@second ~]# cat report.03.second > /dev/ttyS1
[root@second ~]# poweroff
```

   После этого в вашей системе появятся соответствующие файлы. Для проверки корректности убедитесь, что полученные файлы *ненулевого* объёма (в среднем — несколько килобайт).

   В случае использования собственной системы **`Linux`** можно установить утилиту **`report`** из репозитория и проверять корректность записанного файла также прогоном в утилите:

```
[USER@USERsPC] ls | grep report
report.03.first
report.03.second
[USER@USERsPC] report report.03.first
tar: ./CPU.txt: time stamp 2025-10-13 15:25:57 is 8649.335340872 s in the
future
tar: ./IN.txt: time stamp 2025-10-13 15:28:02 is 8774.335237496 s in the
future
tar: ./BOTH.txt: time stamp 2025-10-13 15:28:02 is 8774.335184105 s in the
future
tar: ./TIME.txt: time stamp 2025-10-13 15:28:02 is 8774.335138278 s in the
future
tar: ./OUT.txt: time stamp 2025-10-13 15:28:02 is 8774.335085218 s in the
future
tar: .: time stamp 2025-10-13 15:25:57 is 8649.334813324 s in the future
Replay report.03.first
[root@01-first ~]# ip a show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
   link/ether 08:00:27:da:00:dd brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx080027da00dd
   inet 10.9.0.1/24 scope global eth1
      valid_lft forever preferred_lft forever
[root@01-first ~]# ping -c5 10.9.0.2
PING 10.9.0.2 (10.9.0.2) 56(84) bytes of data.
64 bytes from 10.9.0.2: icmp_seq=1 ttl=64 time=0.531 ms
64 bytes from 10.9.0.2: icmp_seq=2 ttl=64 time=0.635 ms
64 bytes from 10.9.0.2: icmp_seq=3 ttl=64 time=0.712 ms
64 bytes from 10.9.0.2: icmp_seq=4 ttl=64 time=0.558 ms
64 bytes from 10.9.0.2: icmp_seq=5 ttl=64 time=0.927 ms

--- 10.9.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4132ms
rtt min/avg/max/mdev = 0.531/0.672/0.927/0.142 ms
[root@01-first ~]#
exit
Replay report.03.first finished
[USER@USERsPC]
```

3.5. Самостоятельная работа

В качестве отчёта о проделанной лабораторной пришлите преподавателю два отчёта *report.03.first* и *report.03.second* с описанными в последнем разделе командами.

Глава 4. Основные утилиты и команды просмотра настроек и мониторинга сети

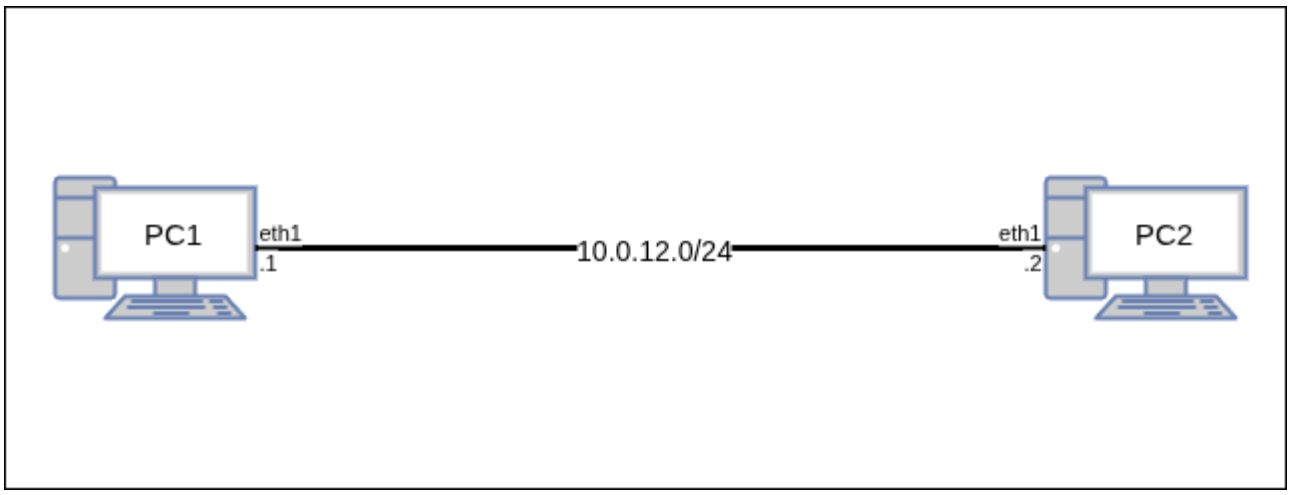
Цель лабораторной работы — познакомить изучающего с основными утилитами и командами просмотра настроек и мониторинга сети.

Задачи лабораторной работы:

- Изучить логику работы последовательных портов;
- Изучить логику работы сетевых интерфейсов;
- Изучить алгоритм генерации отчёта по лабораторным работам.

4.1. Настройка топологии

Для изучения настроек рассмотрим базовую топологию из двух устройств:



Для работы создайте 2 *клона* согласно топологии сети. Для создания соединений между машинами необходимо в VirtualBox настроить сетевые интерфейсы (описание настройки подключения находится в соответствующем разделе второй лабораторной):

- **PC1**:
  - **Adapter2** — **intnet**
- **PC2**:
  - **Adapter2** — **intnet**

4.2. Работа с сетевыми интерфейсами

4.2.1. Просмотр сетевых интерфейсов

Все сетевые интерфейсы, представленные в системе, можно посмотреть с помощью *команды просмотра интерфейса* :new: **`ip link`** или **`ip link show`**. Здесь показывается основная информация о состоянии интерфейса и его базовых параметрах. Для отслеживание конкретного интерфейса необходимо явно указывать его в команде **`ip link show <interface>`**. Для получения расширенной информации необходимо использовать ключ **`-d`** в команде (ВАЖНО: *после* слова **`ip`**, *до* всех остальных слов в команде).

1. С помощью команды **`ip link`** на PC1 выведите информацию о доступных сетевых интерфейсах.

```
[root@PC1 ~]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
DEFAULT group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
   link/ether 08:00:27:f5:b6:8c brd ff:ff:ff:ff:ff:ff
   altname enp0s3
   altname enx080027f5b68c
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
   link/ether 08:00:27:04:e4:05 brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx08002704e405
4: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
   link/ether 08:00:27:16:43:2e brd ff:ff:ff:ff:ff:ff
   altname enp0s9
   altname enx08002716432e
5: eth3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
   link/ether 08:00:27:8b:a9:e1 brd ff:ff:ff:ff:ff:ff
   altname enp0s10
   altname enx0800278ba9e1
```

2. С помощью команды **`ip link show <interface>`** на PC1 выведите информацию об интерфейсе eth1.

```
[root@PC1 ~]# ip link show eth1
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
   link/ether 08:00:27:04:e4:05 brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx08002704e405
```

3. С помощью команды **`ip -d link show <interface>`** на PC1 выведите подробную информацию об интерфейсе lo.

```
[root@PC1 ~]# ip -d link show lo
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
DEFAULT group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 promiscuity 0
allmulti 0 minmtu 0 maxmtu 0 numtxqueues 1 numrxqueues 1 gso_max_size 65536
gso_max_segs 65535 tso_max_size 524280 tso_max_segs 65535 gro_max_size 65536
gso_ipv4_max_size 65536 gro_ipv4_max_size 65536
```

4.2.2. Управление сетевыми интерфейсами

Изначально все доступные интерфейсы находятся в выключенном состоянии. Для включения интерфейса используется команда **`ip link set <interface> UP`**. В панели интерфейсов при этом меняется состояние (поле **`state`**) интерфейса.

1. С помощью команды управления и просмотра интерфейсов посмотрите на PC1 состояние интерфейса eth1, затем включите интерфейс и вновь посмотрите его состояние.

```
[root@PC1 ~]# ip link show eth1
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
   link/ether 08:00:27:04:e4:05 brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx08002704e405

[root@PC1 ~]# ip link set eth1 up

[root@PC1 ~]# ip link show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
mode DEFAULT group default
qlen 1000
   link/ether 08:00:27:04:e4:05 brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx08002704e405
[root@PC1 ~]#
```

4.3. Работа с IP-адресами

4.3.1. Просмотр IP-адресов

Для просмотра IP-адресов, связанных с интерфейсами устройства, используется команда *просмотра интерфейсов с описанием установленных IP-адресов* :new: **`ip addr`** (**`ip a`**, **`ip addr show`**, **`ip a show`**). Вывод представляет собой вывод информации о интерфейсах (**`ip link`**), дополненный информацией об IP-адресах. Команда также поддерживает обращение к конкретному интерфейсу и флаг **`-d`** для просмотра расширенных настроек.

1. С помощью команды **`ip addr`** на PC2 выведите информацию о всех интерфейсах с описанием установленных IP-адресов

```
[root@PC2 ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default 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
```

```
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
   link/ether 08:00:27:37:4f:eb brd ff:ff:ff:ff:ff:ff
   altname enp0s3
   altname enx080027374feb
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
   link/ether 08:00:27:ee:00:c5 brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx080027ee00c5
4: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
   link/ether 08:00:27:f4:dc:7b brd ff:ff:ff:ff:ff:ff
   altname enp0s9
   altname enx080027f4dc7b
5: eth3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
   link/ether 08:00:27:f7:0d:ae brd ff:ff:ff:ff:ff:ff
   altname enp0s10
   altname enx080027f70dae
```

2. С помощью команды **`ip addr show <interface>`** на PC2 выведите информацию об интерфейсе eth1 с описанием установленных IP-адресов

```
[root@PC2 ~]# ip addr show eth1
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
   link/ether 08:00:27:ee:00:c5 brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx080027ee00c5
```

3. С помощью команды **`ip -d addr show <interface>`** на PC2 выведите подробную информацию об интерфейсе eth1 с описанием установленных IP-адресов

```
[root@PC2 ~]# ip -d a show eth1
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
   link/ether 08:00:27:ee:00:c5 brd ff:ff:ff:ff:ff:ff promiscuity 0 allmulti
0 minmtu 46 maxmtu 16110
numtxqueues 1 numrxqueues 1 gso_max_size 65536 gso_max_segs 65535
tso_max_size 65536 tso_max_segs 65535
gro_max_size 65536 gso_ipv4_max_size 65536 gro_ipv4_max_size 65536 parentbus
pci parentdev 0000:00:08.
0
   altname enp0s8
   altname enx080027ee00c5
[root@PC2 ~]#
```

Заметьте, что по умолчанию только LoopBack-интерфейс содержит предустановленный IP-адрес 127.0.0.1 для использования исключительно в качестве локального сетевого адреса.

4.3.2. Управление IP-адресами

Добавление нового адреса на заданный интерфейс производится командой настройки IPѣадресов **`ip addr add dev <interface> <IPv4>/<mask>`**. Заметьте, что в выводе команды **`ip a show eth1`** после добавления IPѣадреса поле **`scope`** имеет значение **`global`** (в отличие от **`scope host`** на LoopBackѣе); это показатель возможности использования данного адреса для сетевого взаимодействия через него с другими абонентами сети.

1. С помощью команды настройки IPѣадресов установите на PC1 IPѣадрес на интерфейсе eth1

```
[root@PC1 ~]# ip addr add dev eth1 10.0.12.1/24
[root@PC1 ~]# ip a show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
   link/ether 08:00:27:04:e4:05 brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx08002704e405
   inet 10.0.12.1/24 scope global eth1
      valid_lft forever preferred_lft forever
[root@PC1 ~]#
```

2. С помощью команд настройки IPѣадресов установите на PC2 на интерфейсе eth1 IPѣадрес из той же сети и выведите информацию о настроенных IPѣадресах

```
[root@PC2 ~]# ip addr add dev eth1 10.0.12.2/24
[root@PC2 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default 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
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
   link/ether 08:00:27:37:4f:eb brd ff:ff:ff:ff:ff:ff
   altname enp0s3
   altname enx080027374feb
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
   link/ether 08:00:27:ee:00:c5 brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx080027ee00c5
   inet 10.0.12.2/24 scope global eth1
      valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
   link/ether 08:00:27:f4:dc:7b brd ff:ff:ff:ff:ff:ff
   altname enp0s9
   altname enx080027f4dc7b
5: eth3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
   link/ether 08:00:27:f7:0d:ae brd ff:ff:ff:ff:ff:ff
   altname enp0s10
   altname enx080027f70dae
[root@PC2 ~]#
```

4.3.3. Проверка настройки сети

Для создания трафика в сети самым простым способом является команда **`ping <dstIP>`**, отправляющая ICMPэхопакеты и получающая на них ответы, что и является подтверждением корректного двунаправленного соединения. Для работы удобно использовать флаги: **`-c<N>`** для регулировки количества отправляемых сообщений (∞ по умолчанию), **`-f`** для мгновенной отправки сразу пачки пакетов (удобно комбинировать с **`-c`**: например, **`ping -fc3 <dstIP>`** мгновенно отправит три ICMPэхопакета).

1. С помощью команды **`ping -c3 <dstIP>`** отправьте три ICMPэхопакета с PC1 на PC2

   ```
   [root@PC1 ~]# ping -c3 10.0.12.2
   PING 10.0.12.2 (10.0.12.2) 56(84) bytes of data.
   From 10.0.12.1 icmp_seq=1 Destination Host Unreachable
   From 10.0.12.1 icmp_seq=2 Destination Host Unreachable
   From 10.0.12.1 icmp_seq=3 Destination Host Unreachable

   --- 10.0.12.2 ping statistics ---
   3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2077ms
   pipe 3
   [root@PC1 ~]#
   ```

   Поскольку интерфейс **`eth1`** на **`PC2`** отключён, при попытке передачи данных между абонентами будет появляться ошибка **`From <srcIP> icmp_seq=<N> Destination Host Unreachable`**. Ошибка показывает, что пакеты успешно отправляются с устройства, однако достигнуть получателя не могут.

2. С помощью команд управления интерфейсами включите на PC2 интерфейс eth1

   ```
   [root@PC2 ~]# ip link set eth1 up
   [root@PC2 ~]#
   ```

3. С помощью команды **`ping -c3 <dstIP>`** повторно отправьте три ICMPэхопакета с PC1 на PC2

   ```
   [root@PC1 ~]# ping -c3 10.0.12.2
   PING 10.0.12.2 (10.0.12.2) 56(84) bytes of data.
   64 bytes from 10.0.12.2: icmp_seq=1 ttl=64 time=0.898 ms
   64 bytes from 10.0.12.2: icmp_seq=2 ttl=64 time=0.435 ms
   64 bytes from 10.0.12.2: icmp_seq=3 ttl=64 time=0.365 ms
   --- 10.0.12.2 ping statistics ---
   3 packets transmitted, 3 received, 0% packet loss, time 2044ms
   rtt min/avg/max/mdev = 0.365/0.566/0.898/0.236 ms
   [root@PC1 ~]#
   ```

4.4. Работа с Таблицами маршрутизации

Для определения выходного интерфейса при отправке или промежуточной маршрутизации пакета используются *таблицы маршрутизации*. На одном устройстве может быть несколько таблиц.

Встроенных по умолчанию таблиц две:

- **`main`** — используется для описания внешних маршрутов;

- **local** — для описания локальных маршрутов (в том числе broadcast маршрутов).

С помощью специальных команд возможно создать любое количество дополнительных таблиц.

4.4.1. Просмотр таблиц маршрутизации

Посмотреть таблицу маршрутизации можно с помощью команды *просмотра таблиц маршрутизации* :new: **ip route** (**ip route list**). Для просмотра конкретной таблицы используется команда **ip route list table <table-name>**. Для просмотра всех таблиц используется **table-name=all**.

1. С помощью команд просмотра таблиц маршрутизации на PC2 выведите базовую информацию таблиц маршрутизации, две основные таблицы маршрутизации (main и local) и все таблицы маршрутизации.

```
[root@PC2 ~]# ip route
10.0.12.0/24 dev eth1 proto kernel scope link src 10.0.12.2

[root@PC2 ~]# ip route list table main
10.0.12.0/24 dev eth1 proto kernel scope link src 10.0.12.2

[root@PC2 ~]# ip route list table local
local 10.0.12.2 dev eth1 proto kernel scope host src 10.0.12.2
broadcast 10.0.12.255 dev eth1 proto kernel scope link src 10.0.12.2
local 127.0.0.0/8 dev lo proto kernel scope host src 127.0.0.1
local 127.0.0.1 dev lo proto kernel scope host src 127.0.0.1
broadcast 127.255.255.255 dev lo proto kernel scope link src 127.0.0.1

[root@PC2 ~]# ip route list table all
10.0.12.0/24 dev eth1 proto kernel scope link src 10.0.12.2
local 10.0.12.2 dev eth1 table local proto kernel scope host src 10.0.12.2
broadcast 10.0.12.255 dev eth1 table local proto kernel scope link src
10.0.12.2
local 127.0.0.0/8 dev lo table local proto kernel scope host src 127.0.0.1
local 127.0.0.1 dev lo table local proto kernel scope host src 127.0.0.1
broadcast 127.255.255.255 dev lo table local proto kernel scope link src
127.0.0.1

[root@PC2 ~]#
```

   Без данных в таблице маршрутизации пакеты не будут передаваться.

2. С помощью команды управления таблицами маршрутизации на PC1 посмотрите на базовые данные таблиц, затем удалите информацию о маршруте в установленную сеть и вновь посмотрите данные.

```
[root@PC1 ~]# ip route
10.0.12.0/24 dev eth1 proto kernel scope link src 10.0.12.1

[root@PC1 ~]# ip route del dev eth1 10.0.12.0/24
[root@PC1 ~]# ip route
```

3. С помощью команды **ping -c3 <dstIP>** отправьте три ICMP пакета с PC1 на PC2.

```
[root@PC1 ~]# ping -c3 10.0.12.2
ping: connect: Network is unreachable
[root@PC1 ~]#
```

Ошибка формата **`ping: connect: Network is unreachable`** обозначает невозможность маршрутизировать пакет на источнике. Добавление адреса обратно возвращает работоспособность сети.

4. С помощью команды управления таблицами маршрутизации на PC1 добавьте информацию о маршруте в установленную сеть, а затем посмотрите базовые данные таблиц маршрутизации.

```
[root@PC1 ~]# ip route add dev eth1 10.0.12.0/24
[root@PC1 ~]# ip route
10.0.12.0/24 dev eth1 scope link
[root@PC1 ~]#
```

5. С помощью команды **`ping -c3 <dstIP>`** повторно отправьте три ICMPэпакета с PC1 на PC2.

```
[root@PC1 ~]# ping -c3 10.0.12.2
PING 10.0.12.2 (10.0.12.2) 56(84) bytes of data.
64 bytes from 10.0.12.2: icmp_seq=1 ttl=64 time=0.582 ms
64 bytes from 10.0.12.2: icmp_seq=2 ttl=64 time=0.549 ms
64 bytes from 10.0.12.2: icmp_seq=3 ttl=64 time=0.488 ms

--- 10.0.12.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2067ms
rtt min/avg/max/mdev = 0.488/0.539/0.582/0.038 ms
[root@PC1 ~]#
```

4.5. Мониторинг сети

Для отслеживания трафика используется утилита **`tcpdump`**, сканирующая сетевые интерфейсы и возвращающая информацию о проходящих через интерфейс пакетах. Вызов утилиты неразрывно связан с параметром **`-i <interface>`** для указания интерфейса, на котором будет проходить сканирование.

1. С помощью *команды мониторинга сети* :new: **`tcpdump -i <interface>`** запустите на PC2 сканирование интерфейса eth1.

```
[root@PC2 ~]# tcpdump -i eth1
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

2. С помощью команды **`ping -c1 <dstIP>`** отправьте один ICMPэпакет с PC1 на PC2.

```
[root@PC1 ~]# ping -c1 10.0.12.2
PING 10.0.12.2 (10.0.12.2) 56(84) bytes of data.
64 bytes from 10.0.12.2: icmp_seq=1 ttl=64 time=0.662 ms
```

```
--- 10.0.12.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.662/0.662/0.662/0.000 ms
[root@PC1 ~]#
```

3. Отключите мониторинг трафика с помощью **Ctrl+C**.

```
[root@PC2 ~]# tcpdump -i eth1
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
19:34:59.183275 IP 10.0.12.1 > PC2: ICMP echo request, id 5, seq 1, length
64
19:34:59.183301 IP PC2 > 10.0.12.1: ICMP echo reply, id 5, seq 1, length 64
19:35:04.320317 ARP, Request who-has PC2 tell 10.0.12.1, length 46
19:35:04.320330 ARP, Reply PC2 is-at 08:00:27:ee:00:c5 (oui Unknown), length
28
19:35:04.610697 ARP, Request who-has 10.0.12.1 tell PC2, length 28
19:35:04.611626 ARP, Reply 10.0.12.1 is-at 08:00:27:04:e4:05 (oui Unknown),
length 46
^C
6 packets captured
6 packets received by filter
0 packets dropped by kernel
[root@PC2 ~]#
```

tcpdump перехватил 6 пакетов: пару запрос⁄ответ ICMP от ping, а также две пары запрос⁄ответов ARP для определения связи MAC⁄адресов устройств и IP⁄адресов.

Полезным ключом **tcpdump** является **-X**, показывающий шестнадцатиричный код пакета, а также (по возможности) его ASCII⁄расшифровку. Ключ полезен для отслеживания передаваемых в пакете данных.

4. С помощью команды мониторинга сети запустите на PC2 сканирование интерфейса eth1 с выводом кода пакета.

```
[root@PC2 ~]# tcpdump -X -i eth1
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

5. С помощью команды **ping -c1 <dstIP>** отправьте один ICMP⁄пакет с PC1 на PC2.

```
[root@PC1 ~]# ping -c1 10.0.12.2
PING 10.0.12.2 (10.0.12.2) 56(84) bytes of data.
64 bytes from 10.0.12.2: icmp_seq=1 ttl=64 time=0.662 ms

--- 10.0.12.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.662/0.662/0.662/0.000 ms
[root@PC1 ~]#
```

6. Отключите мониторинг трафика с помощью **Ctrl+C**.

```
[root@PC2 ~]# tcpdump -X -i eth1
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
19:39:18.001500 IP 10.0.12.1 > PC2: ICMP echo request, id 6, seq 1, length
64
```

```
        0x0000:  4500 0054 ce95 4000 4001 4011 0a00 0c01  E..T..@.@.@.....
        0x0010:  0a00 0c02 0800 8203 0006 0001 e577 e168  .............w.h
        0x0020:  0000 0000 ea41 0600 0000 0000 1011 1213  .....A..........
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  .............!"#
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  $%&'()*+,-./0123
        0x0050:  3435 3637                                4567
19:39:18.001545 IP PC2 > 10.0.12.1: ICMP echo reply, id 6, seq 1, length 64
        0x0000:  4500 0054 553d 0000 4001 f969 0a00 0c02  E..TU=..@..i....
        0x0010:  0a00 0c01 0000 8a03 0006 0001 e577 e168  .............w.h
        0x0020:  0000 0000 ea41 0600 0000 0000 1011 1213  .....A..........
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  .............!"#
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  $%&'()*+,-./0123
        0x0050:  3435 3637                                4567

2 packets captured
2 packets received by filter
0 packets dropped by kernel
[root@PC2 ~]#
```

Примечание

Заметьте, что ICMP‑пакеты несут одинаковое содержание — последовательные байты, стандарт ICMP‑сообщения ping.

Ещё одним полезным ключом является **-n**, который убирает из описания перехваченных пакетов известные имена устройств, заменяя их на IP‑адреса.

7. С помощью команды мониторинга сети запустите на PC2 сканирование интерфейса eth1 с выводом IP‑адресов.

```
[root@PC2 ~]# tcpdump -n -i eth1
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

8. С помощью команды **ping -c1 <dstIP>** отправьте один ICMP‑пакет с PC1 на PC2.

```
[root@PC1 ~]# ping -c1 10.0.12.2
PING 10.0.12.2 (10.0.12.2) 56(84) bytes of data.
64 bytes from 10.0.12.2: icmp_seq=1 ttl=64 time=0.662 ms
--- 10.0.12.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.662/0.662/0.662/0.000 ms
[root@PC1 ~]#
```

9. Отключите мониторинг трафика с помощью **Ctrl+C**.

```
[root@PC2 ~]# tcpdump -n -i eth1
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
19:40:30.751100 IP 10.0.12.1 > 10.0.12.2: ICMP echo request, id 8, seq 1,
length 64
19:40:30.751127 IP 10.0.12.2 > 10.0.12.1: ICMP echo reply, id 8, seq 1,
length 64
```

```
2 packets captured
2 packets received by filter
0 packets dropped by kernel
[root@PC2 ~]#
```

Заметьте, что в предыдущих примерах получателем и источником выступал IP PC2, тогда как сейчас описывается явный IP-адрес.

4.6. Самостоятельная работа

4.6.1. Задание

Запустить отчёты на каждой машине и выполнить соответствующие команды.

```
report 4 pc1
```

1. **`ip a show eth1`**.
2. **`ip route`**.
3. **`ip route del dev eth1 10.0.12.0/24`**.
4. **`ping -fc3 10.0.12.2`**.
5. **`ip route add dev eth1 10.0.12.0/24`**.
6. **`ip route`**.
7. **`tcpdump -c6 -nX -i eth1`**.
8. Дождаться автозавершения **`tcpdump`**.

```
report 4 pc1
```

1. **`ip a`**.
2. **`ip route`**.
3. **`ip link set eth1 down`**.
4. **`ping -fc3 10.0.12.1`**.
5. **`ip link set eth1 up`**.
6. **`ping -c3 10.0.12.1`**.

Полученные отчёты **`report.04.pc1`** и **`report.04.pc2`** через последовательный порт перенести из виртуальной машины и прислать их преподавателю.

Глава 5. Настройка VLAN

5.3. Самостоятельная работа

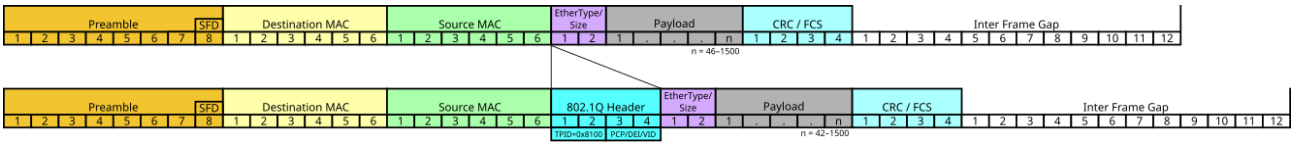
Цель лабораторной работы — познакомить изучающего с основами технологии VLAN.

Задачи лабораторной работы:

- Изучить базовую логику работы протоколов VLAN;
- Реализовать тестовую топологию с применением технологии VLAN.

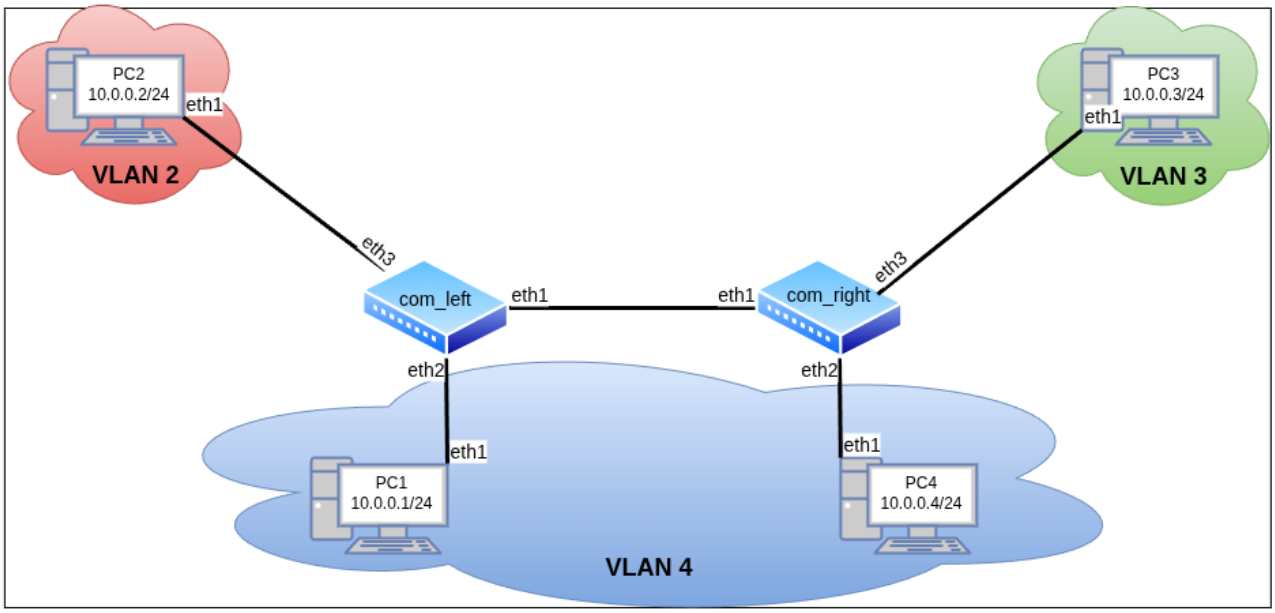
5.1. Технология VLAN

VLAN — Virtual Local Area Network — технология, которая позволяет разделить одну физическую сеть на несколько логических подсетей независимо от их физического расположения. В данной лабораторной для организации VLANов используется протокол IEEE 802.1Q, в котором Ethernetфреймы снабжаются дополнительным идентификатором виртуальной сети (тегом **VID**).



5.2. Построение сети с VLAN

Для изучения технологии VLAN разберём топологию с из шести устройств:



В предлагаемой топологии между любыми двумя соседними узлами **PC** и **com** циркулируют обычные Ethernetфреймы, а между узлами **comleft** и **comright** — любые модифицированные 802.1Q (т. н. **trunk**). При этом фреймы, получаемые от **PC1** и **PC4** помечаются одним **VID**, а от **PC2** и **PC3** — двумя другими. Таким образом **PC1** и **PC4** образуют единый VLAN и доступны друг для друга, а **PC2** и **PC3** находятся в изолированных VLANах с единственным абонентом в каждом.

Для работы создайте 6 *клонов* согласно топологии сети. Для создания соединений между машинами необходимо в VirtualBox настроить сетевые интерфейсы (описание настройки подключения находится в соответствующем разделе второй лабораторной):

- **com_left**:
  - **Adapter2** — **trunk**
  - **Adapter3** — **left4**
  - **Adapter4** — **left2**
- **com_right**:
  - **Adapter2** — **trunk**
  - **Adapter3** — **right4**
  - **Adapter4** — **right3**
- **pc1**:
  - **Adapter2** — **left4**
- **pc2**:
  - **Adapter2** — **left2**
- **pc3**:
  - **Adapter2** — **right3**
- **pc4**:
  - **Adapter2** — **right4**

5.2.1. Настройка абонентов сети

1. С помощью команд управления интерфейсами и настройки IP-адресов настройте интерфейсы и установите на них IP-адреса согласно топологии.

```
[root@pc1 ~]# ip link set eth1 up
[root@pc1 ~]# ip addr add dev eth1 10.0.0.1/24
```

```
[root@pc2 ~]# ip link set eth1 up
[root@pc2 ~]# ip addr add dev eth1 10.0.0.2/24
```

```
[root@pc3 ~]# ip link set eth1 up
[root@pc3 ~]# ip addr add dev eth1 10.0.0.3/24
```

```
[root@pc4 ~]# ip link set eth1 up
[root@pc4 ~]# ip addr add dev eth1 10.0.0.4/24
```

5.2.2. Настройка VLAN

Настройка VLAN производится на маршрутизирующих устройствах сети — коммутаторах или маршрутизаторах. Коммутаторы **comleft** и **comright** выступают в топологии в роли *сетевых мостов* — сетевых устройств, позволяющих обеспечивать целевую маршрутизацию данных *не выше интерфейсного уровня*.

Для реализации поведения сетевого моста на коммутаторах необходимо задать специальные виртуальные интерфейсы типа **bridge**, через которые с помощью *команды связывания интерфейсов* :new: **ip link set <interface> master <master-interface>** будут объединяться интерфейсы разных каналов. Данные сетевые интерфейсы также будут обеспечивать фильтрацию фреймов по тегам VLAN, для этого при их создании необходимо указать параметр **vlan_filtering**.

1. С помощью команд управления интерфейсами создайте на коммутаторах интерфейсы типа **bridge** с указанием параметра **vlan_filtering** и свяжите с ними все используемые физические интерфейсы.

```
[root@comleft ~]# ip link add dev br0 type bridge vlan_filtering 1
[root@comleft ~]# ip link set eth1 master br0
[root@comleft ~]# ip link set eth2 master br0
[root@comleft ~]# ip link set eth3 master br0
```

```
[root@comright ~]# ip link add dev br0 type bridge vlan_filtering 1
[root@comright ~]# ip link set eth1 master br0
[root@comright ~]# ip link set eth2 master br0
[root@comright ~]# ip link set eth3 master br0
```

   Далее необходимо настроить фильтрацию VLAN. Для этого с помощью *команды настройки VLAN* :new: **bridge vlan add vid <vlan-id> dev <interface>** необходимо указать, какой интерфейс будет обрабатывать фреймы с указанным тегом:

   - Для интерфейсов, ведущих к компьютерам непосредственно, отправка фреймов должна быть без тега (при обработке на интерфейсе пропускаться в канал будут лишь помеченные фреймы, но перед самой передачей тег будет сниматься);
   - Интерфейс, объединяющий коммутаторы, должен пересылать только помеченные фреймы.

2. С помощью команд настройки VLAN установите на коммутаторах VLAN на интерфейсы согласно топологии.

```
[root@comleft ~]# bridge vlan add vid 2 dev eth3 pvid untagged
[root@comleft ~]# bridge vlan add vid 4 dev eth2 pvid untagged
[root@comleft ~]# bridge vlan add vid 2 dev eth1
[root@comleft ~]# bridge vlan add vid 3 dev eth1
[root@comleft ~]# bridge vlan add vid 4 dev eth1
```

```
[root@comright ~]# bridge vlan add vid 3 dev eth3 pvid untagged
[root@comright ~]# bridge vlan add vid 4 dev eth2 pvid untagged
[root@comright ~]# bridge vlan add vid 2 dev eth1
[root@comright ~]# bridge vlan add vid 3 dev eth1
[root@comright ~]# bridge vlan add vid 4 dev eth1
```

3. С помощью команд управления интерфейсами включите на коммутаторах все используемые интерфейсы.

   Для упрощения можно воспользоваться коротким shell-сценарием, автоматически включающим все интерфейсы (в том числе **неиспользуемые**).

   ```
   [root@comleft ~]# for I in `ls /sys/class/net`; do ip link set $I up; done
   ```

   ```
   [root@comright ~]# for I in `ls /sys/class/net`; do ip link set $I up; done
   ```

   Все настройки, связанные с VLAN, можно посмотреть специальной командой *просмотра настроек VLAN* :new: **`bridge vlan show`**.

4. С помощью команд настройки VLAN выведите все данные о настройке VLAN на коммутаторе **`comleft`**.

   ```
   [root@comleft ~]# bridge vlan show
   port              vlan-id
   eth1              1 PVID Egress Untagged
                     2
                     3
                     4
   eth2              1 Egress Untagged
                     4 PVID Egress Untagged
   eth3              1 Egress Untagged
                     2 PVID Egress Untagged
   br0               1 PVID Egress Untagged
   [root@comleft ~]#
   ```

   Теперь попробуйте пропустить между абонентами трафик: между **`pc1`** и **`pc4`** будет проходить соединение, между любой другой парой абонентов — нет.

5. С помощью команды **`ping -c5 <dstIP>`** отправьте пять ICMP-пакетов с PC3 на PC2 и пять ICMP-пакетов с PC3 на PC1.

   ```
   [root@pc3 ~]# ping -c5 10.0.0.2
   PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
   From 10.0.0.3 icmp_seq=1 Destination Host Unreachable
   ping: sendmsg: No route to host
   From 10.0.0.3 icmp_seq=2 Destination Host Unreachable
   From 10.0.0.3 icmp_seq=3 Destination Host Unreachable
   From 10.0.0.3 icmp_seq=5 Destination Host Unreachable

   --- 10.0.0.2 ping statistics ---
   5 packets transmitted, 0 received, +4 errors, 100% packet loss, time 4105ms
   pipe 3
   [root@pc3 ~]# ping -c5 10.0.0.1
   PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
   64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.948 ms
   64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.792 ms
   64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.663 ms
   64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=1.08 ms
   64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=1.28 ms
   ```

```
--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4064ms
rtt min/avg/max/mdev = 0.663/0.953/1.282/0.217 ms
[root@pc3 ~]#
```

6. С помощью команды **`ping -c5 <dstIP>`** отправьте пять ICMPэхопакетов с PC4 на PC1.

```
[root@pc4 ~]# ping -c5 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.948 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.792 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.663 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=1.08 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=1.28 ms

--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4064ms
rtt min/avg/max/mdev = 0.663/0.953/1.282/0.217 ms
[root@pc4 ~]#
```

Убедитесь, что данные от абонентов из других VLAN доходят до коммутаторов, но далее не проходят.

Запустите на **`comleft`** команду **`tcpdump -xx -i eth1`**, которая будет отслеживать трафик на интерфейсе **`eth1`**.

7. С помощью команды мониторинга сети запустите на **`comleft`** сканирование интерфейса **`eth1`** с выводом кода пакета.

```
[root@comleft ~]# tcpdump -xx -i eth1
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

8. С помощью команды **`ping -c3 <dstIP>`** отправьте три ICMPэхопакета с PC3 на PC1.

```
[root@pc3 ~]# ping -c3 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
From 10.0.0.3 icmp_seq=1 Destination Host Unreachable
From 10.0.0.3 icmp_seq=2 Destination Host Unreachable
From 10.0.0.3 icmp_seq=3 Destination Host Unreachable

--- 10.0.0.1 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2077ms
pipe 3
[root@pc3 ~]#
```

```
[root@comleft ~]# tcpdump -xx -i eth1
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
22:18:47.881650 ARP, Request who-has 10.0.0.1 tell 10.0.0.3, length 46
       0x0000:  ffff ffff ffff 0800 27cc 8f19 8100 0003
       0x0010:  0806 0001 0800 0604 0001 0800 27cc 8f19
       0x0020:  0a00 0003 0000 0000 0000 0a00 0001 0000
       0x0030:  0000 0000 0000 0000 0000 0000 0000 0000
22:18:48.933866 ARP, Request who-has 10.0.0.1 tell 10.0.0.3, length 46
       0x0000:  ffff ffff ffff 0800 27cc 8f19 8100 0003
       0x0010:  0806 0001 0800 0604 0001 0800 27cc 8f19
```

```
        0x0020:  0a00 0003 0000 0000 0000 0a00 0001 0000
        0x0030:  0000 0000 0000 0000 0000 0000 0000 0000
22:18:49.957684 ARP, Request who-has 10.0.0.1 tell 10.0.0.3, length 46
        0x0000:  ffff ffff ffff 0800 27cc 8f19 8100 0003
        0x0010:  0806 0001 0800 0604 0001 0800 27cc 8f19
        0x0020:  0a00 0003 0000 0000 0000 0a00 0001 0000
        0x0030:  0000 0000 0000 0000 0000 0000 0000 0000
```

5.3. Самостоятельная работа

5.3.1. Задание

Запустить отчёты на каждой машине и выполнить соответствующие команды.

```
report 5 comright
```

1. **`ip a show`**.
2. **`bridge vlan show`**.
3. **`tcpdump -n -i eth1`**.
4. Выполните все остальные отчёты, после чего завершите **`tcpdump`** через **`Ctrl+C`**.

```
report 5 comleft
```

1. **`ip a show`**.
2. **`bridge vlan show`**.
3. **`tcpdump -n -i eth1`**.
4. Выполните все остальные отчёты, после чего завершите **`tcpdump`** через **`Ctrl+C`**.

```
report 5 pc1
```

1. **`ip a show eth1`**.
2. **`ping -fc3 10.0.0.3`**.
3. **`ping -fc3 10.0.0.4`**.

```
report 5 pc2
```

1. **`ip a show eth1`**.
2. **`ping -fc3 10.0.0.3`**.
3. **`ping -fc3 10.0.0.4`**.

```
report 5 pc3
```

1. **`ip a show eth1`**.

2. **`ping -fc3 10.0.0.1`**.
3. **`ping -fc3 10.0.0.2`**.

```
report 5 pc4
```

1. **`ip a show eth1`**.
2. **`ping -fc3 10.0.0.1`**.
3. **`ping -fc3 10.0.0.2`**.

Полученные отчёты **`report.05.comright`**, **`report.05.comleft`**, **`report.05.pc1`**, **`report.05.pc2`**, **`report.05.pc3`**, **`report.05.pc4`** через последовательный порт перенести из виртуальной машины и прислать их преподавателю.

Глава 6. Маршрутизация сетей с VLAN

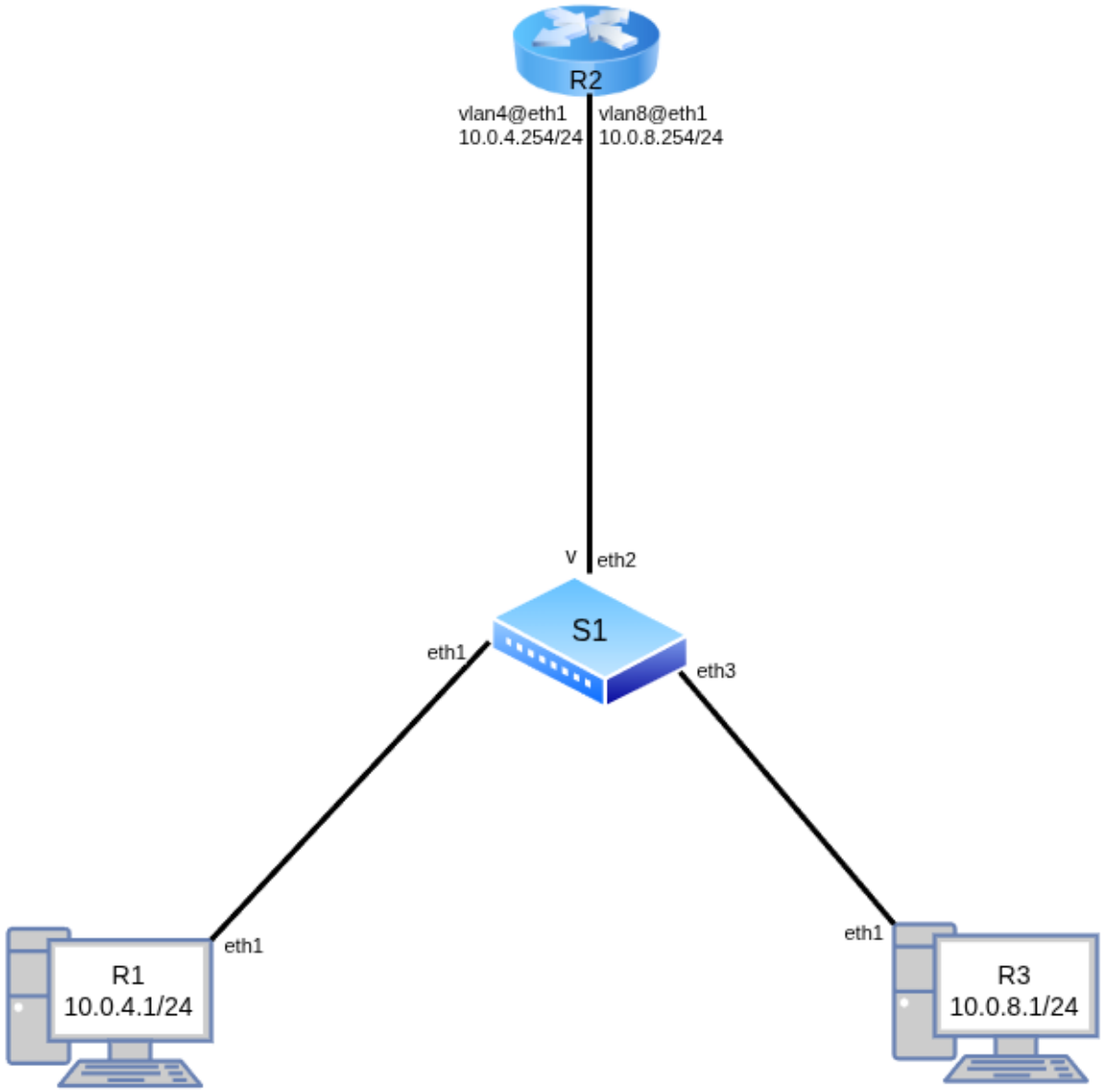
Цель лабораторной работы — познакомить изучающего с возможностями маршрутизации в сетях с VLAN.

Задачи лабораторной работы:

- Изучить возможности маршрутизации пакетов между VLAN;
- Реализовать тестовую топологию с применением технологии VLAN.

6.1. Построение сети с VLAN

Для изучения маршрутизации в сетях с VLAN разберём топологию с двумя VLAN/сетями и настроим трафик между ними через пограничный маршрутизатор, способный передавать пакеты в обеих сетях.



Для работы создайте 4 *клона* согласно топологии сети. Для создания соединений между машинами необходимо в VirtualBox настроить сетевые интерфейсы (описание настройки подключения находится в соответствующем разделе второй лабораторной):

- **R1**:
  - **Adapter2** — **left4**
- **R2**:
  - **Adapter2** — **trunk**
- **R3**:
  - **Adapter2** — **right8**
- **S1**:
  - **Adapter2** — **left4**
  - **Adapter3** — **trunk**

- **Adapter4** — **right8**

6.1.1. Настройка абонентов

1. С помощью команд управления интерфейсами и настройки IP‑адресов на R1 и R3 включите интерфейсы и установите на них IP‑адреса согласно топологии:

```
[root@R1 ~]# ip link set eth1 up
[root@R1 ~]# ip addr add dev eth1 10.0.4.1/24
[root@R1 ~]#
```

```
[root@R3 ~]# ip link set eth1 up
[root@R3 ~]# ip addr add dev eth1 10.0.8.1/24
[root@R3 ~]#
```

6.1.2. Настройка пограничного маршрутизатора

Поскольку трафик необходимо будет направлять через пограничный маршрутизатор, на нём должны быть явные интерфейсы локальных сетей с присвоенными IP‑адресами. Так как физический канал связи по топологии один, необходимо создать виртуальные интерфейсы, каждый из которых будет направлен в свою VLAN‑сеть.

1. *команды создания связанных виртуальных интерфейсов* :new: на R2 создайте два виртуальных VLAN‑интерфейса из VLAN‑сетей согласно топологии

```
[root@R2 ~]# ip link add link eth1 name vlan4 type vlan id 4
[root@R2 ~]# ip link add link eth1 name vlan8 type vlan id 8
[root@R2 ~]#
```

2. С помощью команд управления интерфейсами и настройки IP‑адресов на R2 включите интерфейсы и установите на них IP‑адреса согласно топологии:

```
[root@R2 ~]# ip addr add dev vlan4 10.0.4.254/24
[root@R2 ~]# ip addr add dev vlan8 10.0.8.254/24
[root@R2 ~]# ip link set eth1 up
[root@R2 ~]# ip link set vlan4 up
[root@R2 ~]# ip link set vlan8 up
[root@R2 ~]#
```

   Для передачи пакетов между сетевыми интерфейсами необходимо изменить настройку **`net.ipv4.conf.all.forwarding`**.

3. *перечачи пакетов между интерфейсами* :new: **`sysctl net.ipv4.conf.all.forwarding=1`** разрешите передачу пакетов на R2

```
[root@R2 ~]# sysctl net.ipv4.conf.all.forwarding
net.ipv4.conf.all.forwarding = 0
[root@R2 ~]# sysctl net.ipv4.conf.all.forwarding=1
[root@R2 ~]#
```

6.1.3. Настройка коммутатора

1. С помощью команд управления интерфейсами создайте на S1 интерфейс типа bridge с указанием параметра vlan_filtering и свяжите с ним все используемые физические интерфейсы

```
[root@S1 ~]# ip link add dev br0 type bridge vlan_filtering 1
[root@S1 ~]# ip link set eth1 master br0
[root@S1 ~]# ip link set eth2 master br0
[root@S1 ~]# ip link set eth3 master br0
```

2. С помощью команд настройки VLAN установите на S1 VLAN на интерфейсы согласно топологии

```
[root@S1 ~]# bridge vlan add vid 4 dev eth1 pvid untagged
[root@S1 ~]# bridge vlan add vid 8 dev eth3 pvid untagged
[root@S1 ~]# bridge vlan add vid 4 dev eth2
[root@S1 ~]# bridge vlan add vid 8 dev eth2
```

3. С помощью команд управления интерфейсами включите на S1 все используемые интерфейсы

```
[root@S1 ~]# for I in `ls /sys/class/net`; do ip link set $I up; done
```

6.1.4. Настройка статических маршрутов

1. С помощью команды управления таблицами маршрутизации на R1 и R2 добавьте информацию о маршрутах по умолчанию

```
[root@R1 ~]# ip route add default via 10.0.4.254
```

```
[root@R3 ~]# ip route add default via 10.0.8.254
```

6.1.5. Проверка настройки сети

1. С помощью команды мониторинга сети запустите на R2 сканирование интерфейса eth1 с выводом кода пакета

```
[root@R2 ~]# tcpdump -xx -i eth1
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

2. С помощью команды **`ping -c3 <dstIP>`** отправьте три ICMP-пакета с R1 на R3

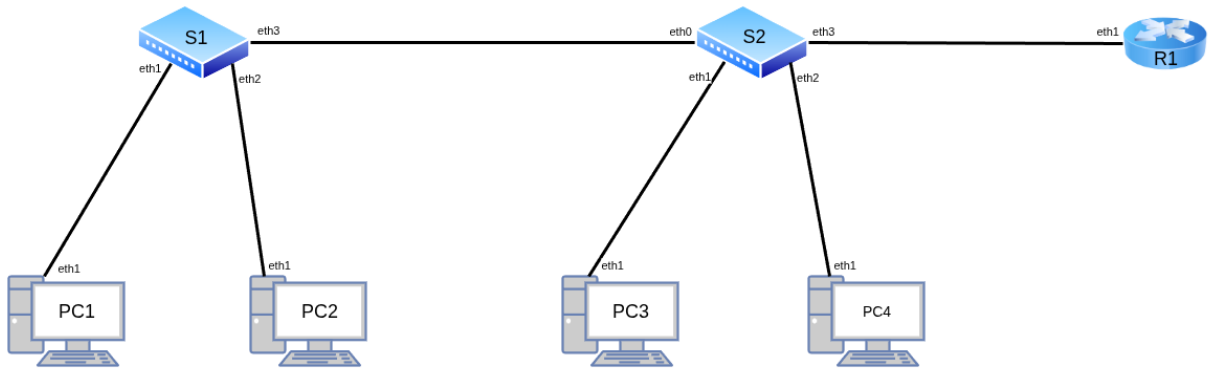
```
[root@R1 ~]# ping -c3 10.0.8.1
PING 10.0.8.1 (10.0.8.1) 56(84) bytes of data.
64 bytes from 10.0.8.1: icmp_seq=1 ttl=63 time=3.28 ms
64 bytes from 10.0.8.1: icmp_seq=2 ttl=63 time=1.87 ms
64 bytes from 10.0.8.1: icmp_seq=3 ttl=63 time=1.97 ms
```

```
--- 10.0.8.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.868/2.371/3.282/0.644 ms
[root@R1 ~]#
```

Каждый ping request/reply в **tcpdump** сопровождается четырьмя отметками: двумя для исходящего пакета и двумя для ответного. Отметки об одном и том же пакете отличаются только в 13–16 байтах (конец первой строки каждого пакета). Это тэг VID: **0x8100** — это информационные байты, сообщающие о наличии тэга в заголовке, **0x0008** и **0x0004** — сами значения идентификаторов тэга.

```
[root@R2 ~]# tcpdump -xx -i eth1
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
20:25:49.360836 IP 10.0.4.1 > 10.0.8.1: ICMP echo request, id 2, seq 1,
length 64
       0x0000:  0800 2719 4dcd 0800 27f5 ce47 8100 0004
       0x0010:  0800 4500 0054 0023 4000 4001 1a85 0a00
       0x0020:  0401 0a00 0801 0800 c6f6 0002 0001 4d5f
       0x0030:  d068 0000 0000 546b 0000 0000 0000 1011
       0x0040:  1213 1415 1617 1819 1a1b 1c1d 1e1f 2021
       0x0050:  2223 2425 2627 2829 2a2b 2c2d 2e2f 3031
       0x0060:  3233 3435 3637
20:25:49.360855 IP 10.0.4.1 > 10.0.8.1: ICMP echo request, id 2, seq 1,
length 64
       0x0000:  0800 27b3 98a5 0800 2719 4dcd 8100 0008
       0x0010:  0800 4500 0054 0023 4000 3f01 1b85 0a00
       0x0020:  0401 0a00 0801 0800 c6f6 0002 0001 4d5f
       0x0030:  d068 0000 0000 546b 0000 0000 0000 1011
       0x0040:  1213 1415 1617 1819 1a1b 1c1d 1e1f 2021
       0x0050:  2223 2425 2627 2829 2a2b 2c2d 2e2f 3031
       0x0060:  3233 3435 3637
20:25:49.361696 IP 10.0.8.1 > 10.0.4.1: ICMP echo reply, id 2, seq 1, length
64
       0x0000:  0800 2719 4dcd 0800 27b3 98a5 8100 0008
       0x0010:  0800 4500 0054 2bb9 0000 4001 2eef 0a00
       0x0020:  0801 0a00 0401 0000 cef6 0002 0001 4d5f
       0x0030:  d068 0000 0000 546b 0000 0000 0000 1011
       0x0040:  1213 1415 1617 1819 1a1b 1c1d 1e1f 2021
       0x0050:  2223 2425 2627 2829 2a2b 2c2d 2e2f 3031
       0x0060:  3233 3435 3637
20:25:49.361705 IP 10.0.8.1 > 10.0.4.1: ICMP echo reply, id 2, seq 1, length
64
       0x0000:  0800 27f5 ce47 0800 2719 4dcd 8100 0004
       0x0010:  0800 4500 0054 2bb9 0000 3f01 2fef 0a00
       0x0020:  0801 0a00 0401 0000 cef6 0002 0001 4d5f
       0x0030:  d068 0000 0000 546b 0000 0000 0000 1011
       0x0040:  1213 1415 1617 1819 1a1b 1c1d 1e1f 2021
       0x0050:  2223 2425 2627 2829 2a2b 2c2d 2e2f 3031
       0x0060:  3233 3435 3637
<...>
```

6.2. Самостоятельная работа



1. Определить свой вариант **X** как свой номер по порядку в упорядоченном по алфавиту списке группы.
2. Создать топологию, изображённую выше.
3. **PC1** и **PC3** должны быть объединены в один VLAN с номером (**X** + 10).

   PC2 и **PC4** должны быть объединены в другой VLAN с номером (**X** + 20).
4. Реализовать пересылку между VLAN через маршрутизатор **R1**.
5. IP-адреса для сетевых устройств в VLAN (**X** + 10) брать из сети **10.0.{X+10}.0/24**.

   IP-адреса для сетевых устройств в VLAN (**X** + 20) брать из сети **10.0.{X+20}.0/24**.

6.2.1. Задание

Запустить отчёты на каждой машине и выполнить соответствующие команды.

Примечание

Поскольку для устройств не зафиксированы номера хостов, их выбор оставляется за исполнителем. При описании команд отчётов используются номера A, B, C, D для IP-адресов абонентов и R для обозначения IP-адресов маршрутизатора. Необходимо на этом месте указывать выбранные IP-адреса.

```
report 6 pc1
```

1. **ip a show <PC1→S1 interface>**.
2. **ip route**.

3. **ping -fc3 10.0.{X+10}.R**.
4. **ping -fc3 10.0.{X+20}.R**.
5. **ping -fc3 10.0.{X+20}.B**.
6. **ping -fc3 10.0.{X+10}.C**.
7. **ping -fc3 10.0.{X+20}.D**.
8. **traceroute 10.0.{X+10}.C**.

```
report 6 pc2
```

1. **ip a show <PC2→S1 interface>**.
2. **ip route**.
3. **ping -fc3 10.0.{X+20}.R**.
4. **ping -fc3 10.0.{X+10}.R**.
5. **ping -fc3 10.0.{X+10}.A**.
6. **ping -fc3 10.0.{X+10}.C**.
7. **ping -fc3 10.0.{X+20}.D**.
8. **traceroute 10.0.{X+20}.D**.

```
report 6 pc3
```

1. **ip a show <PC3→S2 interface>**.
2. **ip route**.
3. **ping -fc3 10.0.{X+10}.R**.
4. **ping -fc3 10.0.{X+20}.R**.
5. **ping -fc3 10.0.{X+10}.A**.
6. **ping -fc3 10.0.{X+20}.B**.
7. **ping -fc3 10.0.{X+20}.D**.
8. **traceroute 10.0.{X+10}.A**.

```
report 6 pc4
```

1. **ip a show <PC4→S2 interface>**.
2. **ip route**.
3. **ping -fc3 10.0.{X+20}.R**.
4. **ping -fc3 10.0.{X+10}.R**.

5. **`ping -fc3 10.0.{X+10}.A`**.
6. **`ping -fc3 10.0.{X+20}.B`**.
7. **`ping -fc3 10.0.{X+10}.C`**.
8. **`traceroute 10.0.{X+20}.B`**.

```
report 6 r1
```

1. **`ip a show <R1→S2 (vlan X+10) interface>`**.
2. **`ip a show <R1→S2 (vlan X+20) interface>`**.
3. **`ip route`**.
4. **`sysctl net.ipv4.conf.all.forwarding`**.

```
report 6 s1
```

1. **`bridge vlan show`**.

```
report 6 s2
```

1. **`bridge vlan show`**.

Полученные отчёты **`report.06.pc1`**, **`report.06.pc2`**, **`report.06.pc3`**, **`report.06.pc4`**, **`report.06.r1`**, **`report.06.s1`**, **`report.06.s2`** через последовательный порт перенести из виртуальной машины и прислать их преподавателю с подписью выполненного варианта.

Глава 7. Работа с протоколом STP

Цель лабораторной работы — познакомить изучающего с основами протокола STP.

Задачи лабораторной работы:

- Изучить логику работы протокола;
- Изучить возможности управления параметрами каналов;
- Реализовать тестовую топологию с применением протокола STP.

7.1. Протокол STP

Протокол STP (*Spanning Tree Protocol*) предназначен для устранения петель в топологиях с избыточными соединениями на интерфейсном (канальном, L2) уровне путём построения полного связного дерева. Из подзадач протокола можно выделить *определение корневого коммутатора* сети и приоритетное *отключение избыточных каналов связи*.

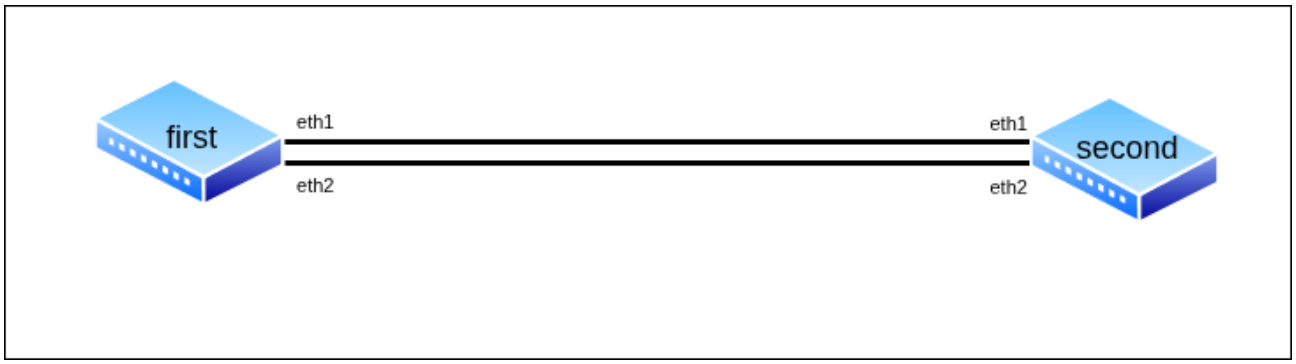
Метрикой выбора канала между узлами в STP является *стоимость пути (path cost)*, которая рассчитывается на основе пропускной способности канала.

Кратко работу протокола можно описать следующей последовательностью действий:

- Все устройства отправляют *BPDU (Bridge Protocol Data Unit)* с собственным идентификатором (*Bridge ID*).
- Выбирается *корневое устройство (Root Bridge)* — устройство с наименьшим Bridge ID.
- Каждое некорневое устройство определяет *корневой порт* — порт с наименьшей стоимостью пути к корневому устройству.
- На каждом сегменте сети выбирается *назначенный порт (Designated Port)* — порт с наименьшей стоимостью пути к корневому устройству на этом сегменте.
- Все остальные порты переходят в *блокированное состояние (Blocking)* для предотвращения петель.
- Корневой и назначенные порты переходят в *состояние передачи (Forwarding)*, обеспечивая связность без петель.

7.2. Пример настройки STP

Рассмотрим систему из двух коммутаторов, объединённых двумя каналами связи, и изучим на ней возможности работы с протоколом:



Для работы создайте 2 *клона* согласно топологии сети. Для создания соединений между машинами необходимо в VirtualBox настроить сетевые интерфейсы (описание настройки подключения находится в соответствующем разделе второй лабораторной):

- **`first`**:
  - **`Adapter2`** — **`intnet`**
  - **`Adapter3`** — **`deepnet`**
- **`second`**:
  - **`Adapter2`** — **`intnet`**
  - **`Adapter3`** — **`deepnet`**

7.2.1. Настройка коммутаторов

1. С помощью команд управления интерфейсами создайте на коммутаторах first и second интерфейсы типа bridge и свяжите с ними все используемые физические интерфейсы.

```
[root@first ~]# ip link add br0 type bridge
[root@first ~]# ip link set eth1 master br0
[root@first ~]# ip link set eth2 master br0
[root@first ~]#
```

```
[root@second ~]# ip link add br0 type bridge
[root@second ~]# ip link set eth1 master br0
[root@second ~]# ip link set eth2 master br0
[root@second ~]#
```

2. С помощью команд управления интерфейсами включите на коммутаторах все используемые интерфейсы.

```
[root@first ~]# ip link set br0 up
[root@first ~]# ip link set eth1 up
[root@first ~]# ip link set eth2 up
[root@first ~]#
```

```
[root@second ~]# ip link set br0 up
[root@second ~]# ip link set eth1 up
[root@second ~]# ip link set eth2 up
[root@second ~]#
```

7.2.2. Параметры сетевого моста

1. С помощью команды управления интерфейсами на first выведите подробную информацию об интерфейсе br0.

```
[root@first ~]# ip -d link show br0
6: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
mode DEFAULT group default ql
en 1000
   link/ether 08:00:27:26:29:ce brd ff:ff:ff:ff:ff:ff promiscuity 0 allmulti
0 minmtu 68 maxmtu 65535
   bridge forward_delay 1500 hello_time 200 max_age 2000 ageing_time 30000
stp_state 0 priority 32768
   vlan_filtering 0 vlan_protocol 802.1Q bridge_id 8000.8:0:27:26:29:ce
designated_root 8000.8:0:27:26:29:ce
   root_port 0 root_path_cost 5 topology_change 0 topology_change_detected 0
hello_timer 0.00 tcn_timer 0.00
   topology_change_timer 0.00 gc_timer 31.90 fdb_n_learned 2 fdb_max_learned
0 vlan_default_pvid 1
   vlan_stats_enabled 0 vlan_stats_per_port 0 group_fwd_mask 0 group_address
01:80:c2:00:00:00 mcast_snooping 1
   no_linklocal_learn 0 mcast_vlan_snooping 0 mst_enabled 0 mcast_router 1
mcast_query_use_ifaddr 0
   mcast_querier 0 mcast_hash_elasticity 16 mcast_hash_max 4096
mcast_last_member_count 2 mcast_startup_query_count 2
   mcast_last_member_interval 100 mcast_membership_interval 26000
mcast_querier_interval 25500
   mcast_query_interval 12500 mcast_query_response_interval 1000
```

```
mcast_startup_query_interval 3125
   mcast_stats_enabled 0 mcast_igmp_version 2 mcast_mld_version 1
nf_call_iptables 0 nf_call_ip6tables 0
   nf_call_arptables 0 numtxqueues 1 numrxqueues 1 gso_max_size 65536
gso_max_segs 65535 tso_max_size 65536
   tso_max_segs 65535 gro_max_size 65536 gso_ipv4_max_size 65536
gro_ipv4_max_size 65536
[root@first ~]#
```

В работе протокола STP учитываются несколько параметров данного сетевого моста:

- *stp_state* — состояние протокола STP на коммутаторе (подключён ли он на данном устройстве или нет);
- *priority* — приоритет данного коммутатора, используется для определения корневого коммутатора;
- *bridge_id* — идентификатор данного коммутатора, совпадает со значением MAC-адреса устройства (при передаче BPDU поле BID состоит из полей priority и bridge_id, однако в описании Linux-интерфейсов они хранятся отдельно);
- *designated_port* — это порт, через который в сегмент сети (который к нему подключен) будут передаваться BPDU. По значению совпадает со значением MAC-адреса на корневом коммутаторе;
- *root_port* — номер корневого порта зависимого (некорневого) коммутатора (0 в случае корневого коммутатора).

Работа протокола подразумевает постоянную передачу служебных сообщений между коммутаторами (BPDU) для синхронизации состояний. За скорость синхронизации отвечают параметры:

- *hello_time* — интервал отправки BPDU между соседними коммутаторами;
- *max_age* — максимальное время актуальности полученного BPDU (определяет размер (диаметр) сети: если значение срока действия сообщения (Message Age; передаётся в BPDU, увеличивается на 1 при прохождении каждого коммутатора) меньше или равно значению максимального срока действия (Max Age), некорневой мост перенаправляет конфигурационный BPDU. Если значение срока действия сообщения превышает значение максимального срока действия, некорневой мост отменяет конфигурационный BPDU. В этом случае сеть считается слишком большой, и некорневой мост отключается от корневого моста);
- *forward_delay* — задержка отправки BPDU после изменения топологии системы (время нахождения портов коммутатора в состоянии **`listening`** и **`learning`**).

7.2.3. Работа с STP

На данный момент в настраиваемых нами системах протокол STP отключён. Оба коммутатора считают себя единственными (и потому корневыми) коммутаторами в системе.

1. С помощью команды **`bridge link`** на коммутаторах выведите параметры всех связанных с сетевым мостом интерфейсов.

```
[root@first ~]# bridge link
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 32 cost 5
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 32 cost 5
[root@first ~]#
```

```
[root@second ~]# bridge link
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 32 cost 5
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 32 cost 5
[root@second ~]#
```

2. С помощью команды управления интерфейсами выведите информацию о параметрах сетевого моста. Обратите внимание на параметры **stp_state**, **priority**, **bridge_id**, **designated_port**, **root_port**.

```
[root@first ~]# ip -d link show br0 | grep root
   bridge <...> stp_state 0 priority 32768 <...>
   bridge_id 8000.8:0:27:26:29:ce designated_root 8000.8:0:27:26:29:ce
root_port 0 <...>
[root@first ~]#
```

```
[root@second ~]# ip -d link show br0 | grep root
   bridge <...> stp_state 0 priority 32768 <...>
   bridge_id 8000.8:0:27:5a:86:5c designated_root 8000.8:0:27:5a:86:5c
root_port 0 <...>
[root@second ~]#
```

3. С помощью команды изменения параметров интерфейса **ip link set dev br0 type bridge stp_state 1** на коммутаторах включите протокол STP. С помощью команд управления интерфейсами и сетевым мостом выведите параметры интерфейсов и моста.

```
[root@first ~]# ip link set dev br0 type bridge stp_state 1
<Some time later>

[root@first ~]# ip -d link show br0 | grep root
   bridge <...> stp_state 1 priority 32768 <...>
   bridge_id 8000.8:0:27:26:29:ce designated_root 8000.8:0:27:26:29:ce
root_port 0 <...>

[root@first ~]# bridge link
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 32 cost 5
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 32 cost 5
[root@first ~]#
```

```
[root@second ~]# ip link set dev br0 type bridge stp_state 1
<Some time later>

[root@second ~]# ip -d link show br0 | grep root
   bridge <...> stp_state 1 priority 32768 <...>
   bridge_id 8000.8:0:27:5a:86:5c designated_root 8000.8:0:27:26:29:ce
root_port 1 <...>
```

```
[root@second ~]# bridge link
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 32 cost 5
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
blocking priority 32 cost 5
[root@second ~]#
```

Поскольку приоритеты коммутаторов одинаковы, корневым автоматически становится коммутатор с меньшим **bridge_id**. У зависимого коммутатора при этом меняются значения **designated_root** и **root_port**.

Примечание

Поскольку MAC-адреса созданных клонов могут отличаться от представленных в лабораторной, внимательно следите за обозначениями виртуальных машин, для которых выполняются команды. Для удобства в тексте далее будут использоваться обозначения «корневой коммутатор» и «зависимый коммутатор».

Для смены корневого коммутатора можно изменить приоритет.

4. С помощью команды изменения параметров интерфейса на *зависимом* коммутаторе установите новое значение приоритета моста.

```
[root@second ~]# ip link set dev br0 type bridge priority 4096
[root@second ~]# ip -d link show br0 | grep priority
   bridge <...> priority 4096 <...>
   bridge_id 1000.8:0:27:5a:86:5c designated_root 1000.8:0:27:5a:86:5c
root_port 0 <...>
[root@second ~]# bridge link
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 32 cost 5
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
listening priority 32 cost 5
<Some time later>
[root@second ~]# bridge link
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 32 cost 5
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 32 cost 5
```

```
<Some time later>
[root@first ~]# ip -d link show br0 | grep priority
   bridge <...> priority 32768 <...>
   bridge_id 8000.8:0:27:26:29:ce designated_root 1000.8:0:27:5a:86:5c
root_port 1 <...>
[root@first ~]# bridge link
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 32 cost 5
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
blocking priority 32 cost 5
[root@first ~]#
```

После синхронизации первый коммутатор стал корневым, а второй — зависимым.

Для управления каналами передачи данных используются настройки приоритета портов и «стоимости» каналов.

5. С помощью команды изменения параметров интерфейса на *корневом* коммутаторе установите новое значение приоритета связанного порта eth1.

```
[root@second ~]# bridge link set dev eth1 priority 63
[root@second ~]# bridge link
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 63 cost 5
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 32 cost 5
[root@second ~]#
```

   На *зависимом* коммутаторе поменяются статусы каналов, активным станет канал с меньшим значением **`priority`** у соседа.

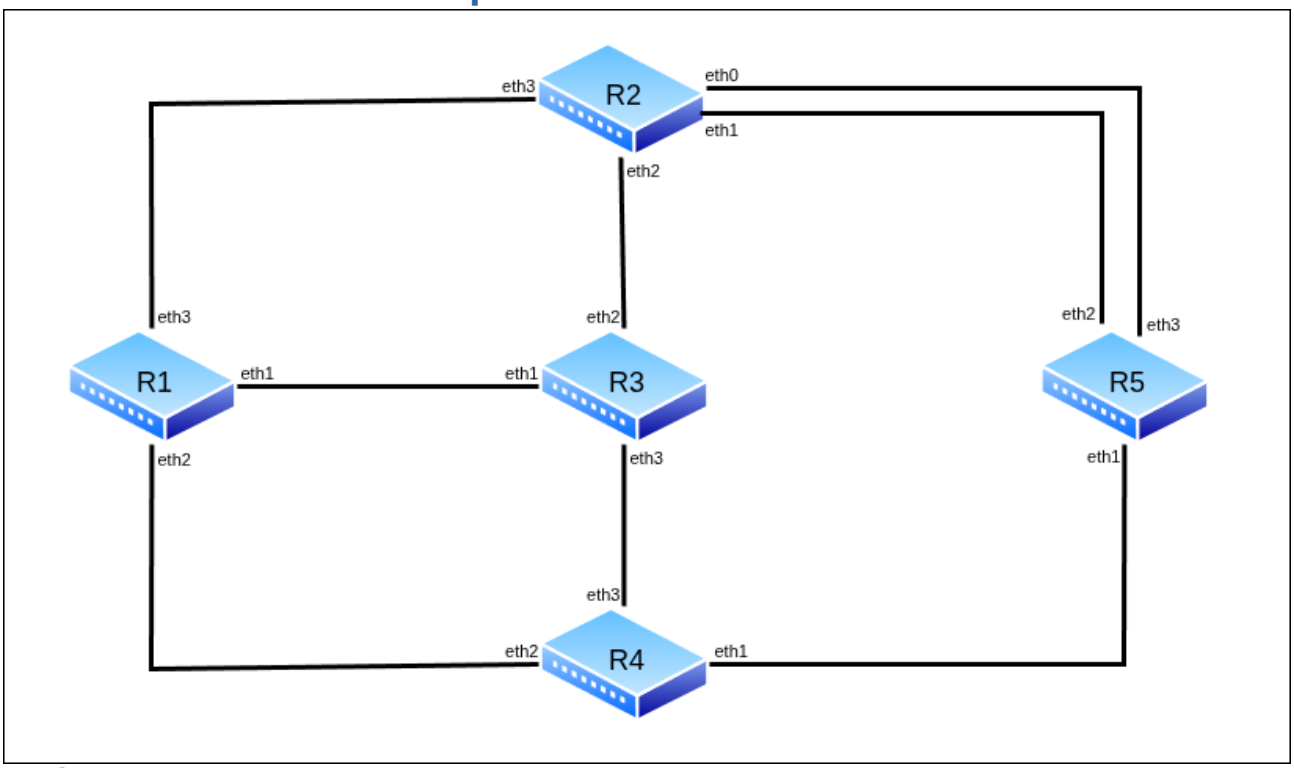
```
<Some time later>
[root@first ~]# ip -d link show br0 | grep priority
   bridge <...> priority 32768 <...>
   bridge_id 8000.8:0:27:26:29:ce designated_root 1000.8:0:27:5a:86:5c
root_port 2 <...>
[root@first ~]# bridge link
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
blocking priority 32 cost 5
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
listening priority 32 cost 5
[root@first ~]#
```

6. С помощью команды изменения параметров интерфейса на *зависимом* коммутаторе установите новое значение стоимости канала связанного порта eth2.

   Статусы каналов вновь поменяются, использоваться будет канал с меньшей стоимостью.

```
[root@first ~]# bridge link set dev eth2 cost 20000
[root@first ~]# bridge link
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
listening priority 32 cost 5
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
blocking priority 32 cost 20000
<Some time later>
[root@first ~]# bridge link
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
forwarding priority 32 cost 5
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state
blocking priority 32 cost 20000
[root@first ~]#
```

7.3. Самостоятельная работа



Важно

Для работы необходимо 5 *клонов* согласно топологии сети. Для создания соединений между машинами необходимо в VirtualBox настроить сетевые интерфейсы (описание настройки подключения находится в соответствующем разделе второй лабораторной):

- **R1**:
  - **Adapter2** — **net13**
  - **Adapter3** — **net14**
  - **Adapter4** — **net12**
- **R2**:
  - **Adapter1** — **intnet25**
  - **Adapter2** — **deepnet25**
  - **Adapter3** — **net23**
  - **Adapter4** — **net12**
- **R3**:
  - **Adapter2** — **net13**
  - **Adapter3** — **net23**
  - **Adapter4** — **net34**

- **R4**:
  - **Adapter2** — **net45**
  - **Adapter3** — **net14**
  - **Adapter4** — **net34**
- **R5**:
  - **Adapter2** — **net45**
  - **Adapter3** — **deepnet25**
  - **Adapter4** — **intnet25**

7.3.1. Варианты заданий

Таблица 7.1. Варианты заданий

Вариант	Задание
1	Настроить устройства (см. топологию) так, чтобы корневым коммутатором было устройство R1, а корневым портом на R5 был eth1
2	Настроить устройства (см. топологию) так, чтобы корневым коммутатором было устройство R2, а корневым портом на R4 был eth3
3	Настроить устройства (см. топологию) так, чтобы корневым коммутатором было устройство R3, а корневым портом на R5 был eth2
4	Настроить устройства (см. топологию) так, чтобы корневым коммутатором было устройство R4, а корневым портом на R1 был eth3
5	Настроить устройства (см. топологию) так, чтобы корневым коммутатором было устройство R5, а корневым портом на R3 был eth1

7.3.1.1. Задание

Запустить отчёты на каждой машине и выполнить соответствующие команды.

```
report 7 r1
```

1. **ip link show**.
2. **ip -d link show <название bridge-интерфейса>**.
3. **bridge link**.

```
report 7 r2
```

1. **ip link show**.
2. **ip -d link show <название bridge-интерфейса>**.
3. **bridge link**.

```
report 7 r3
```

1. **ip link show**.
2. **ip -d link show <название bridge-интерфейса>**.
3. **bridge link**.

```
report 7 r4
```

1. **ip link show**.
2. **ip -d link show <название bridge-интерфейса>**.
3. **bridge link**.

```
report 7 r5
```

1. **ip link show**.
2. **ip -d link show <название bridge-интерфейса>**.
3. **bridge link**.

Полученные отчёты **report.07.r1**, **report.07.r2**, **report.07.r3**, **report.07.r4**, **report.07.r5** через последовательный порт перенести из виртуальной машины и прислать их преподавателю с подписью выполненного варианта.

Глава 8. Статические маршруты и маршруты по умолчанию

Цель лабораторной работы — познакомить изучающего с возможностями статической маршрутизации.

Задачи лабораторной работы:

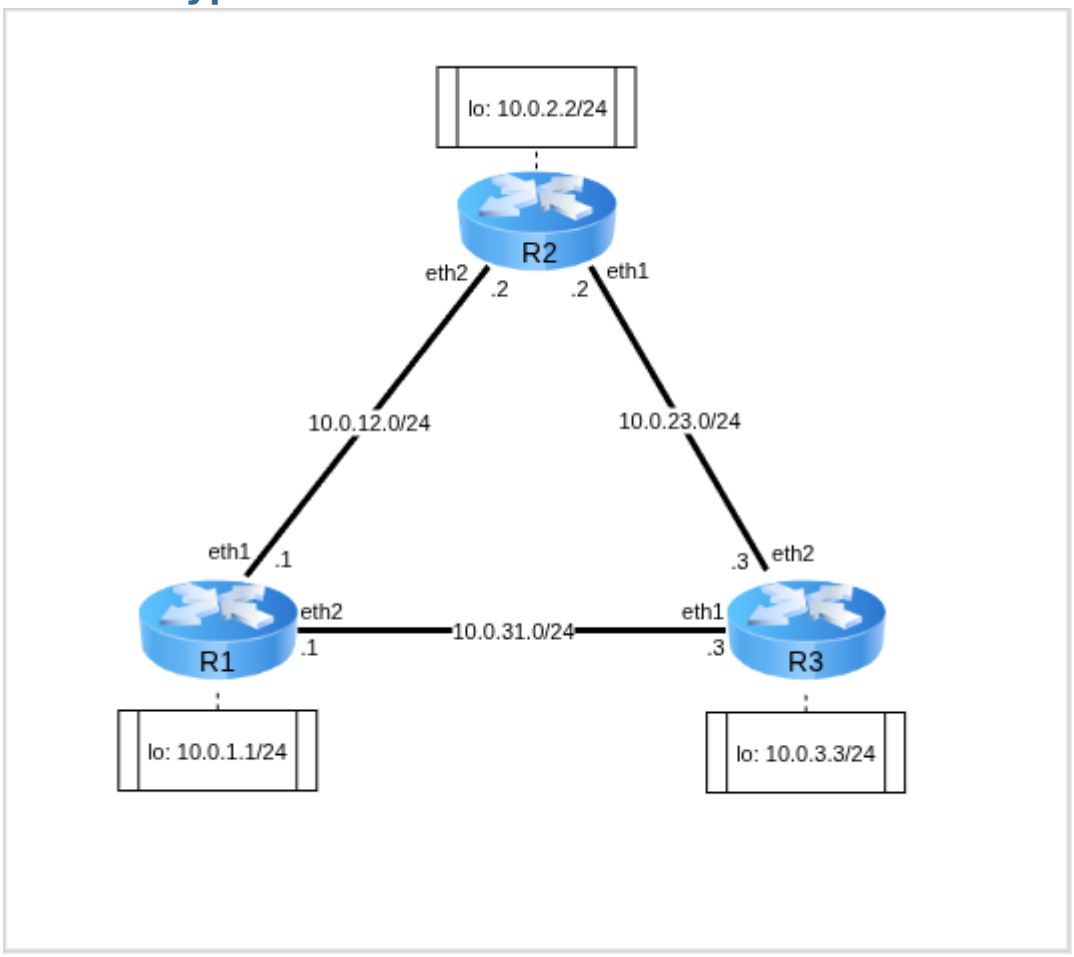
- Изучить работу статической маршрутизации;
- Реализовать тестовую топологию с применением статической маршрутизации на сетевом уровне.

8.1. Статическая маршрутизация

Задание маршрутов на канальном и сетевом уровнях может осуществляться посредством динамического обновления таблиц (*коммутации* на канальном уровне, *маршрутизации* — на сетевом) или задания статических правил в таблицы.

Основным параметром выбора выступает идентификатор получателя; дополнительно с помощью задания правил можно управлять маршрутизацией, ориентируясь на другие параметры.

8.2. Пример настройки статической маршрутизация на сетевом уровне



Для работы создадим 3 *клона* согласно топологии сети. Для создания соединений между машинами необходимо в VirtualBox настроить сетевые интерфейсы (описание настройки подключения находится в соответствующем разделе второй лабораторной):

- **R1**:
  - **Adapter2** — **net12**
  - **Adapter3** — **net31**
- **R2**:
  - **Adapter2** — **net23**
  - **Adapter3** — **net12**
- **R3**:
  - **Adapter2** — **net31**
  - **Adapter3** — **net23**

8.2.1. Базовая настройка виртуальных машин

1. С помощью команд управления интерфейсами создайте и включите интерфейсы согласно топологии.

```
[root@R1 ~]# ip link set eth1 up
[root@R1 ~]# ip link set eth2 up
[root@R1 ~]# ip link set lo up
[root@R1 ~]#
```

```
[root@R2 ~]# ip link set eth1 up
[root@R2 ~]# ip link set eth2 up
[root@R2 ~]# ip link set lo up
[root@R2 ~]#
```

```
[root@R3 ~]# ip link set eth1 up
[root@R3 ~]# ip link set eth2 up
[root@R3 ~]# ip link set lo up
[root@R3 ~]#
```

2. На соответствующих интерфейсах с помощью команд настройки IP-адресов установите адреса согласно топологии.

```
[root@R1 ~]# ip addr add dev eth1 10.0.12.1/24
[root@R1 ~]# ip addr add dev eth2 10.0.31.1/24
[root@R1 ~]# ip addr add dev lo 10.0.1.1/24
[root@R1 ~]#
```

```
[root@R2 ~]# ip addr add dev eth1 10.0.23.2/24
[root@R2 ~]# ip addr add dev eth2 10.0.12.2/24
[root@R2 ~]# ip addr add dev lo 10.0.2.2/24
[root@R2 ~]#
```

```
[root@R3 ~]# ip addr add dev eth1 10.0.31.3/24
[root@R3 ~]# ip addr add dev eth2 10.0.23.3/24
[root@R3 ~]# ip addr add dev lo 10.0.3.3/24
[root@R3 ~]#
```

3. С помощью команды настройки IP-адресов выведите доступные интерфейсы с указанием IP-адресов на них.

```
[root@R1 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default 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
   inet 10.0.1.1/24 scope global lo
      valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
   link/ether 08:00:27:a1:1d:0b brd ff:ff:ff:ff:ff:ff
   altname enp0s3
   altname enx080027a11d0b
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
```

```
group default qlen 1000
   link/ether 08:00:27:15:a8:2d brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx08002715a82d
   inet 10.0.12.1/24 scope global eth1
      valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
   link/ether 08:00:27:18:1f:bc brd ff:ff:ff:ff:ff:ff
   altname enp0s9
   altname enx080027181fbc
   inet 10.0.31.1/24 scope global eth2
      valid_lft forever preferred_lft forever
5: eth3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
   link/ether 08:00:27:3c:0a:dd brd ff:ff:ff:ff:ff:ff
   altname enp0s10
   altname enx0800273c0add
[root@R1 ~]#
```

```
[root@R2 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default 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
   inet 10.0.2.2/24 scope global lo
      valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
   link/ether 08:00:27:a8:1c:34 brd ff:ff:ff:ff:ff:ff
   altname enp0s3
   altname enx080027a81c34
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
   link/ether 08:00:27:49:d4:10 brd ff:ff:ff:ff:ff:ff
   altname enp0s8
   altname enx08002749d410
   inet 10.0.23.2/24 scope global eth1
      valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
   link/ether 08:00:27:45:07:41 brd ff:ff:ff:ff:ff:ff
   altname enp0s9
   altname enx080027450741
   inet 10.0.12.2/24 scope global eth2
      valid_lft forever preferred_lft forever
5: eth3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
   link/ether 08:00:27:49:02:f3 brd ff:ff:ff:ff:ff:ff
   altname enp0s10
   altname enx0800274902f3
[root@R2 ~]#
```

```
[root@R3 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default 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
    inet 10.0.3.3/24 scope global lo
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
    link/ether 08:00:27:71:25:a4 brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    altname enx0800277125a4
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 08:00:27:21:3d:8f brd ff:ff:ff:ff:ff:ff
    altname enp0s8
    altname enx080027213d8f
    inet 10.0.31.3/24 scope global eth1
       valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 08:00:27:cc:73:52 brd ff:ff:ff:ff:ff:ff
    altname enp0s9
    altname enx080027cc7352
    inet 10.0.23.3/24 scope global eth2
       valid_lft forever preferred_lft forever
5: eth3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
    link/ether 08:00:27:90:28:d9 brd ff:ff:ff:ff:ff:ff
    altname enp0s10
    altname enx0800279028d9
[root@`R3` ~]#
```

Примечание

Объединение нескольких абонентов через одну сеть (одно и то же название сети в описании сетевого интерфейса в VirtualBox) позволяет сразу передавать между ними трафик в рамках этой локальной сети. Информация о том, как маршрутизировать пакеты в другие сети, отсутствует.

4. С помощью команды **`ping -c3 <dstIP>`** продемонстрируйте возможность передачи пакетов между абонентами одной сети и *недостижимость* (**`Network is unreachable`**) других сетей

```
[root@R1 ~]# ping -c3 10.0.12.2
PING 10.0.12.2 (10.0.12.2) 56(84) bytes of data.
64 bytes from 10.0.12.2: icmp_seq=1 ttl=64 time=0.347 ms
64 bytes from 10.0.12.2: icmp_seq=2 ttl=64 time=0.307 ms
64 bytes from 10.0.12.2: icmp_seq=3 ttl=64 time=0.410 ms

--- 10.0.12.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2081ms
rtt min/avg/max/mdev = 0.307/0.354/0.410/0.042 ms

[root@R1 ~]# ping -c3 10.0.31.3
PING 10.0.31.3 (10.0.31.3) 56(84) bytes of data.
64 bytes from 10.0.31.3: icmp_seq=1 ttl=64 time=0.377 ms
64 bytes from 10.0.31.3: icmp_seq=2 ttl=64 time=0.461 ms
64 bytes from 10.0.31.3: icmp_seq=3 ttl=64 time=0.376 ms
```

```
--- 10.0.31.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2088ms
rtt min/avg/max/mdev = 0.376/0.404/0.461/0.039 ms

[root@R1 ~]# ping -c3 10.0.3.3
ping: connect: Network is unreachable
[root@R1 ~]# ping -c3 10.0.23.3
ping: connect: Network is unreachable
[root@R1 ~]#
```

8.2.2. Настройка статических маршрутов

Для настройки маршрута необходимо добавить правило в таблицу маршрутизации, явно указав, через какого *известного* абонента необходимо отправлять пакет для достижения сети.

1. С помощью команды управления таблицами маршрутизации установите на R1 статический маршрут в сеть **`10.0.23.0/24`** через маршрутизатор R3

   ```
   [root@R1 ~]# ip route add 10.0.23.0/24 via 10.0.31.3
   [root@R1 ~]#
   ```

2. С помощью команды **`ping -c3 <dstIP>`** с R1 проверьте доступность обоих абонентов сети **`10.0.23.0/24`**

   ```
   [root@R1 ~]# ping -c3 10.0.23.3
   PING 10.0.23.3 (10.0.23.3) 56(84) bytes of data.
   64 bytes from 10.0.23.3: icmp_seq=1 ttl=64 time=0.399 ms
   64 bytes from 10.0.23.3: icmp_seq=2 ttl=64 time=0.425 ms
   64 bytes from 10.0.23.3: icmp_seq=3 ttl=64 time=0.271 ms

   --- 10.0.23.3 ping statistics ---
   3 packets transmitted, 3 received, 0% packet loss, time 2073ms
   rtt min/avg/max/mdev = 0.271/0.365/0.425/0.067 ms

   [root@R1 ~]# ping -c3 10.0.23.2
   PING 10.0.23.2 (10.0.23.2) 56(84) bytes of data.

   --- 10.0.23.2 ping statistics ---
   3 packets transmitted, 0 received, 100% packet loss, time 2065ms

   [root@R1 ~]#
   ```

   Доступ к сети со стороны маршрутизатора R3 появился, однако при попытке отправить пакет в описанную сеть на R2 пакеты теряются.

   Причина этому — запрет маршрутизации по умолчанию между сетевыми интерфейсами. Для передачи необходимо либо настраивать на интерфейсном уровне сетевой мост, явно объединяющий интерфейсы, либо разрешить *пробрасывание пакетов на сетевом уровне* (IP forwarding). Для этого необходимо настроить параметр **`net.ipv4.conf.all.forwarding`**.

3. С помощью команды **`sysctl <*parameters>`** выведите текущее значение IP Forwarding флага, после чего установите его.

```
[root@R3 ~]# sysctl net.ipv4.conf.all.forwarding
net.ipv4.conf.all.forwarding = 0
[root@R3 ~]# sysctl net.ipv4.conf.all.forwarding=1
[root@R3 ~]#
```

Примечание

После данной настройки пакет будет доставлен с **R1:10.0.31.1** на **R2:10.0.23.2**, и, поскольку исходящий IP для **R2** лежит в неизвестной сети, ответ передаться не сможет.

```
[root@R1 ~]# ping -c3 10.0.23.2
PING 10.0.23.2 (10.0.23.2) 56(84) bytes of data.

--- 10.0.23.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2059ms

[root@R1 ~]#
```

4. С помощью команды управления таблицами маршрутизации установите на R2 статический маршрут в сеть **10.0.31.0/24** через маршрутизатор R1

```
[root@R2 ~]# ip route add 10.0.31.0/24 via 10.0.12.1
[root@R2 ~]#
```

5. С помощью команды **ping -c3 <dstIP>** на R1 проверьте доступность R2 из сети **10.0.23.0/24**

```
[root@R1 ~]# ping -c3 10.0.23.2
PING 10.0.23.2 (10.0.23.2) 56(84) bytes of data.
64 bytes from 10.0.23.2: icmp_seq=1 ttl=64 time=0.509 ms
64 bytes from 10.0.23.2: icmp_seq=2 ttl=64 time=0.563 ms
64 bytes from 10.0.23.2: icmp_seq=3 ttl=64 time=0.515 ms

--- 10.0.23.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2063ms
rtt min/avg/max/mdev = 0.509/0.529/0.563/0.024 ms
[root@R1 ~]#
```

8.2.3. Анализ статических маршрутов трафика в сети

Примечание

Заметьте, что пакеты при текущей настройке делают «круг» по топологии: ICMP Request от **R1** идёт по сетям **10.0.31.0** и **10.0.23.0**, а ICMP Reply от **R2** идёт через **10.0.12.0**.

1. С помощью команды мониторинга сети запустите на R3 сканирование интерфейса eth1 с выводом кода пакета

```
[root@R3 ~]# tcpdump -xx -i eth1
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

2. С помощью команды мониторинга сети запустите на R2 сканирование интерфейса eth2 с выводом кода пакета

```
[root@R2 ~]# tcpdump -xx -i eth2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

3. С помощью команды **`ping -c3 <dstIP>`** отправьте три ICMP‑пакета с R1 на R2 в сеть **`10.0.23.0/24`**

```
[root@R1 ~]# ping -c3 10.0.23.2
PING 10.0.23.2 (10.0.23.2) 56(84) bytes of data.
64 bytes from 10.0.23.2: icmp_seq=1 ttl=64 time=0.647 ms
64 bytes from 10.0.23.2: icmp_seq=2 ttl=64 time=0.600 ms
64 bytes from 10.0.23.2: icmp_seq=3 ttl=64 time=0.593 ms

--- 10.0.23.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2033ms
rtt min/avg/max/mdev = 0.593/0.613/0.647/0.023 ms
[root@R1 ~]#
```

Примечание

На R3 будут видны только ICMP‑запросы, на R2 — только ICMP‑ответы:

```
[root@R3 ~]# tcpdump -xx -i eth1
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
15:03:11.208377 IP 10.0.31.1 > 10.0.23.2: ICMP echo request, id 20, seq 1,
length 64
       0x0000:  0800 2721 3d8f 0800 2718 1fbc 0800 4500
       0x0010:  0054 98d3 4000 4001 57d3 0a00 1f01 0a00
       0x0020:  1702 0800 4263 0014 0001 ae13 d068 0000
       0x0030:  0000 6b38 0d00 0000 0000 1011 1213 1415
       0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
       0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
       0x0060:  3637
15:03:12.216848 IP 10.0.31.1 > 10.0.23.2: ICMP echo request, id 20, seq 2,
length 64
       0x0000:  0800 2721 3d8f 0800 2718 1fbc 0800 4500
       0x0010:  0054 9c52 4000 4001 5454 0a00 1f01 0a00
       0x0020:  1702 0800 743f 0014 0002 af13 d068 0000
       0x0030:  0000 385b 0d00 0000 0000 1011 1213 1415
       0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
       0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
       0x0060:  3637
15:03:13.240997 IP 10.0.31.1 > 10.0.23.2: ICMP echo request, id 20, seq 3,
length 64
       0x0000:  0800 2721 3d8f 0800 2718 1fbc 0800 4500
       0x0010:  0054 9f28 4000 4001 517e 0a00 1f01 0a00
       0x0020:  1702 0800 d0e0 0014 0003 b013 d068 0000
```

```
        0x0030:  0000 dab8 0d00 0000 0000 1011 1213 1415
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
        0x0060:  3637
```

```
[root@R2 ~]# tcpdump -xx -i eth2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), snapshot length 262144 bytes
15:03:11.713981 IP R2 > 10.0.31.1: ICMP echo reply, id 20, seq 1, length 64
        0x0000:  0800 2715 a82d 0800 2745 0741 0800 4500
        0x0010:  0054 0c14 0000 4001 2493 0a00 1702 0a00
        0x0020:  1f01 0000 4a63 0014 0001 ae13 d068 0000
        0x0030:  0000 6b38 0d00 0000 0000 1011 1213 1415
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
        0x0060:  3637
15:03:12.722582 IP R2 > 10.0.31.1: ICMP echo reply, id 20, seq 2, length 64
        0x0000:  0800 2715 a82d 0800 2745 0741 0800 4500
        0x0010:  0054 0eff 0000 4001 21a8 0a00 1702 0a00
        0x0020:  1f01 0000 7c3f 0014 0002 af13 d068 0000
        0x0030:  0000 385b 0d00 0000 0000 1011 1213 1415
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
        0x0060:  3637
15:03:13.746411 IP `R2` > 10.0.31.1: ICMP echo reply, id 20, seq 3, length
64
        0x0000:  0800 2715 a82d 0800 2745 0741 0800 4500
        0x0010:  0054 0fb4 0000 4001 20f3 0a00 1702 0a00
        0x0020:  1f01 0000 d8e0 0014 0003 b013 d068 0000
        0x0030:  0000 dab8 0d00 0000 0000 1011 1213 1415
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
        0x0060:  3637
```

4. С помощью команды управления таблицами маршрутизации установите статические маршруты и настройки IP-Forwarding на маршрутизаторах для организации полной связности IP-адресов *не LoopBack-интерфейсов*

```
[root@R1 ~]# sysctl net.ipv4.conf.all.forwarding=1
[root@R1 ~]#
```

```
[root@R2 ~]# sysctl net.ipv4.conf.all.forwarding=1
[root@R1 ~]#
```

```
[root@R3 ~]# ip route add 10.0.12.0/24 via 10.0.23.2
[root@R3 ~]#
```

5. С помощью команды **`ping -c3 <dstIP>`** отправьте три ICMP-пакета с R3 на R1 в сеть **`10.0.12.0/24`**

```
[root@R3 ~]# ping -c3 10.0.12.1
PING 10.0.12.1 (10.0.12.1) 56(84) bytes of data.
64 bytes from 10.0.12.1: icmp_seq=1 ttl=64 time=0.618 ms
64 bytes from 10.0.12.1: icmp_seq=2 ttl=64 time=1.01 ms
64 bytes from 10.0.12.1: icmp_seq=3 ttl=64 time=0.753 ms
```

```
--- 10.0.12.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2039ms
rtt min/avg/max/mdev = 0.618/0.795/1.014/0.164 ms
[root@R3 ~]#
```

8.2.4. Настройка маршрутов по умолчанию

Настройка явных статических маршрутов обеспечивает связность IP-сетей только для указанных маршрутов, в то же время доступ к LoopBack-сетям всё также отсутствует.

1. С помощью команды **`ping -c3 <dstIP>`** попробуйте отправить три ICMP-пакета с R1 на R3 в сеть **`10.0.3.0/24`**

   ```
   [root@R1 ~]# ping -c3 10.0.3.3
   ping: connect: Network is unreachable
   [root@R1 ~]#
   ```

   Для описания маршрутов во все явно не описанные сети используется статическая маршрутизация по умолчанию.

   Поскольку в таблице маршрутизации приоритет маршрутизации определяется маской сети (чем меньше маска, тем меньше приоритет у выбора маршрута), а маршрут по умолчанию описывает сеть вида **`0.0.0.0/0`** с нулевой маской, по этому маршруту будут оправляться пакеты только при отсутствии любых других доступных по таблице маршрутов.

2. С помощью команды управления таблицами маршрутизации установите статический маршрут по умолчанию на R1 через маршрутизатор R2 сети **`10.0.12.0/24`**

   ```
   [root@R1 ~]# ip route add default via 10.0.12.2
   [root@R1 ~]#
   ```

3. С помощью команды управления таблицами маршрутизации установите статический маршрут по умолчанию на R2 через маршрутизатор R3 сети **`10.0.23.0/24`**

   ```
   [root@R2 ~]# ip route add default via 10.0.23.3
   [root@R2 ~]#
   ```

   Примечание

   ICMP-ответы в данном случае будут передаваться по тому же маршруту, по которому передаются ICMP-запросы.

4. С помощью команды мониторинга сети запустите на R2 сканирование интерфейса eth2 с выводом кода пакета

   ```
   [root@R2 ~]# tcpdump -xx -i eth2
   tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
   listening on eth2, link-type EN10MB (Ethernet), snapshot length 262144 bytes
   ```

5. С помощью команды **`ping -c3 <dstIP>`** попробуйте отправить три ICMP-пакета с R1 на R3 в сеть **`10.0.3.0/24`**

```
[root@R1 ~]# ping -c3 10.0.3.3
PING 10.0.3.3 (10.0.3.3) 56(84) bytes of data.
64 bytes from 10.0.3.3: icmp_seq=1 ttl=63 time=1.07 ms
64 bytes from 10.0.3.3: icmp_seq=2 ttl=63 time=1.07 ms
64 bytes from 10.0.3.3: icmp_seq=3 ttl=63 time=0.637 ms

--- 10.0.3.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.637/0.925/1.069/0.203 ms
[root@R1 ~]#
```

```
[root@R2 ~]# tcpdump -xx -i eth2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), snapshot length 262144 bytes
15:14:40.511636 IP 10.0.12.1 > 10.0.3.3: ICMP echo request, id 22, seq 1,
length 64
       0x0000:  0800 2745 0741 0800 2715 a82d 0800 4500
       0x0010:  0054 ef1e 4000 4001 2887 0a00 0c01 0a00
       0x0020:  0303 0800 5774 0016 0001 5f16 d068 0000
       0x0030:  0000 a822 0a00 0000 0000 1011 1213 1415
       0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
       0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
       0x0060:  3637
15:14:40.512008 IP 10.0.3.3 > 10.0.12.1: ICMP echo reply, id 22, seq 1,
length 64
       0x0000:  0800 2715 a82d 0800 2745 0741 0800 4500
       0x0010:  0054 c6ce 0000 3f01 91d7 0a00 0303 0a00
       0x0020:  0c01 0000 5f74 0016 0001 5f16 d068 0000
       0x0030:  0000 a822 0a00 0000 0000 1011 1213 1415
       0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
       0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
       0x0060:  3637
15:14:41.511935 IP 10.0.12.1 > 10.0.3.3: ICMP echo request, id 22, seq 2,
length 64
       0x0000:  0800 2745 0741 0800 2715 a82d 0800 4500
       0x0010:  0054 f081 4000 4001 2724 0a00 0c01 0a00
       0x0020:  0303 0800 346f 0016 0002 6016 d068 0000
       0x0030:  0000 ca26 0a00 0000 0000 1011 1213 1415
       0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
       0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
       0x0060:  3637
15:14:41.512459 IP 10.0.3.3 > 10.0.12.1: ICMP echo reply, id 22, seq 2,
length 64
       0x0000:  0800 2715 a82d 0800 2745 0741 0800 4500
       0x0010:  0054 c7d3 0000 3f01 90d2 0a00 0303 0a00
       0x0020:  0c01 0000 3c6f 0016 0002 6016 d068 0000
       0x0030:  0000 ca26 0a00 0000 0000 1011 1213 1415
       0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
       0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
       0x0060:  3637
15:14:42.513129 IP 10.0.12.1 > 10.0.3.3: ICMP echo request, id 22, seq 3,
length 64
       0x0000:  0800 2745 0741 0800 2715 a82d 0800 4500
       0x0010:  0054 f22e 4000 4001 2577 0a00 0c01 0a00
       0x0020:  0303 0800 476a 0016 0003 6116 d068 0000
       0x0030:  0000 b62a 0a00 0000 0000 1011 1213 1415
       0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
       0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
```

```
        0x0060:  3637
15:14:42.513792 IP 10.0.3.3 > 10.0.12.1: ICMP echo reply, id 22, seq 3,
length 64
        0x0000:  0800 2715 a82d 0800 2745 0741 0800 4500
        0x0010:  0054 cacf 0000 3f01 8dd6 0a00 0303 0a00
        0x0020:  0c01 0000 4f6a 0016 0003 6116 d068 0000
        0x0030:  0000 b62a 0a00 0000 0000 1011 1213 1415
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
        0x0060:  3637
```

6. С помощью команды управления таблицами маршрутизации установите статический маршрут по умолчанию на R3 через маршрутизатор R1 сети **10.0.31.0/24**

```
[root@R3 ~]# ip route add default via 10.0.31.1
[root@R3 ~]#
```

7. С помощью команды **ping -c3 <dstIP>** попробуйте отправить три ICMPпакета с R3 на R2 в сеть **10.0.2.0/24**

```
[root@R3 ~]# ping -c3 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=63 time=0.716 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=63 time=1.60 ms
64 bytes from 10.0.2.2: icmp_seq=3 ttl=63 time=0.588 ms

--- 10.0.2.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2065ms
rtt min/avg/max/mdev = 0.588/0.969/1.604/0.451 ms
[root@R3 ~]#
```

8.2.5. Анализ путей трафика в сети

Для анализа путей трафика удобно воспользоваться утилитой **traceroute**, которая последовательно отправляет пакеты с увеличивающимся **TTL** и по сигналам о сбрасывании пакета на участках сети строит маршрут пакета в сети.

Предупреждение

Строго говоря, поскольку путь каждого отдельного пакета независим, при изменении сети полученный маршрут не будет отвечать действительному и будет лишь охватывать какието маршрутизаторы, отстоящие от нашего абонента на указанное число «прыжков». Однако поскольку наша тестовая сеть неизменяема, «маршрут» в **traceroute** будет отвечать реальному маршруту пакета в сети.

1. С помощью команды *проверки доступности абонентов в сети* :new: и команды *отслеживания «маршрута» пакета в сети* :new: продемонстрируйте связь между R1 и R2 сети **10.0.23.0/24**

```
[root@R1 ~]# ping -c3 10.0.23.2
PING 10.0.23.2 (10.0.23.2) 56(84) bytes of data.
64 bytes from 10.0.23.2: icmp_seq=1 ttl=64 time=1.47 ms
64 bytes from 10.0.23.2: icmp_seq=2 ttl=64 time=0.830 ms
64 bytes from 10.0.23.2: icmp_seq=3 ttl=64 time=0.643 ms
```

```
--- 10.0.23.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.643/0.980/1.468/0.353 ms

[root@R1 ~]# traceroute 10.0.23.2
traceroute to 10.0.23.2 (10.0.23.2), 30 hops max, 60 byte packets
1  10.0.31.3 (10.0.31.3)  0.471 ms  0.697 ms  0.671 ms
2  10.0.23.2 (10.0.23.2)  0.746 ms  0.730 ms  0.742 ms
[root@R1 ~]#
```

2. С помощью команд мониторинга сети продемонстрируйте связь между R1 сети **10.0.1.0/24** и R3 сети **10.0.3.0/24**

```
[root@R1 ~]# ping -c3 -I 10.0.1.1 10.0.3.3
PING 10.0.3.3 (10.0.3.3) from 10.0.1.1 : 56(84) bytes of data.
64 bytes from 10.0.3.3: icmp_seq=1 ttl=64 time=1.10 ms
64 bytes from 10.0.3.3: icmp_seq=2 ttl=64 time=0.620 ms
64 bytes from 10.0.3.3: icmp_seq=3 ttl=64 time=0.768 ms

--- 10.0.3.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2020ms
rtt min/avg/max/mdev = 0.620/0.828/1.098/0.199 ms

[root@R1 ~]# traceroute -s 10.0.1.1 10.0.3.3
traceroute to 10.0.3.3 (10.0.3.3), 30 hops max, 60 byte packets
1  10.0.23.2 (10.0.23.2)  1.340 ms  1.283 ms  1.344 ms
2  10.0.3.3 (10.0.3.3)  1.226 ms  1.296 ms  1.274 ms
[root@R1 ~]#
```

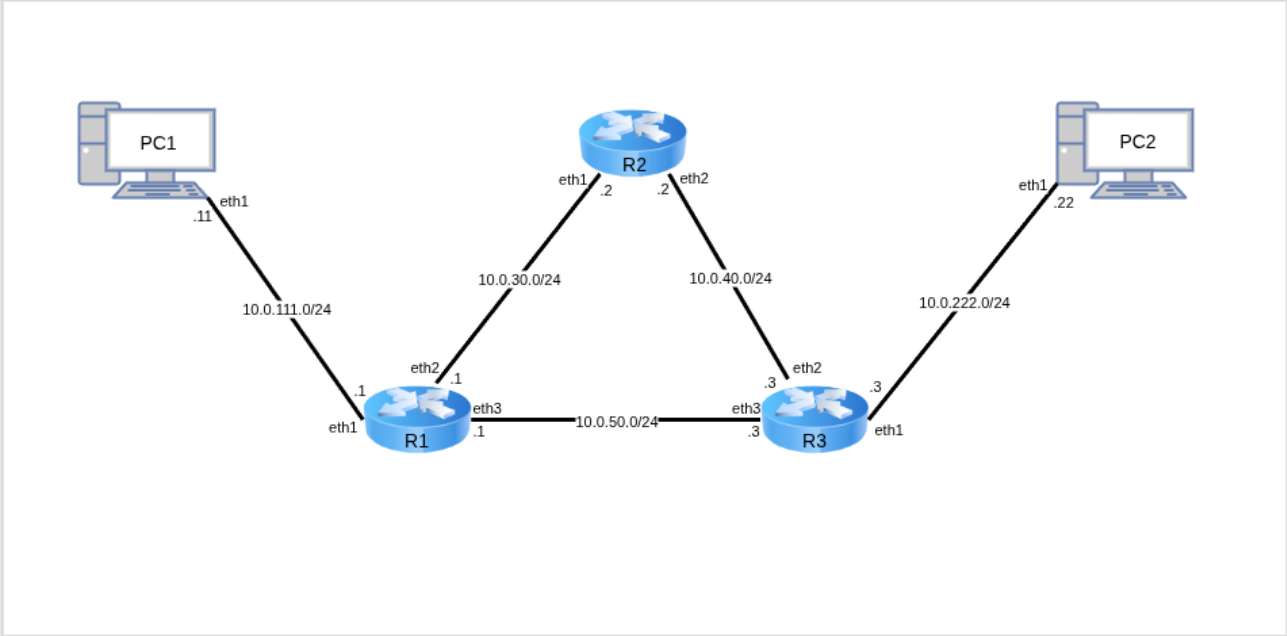
3. С помощью команд управления таблицами маршрутизации продемонстрируйте таблицы маршрутизации на всех абонентах

```
[root@R1 ~]# ip route list
default via 10.0.12.2 dev eth1
10.0.12.0/24 dev eth1 proto kernel scope link src 10.0.12.1
10.0.23.0/24 via 10.0.31.3 dev eth2
10.0.31.0/24 dev eth2 proto kernel scope link src 10.0.31.1
[root@R1 ~]#
```

```
[root@R2 ~]# ip route list
default via 10.0.23.3 dev eth1
10.0.12.0/24 dev eth2 proto kernel scope link src 10.0.12.2
10.0.23.0/24 dev eth1 proto kernel scope link src 10.0.23.2
10.0.31.0/24 via 10.0.12.1 dev eth2
[root@R2 ~]#
```

```
[root@R3 ~]# ip route list
default via 10.0.31.1 dev eth1
10.0.12.0/24 via 10.0.23.2 dev eth2
10.0.23.0/24 dev eth2 proto kernel scope link src 10.0.23.3
10.0.31.0/24 dev eth1 proto kernel scope link src 10.0.31.3
[root@R3 ~]#
```

8.3. Самостоятельная работа



Для работы необходимо 5 *клонов* согласно топологии сети. Для создания соединений между машинами необходимо в VirtualBox настроить сетевые интерфейсы (описание настройки подключения находится в соответствующем разделе второй лабораторной):

- **R1**:
  - **Adapter2** — **net111**
  - **Adapter3** — **net30**
  - **Adapter4** — **net50**
- **R2**:
  - **Adapter2** — **net30**
  - **Adapter3** — **net40**
- **R3**:
  - **Adapter2** — **net222**
  - **Adapter3** — **net40**
  - **Adapter4** — **net50**
- **PC1**:
  - **Adapter2** — **net111**
- **PC2**:
  - **Adapter2** — **net222**

8.3.1. Варианты заданий

Таблица 8.1. Варианты заданий

Вариант	Задание
1	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить статический маршрут между **R1** и **R3** 4. Убедиться, что **PC2** пингуется с **PC1** и наоборот 5. Выполнить **`traceroute`** с **PC1** на **PC2**
2	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить статические маршруты так, чтобы **PC2** не мог бы пинговать **R2**, а **PC1** мог бы 4. Выполнить **`traceroute`** с **PC1** и **PC2** на **R2**
3	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить статические маршруты так, чтобы **PC1** не мог бы пинговать **R2**, а **PC2** мог бы 4. Выполнить **`traceroute`** с **PC1** и **PC2** на **R2**
4	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить статические маршруты так, чтобы **PC1** не мог бы пинговать **R2**, но мог бы пинговать **R3** 4. Выполнить **`traceroute`** с **PC1** на **R2** и **R3**
5	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить статические маршруты так, чтобы **PC2** не мог бы пинговать **R2**, но мог бы пинговать **R1** 4. Выполнить **`traceroute`** с **PC2** на **R2** и **R1**
6	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить статические маршруты так, чтобы **PC1** не мог бы пинговать **R2**, но мог бы пинговать **PC2** 4. Выполнить **`traceroute`** с **PC1** на **PC2** и **R2**

Вариант	Задание
7	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить статические маршруты так, чтобы **PC2** не мог бы пинговать **R2**, но мог бы пинговать **PC1** 4. Выполнить **`traceroute`** с **PC2** на **PC1** и **R2**
8	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить статические маршруты так, чтобы **R2** мог бы пинговать **PC2** и **PC1** 4. Выполнить **`traceroute`** с **R2** на **PC2** и **PC1**
9	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить статические маршруты так, чтобы **R2** мог бы пинговать **PC2**, но не мог бы пинговать **PC1** 4. Выполнить **`traceroute`** с **R2** на **PC2** и **PC1**
10	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить статические маршруты так, чтобы **R2** мог бы пинговать **PC1**, но не мог бы пинговать **PC2** 4. Выполнить **`traceroute`** с **R2** на **PC2** и **PC1**

8.3.1.1. Задание

Запустить отчёты на каждой машине и выполнить соответствующие команды.

```
report 8 pc1
```

1. **`ip a show eth1`**.
2. **`ip route`**.
3. **`ping -fc3 10.0.50.1`**.
4. **`ping -fc3 10.0.30.2`**.
5. **`ping -fc3 10.0.40.3`**.

6. **ping -fc3 10.0.222.22**.
7. **traceroute 10.0.222.22**.
8. **traceroute 10.0.40.2**.

```
report 8 pc2
```

1. **ip a show eth1**.
2. **ip route**.
3. **ping -fc3 10.0.50.1**.
4. **ping -fc3 10.0.30.2**.
5. **ping -fc3 10.0.40.3**.
6. **ping -fc3 10.0.111.11**.
7. **traceroute 10.0.111.11**.
8. **traceroute 10.0.40.2**.

```
report 8 r1
```

1. **ip a show**.
2. **ip route**.
3. **ping -fc3 10.0.30.2**.
4. **ping -fc3 10.0.40.3**.
5. **ping -fc3 10.0.111.11**.
6. **ping -fc3 10.0.222.22**.
7. **traceroute 10.0.222.22**.
8. **traceroute 10.0.111.11**.

```
report 8 r2
```

1. **ip a show**.
2. **ip route**.
3. **ping -fc3 10.0.50.1**.
4. **ping -fc3 10.0.40.3**.
5. **ping -fc3 10.0.111.11**.
6. **ping -fc3 10.0.222.22**.
7. **traceroute 10.0.222.22**.

8. **`traceroute 10.0.111.11`**.

```
report 8 r3
```

1. **`ip a show`**.
2. **`ip route`**.
3. **`ping -fc3 10.0.30.2`**.
4. **`ping -fc3 10.0.50.1`**.
5. **`ping -fc3 10.0.111.11`**.
6. **`ping -fc3 10.0.222.22`**.
7. **`traceroute 10.0.222.22`**.
8. **`traceroute 10.0.111.11`**.

Полученные отчёты **`report.08.pc1`**, **`report.08.pc2`**, **`report.08.r1`**, **`report.08.r2`**, **`report.08.r3`** через последовательный порт перенести из виртуальной машины и прислать их преподавателю с подписью выполненного варианта.

Глава 9. Маршрутизация с использованием RIP

Цель лабораторной работы — познакомить изучающего с основами протокола маршрутизации RIP.

Задачи лабораторной работы:

- Изучить логику работы протокола;
- Изучить агента маршрутизации BIRD для настройки RIP;
- Реализовать тестовую топологию с применением протокола маршрутизации RIP.

9.1. Протокол RIP

Протокол RIP (Routing Information Protocol) — Протокол маршрутизации по вектору расстояния. В рамках протокола каждый узел передаёт вектор расстояний (в качестве метрики расстояния по умолчанию используется количество hop'ов до соответствующего абонента) своим соседям, на основании полученных от них векторов обновляет свой согласно формуле, и в результате получает вектор расстояний до всех абонентов в сети.

Начальный вектор: {A: 1; B: 1; other: None}, где:

- A, B — Известные узлу абоненты/соседи;
- 1 — Метрика расстояния (в терминах количества hop'ов);

- other — поле, описывающее все остальные сети;
- None — Указание недостижимости сетей (по умолчанию значением недостижимости принято 16).

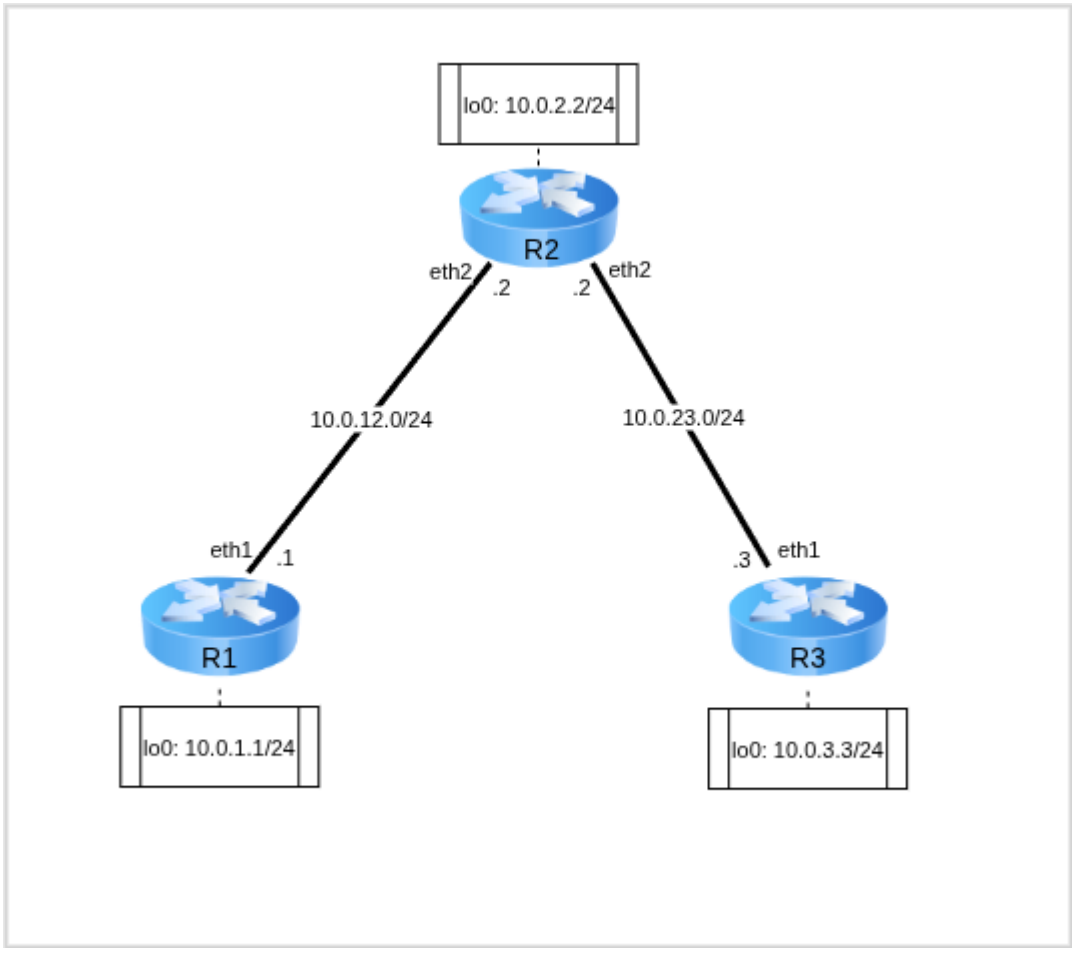
Формула обновления координат:

Dnew(self, j) = min{ D(self, j), D(self, k) + D(k, j) }, где:

- Dnew(self, j) — Новое значение расстояния от данного абонента до абонента j (координата RIP‑вектора self);
- D(self, j) — Текущее значение расстояния от данного абонента до абонента j (координата RIP‑вектора self);
- D(self, k) — Текущее значение расстояния от данного абонента до абонента k (координата RIP‑вектора self);
- D(k, j) — Значение расстояния от абонента k до абонента j (координата RIP‑вектора k).

9.2. Пример настройки RIP

Для изучения маршрутизации с помощью RIP разберём топологию с тремя маршрутизаторами.



Для работы создайте 3 *клона* согласно топологии сети. Для создания соединений между машинами необходимо в VirtualBox настроить сетевые интерфейсы (описание настройки подключения находится в соответствующем разделе второй лабораторной):

- **R1**:
  - **Adapter2** — **intnet**
- **R2**:
  - **Adapter2** — **intnet**
  - **Adapter3** — **deepnet**
- **R3**:
  - **Adapter2** — **deepnet**

9.2.1. Базовая настройка виртуальных машин

1. С помощью команд управления интерфейсами создайте и включите интерфейсы согласно топологии

```
[root@R1 ~]# ip link set eth1 up
[root@R1 ~]# ip link add dev lo0 type veth
[root@R1 ~]# ip link set lo0 up
[root@R1 ~]#
```

```
[root@R2 ~]# ip link set eth1 up
[root@R2 ~]# ip link set eth2 up
[root@R2 ~]# ip link add dev lo0 type veth
[root@R2 ~]# ip link set lo0 up
```

```
[root@R3 ~]# ip link set eth1 up
[root@R3 ~]# ip link add dev lo0 type veth
[root@R3 ~]# ip link set lo up
[root@R3 ~]#
```

2. С помощью команд настройки IP⁄адресов и настройки IP⁄Forwarding на маршрутизаторах установите адреса согласно топологии.

```
[root@R1 ~]# ip addr add dev eth1 10.0.12.1/24
[root@R1 ~]# ip addr add dev lo0 10.0.1.1/24
[root@R1 ~]#
```

```
[root@R2 ~]# ip addr add dev eth1 10.0.12.2/24
[root@R2 ~]# ip addr add dev eth2 10.0.23.2/24
[root@R2 ~]# ip addr add dev lo0 10.0.2.2/24
[root@R2 ~]# sysctl net.ipv4.conf.all.forwarding=1
[root@R2 ~]#
```

```
[root@R3 ~]# ip addr add dev eth1 10.0.23.3/24
[root@R3 ~]# ip addr add dev lo 10.0.3.3/24
[root@R3 ~]#
```

Вместе с описанием IP-адресов в таблице маршрутизации автоматически появятся записи о маршрутах в сетях описанных адресов.

3. С помощью команд управления таблицами маршрутизации на маршрутизаторах выведите данные всех таблиц маршрутизации.

```
[root@R1 ~]# ip route
10.0.1.0/24 dev lo0 proto kernel scope link src 10.0.1.1 linkdown
10.0.12.0/24 dev eth1 proto kernel scope link src 10.0.12.1
[root@R1 ~]#
```

```
[root@R2 ~]# ip route
10.0.2.0/24 dev lo0 proto kernel scope link src 10.0.2.2 linkdown
10.0.12.0/24 dev eth1 proto kernel scope link src 10.0.12.2
10.0.23.0/24 dev eth2 proto kernel scope link src 10.0.23.2
[root@R2 ~]#
```

```
[root@R3 ~]# ip route
10.0.3.0/24 dev lo0 proto kernel scope link src 10.0.3.3 linkdown
10.0.23.0/24 dev eth1 proto kernel scope link src 10.0.23.3
[root@R3 ~]#
```

9.2.2. BIRD Routing Daemon

Для работы протоколов (в частности, протоколов маршрутизации) необходимо использовать специальные программы-менеджеры, которые управляют настройками и параметрами протоколов — Routing Daemons. Для выполнения лабораторной используется демон BIRD. Для настройки демона используется конфигурационный файл **`/etc/bird/bird.conf`** специального вида.

9.2.3. Настройка RIP

1. Опишите (или скопируйте) конфигурационный файл для R1 в соответствующем файле виртуальной машины

   `@R1:/etc/bird/bird.conf`

```
router id 10.0.1.1;

protocol kernel {
   learn all;
   ipv4 { export all; };
}

protocol device {
   scan time 10;
}

protocol rip {
   interface "eth1", "lo0";
   ipv4 {
       import all;
       export where (net = 10.0.1.0/24) || (net = 10.0.12.0/24);
   };
}
```

Для работы демона необходимо указывать ключевые параметры конфигурации:

- Описание уникального идентификатора маршрутизатора в сети, «от имени» которого будут рассылаться данные о маршрутах;
- Структуру **`protocol kernel`** — она описывает действия, связанные с таблицами маршрутизации ядра системы. Необходимо указать сохранение получаемых данных в таблицах маршрутизации устройства (а не просто передачу этих данных), а также частоту обновления таблиц получаемыми данными;
- Структуру **`protocol device`** — она описывает действия самого сетевого устройства. Необходимо указать периодичность сканирования портов на наличие BIRD-данных от других устройств;
- Структуру **`protocol rip`** — она описывает действия, связанные с маршрутизацией с помощью протокола:
  - Указание интерфейсов для приёма/передачи данных RIP;
  - Экспорт и импорт маршрутов по IPv4 согласно описанным правилам.

2. Аналогично опишите (или скопируйте) конфигурационные файлы для R2 и R3.

@R2:/etc/bird/bird.conf

```
router id 10.0.2.2;

protocol kernel {
   learn all;
   ipv4 { export all; };
}

protocol device {
   scan time 10;
}

protocol rip {
   interface "eth*";
   ipv4 {
       import where net != 10.0.1.0/24;
       export all;
   };
}
```

@R3:/etc/bird/bird.conf

```
router id 10.0.3.3;

protocol kernel {
       learn all;
       ipv4 { export all; };
}

protocol device {
   scan time 10;
}

protocol rip {
```

```
    interface "*";
    ipv4 {
        import all;
        export all;
    };
}
```

Заметьте, что **R2** не будет принимать данные, связанные с сетью **10.0.1.0/24**, и, соответственно, не будет отправлять их далее по топологии.

3. С помощью команды **bird** запустите BIRD на *каждом* из устройств.

   С помощью команды **birdc** можно посмотреть параметры работы демона.

4. С помощью команды **birdc show route** посмотрите таблицу RIP-маршрутов, передаваемых BIRD.

```
[root@R1 ~]# bird
[root@R1 ~]# birdc show route
BIRD +detached. ready.
Table master4:
10.0.1.0/24          unicast [kernel1 12:25:54.420] * (10)
       dev lo0
10.0.12.0/24         unicast [kernel1 12:25:54.420] * (10)
       dev eth1
10.0.23.0/24         unicast [rip1 12:26:00.124] * (120/2)
       via 10.0.12.2 on eth1
10.0.2.0/24          unicast [rip1 12:26:00.124] * (120/2)
       via 10.0.12.2 on eth1
10.0.3.0/24          unicast [rip1 12:26:02.896] * (120/3)
       via 10.0.12.2 on eth1
[root@R1 ~]#
```

   В таблице маршрутизации после запуска должны появиться новые записи о доступных маршрутах с указанием **proto bird**, означающим, что маршрут получен с помощью BIRD-демона (поскольку маршруты не приходят мгновенно, может потребоваться время на получение всех данных).

5. С помощью команды управления таблицами маршрутизации убедитесь, что таблицы маршрутизации на устройствах изменились в соответствии с данными от BIRD.

```
[root@R1 ~]# ip route
10.0.1.0/24 dev lo0 proto kernel scope link src 10.0.1.1 linkdown
10.0.2.0/24 via 10.0.12.2 dev eth1 proto bird metric 32
10.0.3.0/24 via 10.0.12.2 dev eth1 proto bird metric 32
10.0.12.0/24 dev eth1 proto kernel scope link src 10.0.12.1
10.0.23.0/24 via 10.0.12.2 dev eth1 proto bird metric 32
[root@R1 ~]#
```

```
[root@R2 ~]# ip route
10.0.2.0/24 dev lo0 proto kernel scope link src 10.0.2.2 linkdown
10.0.3.0/24 via 10.0.23.3 dev eth2 proto bird metric 32
10.0.12.0/24 dev eth1 proto kernel scope link src 10.0.12.2
10.0.12.0/24 via 10.0.12.1 dev eth1 proto bird metric 32
10.0.23.0/24 dev eth2 proto kernel scope link src 10.0.23.2
[root@R2 ~]#
```

```
[root@R3 ~]# ip route
10.0.2.0/24 via 10.0.23.2 dev eth1 proto bird metric 32
10.0.3.0/24 dev lo0 proto kernel scope link src 10.0.3.3 linkdown
10.0.12.0/24 via 10.0.23.2 dev eth1 proto bird metric 32
10.0.23.0/24 dev eth1 proto kernel scope link src 10.0.23.3
10.0.23.0/24 via 10.0.23.2 dev eth1 proto bird metric 32
[root@R3 ~]#
```

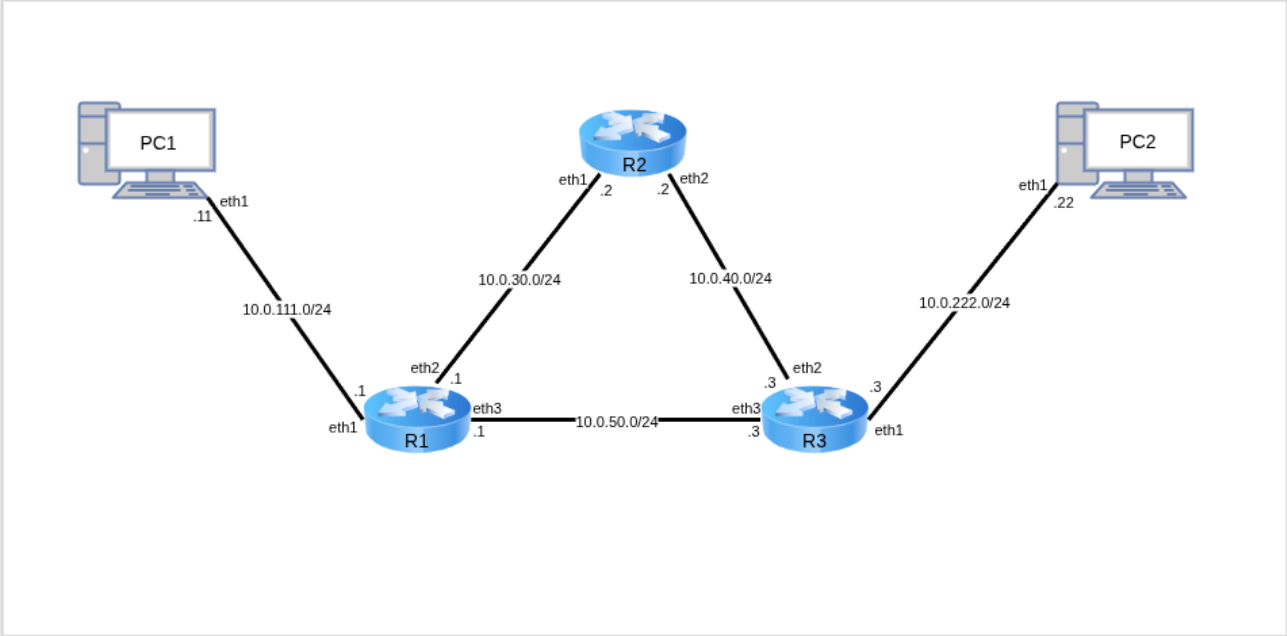
Заметьте, что информация о маршруте в сеть **10.0.1.0/24** не передалась на **R2** и **R3**, поскольку не была указана как допустимая к передаче.

6. С помощью команды **ping -c3 <dstIP>** проверьте достижимость отдельных сетей с разных устройств.

```
[root@R1 ~]# ping -c3 10.0.23.3
PING 10.0.23.3 (10.0.23.3) 56(84) bytes of data.
64 bytes from 10.0.23.3: icmp_seq=1 ttl=63 time=1.00 ms
64 bytes from 10.0.23.3: icmp_seq=2 ttl=63 time=0.872 ms
64 bytes from 10.0.23.3: icmp_seq=3 ttl=63 time=0.854 ms

--- 10.0.23.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.854/0.908/1.000/0.065 ms
[root@R1 ~]# ping -c3 10.0.3.3
PING 10.0.3.3 (10.0.3.3) 56(84) bytes of data.
64 bytes from 10.0.3.3: icmp_seq=1 ttl=63 time=0.903 ms
64 bytes from 10.0.3.3: icmp_seq=2 ttl=63 time=0.838 ms
64 bytes from 10.0.3.3: icmp_seq=3 ttl=63 time=0.990 ms

--- 10.0.3.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2055ms
rtt min/avg/max/mdev = 0.838/0.910/0.990/0.062 ms
[root@R1 ~]#
```

```
[root@R3 ~]# ping -c3 10.0.1.1
ping: connect: Network is unreachable
[root@R3 ~]# ping -c3 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=64 time=0.591 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=64 time=0.448 ms
64 bytes from 10.0.2.2: icmp_seq=3 ttl=64 time=0.567 ms

--- 10.0.2.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2060ms
rtt min/avg/max/mdev = 0.448/0.535/0.591/0.062 ms
[root@R3 ~]#
```

9.3. Самостоятельная работа



Для работы необходимо 5 *клонов* согласно топологии сети. Для создания соединений между машинами необходимо в VirtualBox настроить сетевые интерфейсы (описание настройки подключения находится в соответствующем разделе второй лабораторной):

- **R1**:
  - **Adapter2** — **net111**
  - **Adapter3** — **net30**
  - **Adapter4** — **net50**
- **R2**:
  - **Adapter2** — **net30**
  - **Adapter3** — **net40**
- **R3**:
  - **Adapter2** — **net222**
  - **Adapter3** — **net40**
  - **Adapter4** — **net50**
- **PC1**:
  - **Adapter2** — **net111**
- **PC2**:
  - **Adapter2** — **net222**

9.3.1. Варианты заданий

Таблица 9.1. Варианты заданий

Вариант	Задание
1	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить RIP между **R1** и **R3** 4. Убедиться, что **PC2** пингуется с **PC1** и наоборот 5. Выполнить **`traceroute`** с **PC1** на **PC2**
2	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить RIP так, чтобы **PC2** не мог бы пинговать **R2**, а **PC1** мог бы 4. Выполнить **`traceroute`** с **PC1** и **PC2** на **R2**
3	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить RIP так, чтобы **PC1** не мог бы пинговать **R2**, а **PC2** мог бы 4. Выполнить **`traceroute`** с **PC1** и **PC2** на **R2**
4	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить RIP так, чтобы **PC1** не мог бы пинговать **R2**, но мог бы пинговать **R3** 4. Выполнить **`traceroute`** с **PC1** на **R2** и **R3**
5	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить RIP так, чтобы **PC2** не мог бы пинговать **R2**, но мог бы пинговать **R1** 4. Выполнить **`traceroute`** с **PC2** на **R2** и **R1**
6	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить RIP так, чтобы **PC1** не мог бы пинговать **R2**, но мог бы пинговать **PC2** 4. Выполнить **`traceroute`** с **PC1** на **PC2** и **R2**
7	1. Создать топологию, указанную на рисунке

Вариант	Задание
	2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить RIP так, чтобы **PC2** не мог бы пинговать **R2**, но мог бы пинговать **PC1** 4. Выполнить **traceroute** с **PC2** на **PC1** и **R2**
8	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить RIP так, чтобы **R2** мог бы пинговать **PC2** и **PC1** 4. Выполнить **traceroute** с **R2** на **PC2** и **PC1**
9	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить RIP так, чтобы **R2** мог бы пинговать **PC2**, но не мог бы пинговать **PC1** 4. Выполнить **traceroute** с **R2** на **PC2** и **PC1**
10	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить RIP так, чтобы **R2** мог бы пинговать **PC1**, но не мог бы пинговать **PC2** 4. Выполнить **traceroute** с **R2** на **PC2** и **PC1**

9.3.1.1. Задание

Запустить отчёты на каждой машине и выполнить соответствующие команды.

```
report 9 pc1
```

1. **ip a show eth1**.
2. **ip route**.
3. **cat /etc/bird/bird.conf**.
4. **ping -fc3 10.0.50.1**.
5. **ping -fc3 10.0.30.2**.
6. **ping -fc3 10.0.40.3**.
7. **ping -fc3 10.0.222.22**.
8. **traceroute 10.0.222.22**.
9. **traceroute 10.0.40.2**.

`report 9 pc2`

1. **`ip a show eth1`**.
2. **`ip route`**.
3. **`cat /etc/bird/bird.conf`**.
4. **`ping -fc3 10.0.50.1`**.
5. **`ping -fc3 10.0.30.2`**.
6. **`ping -fc3 10.0.40.3`**.
7. **`ping -fc3 10.0.111.11`**.
8. **`traceroute 10.0.111.11`**.
9. **`traceroute 10.0.40.2`**.

`report 9 r1`

1. **`ip a show`**.
2. **`ip route`**.
3. **`cat /etc/bird/bird.conf`**.
4. **`ping -fc3 10.0.30.2`**.
5. **`ping -fc3 10.0.40.3`**.
6. **`ping -fc3 10.0.111.11`**.
7. **`ping -fc3 10.0.222.22`**.
8. **`traceroute 10.0.222.22`**.
9. **`traceroute 10.0.111.11`**.

`report 9 r2`

1. **`ip a show`**.
2. **`ip route`**.
3. **`cat /etc/bird/bird.conf`**.
4. **`ping -fc3 10.0.50.1`**.
5. **`ping -fc3 10.0.40.3`**.
6. **`ping -fc3 10.0.111.11`**.
7. **`ping -fc3 10.0.222.22`**.
8. **`traceroute 10.0.222.22`**.

9. **`traceroute 10.0.111.11`**.

```
report 9 r3
```

1. **`ip a show`**.
2. **`ip route`**.
3. **`cat /etc/bird/bird.conf`**.
4. **`ping -fc3 10.0.30.2`**.
5. **`ping -fc3 10.0.50.1`**.
6. **`ping -fc3 10.0.111.11`**.
7. **`ping -fc3 10.0.222.22`**.
8. **`traceroute 10.0.222.22`**.
9. **`traceroute 10.0.111.11`**.

Полученные отчёты **`report.09.pc1`**, **`report.09.pc2`**, **`report.09.r1`**, **`report.09.r2`**, **`report.09.r3`** через последовательный порт перенести из виртуальной машины и прислать их преподавателю с подписью выполненного варианта.

Глава 10. Маршрутизация с использованием OSPF

Цель лабораторной работы — познакомить изучающего с основами протокола маршрутизации OSPF.

Задачи лабораторной работы:

- Изучить логику работы протокола;
- Изучить агента маршрутизации BIRD для настройки OSPF;
- Реализовать тестовую топологию с применением протокола маршрутизации OSPF.

10.1. Протокол OSPF

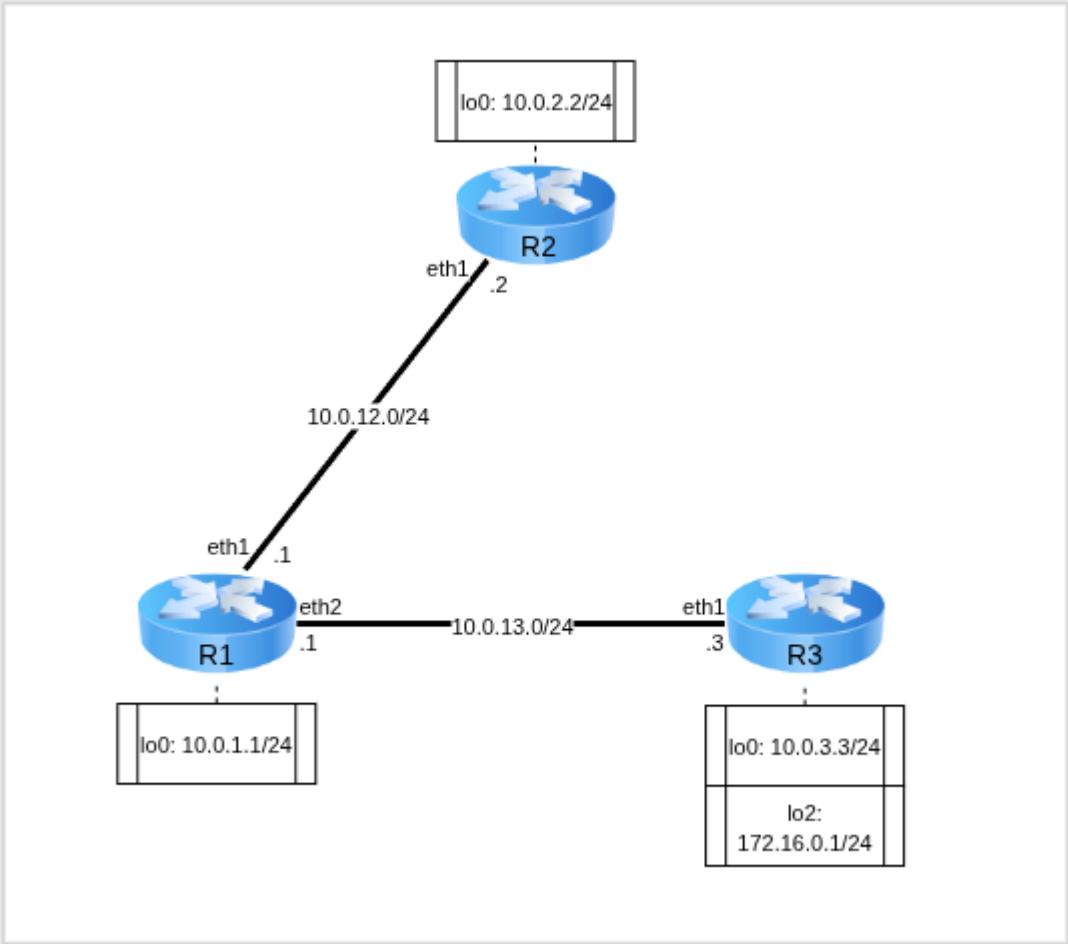
Протокол OSPF (Open Shortest Path First) — протокол маршрутизации по состоянию канала.

Глобально протокол можно разделить на два этапа — *обмен данными* и *построение маршрутов*. Данные этапы могут конвейерным образом сочетаться.

При обмене данными каждый узел передаёт своим соседям данные заголовков LSA (Link State Advertisement) интерфейсов из своей базы данных. При получении новых записей или обновления данных о каком-то канале узел запоминает эти данные и запрашивает полные данные LSA. На основании полных данных строится взвешенный ориентированный остовный граф всей сетевой топологии, веса рассчитываются согласно метрике *стоимости* канала (некоторой величины на базе информации о состоянии канала).

После получения полных данных, на этапе построения маршрутов, используется алгоритм Дейкстры для построения связного ориентированного остовного дерева маршрутов с минимальной стоимостью.

10.2. Пример настройки OSPF



Для работы создайте 3 *клона* согласно топологии сети. Для создания соединений между машинами необходимо в VirtualBox настроить сетевые интерфейсы (описание настройки подключения находится в соответствующем разделе второй лабораторной):

- **R1**:
  - **Adapter2** — **intnet**
  - **Adapter3** — **deepnet**
- **R2**:
  - **Adapter2** — **intnet**

- **R3**:
  - **Adapter2 — deepnet**

10.2.1. Базовая настройка виртуальных машин

1. С помощью команд управления интерфейсами создайте и включите интерфейсы согласно топологии.

```
[root@R1 ~]# ip link set eth1 up
[root@R1 ~]# ip link set eth2 up
[root@R1 ~]# ip link add dev lo0 type veth
[root@R1 ~]# ip link set lo0 up
[root@R1 ~]#
```

```
[root@R2 ~]# ip link set eth1 up
[root@R2 ~]# ip link add dev lo0 type veth
[root@R2 ~]# ip link set lo0 up
[root@R2 ~]#
```

```
[root@R3 ~]# ip link set eth1 up
[root@R3 ~]# ip link add dev lo0 type veth
[root@R3 ~]# ip link add dev lo2 type veth
[root@R3 ~]# ip link set lo0 up
[root@R3 ~]# ip link set lo2 up
[root@R3 ~]#
```

2. С помощью команд настройки IPv4 адресов и настройки IPv4 Forwarding на маршрутизаторах установите адреса согласно топологии.

```
[root@R1 ~]# ip addr add dev eth1 10.0.12.1/24
[root@R1 ~]# ip addr add dev eth2 10.0.13.1/24
[root@R1 ~]# ip addr add dev lo0 10.0.1.1/24
[root@R1 ~]#
[root@R1 ~]# sysctl net.ipv4.conf.all.forwarding=1
[root@R1 ~]#
```

```
[root@R2 ~]# ip addr add dev eth1 10.0.12.2/24
[root@R2 ~]# ip addr add dev lo0 10.0.2.2/24
[root@R2 ~]#
```

```
[root@R3 ~]# ip addr add dev eth1 10.0.13.3/24
[root@R3 ~]# ip addr add dev lo0 10.0.3.3/24
[root@R3 ~]# ip addr add dev lo2 172.16.0.1/24
[root@R3 ~]#
```

3. С помощью команд управления таблицами маршрутизации на маршрутизаторах выведите данные всех таблиц маршрутизации.

```
[root@R1 ~]# ip route
10.0.1.0/24 dev lo0 proto kernel scope link src 10.0.1.1 linkdown
10.0.12.0/24 dev eth1 proto kernel scope link src 10.0.12.1
10.0.13.0/24 dev eth2 proto kernel scope link src 10.0.13.1
```

```
[root@R2 ~]# ip route
10.0.2.0/24 dev lo0 proto kernel scope link src 10.0.2.2 linkdown
10.0.12.0/24 dev eth1 proto kernel scope link src 10.0.12.2
```

```
[root@R3 ~]# ip route
10.0.3.0/24 dev lo0 proto kernel scope link src 10.0.3.3 linkdown
10.0.13.0/24 dev eth1 proto kernel scope link src 10.0.13.3
172.16.0.0/24 dev lo2 proto kernel scope link src 172.16.0.1 linkdown
```

10.2.2. BIRD Routing Daemon

Для работы протоколов (в частности, протоколов маршрутизации) необходимо использовать специальные программы/менеджеры, которые управляют настройками и параметрами протоколов — Routing Daemons. Для выполнения лабораторной используется демон BIRD. Для настройки демона используется конфигурационный файл **`/etc/bird/bird.conf`** специального вида.

10.2.3. Настройка OSPF

1. Опишите (или скопируйте) конфигурационный файл для R1 в соответствующем файле виртуальной машины.

   `@R1:/etc/bird/bird.conf`

   ```
   router id 10.0.1.1;

   protocol kernel {
          scan time 20;
          ipv4 { export all; };
   }

   protocol device {
          scan time 10;
   }

   protocol ospf SIMPLE {
          ipv4 { export all; };
          area 0.0.0.0 {
                  interface "eth1" {
                  };
                  interface "eth2" {
                  };
                  interface "lo0" {
                  };
          };
   }
   ```

   Конфигурационный файл включает в себя:

   - описание уникального идентификатора маршрутизатора в сети, «от имени» которого будут рассылаться данные о маршрутах;
   - структуру **`protocol kernel`** — она описывает действия, связанные с таблицами маршрутизации ядра системы;
   - структуру **`protocol device`** — она описывает действия самого сетевого устройства;

- структуру **`protocol ospf`** — она описывает действия, связанные с маршрутизацией с помощью протокола:
  - экспорт всей OSPF-информации о доступных маршрутах по IPv4;
  - экспорт данных всем устройствам за указанными интерфейсами.

2. Аналогично опишите (или скопируйте) конфигурационные файлы для R2 и R3.

@R2:/etc/bird/bird.conf

```
router id 10.0.2.2;

protocol kernel {
       scan time 20;
       ipv4 { export all; };
}

protocol device {
       scan time 10;
}

protocol ospf SIMPLE {
       ipv4 { export all; };
       area 0.0.0.0 {
               interface "eth1" {
               };
               interface "lo0" {
               };
       };
}
```

@R3:/etc/bird/bird.conf

```
router id 10.0.3.3;

protocol kernel {
       scan time 20;
       ipv4 { export all; };
}

protocol device {
       scan time 10;
}

protocol ospf SIMPLE {
       ipv4 { export all; };
       area 0.0.0.0 {
               interface "eth1" {
               };
               interface "eth2" {
               };
               interface "lo0" {
               };
               interface "lo2" {
               };
       };
}
```

3. С помощью команды **bird** запустите BIRD на *каждом* из устройств.

   С помощью команды **birdc** можно посмотреть параметры работы демона.

4. С помощью команды **birdc show route** посмотрите список OSPF-данных, передаваемых BIRD.

```
[root@R1 ~]# bird
[root@R1 ~]# birdc
BIRD +detached. ready.
bird> show route
Table master4:
10.0.12.0/24         unicast [SIMPLE 01:41:40.630] * I (150/10) [10.0.1.1]
       dev eth1
10.0.13.0/24         unicast [SIMPLE 01:41:40.631] * I (150/10) [10.0.1.1]
       dev eth2
10.0.1.1/32          unicast [SIMPLE 01:41:40.631] * I (150/0) [10.0.1.1]
       dev lo0
bird>
```

   В таблице маршрутизации после запуска должны появиться новые записи о доступных маршрутах с указанием **proto bird**, означающим, что маршрут получен с помощью BIRD-демона (поскольку маршруты не приходят мгновенно, может потребоваться время на получение всех данных).

5. С помощью команды управления таблицами маршрутизации убедитесь, что таблицы маршрутизации на устройствах изменились в соответствии с данными от BIRD.

```
[root@R1 ~]# ip route
10.0.1.0/24 dev lo0 proto kernel scope link src 10.0.1.1 linkdown
10.0.1.1 dev lo0 proto bird scope link metric 32 linkdown
10.0.2.2 via 10.0.12.2 dev eth1 proto bird metric 32
10.0.3.3 via 10.0.13.3 dev eth2 proto bird metric 32
10.0.12.0/24 dev eth1 proto kernel scope link src 10.0.12.1
10.0.12.0/24 dev eth1 proto bird scope link metric 32
10.0.13.0/24 dev eth2 proto kernel scope link src 10.0.13.1
10.0.13.0/24 dev eth2 proto bird scope link metric 32
172.16.0.1 via 10.0.13.3 dev eth2 proto bird metric 32
```

```
[root@R2 ~]# ip route
10.0.2.0/24 dev lo0 proto kernel scope link src 10.0.2.2 linkdown
10.0.2.2 dev lo0 proto bird scope link metric 32 linkdown
10.0.12.0/24 dev eth1 proto kernel scope link src 10.0.12.2
10.0.12.0/24 dev eth1 proto bird scope link metric 32

<Some time later>

[root@R2 ~]# ip route
10.0.1.1 via 10.0.12.1 dev eth1 proto bird metric 32
10.0.2.0/24 dev lo0 proto kernel scope link src 10.0.2.2 linkdown
10.0.2.2 dev lo0 proto bird scope link metric 32 linkdown
10.0.3.3 via 10.0.12.1 dev eth1 proto bird metric 32
10.0.12.0/24 dev eth1 proto kernel scope link src 10.0.12.2
10.0.12.0/24 dev eth1 proto bird scope link metric 32
10.0.13.0/24 via 10.0.12.1 dev eth1 proto bird metric 32
172.16.0.1 via 10.0.12.1 dev eth1 proto bird metric 32
```

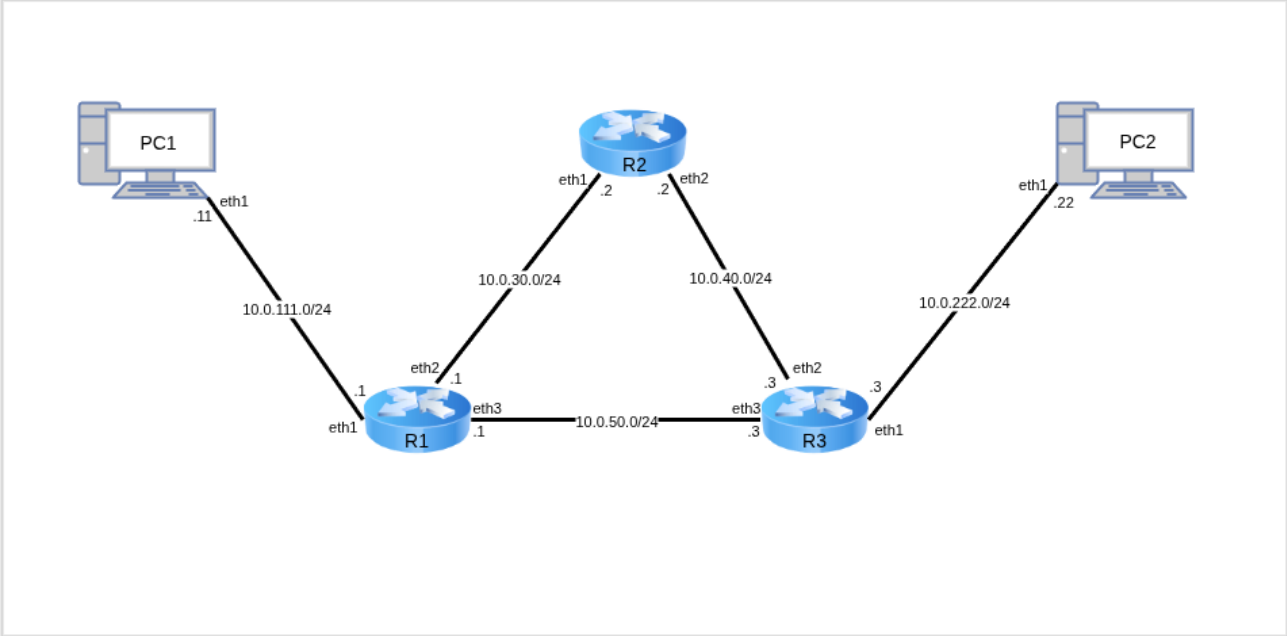
```
[root@R3 ~]# ip route
10.0.1.1 via 10.0.13.1 dev eth1 proto bird metric 32
10.0.2.2 via 10.0.13.1 dev eth1 proto bird metric 32
10.0.3.0/24 dev lo0 proto kernel scope link src 10.0.3.3 linkdown
10.0.3.3 dev lo0 proto bird scope link metric 32 linkdown
10.0.12.0/24 via 10.0.13.1 dev eth1 proto bird metric 32
10.0.13.0/24 dev eth1 proto kernel scope link src 10.0.13.3
10.0.13.0/24 dev eth1 proto bird scope link metric 32
172.16.0.0/24 dev lo2 proto kernel scope link src 172.16.0.1 linkdown
172.16.0.1 dev lo2 proto bird scope link metric 32 linkdown
```

6. С помощью команды **`ping -c3 <dstIP>`** проверьте достижимость отдельных сетей с разных устройств.

```
[root@R1 ~]# ping -c3 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=64 time=0.348 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=64 time=0.409 ms
64 bytes from 10.0.2.2: icmp_seq=3 ttl=64 time=0.392 ms

--- 10.0.2.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2024ms
rtt min/avg/max/mdev = 0.348/0.383/0.409/0.025 ms
[root@R1 ~]#
```

```
[root@R2 ~]# ping -c3 172.16.0.1
PING 172.16.0.1 (172.16.0.1) 56(84) bytes of data.
64 bytes from 172.16.0.1: icmp_seq=1 ttl=63 time=0.665 ms
64 bytes from 172.16.0.1: icmp_seq=2 ttl=63 time=0.489 ms
64 bytes from 172.16.0.1: icmp_seq=3 ttl=63 time=0.711 ms

--- 172.16.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2067ms
rtt min/avg/max/mdev = 0.489/0.621/0.711/0.095 ms
[root@R2 ~]#
```

10.3. Самостоятельная работа



Для работы необходимо 5 *клонов* согласно топологии сети. Для создания соединений между машинами необходимо в VirtualBox настроить сетевые интерфейсы (описание настройки подключения находится в соответствующем разделе второй лабораторной):

- **R1**:
  - **Adapter2** — **net111**
  - **Adapter3** — **net30**
  - **Adapter4** — **net50**
- **R2**:
  - **Adapter2** — **net30**
  - **Adapter3** — **net40**
- **R3**:
  - **Adapter2** — **net222**
  - **Adapter3** — **net40**
  - **Adapter4** — **net50**
- **PC1**:
  - **Adapter2** — **net111**
- **PC2**:
  - **Adapter2** — **net222**

10.3.1. Варианты заданий

Таблица 10.1. Варианты заданий

Вариант	Задание
1	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить OSPF между **R1** и **R3** 4. Убедиться, что **PC2** пингуется с **PC1** и наоборот 5. Выполнить **`traceroute`** с **PC1** на **PC2**
2	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить OSPF так, чтобы **PC2** не мог бы пинговать **R2**, а **PC1** мог бы 4. Выполнить **`traceroute`** с **PC1** и **PC2** на **R2**
3	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить OSPF так, чтобы **PC1** не мог бы пинговать **R2**, а **PC2** мог бы 4. Выполнить **`traceroute`** с **PC1** и **PC2** на **R2**
4	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить OSPF так, чтобы **PC1** не мог бы пинговать **R2**, но мог бы пинговать **R3** 4. Выполнить **`traceroute`** с **PC1** на **R2** и **R3**
5	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить OSPF так, чтобы **PC2** не мог бы пинговать **R2**, но мог бы пинговать **R1** 4. Выполнить **`traceroute`** с **PC2** на **R2** и **R1**
6	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить OSPF так, чтобы **PC1** не мог бы пинговать **R2**, но мог бы пинговать **PC2** 4. Выполнить **`traceroute`** с **PC1** на **PC2** и **R2**
7	

Вариант	Задание
	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить OSPF так, чтобы **PC2** не мог бы пинговать **R2**, но мог бы пинговать **PC1** 4. Выполнить **`traceroute`** с **PC2** на **PC1** и **R2**
8	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить OSPF так, чтобы **R2** мог бы пинговать **PC2** и **PC1** 4. Выполнить **`traceroute`** с **R2** на **PC2** и **PC1**
9	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить OSPF так, чтобы **R2** мог бы пинговать **PC2**, но не мог бы пинговать **PC1** 4. Выполнить **`traceroute`** с **R2** на **PC2** и **PC1**
10	1. Создать топологию, указанную на рисунке 2. Убедиться, что **PC2** не пингуется с **PC1** 3. Настроить OSPF так, чтобы **R2** мог бы пинговать **PC1**, но не мог бы пинговать **PC2** 4. Выполнить **`traceroute`** с **R2** на **PC2** и **PC1**

10.3.1.1. Задание

Запустить отчёты на каждой машине и выполнить соответствующие команды.

```
report 10 pc1
```

1. **`ip a show eth1`**.
2. **`ip route`**.
3. **`cat /etc/bird/bird.conf`**.
4. **`ping -fc3 10.0.50.1`**.
5. **`ping -fc3 10.0.30.2`**.
6. **`ping -fc3 10.0.40.3`**.

7. **ping -fc3 10.0.222.22**.
8. **traceroute 10.0.222.22**.
9. **traceroute 10.0.40.2**.

```
report 10 pc2
```

1. **ip a show eth1**.
2. **ip route**.
3. **cat /etc/bird/bird.conf**.
4. **ping -fc3 10.0.50.1**.
5. **ping -fc3 10.0.30.2**.
6. **ping -fc3 10.0.40.3**.
7. **ping -fc3 10.0.111.11**.
8. **traceroute 10.0.111.11**.
9. **traceroute 10.0.40.2**.

```
report 10 r1
```

1. **ip a show**.
2. **ip route**.
3. **cat /etc/bird/bird.conf**.
4. **ping -fc3 10.0.30.2**.
5. **ping -fc3 10.0.40.3**.
6. **ping -fc3 10.0.111.11**.
7. **ping -fc3 10.0.222.22**.
8. **traceroute 10.0.222.22**.
9. **traceroute 10.0.111.11**.

```
report 10 r2
```

1. **ip a show**.
2. **ip route**.
3. **cat /etc/bird/bird.conf**.
4. **ping -fc3 10.0.50.1**.
5. **ping -fc3 10.0.40.3**.

6. **`ping -fc3 10.0.111.11`**.
7. **`ping -fc3 10.0.222.22`**.
8. **`traceroute 10.0.222.22`**.
9. **`traceroute 10.0.111.11`**.

```
report 10 r3
```

1. **`ip a show`**.
2. **`ip route`**.
3. **`cat /etc/bird/bird.conf`**.
4. **`ping -fc3 10.0.30.2`**.
5. **`ping -fc3 10.0.50.1`**.
6. **`ping -fc3 10.0.111.11`**.
7. **`ping -fc3 10.0.222.22`**.
8. **`traceroute 10.0.222.22`**.
9. **`traceroute 10.0.111.11`**.

Полученные отчёты **`report.10.pc1`**, **`report.10.pc2`**, **`report.10.r1`**, **`report.10.r2`**, **`report.10.r3`** через последовательный порт перенести из виртуальной машины и прислать их преподавателю с подписью выполненного варианта.

Глава 11. Фильтрация трафика с помощью списков контроля доступа

Цель лабораторной работы — познакомить изучающего с основами работы списков контроля доступа.

Задачи лабораторной работы:

- Изучить логику работы списков контроля доступа;
- Реализовать тестовую топологию с применением списков контроля доступа.

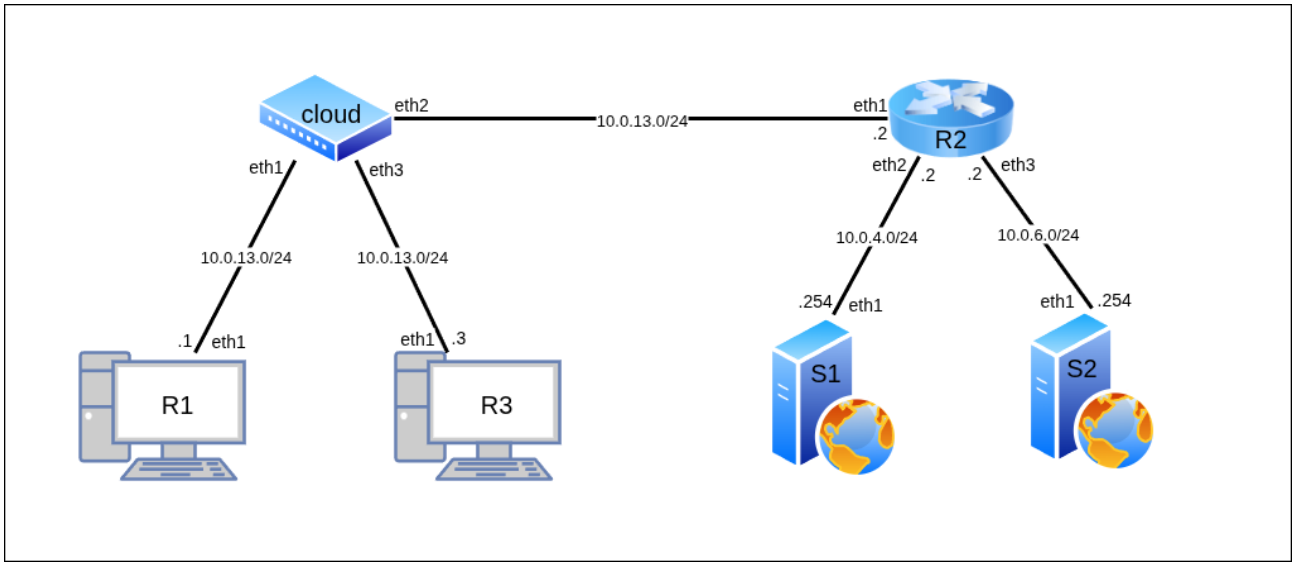
11.1. Списки контроля доступа

Access Control List (ACL) — технология обработки сетевого трафика, занимающаяся *управлением маршрутизации* трафика (как поступающего, так и исходящего или проходящего через устройство).

Списки контроля доступа позволяют как перемаршрутизировать трафик специальным образом на основании данных пакета, так и ограничить прохождение трафика через данное устройство.

11.2. Пример работы со списками контроля доступа

Для изучения списков контроля доступа рассмотрим следующую топологию:



Для работы создадим 6 *клонов* согласно топологии сети. Для создания соединений между машинами необходимо в VirtualBox настроить сетевые интерфейсы (описание настройки подключения находится в соответствующем разделе второй лабораторной):

- **cloud**:
  - **Adapter2** — **cloud1**
  - **Adapter3** — **cloud2**
  - **Adapter4** — **cloud3**
- **R1**:
  - **Adapter2** — **cloud1**
- **R2**:
  - **Adapter2** — **cloud2**
  - **Adapter3** — **wgetnet**
  - **Adapter4** — **sshnet**
- **S1**:
  - **Adapter2** — **wgetnet**
- **S2**:
  - **Adapter2** — **sshnet**

11.2.1. Базовая настройка виртуальных машин

1. Настройте все устройства в сети согласно топологии. Для настройки воспользуйтесь:
   - командами управления интерфейсами
   - командами настройки IP-адресов
   - командами настройки IP-Forwarding

```
[root@cloud ~]# ip link add dev br0 type bridge

[root@cloud ~]# ip link set eth1 master br0
[root@cloud ~]# ip link set eth2 master br0
[root@cloud ~]# ip link set eth3 master br0

[root@cloud ~]# for I in `ls /sys/class/net`; do ip link set $I up; done
[root@cloud ~]#
```

```
[root@S1 ~]# ip link set eth1 up
[root@S1 ~]# ip addr add dev eth1 10.0.4.254/24
[root@S1 ~]#
```

```
[root@S2 ~]# ip link set eth1 up
[root@S2 ~]# ip addr add dev eth1 10.0.6.254/24
[root@S2 ~]#
```

```
[root@R1 ~]# ip link set eth1 up
[root@R1 ~]# ip addr add dev eth1 10.0.13.1/24
[root@R1 ~]#
```

```
[root@R2 ~]# ip link set eth1 up
[root@R2 ~]# ip link set eth2 up
[root@R2 ~]# ip link set eth3 up

[root@R2 ~]# ip addr add dev eth1 10.0.13.2/24
[root@R2 ~]# ip addr add dev eth2 10.0.4.2/24
[root@R2 ~]# ip addr add dev eth3 10.0.6.2/24

[root@R2 ~]# sysctl net.ipv4.conf.all.forwarding=1
[root@R2 ~]#
```

```
[root@R3 ~]# ip link set eth1 up
[root@R3 ~]# ip addr add dev eth1 10.0.13.3/24
[root@R3 ~]#
```

11.2.2. Настройка маршрутизации в сети

1. Опишите (или скопируйте) конфигурационные файлы настройки OSPF-маршрутизации с помощью BIRD Routing Daemon.

   @S1 : **/etc/bird/bird.conf**

```
router id 10.0.4.254;

protocol kernel {
      scan time 20;
      ipv4 { export all; };
}

protocol device {
      scan time 10;
}

protocol ospf SIMPLE {
      ipv4 { export all; };
      area 0.0.0.0 {
             interface "eth1" {
             };
      };
}
```

@S2 : **/etc/bird/bird.conf**

```
router id 10.0.6.254;

protocol kernel {
      scan time 20;
      ipv4 { export all; };
}

protocol device {
      scan time 10;
}

protocol ospf SIMPLE {
      ipv4 { export all; };
      area 0.0.0.0 {
             interface "eth1" {
             };
      };
}
```

@R1 : **/etc/bird/bird.conf**

```
router id 10.0.13.1;

protocol kernel {
      scan time 20;
      ipv4 { export all; };
}

protocol device {
      scan time 10;
}

protocol ospf SIMPLE {
      ipv4 { export all; };
      area 0.0.0.0 {
```

```
              interface "eth1" {
              };
      };
}
```

@R2 : **/etc/bird/bird.conf**

```
router id 10.0.13.2;

protocol kernel {
      scan time 20;
      ipv4 { export all; };
}

protocol device {
      scan time 10;
}

protocol ospf SIMPLE {
      ipv4 { export all; };
      area 0.0.0.0 {
             interface "eth1" {
             };
             interface "eth2" {
             };
             interface "eth3" {
             };
      };
}
```

@R3 : **/etc/bird/bird.conf**

```
router id 10.0.13.3;

protocol kernel {
      scan time 20;
      ipv4 { export all; };
}

protocol device {
      scan time 10;
}

protocol ospf SIMPLE {
      ipv4 { export all; };
      area 0.0.0.0 {
             interface "eth1" {
             };
      };
}
```

2. С помощью команды **bird** запустите BIRD на *каждом* из устройств. С помощью команд управления таблицами маршрутизации убедитесь, что данные о маршрутах успешно добавились в таблицы маршрутизации R≠абонентов.

```
[root@S1 ~]# bird
[root@S1 ~]#
```

```
[root@S2 ~]# bird
[root@S2 ~]#
```

```
[root@R1 ~]# bird
[root@R1 ~]# ip route
10.0.4.0/24 via 10.0.13.2 dev eth1 proto bird metric 32
10.0.6.0/24 via 10.0.13.2 dev eth1 proto bird metric 32
10.0.13.0/24 dev eth1 proto kernel scope link src 10.0.13.1
10.0.13.0/24 dev eth1 proto bird scope link metric 32
[root@R1 ~]#
```

```
[root@R2 ~]# bird
[root@R2 ~]# ip route
10.0.4.0/24 dev eth2 proto kernel scope link src 10.0.4.2
10.0.4.0/24 dev eth2 proto bird scope link metric 32
10.0.6.0/24 dev eth3 proto kernel scope link src 10.0.6.2
10.0.6.0/24 dev eth3 proto bird scope link metric 32
10.0.13.0/24 dev eth1 proto kernel scope link src 10.0.13.2
10.0.13.0/24 dev eth1 proto bird scope link metric 32
[root@R2 ~]#
```

```
[root@R3 ~]# bird
[root@R1 ~]# ip route
10.0.4.0/24 via 10.0.13.2 dev eth1 proto bird metric 32
10.0.6.0/24 via 10.0.13.2 dev eth1 proto bird metric 32
10.0.13.0/24 dev eth1 proto kernel scope link src 10.0.13.3
10.0.13.0/24 dev eth1 proto bird scope link metric 32
[root@R1 ~]#
```

3. С помощью команд мониторинга сети проверьте доступность Sабонентов с Rабонентов.

```
[root@R1 ~]# ping -c3 10.0.4.254
PING 10.0.4.254 (10.0.4.254) 56(84) bytes of data.
64 bytes from 10.0.4.254: icmp_seq=1 ttl=63 time=1.26 ms
64 bytes from 10.0.4.254: icmp_seq=3 ttl=63 time=1.20 ms

--- 10.0.4.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.161/1.207/1.262/0.041 ms

[root@R1 ~]# ping -c3 10.0.6.254
PING 10.0.6.254 (10.0.6.254) 56(84) bytes of data.
64 bytes from 10.0.6.254: icmp_seq=1 ttl=63 time=1.29 ms
64 bytes from 10.0.6.254: icmp_seq=2 ttl=63 time=1.09 ms
64 bytes from 10.0.6.254: icmp_seq=3 ttl=63 time=2.05 ms

--- 10.0.6.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.087/1.475/2.052/0.416 ms
[root@R1 ~]#
```

```
[root@R3 ~]# ping -c3 10.0.4.254
PING 10.0.4.254 (10.0.4.254) 56(84) bytes of data.
64 bytes from 10.0.4.254: icmp_seq=1 ttl=63 time=1.61 ms
64 bytes from 10.0.4.254: icmp_seq=2 ttl=63 time=1.76 ms
64 bytes from 10.0.4.254: icmp_seq=3 ttl=63 time=1.03 ms
```

```
--- 10.0.4.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.027/1.467/1.763/0.317 ms

[root@R3 ~]# ping -c3 10.0.6.254
PING 10.0.6.254 (10.0.6.254) 56(84) bytes of data.
64 bytes from 10.0.6.254: icmp_seq=1 ttl=63 time=1.90 ms
64 bytes from 10.0.6.254: icmp_seq=2 ttl=63 time=1.07 ms
64 bytes from 10.0.6.254: icmp_seq=3 ttl=63 time=1.05 ms

--- 10.0.6.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.046/1.340/1.903/0.398 ms
[root@R3 ~]#
```

11.2.3. Настройка списков контроля доступа

На данный момент никаких ограничений в сети нет, доступ открыт всем абонентам сети для всех операций в сети.

Для наглядности и различия действий будет рассматриваться два вида соединений — HTTP-запрос на загрузку данных и SSH-подключение к абонентам.

1. С помощью команды встроенного модуля Python **`python3 -m http.server`** запустите на S-узлах HTTP-серверы.

```
[root@S1 ~]# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

```
[root@S2 ~]# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

   Для проверки не только IP-связности, но и доступности сервисов будут использоваться команды **`wget`** и **`ssh`**:

   - С помощью **`wget`** будет выполняться HTTP-запрос; скачивание файла означает доступ к устройству, зависание — отсутствие доступа;
   - С помощью **`ssh`** осуществляется подключение к устройству; подключение означает доступ к устройству, зависание — отсутствие доступа.

Примечание

При первом **`ssh`**-подключении может появиться запрос на разрешение подключения. Необходимо *явно* написать **`yes`** в ответ на вопрос в терминал.

2. С помощью команд **`wget <dstIP>:<dstPort>`** и **`ssh <dstIP>`** на R1 и R3 проверьте доступность сервисов S-абонентов.

```
[root@R1 ~]# wget 10.0.4.254:8000
Prepended http:// to '10.0.4.254:8000'
--2025-10-06 19:13:42--  http://10.0.4.254:8000/
Connecting to 10.0.4.254:8000... connected.
```

```
HTTP request sent, awaiting response... 200 OK
Length: 1142 (1.1K) [text/html]
Saving to: 'index.html.1'

index.html.1              100%[===================================>]
1.12K  --.-KB/s    in 0s

2025-10-06 19:13:42 (54.2 MB/s) - 'index.html.1' saved [1142/1142]

[root@R1 ~]#
```

```
[root@R1 ~]# wget 10.0.6.254:8000
Prepended http:// to '10.0.6.254:8000'
--2025-10-06 19:13:57--  http://10.0.6.254:8000/
Connecting to 10.0.6.254:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1087 (1.1K) [text/html]
Saving to: 'index.html.2'

index.html.2              100%[===================================>]
1.06K  --.-KB/s    in 0s

2025-10-06 19:13:57 (33.8 MB/s) - 'index.html.2' saved [1087/1087]

[root@R1 ~]#
```

```
[root@R1 ~]# ssh 10.0.4.254
Last login: Mon Oct  6 13:42:04 2025 from 10.0.13.1
[root@S1 ~]#
<^D>logout
Connection to 10.0.4.254 closed.
[root@R1 ~]#
```

```
[root@R1 ~]# ssh 10.0.6.254
The authenticity of host '10.0.6.254 (10.0.6.254)' can't be established.
ED25519 key fingerprint is
SHA256:BxaYoHAW5ddfM6EwmgSAZ2tKXCH0zoppLfEcQ8YiGdg.
This host key is known by the following other names/addresses:
   ~/.ssh/known_hosts:3: 10.0.4.254
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.6.254' (ED25519) to the list of known
hosts.
Last login: Mon Oct  6 18:58:46 2025 from 10.0.13.3
[root@S2 ~]#
<^D>logout
Connection to 10.0.6.254 closed.
[root@R1 ~]#
```

```
[root@R3 ~]# wget 10.0.4.254:8000
Prepended http:// to '10.0.4.254:8000'
--2025-10-06 19:13:45--  http://10.0.4.254:8000/
Connecting to 10.0.4.254:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1142 (1.1K) [text/html]
Saving to: 'index.html'

index.html                100%[===================================>]
```

```
1.12K  --.-KB/s    in 0s

2025-10-06 19:13:45 (66.8 MB/s) - 'index.html' saved [1142/1142]

[root@R3 ~]#
```

```
[root@R3 ~]# wget 10.0.6.254:8000
Prepended http:// to '10.0.6.254:8000'
--2025-10-06 19:13:54--  http://10.0.6.254:8000/
Connecting to 10.0.6.254:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1087 (1.1K) [text/html]
Saving to: 'index.html.1'

index.html.1              100%[===================================>]
1.06K  --.-KB/s    in 0s

2025-10-06 19:13:54 (31.1 MB/s) - 'index.html.1' saved [1087/1087]
[root@R3 ~]#
```

```
[root@R3 ~]# ssh 10.0.4.254
The authenticity of host '10.0.4.254 (10.0.4.254)' can't be established.
ED25519 key fingerprint is
SHA256:BxaYoHAW5ddfM6EwmgSAZ2tKXCH0zoppLfEcQ8YiGdg.
This host key is known by the following other names/addresses:
   ~/.ssh/known_hosts:3: 10.0.6.254
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.4.254' (ED25519) to the list of known
hosts.
Last login: Mon Oct  6 19:14:00 2025 from 10.0.13.1
[root@S1 ~]#
<^D>logout
Connection to 10.0.4.254 closed.
[root@R3 ~]#
```

```
[root@R3 ~]# ssh 10.0.6.254
Last login: Mon Oct  6 19:14:08 2025 from 10.0.13.1
[root@S2 ~]#
<^D>logout
Connection to 10.0.6.254 closed.
[root@R3 ~]#
```

Как видно из показаний серверов, оба соединения к каждому из абонентов прошли успешно:

```
[root@S1 ~]# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.0.13.1 - - [06/Oct/2025 19:13:35] "GET / HTTP/1.1" 200 -
10.0.13.3 - - [06/Oct/2025 19:13:45] "GET / HTTP/1.1" 200 -
```

```
[root@S2 ~]# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.0.13.1 - - [06/Oct/2025 19:13:50] "GET / HTTP/1.1" 200 -
10.0.13.3 - - [06/Oct/2025 19:13:54] "GET / HTTP/1.1" 200 -
```

Теперь настроим следующие ограничения доступа:

- R1 будет иметь доступ к 8000 порту абонента S1 (для **`wget`**);
- R3 будет иметь доступ к 22 порту абонента S2 (для **`ssh`**);
- *Любые другие* соединения R1 и R3 с S1 и S2 запрещены.

Настройку списков контроля доступа осуществим с помощью команд ip rule, позволяющих манипулировать правилами обращения к таблицам маршрутизации. Правила будут задаваться на промежуточном узле R2.

3. С помощью команды **`ip rule`** выведите текущий список контроля доступа.

```
[root@R2 ~]# ip rule
0:      from all lookup local
32766:  from all lookup main
32767:  from all lookup default
[root@R2 ~]#
```

Для правил в списке контроля доступа может задаваться приоритет. Правила с меньшим значением **`priority`** имеют приоритет над полями с большим значением. Принцип выбора срабатывания правил — сверху вниз до первого совпадения параметров.

4. С помощью команд **`ip rule add <*ip-rule-parameters>`** опишите следующие правила списка контроля доступа:
   - Пакеты от R1 к S1 на порт 8000 обработать согласно стандартной таблице маршрутизации;
   - Остальные пакеты от R1 к S1 сбрасывать без обработки;
   - Пакеты от R1 к S2 сбрасывать без обработки;
   - Пакеты от R3 к S2 на порт 22 обработать согласно стандартной таблице маршрутизации;
   - Остальные пакеты от R3 к S2 сбрасывать без обработки;
   - Пакеты от R3 к S1 сбрасывать без обработки.

Чтобы правила (1), (4) не поглощались правилами (2), (3), (5), (6), необходимо выдать им более высокий приоритет.

```
[root@R2 ~]# ip rule add from 10.0.13.1 to 10.0.4.254 dport 8000 priority 1
table main
[root@R2 ~]# ip rule add blackhole from 10.0.13.1 to 10.0.4.254 priority 2
[root@R2 ~]# ip rule add blackhole from 10.0.13.1 to 10.0.6.254 priority 2
[root@R2 ~]# ip rule add from 10.0.13.3 to 10.0.6.254 dport 22 priority 1
table main
[root@R2 ~]# ip rule add blackhole from 10.0.13.3 to 10.0.6.254 priority 2
[root@R2 ~]# ip rule add blackhole from 10.0.13.3 to 10.0.4.254 priority 2
[root@R2 ~]#
```

5. С помощью команды **`ip rule`** выведите текущий список контроля доступа.

```
[root@R2 ~]# ip rule
0:      from all lookup local
1:      from 10.0.13.1 to 10.0.4.254 dport 8000 lookup main
1:      from 10.0.13.3 to 10.0.6.254 dport 22 lookup main
2:      from 10.0.13.1 to 10.0.4.254 blackhole
2:      from 10.0.13.1 to 10.0.6.254 blackhole
2:      from 10.0.13.3 to 10.0.6.254 blackhole
2:      from 10.0.13.3 to 10.0.4.254 blackhole
32766:  from all lookup main
32767:  from all lookup default
[root@R2 ~]#
```

Проверим, что R1 и R3 не потеряли связь с R2, но не могут организовать поток обычного трафика на S1 и S2.

6. С помощью команд мониторинга сети убедитесь в наличии связности R1 и R3 с R2 и отсутствии связности с S⌀абонентами.

```
[root@R1 ~]# ping -c3 10.0.13.2
PING 10.0.13.2 (10.0.13.2) 56(84) bytes of data.
64 bytes from 10.0.13.2: icmp_seq=1 ttl=64 time=0.539 ms
64 bytes from 10.0.13.2: icmp_seq=2 ttl=64 time=0.596 ms
64 bytes from 10.0.13.2: icmp_seq=3 ttl=64 time=0.435 ms

--- 10.0.13.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2079ms
rtt min/avg/max/mdev = 0.435/0.523/0.596/0.066 ms

[root@R1 ~]# ping -c3 10.0.4.254
PING 10.0.4.254 (10.0.4.254) 56(84) bytes of data.

--- 10.0.4.254 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2033ms

[root@R1 ~]# ping -c3 10.0.6.254
PING 10.0.6.254 (10.0.6.254) 56(84) bytes of data.

--- 10.0.6.254 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2046ms

[root@R1 ~]#
```

```
[root@R3 ~]# ping -c3 10.0.13.2
PING 10.0.13.2 (10.0.13.2) 56(84) bytes of data.
64 bytes from 10.0.13.2: icmp_seq=1 ttl=64 time=0.452 ms
64 bytes from 10.0.13.2: icmp_seq=2 ttl=64 time=0.569 ms
64 bytes from 10.0.13.2: icmp_seq=3 ttl=64 time=0.435 ms

--- 10.0.13.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2081ms
rtt min/avg/max/mdev = 0.435/0.485/0.569/0.059 ms

[root@R3 ~]# ping -c3 10.0.4.254
PING 10.0.4.254 (10.0.4.254) 56(84) bytes of data.

--- 10.0.4.254 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2051ms
```

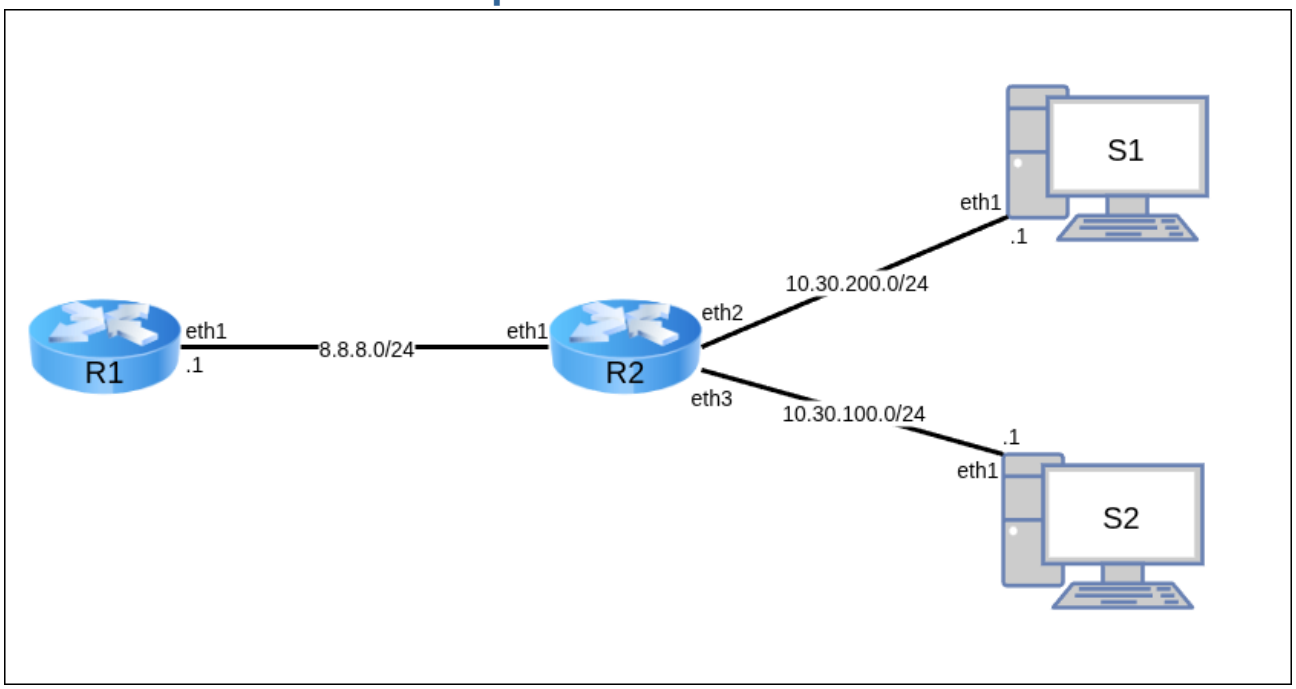
```
[root@R3 ~]# ping -c3 10.0.6.254
PING 10.0.6.254 (10.0.6.254) 56(84) bytes of data.

--- 10.0.6.254 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2063ms

[root@R3 ~]#
```

7. С помощью команд **`wget <dstIP>:<dstPort>`** и **`ssh <dstIP>`** на R1 и R3 проверьте доступность разрешённых сервисов S⁄абонентов и запрет остальных.

```
[root@R1 ~]# wget -t=1 10.0.4.254:8000
Prepended http:// to '10.0.4.254:8000'
--2025-10-06 19:24:48--  http://10.0.4.254:8000/
Connecting to 10.0.4.254:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1142 (1.1K) [text/html]
Saving to: 'index.html'

index.html                100%[====================================>]
1.12K  --.-KB/s    in 0s

2025-10-06 19:24:48 (72.1 MB/s) - 'index.html' saved [1142/1142]

[root@R1 ~]# wget -t=1 10.0.6.254:8000
Prepended http:// to '10.0.6.254:8000'
--2025-10-06 19:24:53--  http://10.0.6.254:8000/
Connecting to 10.0.6.254:8000...^C

[root@R1 ~]# ssh 10.0.4.254
^C

[root@R1 ~]# ssh 10.0.6.254
^C

[root@R1 ~]#
```

```
[root@R3 ~]# wget -t=1 10.0.4.254:8000
Prepended http:// to '10.0.4.254:8000'
--2025-10-06 19:26:32--  http://10.0.4.254:8000/
Connecting to 10.0.4.254:8000...^C

[root@R3 ~]# wget -t=1 10.0.6.254:8000
Prepended http:// to '10.0.6.254:8000'
--2025-10-06 19:26:38--  http://10.0.6.254:8000/
Connecting to 10.0.6.254:8000...^C

[root@R3 ~]# ssh 10.0.4.254
^C

[root@R3 ~]# ssh 10.0.6.254
Last login: Mon Oct  6 19:14:31 2025 from 10.0.13.3
[root@S2 ~]#
<^D>logout
Connection to 10.0.6.254 closed.
[root@R3 ~]#
```

11.3. Самостоятельная работа



Для работы необходимо 4 *клона* согласно топологии сети. Для создания соединений между машинами необходимо в VirtualBox настроить сетевые интерфейсы (описание настройки подключения находится в соответствующем разделе второй лабораторной).

- **R1**:
  - **Adapter2** — **net1**
- **R2**:
  - **Adapter1** — **net1**
  - **Adapter2** — **net2**
  - **Adapter3** — **net3**
- **S1**:
  - **Adapter2** — **net2**
- **S2**:
  - **Adapter2** — **net3**

11.3.1. Варианты заданий

Таблица 11.1. Варианты заданий

Вариант	Задание
1	1. Для маршрутизатора **R2** установить на интерфейсы произвольные IP-адреса из соответствующих локальных сетей

Вариант	Задание
	2. С помощью **OSPF** настроить доступность всех абонентов в сети 3. Разрешить общение **10.30.200.0/24** с **8.8.8.1** 4. Запретить доступ из остальных сетей к сети **8.8.8.1** 5. Сохранить сообщение между сетями **10.30.200.0/24** и **10.30.100.0/24**
2	1. Для маршрутизатора **R2** установить на интерфейсы произвольные IP-адреса из соответствующих локальных сетей 2. С помощью **OSPF** настроить доступность всех абонентов в сети 3. Разрешить общение **10.30.100.0/24** с **8.8.8.1** 4. Запретить доступ из остальных сетей к сети **8.8.8.1** 5. Сохранить сообщение между сетями **10.30.200.0/24** и **10.30.100.0/24**
3	1. Для маршрутизатора **R2** установить на интерфейсы произвольные IP-адреса из соответствующих локальных сетей 2. С помощью **OSPF** настроить доступность всех абонентов в сети 3. Из сетей **10.30.200.0/24** и **10.30.100.0/24** разрешить общение с **8.8.8.1** 4. Запретить сообщение между сетями **10.30.200.0/24** и **10.30.100.0/24**
4	1. Для маршрутизатора **R2** установить на интерфейсы произвольные IP-адреса из соответствующих локальных сетей 2. С помощью **OSPF** настроить доступность всех абонентов в сети 3. Из сети **10.30.200.0/24** разрешить доступ к **8.8.8.1** только на порт **8000** 4. Сохранить сообщение между сетями **10.30.200.0/24** и **10.30.100.0/24**
5	1. Для маршрутизатора **R2** установить на интерфейсы произвольные IP-адреса из соответствующих локальных сетей 2. С помощью **OSPF** настроить доступность всех абонентов в сети 3. Из сети **10.30.100.0/24** разрешить доступ к **8.8.8.1** только на порт **8000** 4. Запретить сообщение между сетями **10.30.200.0/24** и **10.30.100.0/24**
6	1. Для маршрутизатора **R2** установить на интерфейсы произвольные IP-адреса из соответствующих локальных сетей 2. С помощью **OSPF** настроить доступность всех абонентов в сети 3. Из сетей **10.30.200.0/24** и **10.30.100.0/24** разрешить доступ к **8.8.8.1** только на порт **22** 4. Сохранить сообщение между сетями **10.30.200.0/24** и **10.30.100.0/24**

Вариант	Задание
7	1. Для маршрутизатора **R2** установить на интерфейсы произвольные IP-адреса из соответствующих локальных сетей 2. С помощью **OSPF** настроить доступность всех абонентов в сети 3. Из сети **10.30.100.0/24** разрешить доступ к **8.8.8.1** только на порт **22** 4. Сохранить сообщение между сетями **10.30.200.0/24** и **10.30.100.0/24**

11.3.1.1. Задание

Запустить отчёты на каждой машине и выполнить соответствующие команды.

```
report 11 r1
```

1. **ip a show eth1**.
2. **ip route**.
3. **cat /etc/bird/bird.conf**.
4. **ping -fc3 10.30.200.1**.
5. **ping -fc3 10.30.100.1**.
6. **python3 -m http.server**.
7. Завершите работу после выполнения остальных отчётов.

```
report 11 r2
```

1. **ip a show**.
2. **ip route**.
3. **cat /etc/bird/bird.conf**.
4. **ip rule**.
5. **tcpdump -xx -i eth1**.
6. Завершите работу после выполнения остальных отчётов.

```
report 11 s1
```

1. **ip a show eth1**.
2. **ip route**.
3. **cat /etc/bird/bird.conf**.
4. **ping -fc3 10.30.100.1**.

5. **`ping -fc3 8.8.8.1`**.
6. **`wget -t1 -T3 8.8.8.1`**.
7. **`ssh -o ConnectTimeout=5 8.8.8.1`**.

```
report 11 s2
```

1. **`ip a show eth1`**.
2. **`ip route`**.
3. **`cat /etc/bird/bird.conf`**.
4. **`ping -fc3 10.30.200.1`**.
5. **`ping -fc3 8.8.8.1`**.
6. **`wget -t1 -T3 8.8.8.1`**.
7. **`ssh -o ConnectTimeout=5 8.8.8.1`**.

Полученные отчёты **`report.11.r1`**, **`report.11.r2`**, **`report.11.s1`**, **`report.11.s2`** через последовательный порт перенести из виртуальной машины и прислать их преподавателю с подписью выполненного варианта.

Глава 12. VPN и туннелирование

Цель лабораторной работы — познакомить изучающего с основами VPN и туннелирования.

Задачи лабораторной работы:

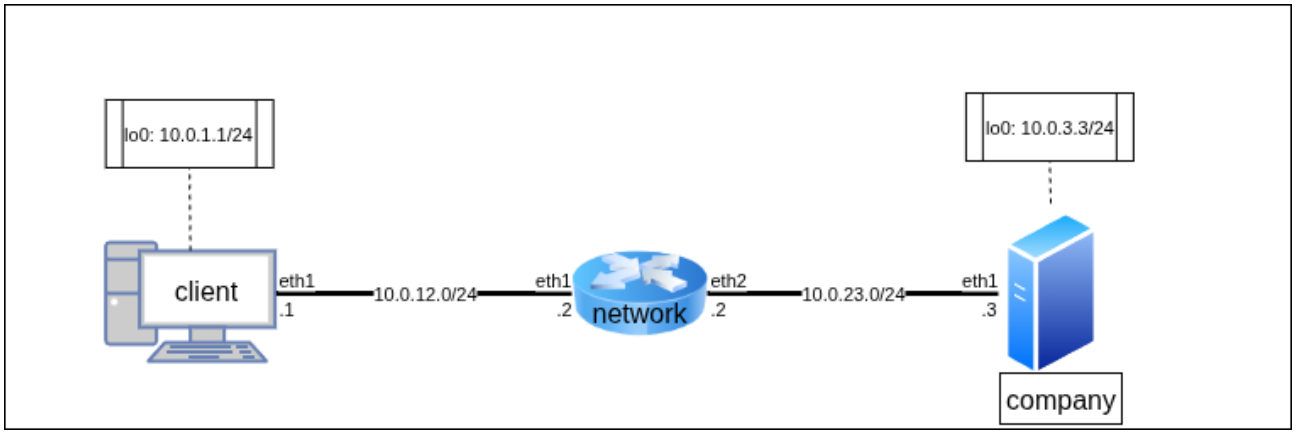
- Изучить логику работы технологий;
- Познакомиться с системой настройки сетевых служб systemd-networkd;
- Реализовать тестовую топологию с построением VPN-соединения WireGuard;
- Реализовать тестовую топологию с построением туннеля IP over IP.

12.1. VPN

VPN (*Virtual Private Network*) — технология, позволяющая объединять в единую адресную «локальную» сеть устройства, физически не находящихся в ней (например, при подключении рабочего ноутбука из командировки в локальную сеть компании). Кроме непосредственного создания туннелей, в VPN возможно настраивать шифрование проходящего трафика. В рамках лабораторной рассматривается один из популярных протоколов VPN — WireGuard

12.2. Настройка базовой топологии

Для изучения VPN и туннелирования разберём топологию с тремя абонентами.



Для работы создайте 3 *клона* согласно топологии сети. Для создания соединений между машинами необходимо в VirtualBox настроить сетевые интерфейсы (описание настройки подключения находится в соответствующем разделе второй лабораторной):

- **`client`**:
  - **`Adapter2`** — **`outnet`**
- **`network`**:
  - **`Adapter2`** — **`outnet`**
  - **`Adapter3`** — **`innet`**
- **`company`**:
  - **`Adapter2`** — **`innet`**

12.2.1. Настройка абонентов

В данной лабораторной вместо привычного метода настройки интерфейсов и адресов используется комплекс инициализирующих программ SystemD, а именно его подсистема **`systemd-networkd`**, отвечающая за автонастройку всех сетевых служб в Linux. Преимущество такого способа заключается в сохранении всех настроек после перезагрузки устройства, поскольку записи о настройке хранятся не в оперативной памяти, а в файлах специального вида.

1. Опишите (или скопируйте) конфигурационный файл для настройки интерфейса eth1 абонента client в соответствующем файле виртуальной машины.

   `@client:/etc/systemd/network/50-outnet.network`

```
[Match]
Name=eth1

[Network]
Address=10.0.12.1/24

[Route]
Gateway=10.0.12.2
Destination=10.0.0.0/8
```

Примечание

Согласно настройкам **.network**-файла будет поднят интерфейс eth1, ему будет присвоен адрес 10.0.12.1/24, а в таблицу маршрутизации будет добавлена запись, согласно которой все пакеты, отправляемые на адреса сети 10.0.0.0/8, будут первоначально направляться на адрес 10.0.12.2.

С помощью **systemd-networkd** можно также создавать и новые интерфейсы, однако это потребует создания отдельного файла, поэтому добавление LoopBack‑интерфейса будет выполнено вручную.

2. С помощью команд управления интерфейсами и настройки IP‑адресов настройте loopback‑интерфейс на абоненте client согласно топологии.

```
[root@client ~]# ip link add dev lo0 type veth
[root@client ~]# ip link set lo0 up
[root@client ~]# ip addr add dev lo0 10.0.1.1/24
```

3. Опишите (или скопируйте) конфигурационный файл для настройки интерфейса eth1 абонента company в соответствующем файле виртуальной машины

   @company:/etc/systemd/network/50-outnet.network

```
[Match]
Name=eth1

[Network]
Address=10.0.23.3/24

[Route]
Gateway=10.0.23.2
Destination=10.0.0.0/8
```

4. С помощью команд управления интерфейсами и настройки IP‑адресов настройте loopback‑интерфейс на абоненте company согласно топологии

```
[root@company ~]# ip link add dev lo0 type veth
[root@company ~]# ip link set lo0 up
[root@company ~]# ip addr add dev lo0 10.0.3.3/24
[root@company ~]#
```

Таким же образом может быть настроен абонент **@network**. Но для закрепления ручной настройки задайте параметры вручную.

5. С помощью команд управления интерфейсами и настройки IP-адресов настройте интерфейсы на абоненте network согласно топологии

```
[root@network ~]# ip link set eth1 up
[root@network ~]# ip link set eth2 up
[root@network ~]# ip addr add dev eth1 10.0.12.2/24
[root@network ~]# ip addr add dev eth2 10.0.23.2/24
[root@network ~]# sysctl net.ipv4.conf.all.forwarding=1
[root@network ~]#
[root@network ~]# ip route add 10.0.1.1 via 10.0.12.1
[root@network ~]# ip route add 10.0.3.3 via 10.0.23.3
[root@network ~]#
```

Для запуска **systemd-networkd** необходимо активировать сервис с помощью служебной команды **systemctl enable --now** (**enable** включает возможность активации сервиса при включении устройства. Флаг **--now** включает сервис непосредственно сейчас).

6. С помощью команды **systemctl enable --now systemd-networkd** запустите сервис автонастройки на client и company

```
[root@client ~]# systemctl enable --now systemd-networkd
Created symlink '/etc/systemd/system/dbus-org.freedesktop.network1.service'
-> '/usr/lib/systemd/system/systemd-networkd.service'.
Created symlink '/etc/systemd/system/multi-user.target.wants/systemd-
networkd.service' -> '/usr/lib/systemd/system/systemd-networkd.service'.
Created symlink '/etc/systemd/system/sockets.target.wants/systemd-
networkd.socket' -> '/usr/lib/systemd/system/systemd-networkd.socket'.
Created symlink '/etc/systemd/system/sysinit.target.wants/systemd-network-
generator.service' -> '/usr/lib/systemd/system/systemd-network-
generator.service'.
Created symlink '/etc/systemd/system/network-online.target.wants/systemd-
networkd-wait-online.service' -> '/usr/lib/systemd/system/systemd-networkd-
wait-online.service'.
[root@client ~]#
```

```
[root@company ~]# systemctl enable --now systemd-networkd
Created symlink '/etc/systemd/system/dbus-org.freedesktop.network1.service'
-> '/usr/lib/systemd/system/systemd-networkd.service'.
Created symlink '/etc/systemd/system/multi-user.target.wants/systemd-
networkd.service' -> '/usr/lib/systemd/system/systemd-networkd.service'.
Created symlink '/etc/systemd/system/sockets.target.wants/systemd-
networkd.socket' -> '/usr/lib/systemd/system/systemd-networkd.socket'.
Created symlink '/etc/systemd/system/sysinit.target.wants/systemd-network-
generator.service' -> '/usr/lib/systemd/system/systemd-network-
generator.service'.
Created symlink '/etc/systemd/system/network-online.target.wants/systemd-
networkd-wait-online.service' -> '/usr/lib/systemd/system/systemd-networkd-
wait-online.service'.
[root@company ~]#
```

7. С помощью команд мониторинга сети проверьте доступность client и company друг для друга

```
[root@company ~]# ping -c5 10.0.12.1
PING 10.0.12.1 (10.0.12.1) 56(84) bytes of data.
64 bytes from 10.0.12.1: icmp_seq=1 ttl=63 time=1.80 ms
64 bytes from 10.0.12.1: icmp_seq=2 ttl=63 time=0.684 ms
```

```
64 bytes from 10.0.12.1: icmp_seq=3 ttl=63 time=1.03 ms
64 bytes from 10.0.12.1: icmp_seq=4 ttl=63 time=0.990 ms
64 bytes from 10.0.12.1: icmp_seq=5 ttl=63 time=1.06 ms

--- 10.0.12.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4038ms
rtt min/avg/max/mdev = 0.684/1.113/1.800/0.369 ms
[root@company ~]#
```

```
[root@client ~]# ping -c3 -I 10.0.1.1 10.0.3.3
PING 10.0.3.3 (10.0.3.3) from 10.0.1.1 : 56(84) bytes of data.
64 bytes from 10.0.3.3: icmp_seq=1 ttl=63 time=0.842 ms
64 bytes from 10.0.3.3: icmp_seq=2 ttl=63 time=0.861 ms
64 bytes from 10.0.3.3: icmp_seq=3 ttl=63 time=0.814 ms

--- 10.0.3.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2048ms
rtt min/avg/max/mdev = 0.814/0.839/0.861/0.019 ms
[root@client ~]#
```

12.2.2. Настройка блокировки сети

Теперь добавьте ограничение на создание TCP‑соединений. Блокировка будет настраиваться с помощью ещё одного сервиса — межсетевого экрана (firewall‑а) nftables. Проверка TCP‑соединения будет осуществляться с помощью SSH‑соединения с **@company** на **@client**.

1. С помощью команды **ssh <dstIP>** убедитесь в доступности SSH‑соединения с company на client

```
[root@company ~]# ssh 10.0.12.1
The authenticity of host '10.0.12.1 (10.0.12.1)' can't be established.
ED25519 key fingerprint is
SHA256:BxaYoHAW5ddfM6EwmgSAZ2tKXCH0zoppLfEcQ8YiGdg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.12.1' (ED25519) to the list of known hosts.
Last login: Sun Oct 19 13:19:43 2025
[root@client ~]#
<^D>logout
Connection to 10.0.12.1 closed.
[root@company ~]#
```

2. С помощью команды **systemctl enable --now nftables.service** запустите сервис поддержки сетевого экрана на network.

3. С помощью команды **nft add rule inet filter forward ip protocol tcp reject** установите блокировку TCP‑соединений в сети

```
[root@network ~]# systemctl enable --now nftables.service
Created symlink '/etc/systemd/system/multi-user.target.wants/
nftables.service' -> '/usr/lib/systemd/system/nftables.service'.
[root@network ~]# nft add rule inet filter forward ip protocol tcp reject
[root@network ~]#
```

4. С помощью команды **ssh <dstIP>** убедитесь в недоступности SSH‑соединения с company на client

```
[root@company ~]# ssh 10.0.12.1
ssh: connect to host 10.0.12.1 port 22: Connection refused
[root@company ~]#
```

Соединение стало невозможным (при этом заметьте, что ping идти не перестал, поскольку ICMP‑пакеты не устанавливают TCP‑соединение между абонентами).

Примечание

Ограничение TCP‑соединения настраивается в лабораторной, так как сетевой трафик кроме полезной нагрузки несёт управляющую информацию, например: данные о состоянии абонентов, каналов, служебные сообщения и т.д. Для корректной передачи этих данных часто используется именно TCP‑соединение, гарантирующее надёжность передачи. Во избежание передачи не регламентированных служебных данных некоторые маршруты в сети могут быть закрыты для несанкционированных TCP‑соединений. При этом передача UDP‑пакетов по этим каналам не ограничивается.

Поскольку большинство протоколов туннелирования и VPN‑протоколов подразумевают обмен именно UDP‑пакетами, таким образом они решают проблему обхода блокировок такого типа, позволяя передавать данные по маршрутам, не доступным для TCP‑соединений.

12.3. WireGuard VPN

Для настройки соединения воспользуемся VPN‑протоколом WireGuard. В качестве транспортного протокола WireGuard использует UDP. Особенностью протокола является схема асимметричного шифрования. Классическая схема с закрытым и открытым ключом, которые генерируются и передаются администратором сети, позволяет безопасно передавать сообщения. Сами ключи можно дополнительно защищать паролями.

12.3.1. Подготовка ключей шифрования

В качестве приватного ключа WireGuard может выступать любая последовательность из 32 байт. Вернее, 31 байта — 32‑ой байт используется в качестве контрольной суммы над остальными. Публичный ключ генерируется на основе приватного для создания асимметрично шифрующей пары.

Предупреждение

Генерация ключей (особенно приватного ключа) должна быть **абсолютно конфиденциальной** для всех сторонних наблюдателей и **строго невоспроизводимой**. Для генерации ключей на основе случайных чисел используются специальные команды **`wg genkey`** для приватного ключа и **`wg pubkey`** для открытого ключа.

Никогда не генерируйте ключи с помощью псевдослучайных последовательностей или на основе каких‑то заведомо известных данных.

Предупреждение

Строго в рамках лабораторной работы для образовательных целей можно воспользоваться заранее сгенерированными парами ключей:

Для **@client**:

- *Приватный* — **ClientPrivateKeyNeverDoThisMethodIRL1234564=**
- *Публичный* — **R2Dq51uWpvn/9wo6IweimQrMAcailb6ZMiJmqFepJmU=**

Для **@company**:

- *Приватный* — **CompanyPrivateKeyNeverDoThisMethodIRL123454=**
- *Публичный* — **fAS3DCTBIIQ3miLTLjuryy0YmZr9HiHTh2sZf9inSi4=**

Однако правильнее (и *рекомендуется*) сгенерировать на каждой машине свою пару ключей и далее пользоваться ими. При этом потребуется копирование из консоли одной ВМ в другую, что может быть невозможно при использовании VirtualBox/консолей управления. При невозможности копирования можно вводить ключи вручную или с помощью последовательных портов передать ключи между виртуальными машинами.

Настройка последовательных портов осуществляется на выключенных виртуальных машинах. При этом все настройки сети, описанные не с помощью systemd/networkd, при перезапуске необходимо будет выполнить повторно.

Для генерации ключей можно воспользоваться удобной командой/конвейером, которая выводит приватный и публичный ключи в консоль:

```
wg genkey | tee /dev/stderr | wg pubkey
```

Не пользуйтесь данной командой **в данном виде** при реальной настройке. Перенаправляйте выводы команд в файлы с ограниченными правами доступа к ним для безопасности хранения данных.

1. С помощью команды генерации ключей (или путём копирования заранее заданных пар) получите пары открытых и закрытых ключей на **@client** и **@company**.

```
[root@client ~]# wg genkey | tee /dev/stderr | wg pubkey
YHbaodB1g6Uh4rXa/Y17gf4t3pjVt68dABYDq79+tVE=
HjxgpnWK367aURR7x6sy9b8wM3UhDJCbs/5XWGWe6CU=
[root@client ~]#
```

```
[root@company ~]# wg genkey | tee /dev/stderr | wg pubkey
OFcPIl8PXbjoPozp9qBH2ZQPOUsr7Qj1ZlyZXeMnMlI=
V87kz+xm9c1liF0WQpBW33Dep2W8B++xby80SrBJ2Cc=
[root@company ~]#
```

12.3.2. Настройка WireGuard-интерфейса

Для создания соединения необходимо создать специальный WireGuard-интерфейс на каждом абоненте. В качестве приватного ключа используется **свой** приватный ключ. В описании доступных соединений указывается публичный ключ того, **к кому** инициируется подключение. Поскольку получателем соединения будет выступать **`@client`**, для него необходимо указать порт, на котором он будет ожидать соединения.

1. Опишите (или скопируйте) конфигурационный файл для *создания* интерфейса wg абонента client в соответствующем файле виртуальной машины

   `@client:/etc/systemd/network/70-wg.netdev`

   ```
   [NetDev]
   Name = wg
   Kind = wireguard

   [WireGuard]
   ListenPort = 51820
   PrivateKey = YHbaodB1g6Uh4rXa/Y17gf4t3pjVt68dABYDq79+tVE=

   [WireGuardPeer]
   AllowedIPs = 192.168.0.0/24
   PublicKey = V87kz+xm9c1liF0WQpBW33Dep2W8B++xby80SrBJ2Cc=
   ```

2. Опишите (или скопируйте) конфигурационный файл для *настройки* интерфейса wg абонента client в соответствующем файле виртуальной машины

   `@client:/etc/systemd/network/70-wg.network`

   ```
   [Match]
   Name = wg

   [Network]
   Address = 192.168.0.1/24
   ```

   Примечание

   При аналогичной настройке интерфейса на **`@company`**, поскольку отсюда будет производиться подключение, необходимо указать итоговый адрес подключения — IP-адрес и порт.

3. Опишите (или скопируйте) конфигурационный файл для *создания* интерфейса wg абонента company в соответствующем файле виртуальной машины

   `@company:/etc/systemd/network/70-wg.netdev`

   ```
   [NetDev]
   Name = wg
   Kind = wireguard

   [WireGuard]
   PrivateKey = OFcPIl8PXbjoPozp9qBH2ZQPOUsr7Qj1ZlyZXeMnMlI=
   ```

```
[WireGuardPeer]
AllowedIPs = 192.168.0.0/24
PublicKey = HjxgpnWK367aURR7x6sy9b8wM3UhDJCbs/5XWGWe6CU=
Endpoint = 10.0.1.1:51820
```

4. Опишите (или скопируйте) конфигурационный файл для *настройки* интерфейса wg абонента company в соответствующем файле виртуальной машины

   @company:/etc/systemd/network/70-wg.network

```
[Match]
Name = wg

[Network]
Address = 192.168.0.2/24
```

Примечание

Поскольку в файл с описанием интерфейса могут обратиться любые процессы/демоны или утилиты (при его создании ему автоматически были выданы права 0644), хранение там приватного ключа в чистом виде запрещено **systemd**. Для решения проблемы можно создать отдельный файл с приватным ключом и указать его в **netdev**-файл через параметр **PrivateKeyFile** или ограничить права для файла только для группы **systemd-network**.

5. С помощью команд управления правами доступа **chgrp** и **chmod** задайте права файлам создания интерфейсов wg, а после перезапустите сервис **systemd-networkd** для применения настроек

```
[root@client ~]# chgrp systemd-network /etc/systemd/network/70-wg.netdev
[root@client ~]# chmod o-r /etc/systemd/network/70-wg.netdev
[root@client ~]# systemctl restart systemd-networkd
[root@client ~]#
```

```
[root@company ~]# chgrp systemd-network /etc/systemd/network/70-wg.netdev
[root@company ~]# chmod o-r /etc/systemd/network/70-wg.netdev
[root@company ~]# systemctl restart systemd-networkd
```

6. С помощью команды **ssh <dstIP>** убедитесь в доступности SSH-соединения с company на client через VPN-соединение

```
[root@company ~]# ssh 192.168.0.1
The authenticity of host '192.168.0.1 (192.168.0.1)' can't be established.
ED25519 key fingerprint is
SHA256:BxaYoHAW5ddfM6EwmgSAZ2tKXCH0zoppLfEcQ8YiGdg.
This host key is known by the following other names/addresses:
   ~/.ssh/known_hosts:3: 10.0.12.1
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.1' (ED25519) to the list of known
hosts.
Last login: Sun Oct 19 13:45:18 2025 from 10.0.23.3
```

```
[root@client ~]#
<^D>logout
Connection to 192.168.0.1 closed.
[root@company ~]#
```

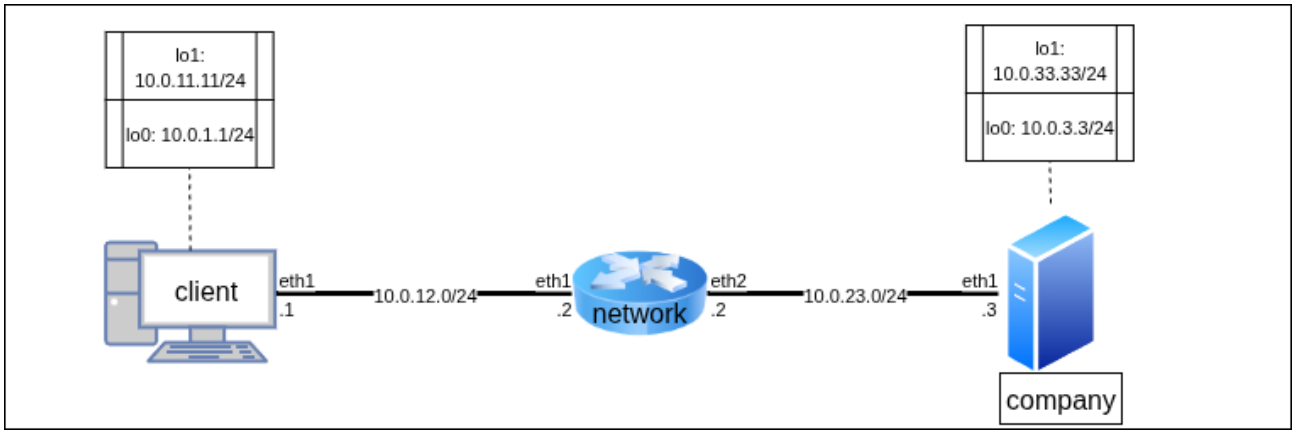
12.4. Туннелирование — IP over IP

`IP over IP`, или IP in IP — простейшая схема туннелирования, основанная на инкапсуляции одного IP‑пакета в другой IP‑пакет. С точки зрения инкапсуляции пакета данный протокол добавляет ещё один заголовок сетевого уровня к уже существующему. Для обработки таких мета‑пакетов необходимо использовать специальный виртуальный интерфейс, который будет обрабатывать специализированный трафик.

Данный вид туннелирования позволяет связать абонентов в одну локальную сеть, когда на самом деле их разделяет глобальная сеть, причём необязательно со статическим фиксированным маршрутом между абонентами. Однако важно заметить, что в данной схеме отсутствует шифрование трафика — туннель выполняет исключительно связывающую функцию.

12.4.1. Настройка обновленной топологии

Продолжим работать с имеющейся топологией, однако добавим новые виртуальные интерфейсы:



1. С помощью команд управления интерфейсами и настройки IP‑адресов настройте loopback‑интерфейсы на абонентах и обеспечьте им связность согласно топологии

```
[root@client ~]# ip link add dev lo1 type veth
[root@client ~]# ip link set lo1 up
[root@client ~]# ip addr add dev lo1 10.0.11.11/24
[root@client ~]#
```

```
[root@company ~]# ip link add dev lo1 type veth
[root@company ~]# ip link set lo1 up
[root@company ~]# ip addr add dev lo1 10.0.33.33/24
[root@company ~]#
```

```
[root@network ~]# ip route add 10.0.11.11 via 10.0.12.1
[root@network ~]# ip route add 10.0.33.33 via 10.0.23.3
[root@network ~]#
```

2. С помощью команд мониторинга сети проверьте доступность client и company друг для друга в заданных сетях

```
[root@client ~]# ping -c3 -I 10.0.11.11 10.0.33.33
PING 10.0.33.33 (10.0.33.33) from 10.0.11.11 : 56(84) bytes of data.
64 bytes from 10.0.33.33: icmp_seq=1 ttl=63 time=1.31 ms
64 bytes from 10.0.33.33: icmp_seq=2 ttl=63 time=0.594 ms
64 bytes from 10.0.33.33: icmp_seq=3 ttl=63 time=0.759 ms

--- 10.0.33.33 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.594/0.886/1.306/0.304 ms
[root@client ~]#
```

3. С помощью команды **ssh <dstIP>** убедитесь в недоступности SSH-соединения с company на client в заданных сетях

```
[root@company ~]# ssh 10.0.11.11
ssh: connect to host 10.0.11.11 port 22: Connection refused
[root@company ~]#
```

12.4.2. Настройка туннеля

Для создания туннеля необходимо просто создать виртуальный интерфейс специального типа **ipip**, связанный с некоторыми уже существующими адресами. После — добавить адреса локальной сети туннеля и использовать его для коммуникации.

1. С помощью команд управления интерфейсами и настройки IP-адресов создайте интерфейс типа **ipip** и настройте его для туннелирования

```
[root@client ~]# ip link add ipip0 type ipip remote 10.0.33.33 local
10.0.11.11
[root@client ~]# ip addr add dev ipip0 172.16.0.1/24
[root@client ~]# ip link set ipip0 up
[root@client ~]#
```

```
[root@company ~]# ip link add ipip0 type ipip remote 10.0.11.11 local
10.0.33.33
[root@company ~]# ip addr add dev ipip0 172.16.0.2/24
[root@company ~]# ip link set ipip0 up
[root@company ~]#
```

2. С помощью команды **ssh <dstIP>** убедитесь в доступности SSH-соединения с company на client в заданных сетях

```
[root@company ~]# ssh 172.16.0.1
The authenticity of host '172.16.0.1 (172.16.0.1)' can't be established.
ED25519 key fingerprint is
SHA256:BxaYoHAW5ddfM6EwmgSAZ2tKXCH0zoppLfEcQ8YiGdg.
This host key is known by the following other names/addresses:
   ~/.ssh/known_hosts:3: 10.0.12.1
   ~/.ssh/known_hosts:6: 192.168.0.1
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.16.0.1' (ED25519) to the list of known
hosts.
Last login: Mon Oct 20 22:30:47 2025
```

```
[root@client ~]#
<^D>logout
Connection to 172.16.0.1 closed.
[root@company ~]#
```

Никакого шифрования трафика в случае туннелирования не происходит. Туннель лишь *объединяет* удалённых абонентов в единую *локальную* сеть.

12.4.3. Анализ передаваемых данных

В передаваемых пакетах используются два IP‑заголовка — *внешний*, по информации из которого производится передача через публичную сеть, и *внутренний*, который, после снятия внешнего заголовка на конечном маршрутизаторе туннеля, используют для передачи пакета по локальной сети.

1. С помощью команд мониторинга сети запустите сканирование интерфейса **`eth1`** на **`@network`**

```
[root@network ~]# tcpdump -c6 -i eth1
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

2. С помощью команд мониторинга сети передайте три ICMP‑пакета по туннелю

```
[root@client ~]# ping -c3 -I 172.16.0.1 172.16.0.2
PING 172.16.0.2 (172.16.0.2) from 172.16.0.1 : 56(84) bytes of data.
64 bytes from 172.16.0.2: icmp_seq=1 ttl=64 time=0.747 ms
64 bytes from 172.16.0.2: icmp_seq=2 ttl=64 time=1.06 ms
64 bytes from 172.16.0.2: icmp_seq=3 ttl=64 time=0.796 ms

--- 172.16.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2045ms
rtt min/avg/max/mdev = 0.747/0.869/1.064/0.139 ms
[root@client ~]#
```

```
[root@network ~]# tcpdump -c6 -i eth1
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
23:01:53.464000 IP 10.0.11.11 > 10.0.33.33: IP 172.16.0.1 > 172.16.0.2: ICMP
echo request, id 4, seq 1, length 64
23:01:53.464411 IP 10.0.33.33 > 10.0.11.11: IP 172.16.0.2 > 172.16.0.1: ICMP
echo reply, id 4, seq 1, length 64
23:01:54.465498 IP 10.0.11.11 > 10.0.33.33: IP 172.16.0.1 > 172.16.0.2: ICMP
echo request, id 4, seq 2, length 64
23:01:54.465932 IP 10.0.33.33 > 10.0.11.11: IP 172.16.0.2 > 172.16.0.1: ICMP
echo reply, id 4, seq 2, length 64
23:01:55.521020 IP 10.0.11.11 > 10.0.33.33: IP 172.16.0.1 > 172.16.0.2: ICMP
echo request, id 4, seq 3, length 64
23:01:55.521376 IP 10.0.33.33 > 10.0.11.11: IP 172.16.0.2 > 172.16.0.1: ICMP
echo reply, id 4, seq 3, length 64
6 packets captured
6 packets received by filter
0 packets dropped by kernel
[root@network ~]#
```

12.5. Самостоятельная работа

12.5.1. Задание

Запустить отчёты на каждой машине и выполнить соответствующие команды.

```
report 12 client
```

1. **cat /etc/systemd/network/50-outnet.network**.
2. **cat /etc/systemd/network/70-wg.netdev**.
3. **cat /etc/systemd/network/70-wg.network**.
4. **ip a show eth1**.
5. **ip route**.
6. **python3 -m http.server**.
7. **<Завершите работу после выполнения остальных отчётов>**.

```
report 12 network
```

1. **ip a show eth1**.
2. **ip a show eth2**.
3. **ip route**.
4. **nft list ruleset**.
5. **tcpdump -xx -i eth1**.
6. **<Завершите работу после выполнения остальных отчётов>**.

```
report 12 company
```

1. **cat /etc/systemd/network/50-outnet.network**.
2. **cat /etc/systemd/network/70-wg.netdev**.
3. **cat /etc/systemd/network/70-wg.network**.
4. **ip a show eth1**.
5. **ip route**.
6. **ssh 10.0.12.1**.
7. **ssh 192.168.0.1**.
8. **ssh 172.16.0.1**.
9. **wget -t1 -T3 10.0.12.1**.
10. **wget -t1 -T3 192.168.0.1**.

11. **`wget -t1 -T3 171.16.0.1`**.

Полученные отчёты **`report.12.client`**, **`report.12.network`**, **`report.12.company`** через последовательный порт перенести из виртуальной машины и прислать их преподавателю.